

Tight Bounds for Noisy Computation of High-Influence Functions, Connectivity, and Threshold

Yuzhou Gu

New York University

YUZHOU@NYU.EDU

Xin Li

Johns Hopkins University

LIXINTS@CS.JHU.EDU

Yinzhan Xu

University of California, San Diego

XYZHAN@UCSD.EDU

Editors: Nika Haghtalab and Ankur Moitra

Abstract

The noisy query model (a.k.a. noisy decision tree model) was formally introduced by (Feige, Raghavan, Peleg and Upfal, SICOMP 1994). In this model, the (binary) return value of every query (possibly repeated) is independently flipped with some fixed probability $p \in (0, 1/2)$. The noisy query complexity (a.k.a. noisy decision tree complexity) is the smallest number of queries an algorithm needs to make to achieve error probability $\leq \frac{1}{3}$.

Previous works often focus on specific problems, and it is of great interest to have a characterization of noisy query complexity for general functions. We show that any Boolean function with total influence $\Omega(n)$ has noisy query complexity $\Omega(n \log n)$, matching the trivial upper bound of $O(n \log n)$. Our result is the first noisy query complexity lower bound of this generality, beyond what was known for random Boolean functions (Reischuk and Schmeltz, FOCS 1991).

Our second contribution is to determine the asymptotic noisy query complexity of the Graph Connectivity problem. In this problem, the goal is to determine whether an undirected graph is connected, where each query asks for the existence of an edge in the graph. A simple algorithm can solve the problem with error probability $o(1)$ using $O(n^2 \log n)$ noisy queries, but no non-trivial lower bounds were known prior to this work. We show that any algorithm that solves the problem with error probability $\leq \frac{1}{3}$ uses $\Omega(n^2 \log n)$ noisy queries.

For the proofs of the above lower bounds, we develop a three-phase method, which is a refinement of the two-phase method of (Feige, Raghavan, Peleg and Upfal, SICOMP 1994). Our three-phase method adds an extra step where carefully designed “free” information is disclosed to the algorithm. This way, we can precisely control the posterior distribution of the input given the query results, empowering a more refined analysis.

Last but not least, we determine the exact number of noisy queries (up to lower order terms) needed to solve the k -Threshold problem and the Counting problem. The k -Threshold problem asks to decide whether there are at least k ones among n bits, given noisy query access to the bits. We prove that $(1 \pm o(1)) \frac{n \log(\min\{k, n-k+1\}/\delta)}{(1-2p) \log \frac{1-p}{p}}$ queries are both sufficient and necessary to achieve error probability $\delta = o(1)$. Previously, such a result was only known when $\min\{k, n-k+1\} = o(n)$ (Wang, Ghaddar, Zhu and Wang, ALT 2025). We also show a similar $(1 \pm o(1)) \frac{n \log(\min\{k+1, n-k+1\}/\delta)}{(1-2p) \log \frac{1-p}{p}}$ bound for the Counting problem, where one needs to count the number of ones among n bits given noisy query access and k denotes the answer.

Keywords: noisy computation, total influence, connectivity, k -threshold, counting

1. Introduction

Fault tolerance is a crucial feature of algorithms that work on large systems, as errors are unavoidable. Hence, previous studies have considered various models to capture the effect of errors, such as the Rényi–Ulam game (Rényi, 1961; Ulam, 1976), independent noise models (Feige et al., 1994), and independent noise models without resampling (Braverman and Mossel, 2008).

(Feige et al., 1994) formally proposed the noisy query model with independent noise, which they call the noisy Boolean decision tree model or the noisy comparison tree model, depending on whether the problem uses point queries to input bits or comparison queries between input elements. In this model, each query returns one bit that is independently flipped with some fixed probability $p \in (0, \frac{1}{2})$ (i.e., independent noise), and repeated queries are allowed. The efficiency of an algorithm is measured in terms of the number of queries it makes. (Feige et al., 1994) showed tight asymptotic bounds for the noisy query complexity for a wide range of problems, including Parity, Threshold, Binary Search and Sorting.

In fact, researchers had studied queries with independent noise even before (Feige et al., 1994) formally defined the model. (Berlekamp, 1964; Horstein, 1963; Burnashev and Zigangirov, 1974) all studied some versions of Binary Search under independent noise. In particular, (Berlekamp, 1964; Horstein, 1963) studied the problem through the lens of channel coding. See (Wang et al., 2022) for a more detailed discussion about the relationship between the channel coding perspective and the noisy query perspective.

Furthermore, many learning problems involve queries with unreliable answers, thus can be naturally considered under the noisy query model. For example, the field of active learning studies problems such as Top- k Selection (e.g., (Mohajer et al., 2017)) in a slightly more general noise model. The Best-Arm Identification problem in Multi-Armed Bandits (e.g., (Audibert et al., 2010)) can also be interpreted under the noisy query model, as observed in (Gu et al., 2025). Noisy group testing (e.g., (Scarlett, 2018)) is yet another notable problem that can be considered as a noisy query problem. These examples further demonstrate that the noisy query model is a natural model to study and have wide applications in learning theory and computer science.

Following (Feige et al., 1994), researchers have studied problems in the noisy query model extensively, including random functions (Reischuk and Schmeltz, 1991; Feige, 1992; Evans and Pippenger, 1998), k -CNF and k -DNF (Kenyon and King, 1994), Binary Search (Ben-Or and Hassidim, 2008; Dereniowski et al., 2021; Gu and Xu, 2023; Zhu et al., 2023b), Sorting (Wang et al., 2022, 2023; Gu and Xu, 2023; Zhu et al., 2023b), Graph Search (Emamjomeh-Zadeh et al., 2016; Dereniowski et al., 2019, 2021) (a generalization of Binary Search), and k -Threshold (Zhu et al., 2023b,a; Wang et al., 2025).

However, despite the popularity and naturality of the model, most research on the noisy query model focus on specific functions instead of general functions. In the above examples, the only exceptions are the lower bounds for random functions (Reischuk and Schmeltz, 1991; Feige, 1992; Evans and Pippenger, 1998), and upper bounds for k -CNF and k -DNF (Kenyon and King, 1994). Furthermore, the specific functions studied in literature are often the ones studied already in (Feige et al., 1994) such as Threshold, Binary Search and Sorting. As a result, the noise query complexity of many important and natural problems are left unexplored. In this paper, we take a first step towards studying the noise query complexity of more general functions and problems.

1.1. High-influence functions

Our first result is a lower bound for the noisy query complexity of high-influence functions, a greatly general family of functions. This is the first result towards understanding the lower bound for general Boolean functions, beyond the lower bound for random functions (Reischuk and Schmeltz, 1991; Feige, 1992; Evans and Pippenger, 1998).

Influence is a central quantity in the analysis of Boolean functions. For a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, the influence of coordinate $i \in [n]$ is defined as

$$\text{Inf}_i(f) = \mathbb{P}_{x \sim \{0,1\}^n} (f(x) \neq f(x \oplus e_i)),$$

where e_i denotes the bit string where the only 1 is in the i -th coordinate, and \oplus denotes exclusive or. That is, $\text{Inf}_i(f)$ is the probability that flipping the i -th coordinate of a uniformly random bit string also flips the function value. The *total influence* is the sum of influences over all coordinates, i.e.,

$$I(f) = \sum_{i \in [n]} \text{Inf}_i(f).$$

We prove that Boolean functions with linear total influence have noisy query complexity $\Omega(n \log n)$.

Theorem 1 (Noisy query complexity of high-influence functions) *For any $c > 0$, there exists $c' > 0$ such that the following holds. For any Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ with $I(f) \geq cn$, any noisy query algorithm computing $f(x)$ with error probability $\leq \frac{1}{3}$ makes at least $c'n \log n$ noisy queries in expectation to the coordinates of the input $x \in \{0, 1\}^n$.*

Note that the error probability $\frac{1}{3}$ can be replaced with any $0 < \epsilon < \frac{1}{2}$ without affecting the asymptotic noisy query complexity. The statement is tight in the sense that any Boolean function on n inputs can be computed with error probability $o(1)$ using $O(n \log n)$ noisy queries: by simply querying each bit $O(\log n)$ times, we can determine the input string with $o(1)$ error probability.

Theorem 1 unifies and generalizes several previous results. For example, it was known that a random Boolean function (with probability $1 - o(1)$) has noisy query complexity $\Omega(n \log n)$ (Reischuk and Schmeltz, 1991; Feige, 1992; Evans and Pippenger, 1998), and computing the parity function requires $\Omega(n \log n)$ noisy queries (Feige et al., 1994). As random Boolean functions and the parity function have total influence $\Omega(n)$, Theorem 1 immediately implies these lower bounds as special cases.

Another central notion related to influence is the sensitivity of Boolean functions¹, as it is well known that the total influence is the same as the *average sensitivity* (the expected sensitivity over a uniformly random input). (Reischuk and Schmeltz, 1991) proved that any non-adaptive algorithm computing a Boolean function f makes at least $\Omega(s(f) \log s(f))$ noisy queries, where $s(f)$ is the (maximum) sensitivity of f . This result is incomparable to Theorem 1, as their lower bound only holds against non-adaptive algorithms. In fact, it is not possible to extend the $\Omega(s(f) \log s(f))$ lower bound against adaptive algorithms in general. For instance, the OR function has sensitivity n and (adaptive) noisy query complexity $O(n)$ (Feige et al., 1994). On the other hand, the average sensitivity of OR is much smaller, which suggests that the average sensitivity of a Boolean function f is a more reasonable measure for lower bounding the *adaptive* noisy query complexity. This

1. The sensitivity of a Boolean function f at input x , denoted by $s(f, x)$, is the number of bits i for which $f(x) \neq f(x \oplus e_i)$.

motivates us to raise the following open question regarding a lower bound for general Boolean functions.

Open Problem 1 *Is it true that every Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ has noisy query complexity $\Omega(I(f) \log I(f))$?*

[Theorem 1](#) resolves the case where $I(f) = \Omega(n)$. We note further evidence supporting the $I(f) \log I(f)$ lower bound. The randomized query complexity $R(f)$ satisfies $R(f) = \Omega(\text{bs}(f)) = \Omega(\text{s}(f)) = \Omega(I(f))$, where $\text{bs}(f)$ denotes block sensitivity and the first step is by ([Nisan, 1989](#), Lemma 4.2). In general, the noisy query complexity of a function f is always between $R(f)$ and $O(R(f) \log R(f))$. Therefore, the $\Omega(I(f) \log I(f))$ lower bound is consistent with these known relationships.

For the proof of [Theorem 1](#), we develop a three-phase lower bound framework, which is based on and refines the two-phase method of ([Feige et al., 1994](#)) for proving a lower bound for the k -Threshold problem. In the three-phase method, we reduce the original problem in the noisy query model to a stronger observation model, where in Phase 1 the algorithm makes non-adaptive noisy observations and in Phase 3 the algorithm makes adaptive exact observations. In Phase 2, the model gives away free information, which can only help the algorithm. By designing this free information carefully, the effect of Phase 1 and 2 combined can be significantly simplified, allowing for a precise analysis in Phase 3.

We note that this idea of giving away free information already appears in ([Feige et al., 1994](#))’s two-phase method. For their problem (k -Threshold), this free information is relatively simple. However, for other problems, the free information could be significantly more involved. We design the free information in a different way in order for the analysis in Phase 3 to be viable. This additional phase to the original two-phase method makes it easier to apply and allows for other applications. As we will soon discuss, the three-phase framework is essential for our result on Graph Connectivity and also leads to a simple proof for the lower bound of k -Threshold.

1.2. Graph Connectivity

Although the noisy query model is quite natural, there has been little prior work studying graph problems in this model. Some prior examples include ([Feige et al., 1994](#)), which briefly mentioned that a lower bound for the noisy query complexity of Bipartite Matching can be achieved by reducing from the other problems they studied; ([Kenyon and King, 1994](#)) designed algorithms for k -CNF and k -DNF using a small number of queries, which imply, for instance, that one can test, up to error probability δ , whether a given n -vertex graph contains a triangle using $O(n^2 \log \frac{1}{\delta})$ noisy queries.

One of the most fundamental graph problems is Graph Connectivity, where we are given an n -vertex undirected graph G , and need to determine whether the graph is connected via edge queries. It is a very basic problem that appears in most first courses in algorithms. For instance, breadth-first search and depth-first search are usually among the first graph algorithms taught in undergraduate algorithm classes, and the simplest application of them is to detect whether a graph is connected. However, to our surprise, we do not even have a good understanding of the noisy query complexity of such an elementary problem.

One simple algorithm for Graph Connectivity is to query every edge in the input graph $O(\log n)$ times to correctly compute the input graph with high probability, and then solve Graph Connectivity on the computed graph. This naive algorithm uses $O(n^2 \log n)$ noisy queries, and is essentially all

what was previously known about Graph Connectivity in the noisy query model. In particular, hardness of Graph Connectivity does not seem to follow from known hardness results.

Using the three-phase method, we prove an $\Omega(n^2 \log n)$ lower bound on the noisy query complexity of Graph Connectivity, showing that the naive $O(n^2 \log n)$ algorithm is actually optimal up to a constant factor:

Theorem 2 (Hardness of Graph Connectivity) *Any algorithm solving the Graph Connectivity problem with error probability $\leq \frac{1}{3}$ uses $\Omega(n^2 \log n)$ noisy queries in expectation.*

Similarly as before, the error probability $\frac{1}{3}$ can be replaced with any $0 < \epsilon < \frac{1}{2}$ without affecting the asymptotic noisy query complexity.

We also show an $\Omega(n^2 \log n)$ lower bound for the related s - t Connectivity problem, where we are given an n -vertex undirected graph G and two fixed vertices s and t , and the goal is to determine whether there is a path in the graph connecting s and t .

As Graph Connectivity and s - t Connectivity are very basic tasks on graphs, their lower bounds immediately imply lower bounds for several other fundamental graph problems as well. For instance, given the lower bounds for Graph Connectivity and s - t Connectivity, it is straightforward to show that Global Min-Cut, s - t Shortest Path, and s - t Max Flow on unweighted undirected graphs all require $\Omega(n^2 \log n)$ noisy queries in expectation.

1.3. Threshold and Counting

In the k -Threshold problem, one is given a length- n Boolean array a and an integer k , and the goal is to determine whether the number of 1's in the array a is at least k . Note that the answer to the input is false if and only if the number of 0's in the input is at least $n - k + 1$. We can thus solve k -Threshold using an algorithm for $(n - k + 1)$ -Threshold: we can flip all input bits, change k to $n - k + 1$, solve the modified instance, and finally flip the result. Therefore, we can assume without loss of generality that $k \leq n - k + 1$, or equivalently, $k \leq (n + 1)/2$.

k -Threshold is one of the first problems studied in the noisy query model. In (Feige et al., 1994), it was shown that $\Theta(n \log \frac{k}{\delta})$ queries are both sufficient and necessary to solve the problem with error probability δ . However, the optimal constant factor was left unknown.²

There has been some progress towards determining the exact constant for k -Threshold. In (Zhu et al., 2023a), it was shown that the noisy query complexity of the OR function on n input bits (equivalent to 1-Threshold) is

$$(1 \pm o(1)) \frac{n \log \frac{1}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}$$

for $\delta = o(1)$, where $D_{\text{KL}}(p \parallel 1 - p) = (1 - 2p) \log \frac{1-p}{p}$ is the Kullback-Leibler divergence between two Bernoulli distributions with heads probabilities p and $1 - p$. This result was later generalized to k -Threshold for all $k = o(n)$ and $\delta = o(1)$ by (Wang et al., 2025), who showed an

$$(1 \pm o(1)) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}$$

2. Studying constant factors is often overlooked in theoretical computer science, but in this research area, determining the optimal constants for noisy query complexities of other fundamental problems such as Binary Search and Sorting has been an active topic (e.g., (Burnashev and Zigangirov, 1974; Ben-Or and Hassidim, 2008; Dereniowski et al., 2021; Gu and Xu, 2023)). See (Gretta and Price, 2024) for more discussions on the importance of studying constants in noisy query complexity.

bound. Compared to (Zhu et al., 2023a), (Wang et al., 2025)’s result works for a much wider range of k . However, their lower bound proof technique unfortunately stops working for the case $k = \Theta(n)$, and this case is frustratingly left open (we remark that their algorithm gives the right upper bound even for $k = \Theta(n)$).

In this work, we complete the last piece of the puzzle, showing a matching bound for all values of k .

Theorem 3 (Noisy query complexity of k -Threshold) *For any $1 \leq k \leq (n+1)/2$ and $\delta = o(1)$, computing k -Threshold on a length- n array with error probability δ needs and only needs*

$$(1 \pm o(1)) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \parallel 1-p)}$$

noisy queries in expectation.

Here the $\delta = o(1)$ assumption is standard and has appeared in several previous works (Gu and Xu, 2023; Zhu et al., 2023a; Wang et al., 2025).

While (Wang et al., 2025) has already given an algorithm achieving the tight upper bound for any k , their algorithm involves calling some extra algorithms such as Noisy Sorting and Noisy Heap, which seems too heavy and unnecessary for the k -Threshold problem. After all, in the classic noiseless setting, the algorithm for k -Threshold is much simpler than algorithms for Sorting or Heap. We provide a much simpler algorithm which involves only checking each bit one by one and completely avoids calling these extra algorithms.

We also provide an alternative and simpler proof of the lower bound for k -Threshold with $k = o(n)$. The proof of (Wang et al., 2025) considers three cases and uses two different methods (the two-phase method from (Feige et al., 1994) and Le Cam’s two point method) for solving them. We show that this casework is unnecessary by providing a uniform and simple proof for all $k = o(n)$ by using our three-phase method.

We also consider a related problem, Counting, where we need to compute the number of 1’s in n input Boolean bits. The lower bound for k -Threshold easily applies to Counting as well (though in a non-black-box way). In addition, we design an algorithm for Counting that matches the lower bound, obtaining the following result.

Theorem 4 (Noisy query complexity of Counting) *Given a sequence $a \in \{0, 1\}^n$, computing $\|a\|_1$ with error probability $\delta = o(1)$ needs and only needs*

$$(1 \pm o(1)) \frac{n \log \frac{\min\{\|a\|_1, n - \|a\|_1\} + 1}{\delta}}{D_{\text{KL}}(p \parallel 1-p)}$$

noisy queries in expectation.

A problem closely-related to k -Threshold is the k -Selection problem, where one is given n items (comparable with each other) and the goal is to select the k -th largest element using noisy comparison queries. It is known that solving k -Selection with error probability $\delta = o(1)$ needs and only needs $\Theta\left(n \log \frac{\min\{k, n-k+1\}}{\delta}\right)$ noisy queries (Feige et al., 1994). Their bounds are only tight up to a constant factor, so the exact value of the leading coefficient remains open.

Open Problem 2 *Determine the exact constant c such that $(c \pm o(1))n \log \frac{\min\{k, n-k+1\}}{\delta}$ noisy queries is both sufficient and necessary to solve the k -Selection problem with error probability $\delta = o(1)$.*

2. Technical overview

2.1. Proof overview for high-influence functions

Recall that the error probability $\frac{1}{3}$ in the statement of [Theorem 1](#) can be replaced with any $0 < \epsilon < \frac{1}{2}$ without loss of generality. Also, the expected number of queries can be replaced by the worst-case number of queries by Markov's inequality.

Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a Boolean function with $I(f) = \Omega(n)$. The hard distribution for the input x will be the uniform distribution over $\{0, 1\}^n$.

Inspired by ([Feige et al., 1994](#)), we prove hardness under the noisy query model by introducing a new problem where the algorithm has more power, and prove hardness of this new problem.

The new problem has three phases, described as follows.

1. In Phase 1, the algorithm makes $m_1 = c_1 \log n$ noisy queries to coordinate x_i for every $i \in [n]$.
2. In Phase 2, the oracle reveals some coordinates of x to the algorithm.
3. In Phase 3, the algorithm makes $m_2 = c_2 n$ adaptive exact queries for some constant c_2 .

The goal of the algorithm is to compute the value $f(x)$.

Note that the first two phases are non-adaptive. The third phase is adaptive but the algorithm makes exact queries. This is the reason why the three-phase problem is easier to analyze than the original noisy query problem.

It is not hard to prove that if no algorithm can solve the three-phase problem with ϵ error probability, then no algorithm can solve the original problem with ϵ error probability using no more than $m_1 m_2 = c_1 c_2 n \log n$ noisy queries. Therefore we only need to prove hardness of the three-phase problem.

Phase 1. Let a_i be the number of times where a query to x_i returns 1 in Phase 1. The posterior distribution of the input x given observations made in Phase 1 depends only on the variables a_i . In other words, $(a_i)_{i \in [n]}$ is a sufficient statistic for x . Conditioned on x , the variables $(a_i)_{i \in [n]}$ are independent, and the distribution of each a_i is a binomial distribution depending only on whether $x_i = 1$. That is, if $x_i = 1$, then $a_i \sim \text{Bin}(m_1, 1 - p)$; otherwise, $a_i \sim \text{Bin}(m_1, p)$ ($\text{Bin}(\cdot, \cdot)$ denotes the binomial distribution). Moreover, for $x_i = 0$, a_i is in an interval I around pm_1 with probability $1 - o(1)$; for $x_i = 1$, a_i is in the interval I with probability $n^{-c_3 \pm o(1)}$, for some constant $c_3 > 0$ depending on c_1 and p . Because the observations are independent for different coordinates, each index has a non-negative weight, such that the posterior probability of the input being x is proportional to the product of weights of the coordinates i where $x_i = 1$.

Phase 2. In Phase 2, the oracle reveals some coordinates of x to the algorithm. This information is revealed in two steps:

- 2a. All coordinates with $a_i \notin I$ are revealed.
- 2b. Every x_i with $a_i \in I$ is revealed independently with probability q_{a_i} for some real numbers $(q_k \in [0, 1])_{k \in I}$.

Because of the observations in Phase 1, the unrevealed coordinates can have different weights. That is, given observations up to Step 2a, the posterior probabilities for different coordinates being 1 can

be different, which is undesirable. Step 2b is a subsampling procedure, with the goal of reweighting the unrevealed coordinates so that all of them have the same weight. If the interval I is not too large, then the probabilities q_k for $k \in I$ will not be too small. Because observations made up to Step 2b are independent for different coordinates, they have the same effect as the following procedure (if the real numbers q_k are chosen carefully): every x_i with $x_i = 1$ is revealed independently with probability $p_+ = 1 - n^{-c_3 \pm o(1)}$ and x_i with $x_i = 0$ is revealed independently with probability $p_- = o(1)$.

Phase 3. At this stage, let the set of unrevealed coordinates be U . Conditioned on the revealed coordinates, x_i for $i \in U$ are i.i.d. $\text{Ber}(q)$ variables, where $\text{Ber}(q)$ denotes the Bernoulli distribution with head probability $q = \frac{1-p_+}{1-p_++1-p_-} = n^{-c_3 \pm o(1)}$. In other words, the distribution of x_U is $\text{Ber}(q)^{\otimes U}$. Let g be a restriction of f where the revealed coordinates of x are fixed to be the revealed values. Then we need to show that on average, computing g with error probability $\leq \epsilon$ requires $\Omega(n)$ (adaptive) exact queries, for some sufficiently small $\epsilon > 0$. To this end, we consider a biased version of total influence l_q :

$$l_q(g) = \sum_{i \in U} \mathbb{P}_{x \sim \text{Ber}(q)^{\otimes U}}(g(x) \neq g(x \oplus e_i)).$$

Our proof strategy consists of the following three steps.

- 3a. First, we show that $l_q(g) = \Omega(n)$ in expectation.
- 3b. After we make a query, we further fix the value of the queried coordinate, and replace g with the new restricted function. We show that each exact query can only decrease $l_q(g)$ by at most 1 in expectation.
- 3c. Finally, we show that if $l_q(g) = \Omega(n)$, then it is impossible to guess $g(x)$ (where x follows a biased product distribution) with error probability $\leq \epsilon$.

Combining the three steps, we obtain that making $c_2 n$ exact queries (in expectation) cannot compute g with $\leq \epsilon$ error probability, for sufficiently small c_2 and $\epsilon > 0$.

We note that Step 3c is where this approach fails to extend to general functions with sublinear total influence. In fact, for $l_q(g) = o(n)$, it might be possible to guess $g(x)$ (where x follows a biased product distribution) with $o(1)$ error probability. For instance, if $g(x)$ is a function where $o(1)$ random fraction of the values are 0, and the other values are 1, it is straightforward to show that $l_q(g) = o(n)$ (in expectation), and we can guess $g(x) = 1$ to achieve $o(1)$ error probability, without using any queries.

2.2. Proof overview for Graph Connectivity

In this section we present an overview of our proof of [Theorem 2](#), hardness of Graph Connectivity. Again, the error probability $\frac{1}{3}$ can be replaced with any $0 < \epsilon < \frac{1}{2}$ and the expected number of queries can be replaced by the worst-case number of queries, without loss of generality.

Hard distribution. To prove the lower bound, we design a hard distribution of inputs. The distribution is based on uniform random spanning trees (USTs) of the complete graph. Let T be a UST. We say an edge $e \in T$ is β -balanced if both components of $T \setminus e$ have size at least βn . Let e be a

uniform random β_0 -balanced edge of T , where $\beta_0 = \frac{1}{21}$. (If such e does not exist, we restart.) We throw a fair coin and decide whether to erase edge e from T . That is, conditioned on (T, e) , the input graph G is T (connected) with probability $\frac{1}{2}$, and is $T \setminus e$ (disconnected) with probability $\frac{1}{2}$.

Three-phase problem. Following the three-phase method, it suffices to show the hardness of the three-phase problem described as follows.

1. In Phase 1, the algorithm makes $m_1 = c_1 \log n$ noisy queries to every unordered pair (called “potential edge”) $(u, v) \in \binom{V}{2}$ for some constant c_1 .
2. In Phase 2, the oracle reveals some edges and non-edges of G to the algorithm.
3. In Phase 3, the algorithm makes $m_2 = c_2 n^2$ adaptive exact queries for some constant c_2 .

The goal of the algorithm is to determine whether the graph is connected.

Phase 1. This phase is similar to Phase 1 in [Section 2.1](#). In Phase 1, the algorithm makes m_1 noisy queries to every potential edge $e \in \binom{V}{2}$. Let a_e be the number of times where a query to a potential edge e returns 1 in Phase 1. Similar to [Section 2.1](#), if $e \in G$, then $a_e \sim \text{Bin}(m_1, 1 - p)$; otherwise $a_e \sim \text{Bin}(m_1, p)$. Specifically, for $e \notin G$, a_e is in an interval I around pm_1 with probability $1 - o(1)$; for $e \in G$, a_e is in the interval I with probability $n^{-c_3 \pm o(1)}$, for some constant $c_3 > 0$ depending on c_1 and p . Because the observations are independent for different edges, each edge has a non-negative weight, such that the posterior probability of $G = T$ for a tree T is proportional to the product of weights of edges in T .

Phase 2. In Phase 2, the oracle reveals some edges and non-edges of G . This information is revealed in three steps (the first two steps are similar to those in [Section 2.1](#)).

- 2a. In Step 2a, the potential edges e with $a_e \notin I$ are revealed. (Recall that I is an interval around pm_1 .) That is, the algorithm now knows which potential edges e with $a_e \notin I$ are edges.
- 2b. In Step 2b, every edge e with $a_e \in I$ is revealed independently with probability q_{a_e} , for some real numbers $(q_k \in [0, 1])_{k \in I}$.
- 2c. In Step 2c, $n - 2$ edges are revealed as follows. If G is disconnected, reveal all edges of G ; otherwise, if there is a β_0 -balanced edge that has not been revealed so far, uniformly randomly choose one (say e^*) from all such edges, and reveal all edges of $G \setminus e^*$. If G is connected but all β_0 -balanced edges have been revealed, report failure.

Similar to [Section 2.1](#), if the real numbers q_k are chosen carefully, the observations made up to Step 2b have the same effect as the following procedure: observe every edge independently with probability $p_+ = 1 - n^{-c_3 \pm o(1)}$; observe every non-edge independently with probability $p_- = o(1)$.

We can show that Step 2c reports failure with $1 - \Omega(1)$ probability. In the following, we condition on the event that the oracle does not report failure in Step 2c. In this case, Step 2c reveals all except for one edge, which forms two connected components T_1 and T_2 . By the construction, T_1 and T_2 both have size at least $\beta_0 n$. The posterior distribution is supported on $\{T_1 \cup T_2, T_1 \cup T_2 \cup \{e\} : e \in E(T_1, T_2)\}$. In our full analysis, we compute an exact formula for the posterior probability of every graph in the support.

Phase 3. In Phase 3 the algorithm makes $m_2 = c_2 n^2$ adaptive exact queries. Let $G_0 = T_1 \cup T_2$, $G_e = T_1 \cup T_2 \cup \{e\}$ for $e \in E(T_1, T_2)$. Let $\mathbb{P}^{(2)}$ denote the posterior distribution of G after Phase 2. Because T_1 and T_2 are already revealed, we can w.l.o.g. assume that the algorithm makes queries only to edges in $E(T_1, T_2)$. We prove that after Phase 2, for most edges $e \in E(T_1, T_2)$, we have

$$\frac{\mathbb{P}^{(2)}(G_e)}{\mathbb{P}^{(2)}(G_0)} = \Theta\left(\frac{1}{|T_1||T_2|}\right) = \Theta\left(\frac{1}{n^2}\right)$$

Let $E^{(3)}$ be the set of potential edges queried in Phase 3. For small enough c_2 , we have

$$\sum_{e \in E^{(3)}} \mathbb{P}^{(2)}(G_e) = O(1) \cdot \mathbb{P}^{(2)}(G_0).$$

Therefore, with constant probability, all queries in Phase 3 return 0. Furthermore, if this happens, then the posterior probability of G_0 and $\{G_e : e \in E(T_1, T_2) \setminus E^{(3)}\}$ are within a constant factor of each other. In this situation, outputting anything will result in a constant error probability. This concludes that for some $\epsilon > 0$, no algorithm can solve the three-phase problem with error probability ϵ .

2.3. Proof overview for k -Threshold and Counting

Before we discuss our techniques for k -Threshold and Counting, we briefly discuss the previous work of (Wang et al., 2025), who showed a

$$(1 \pm o(1)) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \parallel 1-p)}$$

bound for k -Threshold where $k = o(n)$. Their lower bound stops working for $k = \Theta(n)$ because one step in their lower bound proof reveals the locations of $k - 1$ 1's to the algorithm, leaving only $n - k + 1 = (1 - \Omega(1))n$ unknown bits. This case can be solved by an algorithm using $(1 - \Omega(1)) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \parallel 1-p)}$ noisy queries, meaning that this approach cannot be used to show a tight lower bound.

Lower bound for k -Threshold. Our lower bound for k -Threshold for general values of k is a reduction from (Wang et al., 2025)'s lower bound for $k = o(n)$. In the overview, we focus on the case where $k = (n + 1)/2$ for odd n . In this case, the problem is equivalent to computing the majority of n input bits.

Given any instance of k -Threshold on a length- n array for $k = n/\log n$, we first add L artificial 1's to the array to obtain a new instance where $n' = n + L$ and $k' = k + L$. We set L so that $k' = (n' + 1)/2$ (or equivalently, $L = n - 2k + 1$). Now suppose we have an algorithm for the new instance that uses only $(1 - \epsilon) \frac{n' \log \frac{k'}{\delta}}{D_{\text{KL}}(p \parallel 1-p)}$ noisy queries for some $\epsilon > 0$. Whenever the algorithm queries an artificial 1, it can be simulated without incurring an actual noisy query; instead, we only need to flip biased coin with head probability $1 - p$ and return its value. Because the algorithm is for computing the majority, intuitively, by symmetry, the expected number of queries it spends on an input 0 and an input 1 should be the same. Furthermore, if we add the artificial 1's to random positions, the algorithm should not be able to distinguish an artificial 1 with an actual 1. Therefore,

in expectation, L/n' fraction of the algorithm's queries are to artificial 1's, so the actual query complexity for solving the original k -Threshold instance is

$$\left(1 - \frac{L}{n'}\right) \cdot (1 - \epsilon) \frac{n' \log \frac{k'}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} = (1 - \epsilon) \frac{n \log \frac{k'}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}.$$

Because $k = n/\log n$, we have $\log k' \leq \log(n+1) = (1+o(1)) \log k$, so the above bound becomes

$$(1 - \epsilon + o(1)) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)},$$

which contradicts the lower bound from (Wang et al., 2025).

Our lower bound for more general values of k is proved using a similar idea. However, we no longer have the symmetry between 0 and 1, so we need to reduce from the case $k = n/\log n$ or $k = (n+1)/2$ depending on whether the algorithm spends more queries on an input 0 or 1.

Upper bound for k -Threshold. For the upper bound, (Wang et al., 2025)'s algorithm already works also for the $k = \Theta(n)$ case. Nevertheless, we provide a much simpler algorithm that achieves the same tight upper bound. (Wang et al., 2025) used a standard CHECK-BIT procedure to estimate the value of each input bit, and then used established machinery on Noisy Sorting and Noisy Heap studied earlier (Feige et al., 1994). In comparison, our algorithm uses an asymmetric version of the CHECK-BIT procedure that estimates the value of each input bit. Using this asymmetric procedure, we essentially only need to check each input bit one by one, avoiding calling extra algorithms such as Noisy Sorting or Noisy Heap.

Upper bound for Counting. Our algorithm for Counting is based on the idea of our algorithm for k -Threshold. Our algorithm for k -Threshold can additionally count the number of 1's in the input if it is at most k , so one natural idea for Counting is to first compute an estimation k of the answer $\|a\|_1$, then use our algorithm for k -Threshold to compute the exact answer. However, this approach does not work when $\|a\|_1$ is very small compared to n , as there is no reliable way to estimate the answer within a constant factor using $o\left(n \frac{\log \frac{\|a\|_1}{\delta}}{D_{\text{KL}}(p \parallel 1-p)}\right)$ queries. We circumvent this issue by gradually increasing k during the algorithm and simulate the asymmetric CHECK-BIT procedure on each input bit. We can view the asymmetric CHECK-BIT procedure for each bit as a biased random walk on \mathbb{Z} , and for different k the procedure only changes the stopping condition, but not the random walk. In this way we show that k will eventually stop at the correct answer with desired error probability.

Acknowledgments

Part of the work was done while Y. G. was supported by the National Science Foundation under Grant No. DMS-1926686. X. L. is supported by NSF CAREER Award CCF-1845349 and NSF Award CCF-2127575. Y. X. is supported by NSF Grant CCF-2330048, HDR TRIPODS Phase II grant 2217058 and a Simons Investigator Award.

References

David Aldous. The Continuum Random Tree. I. *The Annals of Probability*, 19(1):1 – 28, 1991. doi: 10.1214/aop/1176990534. URL <https://doi.org/10.1214/aop/1176990534>.

- Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *Conference on Learning Theory*, 2010.
- Michael Ben-Or and Avinatan Hassidim. The bayesian learner is optimal for noisy binary search (and pretty good for quantum as well). In *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 221–230, 2008. doi: 10.1109/FOCS.2008.58.
- Elwyn R. Berlekamp. *Block coding with noiseless feedback*. PhD thesis, Massachusetts Institute of Technology, 1964.
- Mark Braverman and Elchanan Mossel. Noisy sorting without resampling. In *Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 268–276, 2008.
- M. V. Burnashev and K. Sh. Zigangirov. An interval estimation problem for controlled observations. *Probl. Peredachi Inf.*, 10(3):51–61, 1974.
- Dariusz Dereniowski, Stefan Tiegel, Przemysław Uznański, and Daniel Wolleb-Graf. A framework for searching in graphs in the presence of errors. In *Proceedings of the 2nd Symposium on Simplicity in Algorithms (SOSA)*, volume 69 of *OASICS*, pages 4:1–4:17, 2019. doi: 10.4230/OASICS.SOSA.2019.4. URL <https://doi.org/10.4230/OASICS.SOSA.2019.4>.
- Dariusz Dereniowski, Aleksander Łukasiewicz, and Przemysław Uznański. Noisy searching: simple, fast and correct. *arXiv preprint arXiv:2107.05753*, 2021. doi: 10.48550/arXiv.2107.05753.
- Ehsan Emamjomeh-Zadeh, David Kempe, and Vikrant Singhal. Deterministic and probabilistic binary search in graphs. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 519–532, 2016. doi: 10.1145/2897518.2897656. URL <https://doi.org/10.1145/2897518.2897656>.
- William Evans and Nicholas Pippenger. Average-case lower bounds for noisy boolean decision trees. *SIAM J. Comput.*, 28(2):433–446, 1998.
- Uriel Feige. On the complexity of finite random functions. *Inf. Process. Lett.*, 44(6):295–296, 1992.
- Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with noisy information. *SIAM J. Comput.*, 23(5):1001–1018, 1994. doi: 10.1137/S0097539791195877.
- William Feller. *An Introduction to Probability Theory and Its Applications, 3rd Edition*. Wiley, 1970.
- Lucas Gretta and Eric Price. Sharp noisy binary search with monotonic probabilities. In *Proceedings of the 51st International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 297, pages 75:1–75:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024. doi: 10.4230/LIPICS.ICALP.2024.75. URL <https://doi.org/10.4230/LIPICS.ICALP.2024.75>.
- Yuzhou Gu and Yinzhan Xu. Optimal bounds for noisy sorting. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1502–1515, 2023.

- Yuzhou Gu, Yanjun Han, and Jian Qian. Evolution of information in interactive decision making: A case study for multi-armed bandits. *arXiv preprint arXiv:2503.00273*, 2025.
- Michael Horstein. Sequential transmission using noiseless feedback. *IEEE Trans. Inf. Theory*, 9(3): 136–143, 1963. doi: 10.1109/TIT.1963.1057832.
- Claire Kenyon and Valerie King. On boolean decision trees with faulty nodes. *Random Struct. Algorithms*, 5(3):453–464, 1994. doi: 10.1002/RSA.3240050306. URL <https://doi.org/10.1002/rsa.3240050306>.
- Soheil Mohajer, Changho Suh, and Adel Elmahdy. Active learning for top- k rank aggregation from noisy comparisons. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2488–2497. PMLR, 06–11 Aug 2017. URL <https://proceedings.mlr.press/v70/mohajer17a.html>.
- Noam Nisan. Crew prams and decision trees. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 327–335, 1989.
- Rüdiger Reischuk and Bernd Schmeltz. Reliable computation with noisy circuits and decision trees—a general $n \log n$ lower bound. In *Proceedings 32nd Annual Symposium of Foundations of Computer Science (FOCS)*, pages 602–611, 1991.
- Alfréd Rényi. On a problem of information theory. *MTA Mat. Kut. Int. Kozl. B*, 6(MR143666): 505–516, 1961.
- Jonathan Scarlett. Noisy adaptive group testing: Bounds and algorithms. *IEEE Transactions on Information Theory*, 65(6):3646–3661, 2018.
- Stanislaw M. Ulam. *Adventures of a mathematician*. New York: Scribner, 1976.
- Ziao Wang, Nadim Ghaddar, and Lele Wang. Noisy sorting capacity. In *Proceedings of the 2022 IEEE International Symposium on Information Theory (ISIT)*, pages 2541–2546. IEEE, 2022. doi: 10.1109/ISIT50566.2022.9834370.
- Ziao Wang, Nadim Ghaddar, Banghua Zhu, and Lele Wang. Variable-length insertion-based noisy sorting. In *Proceedings of the 2023 IEEE International Symposium on Information Theory (ISIT)*, pages 1782–1787, 2023. doi: 10.1109/ISIT54713.2023.10206482.
- Ziao Wang, Nadim Ghaddar, Banghua Zhu, and Lele Wang. Noisy computing of the threshold function. In *36th International Conference on Algorithmic Learning Theory*, 2025. URL <https://openreview.net/forum?id=Rpy7ViuSNC>.
- Banghua Zhu, Ziao Wang, Nadim Ghaddar, Jiantao Jiao, and Lele Wang. Noisy computing of the OR and MAX functions. *arXiv preprint arXiv:2309.03986*, 2023a.
- Banghua Zhu, Ziao Wang, Nadim Ghaddar, Jiantao Jiao, and Lele Wang. On the optimal bounds for noisy computing. In *2023 IEEE International Symposium on Information Theory (ISIT)*, pages 1788–1793, 2023b. doi: 10.1109/ISIT54713.2023.10206881.

Appendix A. Preliminaries

Throughout the paper, we use $p \in (0, 1/2)$ to denote the flipping probability of each noisy query, i.e., for a bit x , $\text{QUERY}(x)$ returns x with probability $1 - p$ and returns $1 - x$ with probability p . For $0 \leq q \leq 1$, $\text{Ber}(q)$ denotes the Bernoulli distribution with head probability q . Throughout the paper, all logs have base e . For two sequences $(f_n)_n, (g_n)_n$, we write $f_n \asymp g_n$ if $f_n = \Theta(g_n)$, i.e., there exists $\epsilon > 0$ such that $\epsilon f_n \leq g_n \leq \epsilon^{-1} f_n$ for all n large enough.

Let $\text{Bin}(n, p)$ denote the binomial distribution. The following large deviation bound is useful.

Lemma 5 *Let $0 < p < \frac{1}{2}$ and $0 < q < 1$. Then for large enough m and integer $k = (q \pm o(1))m$, we have*

$$\mathbb{P}(\text{Bin}(m, p) = k) = \exp(-(D_{\text{KL}}(q \parallel p) \pm o(1))m),$$

where

$$D_{\text{KL}}(a \parallel b) = a \log \frac{a}{b} + (1 - a) \log \frac{1 - a}{1 - b}$$

is the binary KL divergence function.

Proof

$$\begin{aligned} \mathbb{P}(\text{Bin}(m, p) = k) &= \binom{m}{k} p^k (1 - p)^{m-k} \\ &= \exp(m \log m - k \log k - (m - k) \log(m - k) \pm o(m)) \\ &\quad \cdot \exp(k \log p + (m - k) \log(1 - p)) \\ &= \exp\left(-\left(q \log \frac{q}{p} + (1 - q) \log \frac{1 - q}{1 - p} \pm o(1)\right)m\right) \\ &= \exp(-(D_{\text{KL}}(q \parallel p) \pm o(1))m). \end{aligned}$$

■

Appendix B. High-influence functions

We prove [Theorem 1](#) in this section. Let us recall the theorem statement.

Theorem 1 (Noisy query complexity of high-influence functions) *For any $c > 0$, there exists $c' > 0$ such that the following holds. For any Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ with $I(f) \geq cn$, any noisy query algorithm computing $f(x)$ with error probability $\leq \frac{1}{3}$ makes at least $c'n \log n$ noisy queries in expectation to the coordinates of the input $x \in \{0, 1\}^n$.*

B.1. Three-phase problem

Let f be a Boolean function with total influence $I(f) \geq cn$ and x be uniformly chosen from $\{0, 1\}^n$. Our goal is to show that any algorithm computing $f(x)$ with error probability ϵ makes at least $c'n \log n$ queries in expectation. (The error probability $1/3$ in [Theorem 1](#) can be replaced with any $\epsilon > 0$ without affecting the asymptotic noisy query complexity, by repeating the algorithm constantly many times and taking the majority vote.)

Let $c_1, c_2 > 0$ be two absolute constants. We define a three-phase problem as follows.

1. In Phase 1, the algorithm makes $m_1 = c_1 \log n$ queries to every coordinate.
2. In Phase 2, the oracle reveals some elements to the algorithm.
3. In Phase 3, the algorithm makes $m_2 = c_2 n$ adaptive exact queries.

The goal of the algorithm is to determine the value of $f(x)$.

Lemma 6 *If no algorithm can solve the three-phase problem with error probability $\epsilon > 0$, then no algorithm can compute $f(x)$ with error probability ϵ using at most $c_1 c_2 n \log n$ noisy queries.*

Proof Suppose there is an algorithm \mathcal{A} that computes $f(x)$ with error probability ϵ using at most $m_1 m_2$ noisy queries. We define an algorithm \mathcal{A}' that solves the three-phase problem with the same error probability.

Let $x \in \{0, 1\}^n$ be chosen uniformly randomly. Algorithm \mathcal{A}' receives query results in Phase 1 and 2 and enters Phase 3. It simulates algorithm \mathcal{A} on the same input x as follows.

- (1) Initially, $x_i \leftarrow *$ for all $i \in [n]$.
- (2) When \mathcal{A} queries coordinate i :
 - (a) If this is the k -th time coordinate i is queried for $k \leq m_1$, return the k -th noisy query result on coordinate i in Phase 1 to \mathcal{A} .
 - (b) Suppose coordinate i has been queried more than m_1 times. If $x_i = *$, make an exact query to coordinate i in Phase 3 and let $x_i \leftarrow$ the query result. Return $\text{BSC}_p(x_i)$ to \mathcal{A} .
- (3) When \mathcal{A} returns, \mathcal{A}' returns the same result.

Because \mathcal{A} has error probability ϵ , \mathcal{A}' also has error probability ϵ . It suffices to prove that in Phase 3, \mathcal{A}' makes at most m_2 exact queries. Note that the number of exact queries \mathcal{A}' makes in Phase 3 is equal to the number of coordinates i to which \mathcal{A} makes more than m_1 queries. Because \mathcal{A} makes $m_1 m_2$ queries, the expected number of such coordinates is at most m_2 . ■

By Lemma 6, to prove Theorem 1, it suffices to prove hardness of the three-phase problem.

Proposition 7 *For some $c_1, c_2, \epsilon > 0$, no algorithm can solve the three-phase problem with error probability ϵ , where the input is uniformly chosen from $\{0, 1\}^n$.*

Theorem 1 follows by combining Lemma 6 and Proposition 7. The rest of the section is devoted to the proof of Proposition 7.

B.2. Phase 1

In Phase 1, the algorithm makes $m_1 = c_1 \log n$ queries to every element $i \in [n]$. Let $A = \{i \in [n] : x_i = 1\}$, where x is the input bit string. Let a_i denote the number of times where a query to i returns 1. Then for $i \in A$, $a_i \sim \text{Bin}(m_1, 1 - p)$; for $i \notin A$, $a_i \sim \text{Bin}(m_1, p)$. For $0 \leq j \leq m_1$, define

$$p_j = \mathbb{P}(\text{Bin}(m_1, 1 - p) = j) = \binom{m_1}{j} (1 - p)^j p^{m_1 - j}. \quad (1)$$

Let $I = [pm_1 - m_1^{0.6}, pm_1 + m_1^{0.6}]$.

Lemma 8 *Let $x \sim \text{Bin}(m_1, 1 - p)$, $y \sim \text{Bin}(m_1, p)$. Then*

$$\mathbb{P}(x \in I) = n^{-c_3 \pm o(1)}, \quad (2)$$

$$\mathbb{P}(y \in I) = 1 - o(1), \quad (3)$$

where $c_3 = c_1(1 - 2p) \log \frac{1-p}{p}$.

Proof By Lemma 5, for $k \in I$, we have

$$\mathbb{P}(x = k) = \exp(-(D_{\text{KL}}(p \parallel 1 - p) \pm o(1))m_1) = n^{-c_1 D_{\text{KL}}(p \parallel 1 - p) \pm o(1)}.$$

So

$$\mathbb{P}(x \in I) = \sum_{k \in I} \mathbb{P}(x = k) = n^{-c_1 D_{\text{KL}}(p \parallel 1 - p) \pm o(1)}.$$

This proves Eq. (2).

Eq. (3) follows from the Chernoff bound. ■

Let $\mathbb{P}^{(0)} = \text{Ber}(1/2)^{\otimes n}$ denote the prior distribution of A and $\mathbb{P}^{(1)}$ denote the posterior distribution of A conditioned on observations in Phase 1. Then for any set $B \subseteq [n]$ we have

$$\begin{aligned} \mathbb{P}^{(1)}(B) &\propto \mathbb{P}((a_i)_{i \in [n]} | B) \mathbb{P}^{(0)}(B) \\ &= \left(\prod_{i \in B} p_{a_i} \right) \left(\prod_{i \in [n] \setminus B} p_{m_1 - a_i} \right) \mathbb{P}^{(0)}(B) \\ &\propto \left(\prod_{i \in B} p_{a_i} \right) \left(\prod_{i \in [n] \setminus B} p_{m_1 - a_i} \right). \end{aligned}$$

B.3. Phase 2

In Phase 2, the oracle reveals some elements in A and not in A as follows.

- 2a. In Step 2a, the oracle reveals elements i with $a_i \notin I$.
- 2b. In Step 2b, the oracle reveals every $i \in A$ independently with probability q_{a_i} , for some constants $(q_k)_{k \in I}$ to be chosen later.

B.3.1. Step 2a

Let $S_+^{(2a)}$ (resp. $S_-^{(2a)}$) denote the set of elements in A (resp. in A^c) revealed in Step 2a. For $B \subseteq [n]$, we say B is consistent with the observations $(S_+^{(2a)}, S_-^{(2a)})$ if $S_+^{(2a)} \subseteq B$ and $S_-^{(2a)} \cap B = \emptyset$. Let $\mathbb{P}^{(2a)}$ denote the posterior distribution at the end of Step 2a, and $\mathcal{C}^{(2a)}$ be its support. Then $B \subseteq [n]$ is in $\mathcal{C}^{(2a)}$ if and only if it is consistent with $(S_+^{(2a)}, S_-^{(2a)})$. For $B \in \mathcal{C}^{(2a)}$, the posterior probability $\mathbb{P}^{(2a)}(B)$ is given by

$$\mathbb{P}^{(2a)}(B) \propto \mathbb{P}^{(1)}(B) \propto \left(\prod_{i \in B} p_{a_i} \right) \left(\prod_{i \in B^c} p_{m_1 - a_i} \right). \quad (4)$$

B.3.2. Step 2b

Let $S_+^{(2b)}$ be the set of elements of A revealed in Step 2b that were not revealed in Step 2a. Define $S_+^{(\leq 2b)} = S_+^{(2a)} \cup S_+^{(2b)}$. Define $\mathbb{P}^{(2b)}, \mathcal{C}^{(2b)}$ similarly to [Section B.3.1](#). Then $\mathcal{C}^{(2b)}$ is the set $B \in \mathcal{C}^{(2a)}$ that are consistent with $(S_+^{(2b)}, \emptyset)$. For any $B \in \mathcal{C}^{(2b)}$, we have

$$\begin{aligned} \mathbb{P}^{(2b)}(B) &\propto \mathbb{P}\left(S_+^{(2b)} | B, S_+^{(2a)}\right) \mathbb{P}^{(2a)}(B) \\ &\propto \left(\prod_{i \in B \setminus S_+^{(\leq 2b)}} (1 - q_{a_i}) \right) \mathbb{P}^{(2a)}(B) \\ &\propto \left(\prod_{i \in B \setminus S_+^{(\leq 2b)}} (1 - q_{a_i}) p_{a_i} \right) \left(\prod_{i \in B^c \setminus S_-^{(2a)}} p_{m_1 - a_i} \right) \\ &\propto \left(\prod_{i \in B \setminus S_+^{(\leq 2b)}} \frac{(1 - q_{a_i}) p_{a_i}}{p_{m_1 - a_i}} \right). \end{aligned} \tag{5}$$

Note that for any $B \in \mathcal{C}^{(2b)}$ and any $i \in B \setminus S_+^{(\leq 2b)}$, we have $a_i \in I$.

Let us now choose the values of q_k for $k \in I$. For $k \in I$, define

$$q_k = 1 - \frac{p_{m_1 - k} p_{k_l}}{p_k p_{m_1 - k_l}}$$

where $k_l = pm_1 - \log^{0.6} n$ is the left endpoint of I . Because $\frac{p_{m_1 - k}}{p_k} = \left(\frac{1-p}{p}\right)^{m_1 - 2k}$ is decreasing in k , we have $q_k \in [0, 1]$ for $k \in I$. So this choice of q_k 's is valid.

With this choice of q_k , we can simplify [Eq. \(5\)](#) as

$$\mathbb{P}^{(2b)}(B) \propto \left(\frac{p_{k_l}}{p_{m_1 - k_l}} \right)^{|B|}. \tag{6}$$

[Eq. \(6\)](#) is very useful and greatly simplifies the posterior distribution. Importantly, $(a_i)_{i \in [n]}$ does not appear directly in the expression. Therefore, $(S_+^{(\leq 2b)}, S_-^{(2a)}, |A|)$ is a sufficient statistic for A at the end of Step 2b.

Let us now consider the distribution of $(S_+^{(\leq 2b)}, S_-^{(2a)})$ conditioned on A .

Let p_+ be the probability that a coordinate $i \in A$ is in $S_+^{(\leq 2b)}$. Every coordinate $i \in A$ is independently in $S_+^{(2a)}$ with probability $\mathbb{P}(\text{Bin}(m_1, 1-p) \notin I) = 1 - n^{-c_3 \pm o(1)}$ ([Lemma 8](#)). Every coordinate $i \in A$ is independently in $S_+^{(2b)}$ with probability

$$\sum_{k \in I} p_k q_k = \sum_{k \in I} p_k \left(1 - \frac{p_{m_1 - k} p_{k_l}}{p_k p_{m_1 - k_l}} \right).$$

For a fixed $i \in A$, the events $i \in S_+^{(2a)}$ and $i \in S_+^{(2b)}$ are disjoint, so

$$p_+ = \mathbb{P}(\text{Bin}(m_1, 1-p) \notin I) + \sum_{k \in I} p_k q_k \geq \mathbb{P}(\text{Bin}(m_1, 1-p) \notin I) = 1 - n^{-c_3 \pm o(1)}.$$

On the other hand,

$$p_+ = 1 - \sum_{k \in I} p_k (1 - q_k) \leq 1 - p_{k_l} (1 - q_{k_l}) = 1 - p_{k_l} = 1 - n^{-c_3 \pm o(1)},$$

where the last step is by [Lemma 5](#). Thus,

$$p_+ = 1 - n^{-c_3 \pm o(1)}. \quad (7)$$

Let p_- be the probability that a coordinate $i \in A^c$ is in $S_-^{(2a)}$. By [Lemma 8](#),

$$p_- = \mathbb{P}(\text{Bin}(m_1, p) \notin I) = o(1). \quad (8)$$

Therefore, observations up to Step 2b have the same effect as the following procedure:

Definition 9 (Alternative observation procedure) *Let $A \sim \text{Ber}(1/2)^{\otimes n}$.*

- (1) *Observe every coordinate $i \in A$ independently with probability p_+ ([Eq. \(7\)](#)).*
- (2) *Observe every coordinate $i \in A^c$ independently with probability p_- ([Eq. \(8\)](#)).*

By [Eq. \(6\)](#), the posterior distribution of A after Phase 2 is a biased product distribution on the unrevealed coordinates.

B.4. Phase 3

In Phase 3, the algorithm makes at most $c_2 n$ adaptive exact queries. We will show that for c_2 small enough, no algorithm is able to determine $f(x)$ with very small error probability.

Our proof strategy is as follows.

- 3a. Because f has linear total influence, after Phase 2 ends, the Boolean function on the unrevealed coordinates will have a biased version of total influence l_q at least $\Omega(n)$ in expectation.
- 3b. Every (adaptive) query made in Phase 3 decreases l_q by at most 1. Therefore, after Phase 3, the Boolean function on the remaining unrevealed coordinates has $\mathsf{l}_q = \Omega(n)$ in expectation.
- 3c. Finally, we show that if a Boolean function has $\mathsf{l}_q = \Omega(n)$, then it is impossible to guess $f(x)$ (where x follows a biased product distribution) with very small error probability.

Because observations up to Phase 2 have been simplified by our analysis, we make some definitions and restate the problem we need to solve in Phase 3.

B.4.1. Preliminaries

We have a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and an input $x \sim \text{Ber}(1/2)^{\otimes n}$, and the goal is to determine $f(x)$. Then we independently observe each $i \in [n]$ with probability $p_+ = 1 - n^{-c_3 \pm o(1)}$ for $x_i = 1$ and $p_- = o(1)$ for $x_i = 0$. Let $s \in \{0, 1, *\}^n$ be the observations. That is, if coordinate i is revealed, then s_i is the revealed value; otherwise $s_i = *$. Let $U = \{i \in [n] : s_i = *\}$ be the unrevealed coordinates. Conditioned on s , the distribution of x_U is a product of $\text{Ber}(q)$, where

$$q = \frac{1 - p_+}{1 - p_+ + 1 - p_-} = n^{-c_3 \pm o(1)}. \quad (9)$$

Let $f_s : \{0, 1\}^U \rightarrow \{0, 1\}$ be the function $f_s(x_U) = f(s \triangleleft x_U)$ for all $x_U \in \{0, 1\}^U$, where $s \triangleleft x_U$ denotes the bit string where all $*$ in s are replaced with the corresponding value in x_U . Let ρ be the distribution $(\frac{p_-}{2}, \frac{p_+}{2}, 1 - \frac{p_-}{2} - \frac{p_+}{2})$ on $\{0, 1, *\}$. Then without conditioning on x , s has distribution $\rho^{\otimes n}$.

For any Boolean function $g : \{0, 1\}^S \rightarrow \{0, 1\}$, define the q -biased influence of coordinate $i \in S$ as

$$\text{Inf}_{q,i}(f) = \mathbb{P}_{x \sim \text{Ber}(q)^{\otimes S}}(f(x) \neq f(x \oplus e_i))$$

and the q -biased total influence as

$$\text{I}_q(f) = \sum_{i \in S} \text{Inf}_{q,i}(f).$$

When we mention the q -biased total influence of the function f_s , the sum is over the unrevealed coordinates $i \in U$.

For a string $y \in \{0, 1\}^S$, let D_y denote the distribution of $t \in \{0, 1, *\}^S$ where for $i \in S$ with $y_i = 1$, $t_i = 1$ with probability p_+ and $t_i = *$ with probability $1 - p_+$; for $i \in S$ with $y_i = 0$, $t_i = 0$ with probability p_- and $t_i = *$ with probability $1 - p_-$. For a string $t \in \{0, 1, *\}^S$, let E_s denote the distribution of $y \in \{0, 1\}^S$ where $y_i = t_i$ if $t_i \in \{0, 1\}$ and $y_i \sim \text{Ber}(q)$ independently for $i \in S$ with $t_i = *$, where q is defined in Eq. (9). With these definitions, we have $x \sim \text{Ber}(1/2)^{\otimes n}$, $s \sim \rho^{\otimes n}$, $x \sim E_s$ conditioned on s , $s \sim D_x$ conditioned on x .

B.4.2. Step 3a

We connect the total influence $\text{I}(f)$ with the biased total influence $\text{I}_q(f_s)$.

Fix $i \in [n]$, we have (in the following, $x \cup 0_i$ and $x \cup 1_i$ denotes setting the i -th bit of x as 0 and 1, respectively)

$$\begin{aligned}
\text{Inf}_i(f) &= \mathbb{P}_{x \sim \text{Ber}(1/2)^{\otimes ([n] \setminus i)}} (f(x \cup 0_i) \neq f(x \cup 1_i)) \\
&= \mathbb{P}_{\substack{x \sim \text{Ber}(1/2)^{\otimes ([n] \setminus i)} \\ s \sim D_x}} (f(x \cup 0_i) \neq f(x \cup 1_i)) \\
&= \mathbb{P}_{\substack{s \sim \rho^{\otimes ([n] \setminus i)} \\ x \sim E_s}} (f(x \cup 0_i) \neq f(x \cup 1_i)) \\
&= \frac{1}{\mathbb{P}_{s_i \sim \rho}(s_i = *)} \cdot \mathbb{P}_{\substack{s \sim \rho^{\otimes [n]} \\ x \sim E_{s_{[n] \setminus i}}}} (f(x \cup 0_i) \neq f(x \cup 1_i) \wedge s_i = *) \\
&= \frac{1}{\mathbb{P}_{s_i \sim \rho}(s_i = *)} \cdot \mathbb{P}_{\substack{s \sim \rho^{\otimes [n]} \\ x \sim E_s}} (f(x) \neq f(x \oplus e_i) \wedge s_i = *) \\
&= \frac{1}{\rho(*)} \cdot \mathbb{E}_{s \sim \rho^{\otimes [n]}} [\mathbb{1}_{s_i = *} \cdot \mathbb{P}_{x \sim E_s} (f(x) \neq f(x \oplus e_i))] .
\end{aligned}$$

The first step is by definition of Inf_i . The second step is because s is not involved in the condition. The third step is by considering the joint distribution between x and s . The fourth step is because the two conditions $f(x \cup 0_i) \neq f(x \cup 1_i)$ and $s_i = *$ are independent: the former depends on $s_{[n] \setminus i}$ and the latter depends on s_i . The fifth step rewrites the condition. The sixth step changes the order of checking $s_i = *$ and choosing $x \sim E_s$.

Summing over $i \in [n]$, we have

$$\begin{aligned}
\rho(*) \text{I}(f) &= \sum_{i \in [n]} \mathbb{E}_{s \sim \rho^{\otimes [n]}} [\mathbb{1}_{s_i = *} \cdot \mathbb{P}_{x \sim E_s} (f(x) \neq f(x \oplus e_i))] \\
&= \mathbb{E}_{s \sim \rho^{\otimes [n]}} \left[\sum_{i \in [n]: s_i = *} \mathbb{P}_{x \sim E_s} (f(x) \neq f(x \oplus e_i)) \right] \\
&= \mathbb{E}_{s \sim \rho^{\otimes [n]}} \text{I}_q(f_s).
\end{aligned}$$

The second step is by linearity of expectation. The third step is by definition of I_q .

Note that $\rho(*) = 1 - \frac{p_-}{2} - \frac{p_+}{2} = \frac{1}{2} \pm o(1)$. Because $\text{I}(f) \geq cn$, we have

$$\mathbb{E}_{s \sim \rho^{\otimes [n]}} \text{I}_q(f_s) \geq (2c \pm o(1))n. \quad (10)$$

That is, f_s has expected q -biased total influence at least $(2c \pm o(1))n$.

B.4.3. Step 3b

We prove the following lemma, which essentially says that adaptive exact queries in Phase 3 can only decrease the q -biased total influence I_q by a certain amount. In the following, recall the definition of f_t for a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ and $t \in \{0, 1, *\}^n$ is f where we restrict all input coordinates i with $t_i \neq *$ to be equal to t_i .

Lemma 10 *Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a Boolean function. Suppose the input x follows distribution $\text{Ber}(q)^{\otimes n}$. Consider an algorithm which adaptively makes at most m exact queries in expectation. Let $t \in \{0, 1, *\}^n$ be the random variable denoting the query results. Then $\mathbb{E}[\text{I}_q(f_t)] \geq \text{I}_q(f) - m$, where \mathbb{E} is over the randomness of the revealed coordinates and the randomness of the algorithm.*

Proof By induction it suffices to prove the case where the algorithm makes exactly one query. Without loss of generality, assume that the algorithm makes an query to coordinate 1. Then $t = 1*^{n-1}$ with probability q and $t = 0*^{n-1}$ with probability $1 - q$.

$$\begin{aligned}
\mathbb{E}[I_q(f_t)] &= q I_q(f_{1*^{n-1}}) + (1 - q) I_q(f_{0*^{n-1}}) \\
&= q \sum_{2 \leq i \leq n} \text{Inf}_{q,i}(f_{1*^{n-1}}) + (1 - q) \sum_{2 \leq i \leq n} \text{Inf}_{q,i}(f_{0*^{n-1}}) \\
&= \sum_{2 \leq i \leq n} \text{Inf}_{q,i}(f) \\
&= I_q(f) - \text{Inf}_{q,1}(f) \\
&\geq I_q(f) - 1.
\end{aligned}$$

The first step is by expanding the expectation. The second step is by definition of I_q . The third step is because $q \text{Inf}_{q,i}(f_{1*^{n-1}}) + (1 - q) \text{Inf}_{q,i}(f_{0*^{n-1}}) = \text{Inf}_{q,i}(f)$. The fourth step is by definition of I_q . The fifth step is because $\text{Inf}_{q,i}(f) \leq 1$. \blacksquare

We now apply [Lemma 10](#) to Phase 3. Let $t \in \{0, 1, *\}^U$ be the observations made in Phase 3, where $U = \{i \in [n] : s_i = *\}$ is the set of unrevealed coordinates at the end of Phase 2. Then [Eq. \(10\)](#) together with [Lemma 10](#) implies that

$$\mathbb{E}_{s \sim \rho^{\otimes [n]}} [\mathbb{E}_t [I_q((f_s)_t)]] \geq (2c - c_2 \pm o(1))n. \quad (11)$$

B.4.4. Step 3c

After Phase 3, the Boolean function on the unrevealed coordinates has q -biased total influence at least $(2c - c_2 \pm o(1))n = \Omega(n)$ in expectation. In particular, with probability $\geq c - c_2/2 \pm o(1)$, the function has $I_q((f_s)_t) \geq (c - c_2/2 \pm o(1))n$. Any algorithm now needs to output an answer in $\{0, 1\}$. The following result shows that the error probability will be $\Omega(1)$ no matter what the algorithm outputs.

Lemma 11 *For any $0 < c \leq 1$, there exists $c' > 0$ such that the following holds. Let $f : \{0, 1\}^n \rightarrow \{0, 1\}$ be a Boolean function and $\frac{1}{3n} \leq q \leq c/6$ be a parameter. If $I_q(f) \geq cn$, then*

$$c' \leq \mathbb{E}_{x \sim \text{Ber}(q)^{\otimes n}} [f(x)] \leq 1 - c'.$$

Proof Since $I_q(f) \geq cn$ and $\text{Inf}_{q,i}(f) \leq 1$ for every i , there exist at least $cn/2$ indices $i \in [n]$ with $\text{Inf}_{q,i}(f) \geq c/2$. As $\lfloor c/6q \rfloor \leq cn/2$, there are $m = \lfloor c/6q \rfloor$ indices $i \in [n]$ with $\text{Inf}_{q,i}(f) \geq c/2$. Without loss of generality, assume that $\text{Inf}_{q,i}(f) \geq c/2$ for $i \in [m]$.

Let 0^m denote the length- m bit string with all 0's. By union bound, we have

$$\mathbb{P}_{x \sim \text{Ber}(q)^{\otimes n}} (x_{[m]} = 0^m) \geq 1 - \sum_{i \in [m]} \mathbb{P}_{x \sim \text{Ber}(q)^{\otimes n}} (x_i = 1) = 1 - mq \geq 1 - c/6. \quad (12)$$

Let \diamond denote the concatenation operation of two bit strings. For any $i \in [m]$, we have

$$\begin{aligned}
 & \mathbb{P}_{y \sim \text{Ber}(q)^{\otimes ([n] \setminus [m])}} (f(0^m \diamond y) \neq f(e_i \diamond y)) \\
 &= \mathbb{P}_{x \sim \text{Ber}(q)^{\otimes n}} (f(x) \neq f(x \oplus e_i) \mid x_{[m]} = 0^m) \\
 &\geq \mathbb{P}_{x \sim \text{Ber}(q)^{\otimes n}} (f(x) \neq f(x \oplus e_i) \wedge x_{[m]} = 0^m) \\
 &\geq \mathbb{P}_{x \sim \text{Ber}(q)^{\otimes n}} (f(x) \neq f(x \oplus e_i)) + \mathbb{P}_{x \sim \text{Ber}(q)^{\otimes n}} (x_{[m]} = 0^m) - 1 \\
 &\geq \inf_{q,i} (f) + (1 - c/6) - 1 \quad (\text{By Eq. (12)}) \\
 &\geq c/3. \quad (13)
 \end{aligned}$$

Finally, we have

$$\begin{aligned}
 & \mathbb{E}_{x \sim \text{Ber}(q)^{\otimes n}} (f(x)) \\
 &\geq \mathbb{P}_{x \sim \text{Ber}(q)^{\otimes n}} (f(x) = 1 \wedge x_1 + \dots + x_m = 1) \\
 &= \sum_{i \in [m]} \mathbb{P}_{x \sim \text{Ber}(q)^{\otimes n}} (f(x) = 1 \wedge x_{[m]} = e_i) \\
 &= q(1 - q)^{m-1} \cdot \sum_{i \in [m]} \mathbb{P}_{y \sim \text{Ber}(q)^{\otimes ([n] \setminus [m])}} (f(e_i \diamond y) = 1) \\
 &\geq \Theta_c(q) \cdot \sum_{i \in [m]} \mathbb{P}_{y \sim \text{Ber}(q)^{\otimes ([n] \setminus [m])}} (f(0^m \diamond y) = 0 \wedge f(0^m \diamond y) \neq f(e_i \diamond y)) \\
 &\geq \Theta_c(q) \cdot \sum_{i \in [m]} \left(\mathbb{P}_{y \sim \text{Ber}(q)^{\otimes ([n] \setminus [m])}} (f(0^m \diamond y) = 0) + \mathbb{P}_{y \sim \text{Ber}(q)^{\otimes ([n] \setminus [m])}} (f(0^m \diamond y) \neq f(e_i \diamond y)) - 1 \right) \\
 &\geq \Theta_c(1) \cdot \left(\mathbb{P}_{y \sim \text{Ber}(q)^{\otimes ([n] \setminus [m])}} (f(0^m \diamond y) = 0) + c/3 - 1 \right) \quad (\text{By Eq. (13)}) \\
 &\geq \Theta_c(1) \cdot (\mathbb{P}_{x \sim \text{Ber}(q)^{\otimes n}} (f(x) = 0 \wedge x_{[m]} = 0^m) + c/3 - 1) \\
 &\geq \Theta_c(1) \cdot (\mathbb{P}_{x \sim \text{Ber}(q)^{\otimes n}} (f(x) = 0) + (1 - c/6) - 1 + c/3 - 1) \quad (\text{By Eq. (12)}) \\
 &= \Theta_c(1) \cdot (c/6 - \mathbb{E}_{x \sim \text{Ber}(q)^{\otimes n}} (f(x))).
 \end{aligned}$$

Therefore, there exists c' depending only on c such that $\mathbb{E}_{x \sim \text{Ber}(q)^{\otimes n}} [f(x)] \geq c'$.

By symmetry, we also have $\mathbb{E}_{x \sim \text{Ber}(q)^{\otimes n}} (f(x)) \leq 1 - c'$. ■

Applying [Lemma 11](#) to the Boolean function $(f_s)_t$ on the unrevealed coordinates finishes the proof.

Appendix C. Graph Connectivity

Recall the Graph Connectivity problem, where the input is an unknown undirected graph on n labeled vertices. In each query, the algorithm picks an unordered pair $e = (u, v) \in \binom{V}{2}$, and the oracle returns whether e is an edge of G , flipped independently with probability $0 < p < \frac{1}{2}$. The goal of the algorithm is to determine whether G is connected or not.

In this section, we prove [Theorem 2](#), which we recall below:

Theorem 2 (Hardness of Graph Connectivity) *Any algorithm solving the Graph Connectivity problem with error probability $\leq \frac{1}{3}$ uses $\Omega(n^2 \log n)$ noisy queries in expectation.*

C.1. Preliminaries

Lemma 12 (Stirling's formula)

1. There exists an absolute constant $\epsilon > 0$ such that for all $n \geq 1$,

$$\epsilon \sqrt{n} \left(\frac{n}{e}\right)^n \leq n! \leq \epsilon^{-1} \sqrt{n} \left(\frac{n}{e}\right)^n.$$

2. Fix $m \geq 2$. There exists an constant $\epsilon = \epsilon(m) > 0$ such that for all $n \geq 1$ and all $k_1, \dots, k_m \geq 1$ with $k_1 + \dots + k_m = n$,

$$\epsilon \cdot \frac{n^{n+1/2}}{\prod_{i \in [m]} k_i^{k_i+1/2}} \leq \binom{n}{k_1, \dots, k_m} \leq \epsilon^{-1} \cdot \frac{n^{n+1/2}}{\prod_{i \in [m]} k_i^{k_i+1/2}}.$$

Let $(\mathbb{P}_n)_n$ and $(\mathbb{Q}_n)_n$ be two sequences of probability measures such that \mathbb{P}_n and \mathbb{Q}_n are defined on the same measurable spaces Ω_n . We say \mathbb{Q}_n is contiguous with respect to \mathbb{P}_n , denoted by $\mathbb{Q}_n \triangleleft \mathbb{P}_n$, if for every sequence $(\mathcal{A}_n)_n$ of measurable sets $\mathcal{A}_n \subseteq \Omega_n$, $\mathbb{P}_n(\mathcal{A}_n) \rightarrow 0$ implies $\mathbb{Q}_n(\mathcal{A}_n) \rightarrow 0$.

Lemma 13 (Cayley's formula) *The number of spanning trees on n labeled vertices is n^{n-2} .*

C.2. Structure of a uniform spanning tree

Our proof of [Theorem 2](#) uses several properties of the uniform random spanning tree (UST) of the complete graph. In this section we state and prove these properties.

The notion of balanced edges is crucial to our construction and analysis.

Definition 14 (Balanced edges) *Let $\beta > 0$ be a constant. Let T be a spanning tree on n labeled vertices. An edge $e \in T$ is called β -balanced if both sides of e has at least βn vertices. Let $B_\beta(T)$ denote the set of β -balanced edges of T .*

Proposition 15 *Let T be a UST on n labeled vertices. There exist absolute constants $\epsilon, \gamma_1, \gamma_2, \gamma_3 > 0$ such that with probability at least ϵ , the following are true simultaneously.*

(i)

$$\gamma_1 \sqrt{n} \leq |B_{1/3}(T)| \leq \gamma_2 \sqrt{n}.$$

(ii) *For all $e \in B_{1/3}(T)$, if T_1, T_2 are the two connected components of $T \setminus e$, then*

$$\min\{|B_{1/7}(T_1)|, |B_{1/7}(T_2)|\} \geq \gamma_3 \sqrt{n}.$$

Our proof of [Proposition 15](#) uses the following lemmas.

Lemma 16 (Balanced edges form a chain) *Let T be a tree on n labeled vertices. Then the subgraph formed by all edges in $B_{1/3}(T)$ is either empty or a chain.*

Proof Suppose $B_{1/3}(T)$ is non-empty. Let H be the subgraph formed by all edges in $B_{1/3}(T)$.

Step 1. We prove that H is connected. Let $e_1, e_2 \in B_{1/3}(T)$ and e_3 be an edge on the path (in T) between e_1 and e_2 . Let $T_{i,1}, T_{i,2}$ ($i = 1, 2, 3$) be the two connected components of $T \setminus e_i$, such that $e_3 \in T_{1,1}, e_3 \in T_{2,1}, e_1 \in T_{3,1}, e_2 \in T_{3,2}$. Then $|T_{i,j}| \geq \frac{n}{3}$ for $i, j \in \{1, 2\}$. We have $T_{3,i} \supseteq T_{i,2}$ ($i = 1, 2$). So $|T_{3,i}| \geq \frac{n}{3}$ for $i = 1, 2$ and e_3 is $\frac{1}{3}$ -balanced. This shows that H is connected.

Step 2. We prove that H is a chain. By Step 1, H is a connected subgraph of T , so it is a tree. Suppose H has a vertex v of degree at least three. Then there exist three distinct $\frac{1}{3}$ -balanced edges e_1, e_2, e_3 containing v . Say $e_i = (v, u_i)$ ($i = 1, 2, 3$). Let T_i ($i = 1, 2, 3$) be the connected component of $T \setminus e_i$ not containing v . Then $|T_i| \geq \frac{n}{3}$ and T_i ($i = 1, 2, 3$) are all disjoint. This implies

$$n \geq |T_1| + |T_2| + |T_3| + 1 \geq n + 1,$$

which is a contradiction. So all vertices of H have degree at most two, and H must be a chain. \blacksquare

The following statement about typical distances in a UST is well-known (e.g., (Aldous, 1991)).

Lemma 17 (Typical distance in UST) *Let T be a UST on n labeled vertices and u, v be two fixed vertices (not dependent on T). Then for any $\epsilon > 0$, there exist absolute constants $\gamma_1, \gamma_2 > 0$, such that*

$$\mathbb{P} [\gamma_1 \sqrt{n} \leq \text{dist}_T(u, v) \leq \gamma_2 \sqrt{n}] \geq 1 - \epsilon,$$

where dist_T denotes the graph distance in T .

To prove Proposition 15, we define a probability measure \mathbb{Q} on the space of spanning trees on n labeled vertices that is contiguous with respect to the UST measure, and Proposition 15 holds under \mathbb{Q} .

Definition 18 (Measure \mathbb{Q}) *We define the measure \mathbb{Q} as follows. A tree tuple \mathcal{D} is a tuple which consists of the following data:*

- (i) Six distinct vertices $u_0, u_1, u_2, v_0, v_1, v_2 \in [n]$.
- (ii) Integers $\frac{n}{3} - \frac{n}{100} \leq L_0, R_0 \leq \frac{n}{3} - 1$. Integers $\frac{n}{3} - L_0 \leq L_1 \leq \frac{n}{100}, \frac{n}{3} - R_0 \leq R_1 \leq \frac{n}{100}$. Let $W = n - L_0 - L_1 - R_0 - R_1$.
- (iii) A partition of $V = [n]$ into five subsets $V = V_{L_0} \sqcup V_{L_1} \sqcup V_{R_0} \sqcup V_{R_1} \sqcup V_W$, where $u_i \in V_{L_i}, v_i \in V_{R_i}, u_2, v_2 \in V_W$, and $|V_{L_i}| = L_i, |V_{R_i}| = R_i, |V_W| = W$ ($i = 1, 2$).
- (iv) Five spanning trees: T_{L_i} on V_{L_i}, T_{R_i} on V_{R_i}, T_W on V_W ($i = 1, 2$).

Given a tree tuple \mathcal{D} , it produces a spanning tree

$$T(\mathcal{D}) = T_{L_0} \cup T_{L_1} \cup T_{R_0} \cup T_{R_1} \cup T_W \cup \{(u_0, u_1), (u_1, u_2), (v_0, v_1), (v_1, v_2)\}.$$

The measure \mathbb{Q} is the distribution of $T(\mathcal{D})$, where \mathcal{D} is uniformly chosen from all tree tuples.

Lemma 19 (Contiguity) *Let \mathbb{P} be the UST measure on n labeled vertices. Let \mathbb{Q} be as defined in Definition 18. Then $\mathbb{Q} \triangleleft \mathbb{P}$.*

Proof Let us study the support of \mathbb{Q} . By Lemma 16, $\frac{1}{3}$ -balanced edges in T form a chain. By the construction (Definition 18), for $T = T(\mathcal{D})$, u_1 and v_1 are the two endpoints of the chain $B_{1/3}(T)$, and u_2 (resp. v_2) is the second vertex on the path from u_1 to v_1 (resp. v_1 to u_1). Furthermore, u_0 (resp. v_0) is the only neighbor x of u_1 (resp. v_1) not on the path between u_1 and v_1 , such that u_0 's connected component in $T \setminus (u_0, x)$ (resp. v_0 's connected component in $T \setminus (v_0, x)$) has size at least $\frac{n}{3} - \frac{n}{100}$. This means that \mathcal{D} can be reconstructed given $T(\mathcal{D})$, up to swapping (u_0, u_1, u_2) and other data with (v_0, v_1, v_2) and the corresponding data. Specifically, every T in the support of \mathbb{Q} is realized by exactly two \mathcal{D} s. Therefore, \mathbb{Q} is equal to the conditional measure $\mathbb{P}(\cdot \mid T \in \text{supp } \mathbb{Q})$.

Now, to prove that $\mathbb{Q} \triangleleft \mathbb{P}$, it suffices to prove that $|\text{supp } \mathbb{Q}| = \Theta(|\text{supp } \mathbb{P}|) = \Theta(n^{n-2})$. Because every T is realized exactly twice, it suffices to prove that the number of different tree tuples is $\Theta(n^{n-2})$. We construct a tree tuple according to the following procedure.

- (i) Choose six distinct vertices $u_0, u_1, u_2, v_0, v_1, v_2 \in [n]$. There are $n!/(n-6)! = \Theta(n^6)$ ways to do this.
- (ii) Choose L_1 and R_1 from $[1, \frac{n}{100}]$. Choose L_0 from $[\frac{n}{3} - L_1, \frac{n}{3} - 1]$ and R_0 from $[\frac{n}{3} - R_1, \frac{n}{3} - 1]$.
- (iii) Choose the partition $V = V_{L_0} \sqcup V_{L_1} \sqcup V_{R_0} \sqcup V_{R_1} \sqcup V_W$. There are

$$\binom{n-6}{L_0-1, L_1-1, R_0-1, R_1-1, W-2}$$

ways to do this.

- (iv) Choose the five spanning trees, $T_{L_0}, T_{L_1}, T_{R_0}, T_{R_1}, T_W$. There are

$$L_0^{L_0-2} L_1^{L_1-2} R_0^{R_0-2} R_1^{R_1-2} W^{W-2}$$

ways to do this.

Summarizing the above, for large enough n , the number of different tree tuples is

$$\begin{aligned}
& \frac{n!}{(n-6)!} \cdot \sum_{1 \leq L_1, R_1 \leq \frac{n}{100}} \sum_{\substack{\frac{n}{3} - L_1 \leq L_0 \leq \frac{n}{3} - 1 \\ \frac{n}{3} - R_1 \leq R_0 \leq \frac{n}{3} - 1}} \binom{n-6}{L_0-1, L_1-1, R_0-1, R_1-1, W-2} \\
& \cdot L_0^{L_0-2} L_1^{L_1-2} R_0^{R_0-2} R_1^{R_1-2} W^{W-2} \\
& \asymp \sum_{1 \leq L_1, R_1 \leq \frac{n}{100}} \sum_{\substack{\frac{n}{3} - L_1 \leq L_0 \leq \frac{n}{3} - 1 \\ \frac{n}{3} - R_1 \leq R_0 \leq \frac{n}{3} - 1}} \binom{n}{L_0, L_1, R_0, R_1, W} L_0^{L_0-1} L_1^{L_1-1} R_0^{R_0-1} R_1^{R_1-1} W^W \\
& \asymp \sum_{1 \leq L_1, R_1 \leq \frac{n}{100}} \sum_{\substack{\frac{n}{3} - L_1 \leq L_0 \leq \frac{n}{3} - 1 \\ \frac{n}{3} - R_1 \leq R_0 \leq \frac{n}{3} - 1}} \frac{n^{n+1/2}}{(L_0 L_1 R_0 R_1)^{3/2} W^{1/2}} \\
& \asymp \sum_{1 \leq L_1, R_1 \leq \frac{n}{100}} \sum_{\substack{\frac{n}{3} - L_1 \leq L_0 \leq \frac{n}{3} - 1 \\ \frac{n}{3} - R_1 \leq R_0 \leq \frac{n}{3} - 1}} \frac{n^{n-3}}{(L_1 R_1)^{3/2}} \\
& \asymp \sum_{1 \leq L_1, R_1 \leq \frac{n}{100}} \frac{n^{n-3}}{(L_1 R_1)^{1/2}} \\
& \asymp n^{n-2}.
\end{aligned}$$

This finishes the proof. ■

Lemma 20 (Proposition 15, Item (i) for \mathbb{Q}) *Let T be sampled from \mathbb{Q} (Definition 18). For any $\epsilon > 0$, there exist $\gamma_1, \gamma_2 > 0$ such that with probability at least $1 - \epsilon$, Proposition 15, Item (i) holds for T .*

Proof As we discussed in the proof of Lemma 19, all edges on the path between u_1 and v_1 are $\frac{1}{3}$ -balanced. Conditioned on V_W, T_W being a UST on V_W , by Lemma 17,

$$\mathbb{P} \left[\gamma_1 \sqrt{W} \leq \text{dist}_{T_W}(u_1, v_1) \leq \gamma_2 \sqrt{W} \right] \geq 1 - \epsilon.$$

Noting that $\text{dist}_T(u_1, v_1) = \text{dist}_{T_W}(u_1, v_1)$ and $W = \Theta(n)$, we finish the proof. ■

Corollary 21 (Proposition 15, Item (i)) *Let T be a UST on n labeled vertices. There exist absolute constants $\epsilon, \gamma_1, \gamma_2 > 0$ such that with probability at least ϵ , Proposition 15, Item (i) holds for T .*

Proof By Lemmas 19 and 20. ■

Using Corollary 21, we can prove that Proposition 15 holds for \mathbb{Q} .

Lemma 22 (Proposition 15 for \mathbb{Q}) *Let T be sampled from \mathbb{Q} (Definition 18). There exist absolute constants $\epsilon, \gamma_1, \gamma_2, \gamma_3 > 0$ such that with probability at least ϵ , both items in Proposition 15 hold for T .*

Proof Let \mathcal{E}_{L_0} (resp. $\mathcal{E}_{R_0}, \mathcal{E}_W$) be the event that $|B_{1/3}(T_{L_0})|$ (resp. $|B_{1/3}(T_{R_0})|, \text{dist}_{T_W}(u_1, v_1)$) is in $[\gamma_1\sqrt{n}, \gamma_2\sqrt{n}]$. Let $\mathcal{E} = \mathcal{E}_{L_0} \cap \mathcal{E}_{R_0} \cap \mathcal{E}_W$. Let \mathcal{A} be the σ -algebra generated by V_{L_0}, V_{R_0}, V_W . Conditioned on \mathcal{A} , the three subtrees T_{L_0}, T_{R_0}, T_W are independent and are USTs on the respective vertex sets. By Lemma 17, Corollary 21, and $L_0, R_0, W = \Theta(n)$, there exist $\epsilon, \gamma_1, \gamma_2 > 0$ such that

$$\mathbb{P}(\mathcal{E}_{L_0}|\mathcal{A}), \mathbb{P}(\mathcal{E}_{R_0}|\mathcal{A}), \mathbb{P}(\mathcal{E}_W|\mathcal{A}) \geq \epsilon^{1/3}.$$

By independence, we have $\mathbb{P}(\mathcal{E}|\mathcal{A}) \geq \epsilon$. Therefore $\mathbb{P}(\mathcal{E}) \geq \epsilon$.

Now we show that when \mathcal{E} happens, both items in Proposition 15 hold. As we discussed in the proof of Lemma 19, $B_{1/3}(T)$ is the set of edges on the path between u_1 and v_1 . So Item (i) holds because \mathcal{E}_W happens. Now consider Item (ii). Let e be an edge on the path between u_1 and v_1 . Let T_1, T_2 be the two components of $T \setminus e$, where $u_1 \in T_1, v_1 \in T_2$. Then $T_{L_0} \subseteq T_1, T_{R_0} \subseteq T_2$. Every $\frac{1}{3}$ -balanced edge of T_{L_0} divides T_{L_0} into components of size at least $\frac{|T_{L_0}|}{3}$, so it is a β -balanced edge of T_1 with $\beta = \frac{|T_{L_0}|}{3|T_1|} \geq \frac{\frac{n}{3} - \frac{n}{100}}{3 \cdot \frac{2}{3}n} \geq \frac{1}{7}$. The same discussion holds for T_{R_0} and T_2 . Because \mathcal{E}_{L_0} and \mathcal{E}_{R_0} happen, Item (ii) holds. \blacksquare

By Lemmas 19 and 22, we complete the proof of Proposition 15.

Lemma 23 *Let T be a UST. Then for any $\epsilon > 0, 0 < \beta < \frac{1}{2}$, there exists $\gamma > 0$ such that*

$$\mathbb{P}[|B_\beta(T)| \leq \gamma\sqrt{n}] \geq 1 - \epsilon.$$

Proof The proof is by the first moment method. Let $e = (u, v)$ be an unordered pair of vertices. For any integer $\beta n \leq k \leq (1 - \beta)n$, the number of spanning trees containing e such that u 's side contains exactly k (including u) vertices is

$$\binom{n-2}{k-1} k^{k-2} (n-k)^{n-k-2} \asymp n^{-2} \binom{n}{k} k^{k-1} (n-k)^{n-k-1} \asymp \frac{n^{n-3/2}}{(k(n-k))^{3/2}} \asymp n^{n-9/2}.$$

So the number of trees T with $e \in B_\beta(T)$ is

$$\asymp \sum_{k \in [\beta n, (1-\beta)n]} n^{n-9/2} \asymp n^{n-7/2}.$$

Therefore,

$$\mathbb{E}[|B_\beta(T)|] \asymp \frac{\binom{n}{2} \cdot n^{n-7/2}}{n^{n-2}} \asymp \sqrt{n}.$$

The result then follows from Markov's inequality. \blacksquare

Lemma 24 *Let T be a UST. For $e \in T$, let $s_T(e)$ denote the size of the smaller component of $T \setminus e$. Then for any $\epsilon > 0$, there exists $\gamma > 0$ such that*

$$\mathbb{P}\left[\sum_{e \in T} s_T(e) \leq \gamma n^{3/2}\right] \geq 1 - \epsilon.$$

Proof The proof is by first moment method. Let $e = (u, v)$ be an unordered pair of vertices. We define $s_T(e) = 0$ if $e \notin T$. Then

$$\begin{aligned}
\mathbb{E}[s_T(e)] &\asymp \frac{1}{n^{n-2}} \cdot \sum_{1 \leq k \leq \frac{n}{2}} k \cdot \binom{n-2}{k-1} k^{k-2} (n-k)^{n-k-2} \\
&\asymp \frac{1}{n^n} \sum_{1 \leq k \leq \frac{n}{2}} \binom{n}{k} k^k (n-k)^{n-k-1} \\
&\asymp \sum_{1 \leq k \leq \frac{n}{2}} \frac{n^{1/2}}{k^{1/2} (n-k)^{3/2}} \\
&\asymp \sum_{1 \leq k \leq \frac{n}{2}} \frac{1}{k^{1/2} n} \\
&\asymp n^{-1/2}.
\end{aligned}$$

Therefore,

$$\mathbb{E} \left[\sum_{e \in T} s_T(e) \right] \asymp \binom{n}{2} \cdot n^{-1/2} \asymp n^{3/2}.$$

The result then follows from Markov's inequality. ■

Combining [Proposition 15](#), [Lemma 23](#) and [Lemma 24](#), we get the following corollary.

Corollary 25 *Let T be a UST on n labeled vertices. There exist absolute constants $\epsilon, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5 > 0$ such that the following are true simultaneously.*

(i)

$$\gamma_1 \sqrt{n} \leq |B_{1/3}(T)| \leq |B_{1/42}(T)| \leq \gamma_2 \sqrt{n}.$$

(ii) *For all $e \in B_{1/3}(T)$, if T_1 and T_2 are the two connected components of $T \setminus e$, then*

$$\gamma_3 \sqrt{n} \leq \min\{|B_{1/7}(T_1)|, |B_{1/7}(T_2)|\} \leq \max\{|B_{1/14}(T_1)|, |B_{1/14}(T_2)|\} \leq \gamma_4 \sqrt{n}.$$

(iii)

$$\sum_{e \in T} s_T(e) \leq \gamma_5 n^{3/2}.$$

Proof By [Proposition 15](#), there exist $\epsilon, \gamma_1, \gamma_3 > 0$ such that with probability at least ϵ , the lower bounds in [Corollary 25](#), [Items \(i\) and \(ii\)](#) hold. By [Lemma 23](#), there exists $\gamma_2 > 0$ such that

$$\mathbb{P} \left[|B_{1/42}(T)| \leq \gamma_2 \sqrt{n} \right] \geq 1 - \epsilon/3.$$

By Lemma 24, there exists $\gamma_5 > 0$ such that

$$\mathbb{P} \left[\sum_{e \in T} s_T(e) \leq \gamma_5 n^{3/2} \right] \geq 1 - \epsilon/3.$$

By union bound, with probability at least $\epsilon/3$, Corollary 25, Items (i) and (iii), and the lower bound in Corollary 25, Item (ii) hold. Now notice that $B_{1/14}(T_1) \cup B_{1/14}(T_2) \subseteq B_{1/42}(T)$, so the upper bound in Corollary 25, Item (ii) holds with $\gamma_4 = \gamma_2$. ■

C.3. Hard distribution and three-phase problem

We now start the proof of Theorem 2. Our input distribution is defined as follows.

Definition 26 (Hard distribution for graph connectivity) Let $\beta_0 = \frac{1}{21}$. We generate the input graph G using the following procedure.

- (1) Let T be a UST on vertex set $V = [n]$.
- (2) If $B_{\beta_0}(T) = \emptyset$, return to the previous step. Otherwise, let $e_0 \sim \text{Unif}(B_{\beta_0}(T))$.
- (3) Throw a fair coin $z \sim \text{Ber}(1/2)$. If $z = 1$, output $G = T$; if $z = 0$, output $G = T \setminus e_0$.

We define the following three-phase problem, where the algorithm has more power than in the noisy query model.

Definition 27 (Three-phase problem) Let G be generated from Definition 26. Let $c_1, c_2 > 0$ be constants to be determined later. Consider the following three-phase problem.

1. Phase 1: The algorithm makes $m_1 = c_1 \log n$ noisy queries to every unordered pair of vertices $(u, v) \in \binom{V}{2}$.
2. Phase 2: The oracle reveals some edges and non-edges of G . The choice of these edges and non-edges will be described later.
3. Phase 3: The algorithm makes up to $m_2 = c_2 n^2$ (adaptive) exact queries.

The goal of the algorithm is to determine whether the input graph G is connected.

Lemma 28 If no algorithm can solve the three-phase problem (Definition 27) with error probability $\epsilon > 0$, then no algorithm can solve the graph connectivity problem with error probability ϵ using at most $m_1 m_2 = c_1 c_2 n^2 \log n$ noisy queries.

The proof is similar to Lemma 6 and omitted.

Proposition 29 (Hardness of the three-phase problem) For some choices of c_1, c_2 , and Phase 2 strategy (Definition 27), the following is true: there exists $\epsilon > 0$ such that no algorithm can solve the three-phase problem (Definition 27) with error probability ϵ .

Proof [Proof of Theorem 2] Combining Proposition 29 and Lemma 28. ■

The following sections are devoted to the proof of Proposition 29.

C.4. Phase 1

In Phase 1, the algorithm makes $m_1 = c_1 \log n$ noisy queries to every potential edge $e \in \binom{V}{2}$. Let a_e denote the number of times where a query to e returns 1. Then for $e \in G$, $a_e \sim \text{Bin}(m_1, 1 - p)$; for $e \notin G$, $a_e \sim \text{Bin}(m_1, p)$. For $0 \leq k \leq m_1$, define

$$p_k = \mathbb{P}(\text{Bin}(m_1, 1 - p) = k) = \binom{m_1}{k} (1 - p)^k p^{m_1 - k}.$$

Let $I = [pm_1 - \log^{0.6} n, pm_1 + \log^{0.6} n]$.

Lemma 30 *Let $x \sim \text{Bin}(m_1, 1 - p)$, $y \sim \text{Bin}(m_1, p)$. Then*

$$\mathbb{P}(x \in I) = n^{-c_3 \pm o(1)}, \quad (14)$$

$$\mathbb{P}(y \in I) = 1 - o(1), \quad (15)$$

where $c_3 = c_1(1 - 2p) \log \frac{1-p}{p}$.

The proof is the same as [Lemma 8](#) and omitted.

Let $\mathbb{P}^{(0)}$ denote the graph distribution in [Definition 26](#). Let $\mathbb{P}^{(1)}$ denote the posterior distribution of G conditioned on observations in Phase 1. Let $\mathcal{C}^{(0)}$ (resp. $\mathcal{C}^{(1)}$) denote the support of $\mathbb{P}^{(0)}$ (resp. $\mathbb{P}^{(1)}$). Then $\mathcal{C}^{(1)} = \mathcal{C}^{(0)}$ and for any graph $H \in \mathcal{C}^{(0)}$, we have

$$\begin{aligned} \mathbb{P}^{(1)}(H) &\propto \mathbb{P}\left((a_e)_{e \in \binom{V}{2}} | H\right) \mathbb{P}^{(0)}(H), \\ &= \left(\prod_{e \in H} p_{a_e}\right) \left(\prod_{e \in H^c} p_{m_1 - a_e}\right) \mathbb{P}^{(0)}(H). \end{aligned} \quad (16)$$

where H^c denotes the complement $\binom{V}{2} \setminus H$.

C.5. Phase 2

In Phase 2, the oracle reveals some edges and non-edges of G as follows.

- 2a. In Step 2a, the oracle reveals potential edges e with $a_e \notin I$.
- 2b. In Step 2b, the oracle reveals every $e \in G$ independently with probability q_{a_e} . We choose $q_j = 1 - \frac{pm_1 - j p_{j_l}}{p_j pm_1 - j_l}$ for $j \in I$ where $j_l = pm_1 - m_1^{0.6}$.
- 2c. In Step 2c, the oracle reveals $n - 2$ edges of G as follows. If G is disconnected, reveal all edges of G . Otherwise, G is connected and is some tree T . If G has a β_0 -balanced edge that is not revealed yet, uniformly randomly choose an edge e^* from all such edges, and reveal all edges of $G \setminus e^*$. If all β_0 -balanced edges of G have been revealed, report failure.

C.5.1. Step 2a and Step 2b

By the same analysis as in [Sections B.3.1](#) and [B.3.2](#), observations up to Step 2b have the same effect as the following procedure:

Definition 31 (Alternative observation procedure) *Let G be generated as in [Definition 26](#).*

(1) Observe every edge $e \in G$ independently with probability

$$\begin{aligned} p_+ &= 1 - \sum_{j \in I} p_j(1 - q_j) = 1 - \frac{p_{j_l}}{p_{m_1 - j_l}} \cdot \sum_{j \in I} p_{m_1 - j} \\ &= 1 - (1 \pm o(1)) \frac{p_{j_l}}{p_{m_1 - j_l}} = 1 - n^{-c_3 \pm o(1)}. \end{aligned}$$

(2) Observe every non-edge $e \in G^c$ independently with probability

$$p_- = \mathbb{P}(\text{Bin}(m_1, p) \notin I) = o(1).$$

Let $\mathbb{P}^{(2b)}$ be the posterior distribution of G after Step 2b and $\mathcal{C}^{(2b)}$ be its support. Let $E_+^{(2a)}$ (resp. $E_-^{(2a)}$) denote the set of edges (resp. non-edges) revealed in Step 2a. Let $E_+^{(2b)}$ be the set of edges revealed in Step 2b that were not revealed in Step 2a. Define $E_+^{(\leq 2b)} = E_+^{(2a)} \cup E_+^{(2b)}$. Then $\mathcal{C}^{(2b)}$ is the set of graphs $H \in \mathcal{C}^{(0)}$ satisfying $E_+^{(\leq 2b)} \subseteq H$ and $E_-^{(2a)} \cap H = \emptyset$.

By the same analysis as in [Sections B.3.1](#) and [B.3.2](#), the posterior distribution satisfies

$$\mathbb{P}^{(2b)}(H) \propto \left(\frac{p_{k_l}}{p_{m_1 - k_l}} \right)^{|H| - (n-2)} \mathbb{P}^{(0)}(H). \quad (17)$$

for $H \in \mathcal{C}^{(2b)}$.

C.5.2. Step 2c

Let $E_+^{(2c)}$ be the set of edges revealed in Step 2c that were not revealed in previous steps. Let $\mathbb{P}^{(2c)}$ be the posterior distribution of G and $\mathcal{C}^{(2c)}$ be the support of $\mathbb{P}^{(2c)}$.

Lemma 32 *Conditioned on G being connected, with probability $\Omega(1)$, Step 2c does not report failure, and e^* is $\frac{1}{3}$ -balanced.*

Proof Step 2c reports failure when G is connected and all β_0 -balanced edges have been revealed in previous steps. Let T be a UST. By [Corollary 25](#), for some $\epsilon, \gamma_1, \gamma_2 > 0$, we have

$$\mathbb{P}[\gamma_1 \sqrt{n} \leq |B_{1/3}(T)| \leq |B_{\beta_0}(T)| \leq \gamma_2 \sqrt{n}] > \epsilon.$$

When constructing the input distribution ([Definition 14](#)), conditioned on G being connected, the distribution of G is uniform over all spanning trees with at least one β_0 -balanced edge. Let \mathcal{E}_1 be the event that G is connected, has at least $\gamma_1 \sqrt{n}$ $\frac{1}{3}$ -balanced edges, and at most $\gamma_2 \sqrt{n}$ β_0 -balanced edges. By the above discussion, $\mathcal{P}(\mathcal{E}_1) = \Omega(1)$. In [Definition 31](#), every edge is observed independently with probability $p_{\pm} = 1 - n^{-c_3 \pm o(1)}$. By choosing $c_1 > 0$ small enough, we can let $c_3 > 0$ be arbitrarily small. Let $\tilde{B}_{1/3}$ (resp. \tilde{B}_{β_0}) be the set of $\frac{1}{3}$ -balanced (resp. β_0 -balanced) edges of G not revealed in previous steps. Then

$$\tilde{B}_{1/3} \subseteq \tilde{B}_{\beta_0}, \quad \left| \tilde{B}_{1/3} \right| \sim \text{Bin}(|B_{1/3}(G)|, 1 - p_+), \quad \left| \tilde{B}_{\beta_0} \right| \sim \text{Bin}(|B_{\beta_0}(G)|, 1 - p_+).$$

Note that

$$\begin{aligned}\mathbb{E} [\text{Bin}(|B_{1/3}(G)|, 1 - p_+)] &= |B_{1/3}(G)| \cdot (1 - p_+) = n^{1/2 - c_3 \pm o(1)}, \\ \mathbb{E} [\text{Bin}(|B_{\beta_0}(G)|, 1 - p_+)] &= |B_{\beta_0}(G)| \cdot (1 - p_+) = n^{1/2 - c_3 \pm o(1)}.\end{aligned}$$

By Bernstein's inequality, there exists $\gamma_3, \gamma_4 > 0$ such that conditioned on \mathcal{E}_1 , we have

$$\begin{aligned}\mathbb{P} \left[\left| \tilde{B}_{1/3} \right| \geq \gamma_3 |B_{1/3}(G)| \cdot (1 - p_+) \mid \mathcal{E}_1 \right] &\geq 1 - \exp \left(-n^{1/2 - c_3 \pm o(1)} \right), \\ \mathbb{P} \left[\left| \tilde{B}_{\beta_0} \right| \leq \gamma_4 |B_{\beta_0}(G)| \cdot (1 - p_+) \mid \mathcal{E}_1 \right] &\geq 1 - \exp \left(-n^{1/2 - c_3 \pm o(1)} \right).\end{aligned}$$

Let \mathcal{E}_2 be the event that $\left| \tilde{B}_{1/3} \right| \geq \gamma_3 |B_{1/3}(G)| \cdot (1 - p_+)$ and $\left| \tilde{B}_{\beta_0} \right| \leq \gamma_4 |B_{\beta_0}(G)| \cdot (1 - p_+)$. The above discussion shows that $\mathbb{P}(\mathcal{E}_2 | \mathcal{E}_1) = 1 - o(1)$.

When \mathcal{E}_1 and \mathcal{E}_2 both happen, we have $\frac{|\tilde{B}_{1/3}|}{|\tilde{B}_{\beta_0}|} \geq \frac{\gamma_3 \gamma_1}{\gamma_4 \gamma_2} > 0$. So conditioned on $\mathcal{E}_1 \cap \mathcal{E}_2$, the probability that e^* is $\frac{1}{3}$ -balanced is $\Omega(1)$. This finishes the proof. \blacksquare

From now on we condition on the event that Step 2c does not report failure, and e^* is $\frac{1}{3}$ -balanced. Let us consider the posterior distribution $\mathbb{P}^{(2c)}$. Let $E_+^{(\leq 2c)} = E_+^{(\leq 2b)} \cup E_+^{(2c)}$ be the set of observed edges at the end of Step 2c. Then $E_+^{(\leq 2c)}$ consists of $n - 2$ edges, which is a forest with two components T_1 and T_2 , each containing at least $\frac{n}{3}$ and at most $\frac{2n}{3}$ vertices. The support $\mathcal{C}^{(2c)}$ is easy to describe. Let $G_0 = T_1 \cup T_2$ and $G_e = T_1 \cup T_2 \cup \{e\}$ for $e \in E(T_1, T_2)$. Then

$$\mathcal{C}^{(2c)} = \{G_0\} \cup \left\{ G_e : e \in E(T_1, T_2) \setminus E_-^{(2a)} \right\}.$$

The posterior distribution $\mathbb{P}^{(2c)}$ is not simply the distribution $\mathbb{P}^{(2b)}$ restricted to $\mathcal{C}^{(2c)}$. For $H \in \mathcal{C}^{(2c)}$, we have

$$\begin{aligned}\mathbb{P}^{(2c)}(H) &\propto \mathbb{P} \left(E_+^{(2c)} \mid H, E_+^{(\leq 2b)} \right) \mathbb{P}^{(2b)}(H) \\ &\propto \mathbb{P} \left(E_+^{(2c)} \mid H, E_+^{(\leq 2b)} \right) \left(\frac{p_{k_l}}{p_{m_1 - k_l}} \right)^{|H| - (n-2)} \mathbb{P}^{(0)}(H).\end{aligned}\tag{18}$$

For $H = G_0$, Eq. (18) simplifies to

$$\mathbb{P}^{(2c)}(G_0) \propto \mathbb{P}^{(0)}(G_0).\tag{19}$$

For $H = G_e$, Eq. (18) simplifies to

$$\mathbb{P}^{(2c)}(G_e) \propto \frac{1}{|B_{\beta_0}(G_e) \setminus E_+^{(\leq 2b)}|} \cdot \frac{p_{k_l}}{p_{m_1 - k_l}} \mathbb{P}^{(0)}(G_e).\tag{20}$$

Note that the \propto symbols in Eqs. (19) and (20) hide the same factor.

Further simplifying Eqs. (19) and (20), we get

$$\mathbb{P}^{(2c)}(G_0) \propto \sum_{e \in E(T_1, T_2)} \frac{1}{|B_{\beta_0}(G_e)|},\tag{21}$$

$$\mathbb{P}^{(2c)}(G_e) \propto \frac{1}{|B_{\beta_0}(G_e) \setminus E_+^{(\leq 2b)}|} \cdot \frac{p_{k_l}}{p_{m_1 - k_l}}, \quad \forall e \in E(T_1, T_2) \setminus E_-^{(2a)}.\tag{22}$$

We now consider the set $B_{\beta_0}(G_e)$ for $e \in E(T_1, T_2)$. Let $\beta_1 = \frac{\beta_0 n}{|T_1|}$, $\beta_2 = \frac{\beta_0 n}{|T_2|}$. Then $\frac{1}{14} \leq \beta_1, \beta_2 \leq \frac{1}{7}$ and $B_{\beta_1}(T_1) \cup B_{\beta_2}(T_2) \subseteq B_{\beta_0}(G_e)$ for all $e \in E(T_1, T_2)$. For $e_i \in T_i$ ($i = 1, 2$), let $S_{T_i}(e_i)$ be the set of vertices in the smaller component of $T_i \setminus e_i$. (If the two components have the same size, choose a side arbitrarily.) For $e = (u_1, u_2) \in E(T_1, T_2)$ (with $u_i \in T_i$, $i = 1, 2$), an edge $e_i \in T_i \setminus B_{\beta_i}(T_i)$ ($i = 1, 2$) is in $B_{\beta_0}(G_e)$ if and only if $u_i \in S_{T_i}(e_i)$. For $i \in \{1, 2\}$ and $e = (u_1, u_2) \in E(T_1, T_2)$, define

$$B'_{T_i}(e) = \{e_i : e_i \in T_i \setminus B_{\beta_i}(T_i), u_i \in S_{T_i}(e_i)\}.$$

Then for $e \in E(T_1, T_2)$, we have

$$B_{\beta_0}(G_e) = \{e\} \cup B_{\beta_1}(T_1) \cup B_{\beta_2}(T_2) \cup B'_{T_1}(e) \cup B'_{T_2}(e). \quad (23)$$

Note that the union is a disjoint union.

Lemma 33 *Conditioned on G being connected, there exist constants $\epsilon, \gamma_1, \gamma_2, \gamma_3 > 0$ such that with probability at least ϵ , the following are true simultaneously.*

(i) *Step 2c does not report failure and $e^* \in B_{1/3}(G)$.*

(ii)

$$\gamma_1 \sqrt{n} \leq |B_{\beta_1}(T_1)| + |B_{\beta_2}(T_2)| \leq \gamma_2 \sqrt{n}.$$

(iii)

$$\sum_{e_1 \in T_1 \setminus B_{\beta_1}(T_1)} |S_{T_1}(e_1)| + \sum_{e_2 \in T_2 \setminus B_{\beta_2}(T_2)} |S_{T_2}(e_2)| \leq \gamma_3 n^{3/2}.$$

Proof Let T be a UST. Let \mathcal{E}_1 be the event that all items in [Corollary 25](#) hold. Then conditioned on G being connected, \mathcal{E}_1 happens with probability $\Omega(1)$. In the following, condition on that G is connected and \mathcal{E}_1 happens.

Let \mathcal{E}_2 be the event that [Lemma 33, Item \(i\)](#) holds. By the proof of [Lemma 32](#), conditioned on \mathcal{E}_1 , \mathcal{E}_2 happens with probability $\Omega(1)$. In the following, condition on that \mathcal{E}_1 and \mathcal{E}_2 both happen.

By [Corollary 25, Item \(ii\)](#), and because $e^* \in B_{1/3}(T)$,

$$\begin{aligned} |B_{\beta_1}(T_1)| + |B_{\beta_2}(T_2)| &\geq |B_{1/7}(T_1)| + |B_{1/7}(T_2)| \geq \gamma_3 \sqrt{n}, \\ |B_{\beta_1}(T_1)| + |B_{\beta_2}(T_2)| &\leq |B_{1/14}(T_1)| + |B_{1/14}(T_2)| \leq \gamma_4 \sqrt{n}. \end{aligned}$$

Therefore conditioned on \mathcal{E}_1 and \mathcal{E}_2 , [Lemma 33, Item \(ii\)](#) holds.

By [Corollary 25, Item \(iii\)](#), we have

$$\sum_{e_1 \in T_1 \setminus B_{\beta_1}(T_1)} |S_{T_1}(e_1)| + \sum_{e_2 \in T_2 \setminus B_{\beta_2}(T_2)} |S_{T_2}(e_2)| \leq \sum_{e \in T} |S_T(e)| \leq \gamma_5 n^{3/2}.$$

Therefore conditioned on \mathcal{E}_1 and \mathcal{E}_2 , [Lemma 33, Item \(iii\)](#) holds. ■

Corollary 34 *Conditioned on G being connected, there exist constants $\epsilon, \gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5 > 0$ such that with probability at least ϵ , the following are true simultaneously.*

(i) *For all $e \in E(T_1, T_2)$,*

$$|B_{\beta_0}(G_e)| \geq \gamma_1 \sqrt{n}.$$

(ii) *For all $e \in E(T_1, T_2)$,*

$$\gamma_2 |B_{\beta_0}(G_e)| \cdot (1 - p_+) \leq \left| B_{\beta_0}(G_e) \setminus E_+^{(\leq 2b)} \right| \leq \gamma_3 |B_{\beta_0}(G_e)| \cdot (1 - p_+). \quad (24)$$

(iii)

$$\gamma_4 n^{3/2} \leq \sum_{e \in E(T_1, T_2)} \frac{1}{|B_{\beta_0}(G_e)|} \leq \gamma_5 n^{3/2}.$$

Proof Let T be a UST. Let \mathcal{E} be the event that all items in Lemma 33 hold. Then conditioned G being connected, \mathcal{E} happens with probability $\Omega(1)$. In the following, condition on that G is connected and \mathcal{E} happens.

By Lemma 33, Item (ii) and Eq. (23), we have

$$|B_{\beta_0}(G_e)| \geq |B_{\beta_1}(T_1)| + |B_{\beta_2}(T_2)| \geq \gamma_1 \sqrt{n}.$$

So Corollary 34, Item (i) holds. This implies the upper bound in Corollary 34, Item (iii) as

$$\sum_{e \in E(T_1, T_2)} \frac{1}{|B_{\beta_0}(G_e)|} \leq n^2 \cdot \frac{1}{\gamma_1 \sqrt{n}} = \gamma_1^{-1} n^{3/2}.$$

By Corollary 34, Item (i) and Bernstein's inequality, for every $e \in E(T_1, T_2)$, with probability $1 - \exp(-n^{1/2 - c_3 \pm o(1)})$, Eq. (24) holds. By union bound, with probability $1 - o(1)$, Eq. (24) holds for all $e \in E(T_1, T_2)$. This proves Corollary 34, Item (ii).

It remains to prove the lower bound in Corollary 34, Item (iii). By Lemma 33, Items (ii) and (iii), and Eq. (23), we have

$$\begin{aligned} & \sum_{e \in E(T_1, T_2)} |B_{\beta_0}(G_e)| \\ &= \sum_{e \in E(T_1, T_2)} (1 + |B_{\beta_1}(T_1)| + |B_{\beta_2}(T_2)| + |B'_{T_1}(e)| + |B'_{T_2}(e)|) \\ &\leq \sum_{e \in E(T_1, T_2)} (1 + \gamma_2 \sqrt{n} + |B'_{T_1}(e)| + |B'_{T_2}(e)|) \\ &\leq (1 + \gamma_2) \sqrt{n} |T_1| |T_2| + \sum_{e_1 \in T_1 \setminus B_{\beta_1}(T_1)} |S_{T_1}(e_1)| \cdot |T_2| + \sum_{e_2 \in T_2 \setminus B_{\beta_2}(T_2)} |S_{T_2}(e_2)| \cdot |T_1| \\ &\leq (1 + \gamma_2 + \gamma_3) n^{5/2}. \end{aligned}$$

On the other hand, by Cauchy-Schwarz inequality,

$$\left(\sum_{e \in E(T_1, T_2)} \frac{1}{|B_{\beta_0}(G_e)|} \right) \left(\sum_{e \in E(T_1, T_2)} |B_{\beta_0}(G_e)| \right) \geq (|T_1||T_2|)^2 \geq \frac{2}{9}n^2.$$

So

$$\sum_{e \in E(T_1, T_2)} \frac{1}{|B_{\beta_0}(G_e)|} \geq \frac{2}{9(\gamma_2 + \gamma_3)} n^{3/2}.$$

■

Now we are able to further utilize [Eq. \(21\)](#) and [Eq. \(22\)](#). Write

$$Z^{(2c)} = \sum_{e \in E(T_1, T_2)} \frac{1}{|B_{\beta_0}(G_e)|} + \sum_{e \in E(T_1, T_2) \setminus E_-^{(2a)}} \frac{1}{|B_{\beta_0}(G_e) \setminus E_+^{(\leq 2b)}|} \cdot \frac{p_{k_l}}{p_{m_1 - k_l}}.$$

Conditioned on that all items in [Corollary 34](#) hold, for all $e \in E(T_1, T_2) \setminus E_-^{(2a)}$, we have

$$\frac{1}{|B_{\beta_0}(G_e) \setminus E_+^{(\leq 2b)}|} \cdot \frac{p_{k_l}}{p_{m_1 - k_l}} \asymp \frac{1}{|B_{\beta_0}(G_e)| (1 - p_+)} \cdot \frac{p_{k_l}}{p_{m_1 - k_l}} \asymp \frac{1}{|B_{\beta_0}(G_e)|},$$

where the second step holds because

$$\frac{p_{m_1 - k_l}}{p_{k_l}} \cdot (1 - p_+) = \frac{p_{m_1 - k_l}}{p_{k_l}} \cdot \sum_{k \in I} p_k (1 - q_k) = \sum_{k \in I} p_{m_1 - k} = 1 - p_- = 1 - o(1).$$

Therefore,

$$Z^{(2c)} \asymp \sum_{e \in E(T_1, T_2)} \frac{1}{|B_{\beta_0}(G_e)|} \asymp n^{3/2},$$

$$\mathbb{P}^{(2c)}(G_0) = \frac{1}{Z^{(2c)}} \sum_{e \in E(T_1, T_2)} \frac{1}{|B_{\beta_0}(G_e)|} \asymp 1.$$

By Bernstein's inequality, for any $\delta_1 > 0$, we have

$$\mathbb{P} \left[\left| E_-^{(2a)} \right| \geq \delta_1 n^2 \right] = o(1).$$

So for small enough $\delta_1 > 0$, with probability $1 - o(1)$, we have

$$\begin{aligned}
& \sum_{e \in E(T_1, T_2) \setminus E_-^{(2a)}} \mathbb{P}^{(2c)}(G_e) \\
&= \frac{1}{Z^{(2c)}} \sum_{e \in E(T_1, T_2) \setminus E_-^{(2a)}} \frac{1}{|B_{\beta_0}(G_e) \setminus E_+^{(\leq 2b)}|} \cdot \frac{p_{k_l}}{p_{m_1 - k_l}} \\
&\asymp n^{-3/2} \sum_{e \in E(T_1, T_2) \setminus E_-^{(2a)}} \frac{1}{|B_{\beta_0}(G_e)|} \\
&\asymp n^{-3/2} \left(\sum_{e \in E(T_1, T_2)} \frac{1}{|B_{\beta_0}(G_e)|} - \sum_{e \in E_-^{(2a)}} \frac{1}{|B_{\beta_0}(G_e)|} \right) \\
&\asymp n^{-3/2} \left(n^{3/2} - \delta_1 n^2 \cdot n^{-1/2} \right) \\
&\asymp 1,
\end{aligned}$$

where in the second-to-last step we used [Corollary 34, Items \(i\) and \(iii\)](#), and $|E_-^{(2a)}| \leq \delta_1 n^2$.

Summarizing the above, at the end of Step 2c, with probability $\Omega(1)$, we have

$$\begin{aligned}
\mathbb{P}^{(2c)}(\text{disconnected}) &= \mathbb{P}^{(2c)}(G_0) = \Theta(1), \\
\mathbb{P}^{(2c)}(\text{connected}) &= \sum_{e \in E(T_1, T_2) \setminus E_-^{(2a)}} \mathbb{P}^{(2c)}(G_e) = \Theta(1).
\end{aligned}$$

C.6. Phase 3

In Phase 3, the algorithm makes $c_2 n^2$ adaptive exact queries. We show that for $c_2 > 0$ small enough, with probability $\Omega(1)$, the algorithm will not be able to return the correct answer.

Let $E^{(3)}$ be the set of edges queried in Phase 3. We can w.l.o.g. assume that $E^{(3)} \subseteq E(T_1, T_2) \setminus E_-^{(2a)}$, because only queries in this set are useful. Conditioned on G being connected, the probability that $E^{(3)}$ hits the edge e^* is

$$\frac{\sum_{e \in E^{(3)}} \mathbb{P}^{(2c)}(G_e)}{\sum_{e \in E(T_1, T_2) \setminus E_-^{(2a)}} \mathbb{P}^{(2c)}(G_e)} \leq c_2 n^2 \cdot \Theta(n^{-2}) \asymp c_2,$$

which is $1 - \Omega(1)$ for $c_2 > 0$ small enough. Therefore, for small enough c_2 , with probability $\Omega(1)$, $E^{(3)}$ does not hit the edge e^* .

Conditioned on $E^{(3)}$ does not hit e^* , let $\mathbb{P}^{(3)}$ denote the posterior distribution of the original graph G given all observations. We have

$$\begin{aligned}
\mathbb{P}^{(3)}(\text{disconnected}) &= \mathbb{P}^{(3)}(G_0), \\
\mathbb{P}^{(3)}(\text{connected}) &= \sum_{e \in E(T_1, T_2) \setminus (E_-^{(2a)} \cup E^{(3)})} \mathbb{P}^{(3)}(G_e),
\end{aligned}$$

where

$$\begin{aligned}\mathbb{P}^{(2c)}(G_0) &= \frac{1}{Z^{(3)}} \sum_{e \in E(T_1, T_2)} \frac{1}{|B_{\beta_0}(G_e)|}, \\ \mathbb{P}^{(2c)}(G_e) &= \frac{1}{Z^{(3)}} \frac{1}{|B_{\beta_0}(G_e) \setminus E_+^{(\leq 2b)}|} \cdot \frac{p_{k_l}}{p_{m_1 - k_l}}, \\ Z^{(3)} &= \sum_{e \in E(T_1, T_2)} \frac{1}{|B_{\beta_0}(G_e)|} + \sum_{e \in E(T_1, T_2) \setminus (E_-^{(2a)} \cup E^{(3)})} \frac{1}{|B_{\beta_0}(G_e) \setminus E_+^{(\leq 2b)}|} \cdot \frac{p_{k_l}}{p_{m_1 - k_l}}.\end{aligned}$$

By the same discussion as in the end of Phase 2, Step 2c, for $c_2 > 0$ small enough, with probability $\Omega(1)$, we have

$$\begin{aligned}\mathbb{P}^{(3)}(\text{disconnected}) &= \Theta(1), \\ \mathbb{P}^{(3)}(\text{connected}) &= \Theta(1).\end{aligned}$$

In this case, any return value would lead to an error probability of $\Omega(1)$.

This concludes the proof of [Proposition 29](#).

C.7. s - t Connectivity

In this section we modify the proof of [Theorem 2](#) to show hardness of s - t Connectivity. Recall the s - t Connectivity problem, where the input is an unknown undirected graph on n labeled vertices, and a pair of vertices $s, t \in V$. An algorithm can make noisy queries to edge membership and the goal is to determine whether s and t are in the same connected component of G .

Proposition 35 (Hardness of s - t Connectivity) *Any algorithm that solves the s - t Connectivity problem with $\frac{1}{3}$ error probability uses $\Omega(n^2 \log n)$ noisy queries in expectation.*

Proof As discussed in [Section 2.2](#), the error probability in the proposition statement can be replaced with any $0 < \epsilon < \frac{1}{2}$, and the expected number of queries can be replaced with worst-case number of queries.

We design an input distribution for s - t Connectivity by generating G from [Definition 26](#), and choosing s, t i.i.d. $\sim \text{Unif}(V)$.

Then we run the same proof as [Theorem 2](#). That is, we define a three-phase problem for s - t Connectivity, where the oracle uses the same strategy in Phase 2 as in Graph Connectivity. Because s and t are independent with G , in the end of Phase 2, conditioned on Step 2c does not report failure, with probability $\Omega(1)$, $s \in T_1$ and $t \in T_2$. In this case, s - t Connectivity is equivalent to Graph Connectivity. In the proof of Graph Connectivity, we have shown that with probability $\Omega(1)$ (over the randomness of the graph and Phase 1 and 2), any algorithm that uses at most $c_2 n^2$ queries in Phase 3 has $\Omega(1)$ error probability. This implies that the same holds for s - t Connectivity. \blacksquare

Appendix D. Threshold and Counting

In this section we present our proof for [Theorem 3](#) and [Theorem 4](#). We use TH_k^n to denote k -Threshold problem with input length n .

D.1. Lower bound for k -Threshold

In this section we prove the lower bound part of [Theorem 3](#). That is, solving TH_k^n for $k \leq 2n - 1$ requires at least $(1 - o(1)) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$ noisy queries in expectation. The $k = o(n)$ case has been proved in ([Wang et al., 2025](#)) (see also [Section E](#) for our alternative and simpler proof).

Theorem 36 *For $k = o(n)$, solving TH_k^n with $\delta = o(1)$ error probability requires*

$$(1 - o(1)) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$$

noisy queries in expectation, even when the input is uniformly chosen from $\binom{[n]}{k}$ with probability $1/2$ and uniformly chosen from $\binom{[n]}{k-1}$ with probability $1/2$.

In our lower bound proof, we will use [Theorem 36](#) with $k = O(n/\log n)$.

We first prove the case where $k = (n + 1)/2$.

Lemma 37 *Solving TH_k^{2k-1} with $\delta = o(1)$ error probability requires*

$$(1 - o(1)) \frac{2k \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$$

noisy queries in expectation, even when the input is uniformly chosen from $\binom{[2k-1]}{k}$ with probability $1/2$ and uniformly chosen from $\binom{[2k-1]}{k-1}$ with probability $1/2$.

Proof Suppose for the sake of contradiction that we have an algorithm \mathcal{A} that solves TH_k^{2k-1} with error probability $\delta = o(1)$ and uses only $(1 - \epsilon) \frac{2k \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$ noisy queries in expectation, for some absolute constant $\epsilon > 0$. Let \mathcal{D} be a distribution of inputs where with $1/2$ probability the input is chosen uniformly from $\binom{[2k-1]}{k}$, and with $1/2$ probability the input is chosen uniformly from $\binom{[2k-1]}{k-1}$. Now we consider two cases, depending on whether the expected number of queries \mathcal{A} make on indices with 1's is larger or not. In either case, we will use \mathcal{A} to obtain an algorithm more efficient than the lower bound in [Theorem 36](#), thus reaching a contradiction.

First, suppose \mathcal{A} makes more queries in expectation on indices with 1's under input distribution \mathcal{D} . Let $k' = \Theta(k/\log k)$ and let $n = k - 1 + k'$. Consider an instance of $\text{TH}_{k'}^n$ where the input is uniformly chosen from $\binom{[n]}{k'}$ with probability $1/2$ and uniformly chosen from $\binom{[n]}{k'-1}$ with probability $1/2$. By [Theorem 36](#), this instance requires $(1 - o(1)) \frac{n \log \frac{k'}{\delta}}{D_{\text{KL}}(p \| 1-p)} = (1 - o(1)) \frac{k \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$ queries in expectation. We will design an algorithm \mathcal{B} solving such an instance utilizing \mathcal{A} .

When \mathcal{B} gets the input, it first adds $2k - 1 - n$ 1's to the input, and then randomly shuffle the indices. Then \mathcal{B} sends this input to \mathcal{A} . Whenever \mathcal{A} makes a query to a 1 that is artificially added, \mathcal{B} simulates a noisy query using random bits; when \mathcal{A} makes a query to an actual input, \mathcal{B} makes a query as well and pass the result to \mathcal{A} . When \mathcal{A} returns a result, \mathcal{B} returns the same result. It is not difficult to verify that the input distribution for \mathcal{A} is exactly \mathcal{D} . Also, whenever the input to \mathcal{A} has at least k 1's, the input to \mathcal{B} has k' 1's, and vice versa, so the correct output of \mathcal{A} is the same as the correct output of \mathcal{B} . Therefore, \mathcal{B} is correct whenever \mathcal{A} is correct, which happens with probability δ .

Next, we analyze the expected number of queries \mathcal{B} makes, which consist of two parts:

- The number of queries \mathcal{A} makes to an actual 0 in the input: Because we are in the case where \mathcal{A} makes more queries in expectation on indices with 1's in the input than indices with 0's, the expected number of this type of queries is at most half of the expected total number of queries \mathcal{A} makes. Thus, the number of queries in this case is at most $(1 - \epsilon) \frac{k \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$.
- The number of queries \mathcal{A} makes to an actual 1 in the input: Each actual 1 in the input to \mathcal{B} is later permuted to a random position in the sequence. By symmetry, a random 1 in the input to \mathcal{A} under the input distribution \mathcal{D} is queried at most $\frac{1}{k-1} \cdot (1 - \epsilon) \frac{2k \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$ times in expectation. The number of actual 1's is $k' = \Theta(k / \log k)$, so the expected number of queries \mathcal{A} makes on them is $\frac{k'}{k-1} \cdot (1 - \epsilon) \frac{2k \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)} = o(1) \cdot \frac{k \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$.

As a result, the total number of queries \mathcal{B} makes is $(1 - \epsilon) \frac{k \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$, which contradicts [Theorem 36](#).

For the second case where \mathcal{A} makes more (or equal number of) queries in expectation on indices with 0's under input distribution \mathcal{D} . The only difference is that, when \mathcal{B} sends the input to \mathcal{A} , it has to flip the roles of 0's and 1's. Additionally, it has to flip the result \mathcal{A} returns. We omit the details. ■

Given [Theorem 36](#) and [Lemma 37](#), we are ready to prove the lower bound for general k .

Theorem 38 *Solving TH_k^n with $\delta = o(1)$ error probability requires*

$$(1 - o(1)) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$$

noisy queries in expectation for $n / \log n \leq k \leq n/2$.

Proof The high-level proof strategy is similar to that of [Lemma 37](#), by reducing from a hard instance to TH_k^n . However, the difference is that we need to reduce from two different cases depending on whether the average number of queries per 1 or per 0 is larger. In [Lemma 37](#) we did not have to do it because we can simply flip all the 0's and 1's in the input and retain the same problem as $n = 2k - 1$.

Let \mathcal{D} be the input distribution where with $1/2$ probability the input is chosen uniformly from $\binom{[n]}{k}$ and with $1/2$ probability the input is chosen uniformly from $\binom{[n]}{k-1}$. Let \mathcal{A} be an algorithm solving TH_k^n under input distribution \mathcal{D} using $(1 - \epsilon) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$ noisy queries in expectation, for some absolute constant $\epsilon > 0$. Let q_0 denote the expected number of queries \mathcal{A} makes on a random index with input value 0 (under input distribution \mathcal{D}), and let q_1 denote the expected number of queries \mathcal{A} makes on a random index with input value 1 (under input distribution \mathcal{D}). Let Q be the expected number of queries \mathcal{A} makes.

Consider the following two cases: $q_0 \leq q_1$ and $q_0 > q_1$.

Case $q_0 \leq q_1$. Let $k' = \Theta(n / \log n) \leq k$ and let $n' = n - k + k'$. Note that $n' \in [n/2, n]$, so we have $k' = \Theta(n' / \log n')$. By [Theorem 36](#), solving $\text{TH}_{k'}^{n'}$ under input distribution where with $1/2$ probability the input is uniformly from $\binom{[n']}{k'}$ and with $1/2$ probability the input is uniformly from $\binom{[n']}{k'-1}$ with error probability δ requires $(1 - o(1)) \frac{n' \log \frac{k'}{\delta}}{D_{\text{KL}}(p \| 1-p)} = (1 - o(1)) \frac{n' \log \frac{n}{\delta}}{D_{\text{KL}}(p \| 1-p)}$ noisy queries in expectation.

Given an instance of $\text{TH}_{k'}^{n'}$, we add $n - n'$ artificial 1's to the input, and then randomly permute the input, and feed it to \mathcal{A} . If \mathcal{A} queries an actual input, we also make an actual query; if \mathcal{A} queries an artificial input, we can simulate a query without making an actual query. Finally, we use the result returned by \mathcal{A} as our answer. It is not difficult to verify that this algorithm is correct with error probability δ , and the input distribution to \mathcal{A} is \mathcal{D} .

Let us analyze the expected number of queries Q' used by the algorithm, which can be expressed as follows:

$$\frac{1}{2} \cdot (q_1(k' - 1) + q_0(n - k + 1)) + \frac{1}{2} \cdot (q_1 k' + q_0(n - k)).$$

Also, notice that the number of queries Q made by \mathcal{A} under input \mathcal{D} is

$$\frac{1}{2} \cdot (q_1(k - 1) + q_0(n - k + 1)) + \frac{1}{2} \cdot (q_1 k + q_0(n - k)).$$

As $q_0 \leq q_1$, the above implies that $q_1 \geq Q/n$. Furthermore, we have that $Q - Q' = q_1 \cdot (k - k') = q_1 \cdot (n - n') \geq \frac{n - n'}{n} \cdot Q$. Therefore,

$$Q' \leq \frac{n'}{n} \cdot Q \leq (1 - \epsilon) \frac{n' \log \frac{k}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} \leq (1 - \epsilon) \frac{n' \log \frac{n}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)},$$

which contradicts [Theorem 36](#).

Case $q_0 > q_1$. By [Lemma 37](#), solving TH_k^{2k-1} under input distribution where with $1/2$ probability the input is uniformly from $\binom{[2k-1]}{k}$ and with $1/2$ probability the input is uniformly from $\binom{[2k-1]}{k-1}$ with error probability δ requires $(1 - o(1)) \frac{2k \log \frac{k}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} = (1 - o(1)) \frac{2k \log \frac{n}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}$ noisy queries in expectation.

Given such an input to TH_k^{2k-1} , we add $n - 2k + 1$ artificial 0's to the input, and then randomly permute the input, and feed it to \mathcal{A} . Similar as before, we make an actual query if \mathcal{A} queries an actual input, and we simulate a query otherwise. It is not difficult to verify that this algorithm is correct with error probability δ , and the input distribution to \mathcal{A} is \mathcal{D} .

The expected number of actual queries used by the algorithm can be similarly analyzed as the previous case, which can be upper bounded by $(1 - \epsilon) \frac{2k \log \frac{n}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}$, contradicting [Lemma 37](#). \blacksquare

D.2. Upper bound for k -Threshold

In this section we prove our upper bound for k -Threshold, stated as follows.

Theorem 39 *Given a sequence $a \in \{0, 1\}^n$ and an integer $1 \leq k \leq n$, there is an algorithm that can output $\min\{k, \|a\|_1\}$ with error probability $\delta = o(1)$ using*

$$(1 + o(1)) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}$$

noisy queries in expectation.

D.2.1. Preliminaries

The following lemmas are standard.

Lemma 40 (e.g., (Gu and Xu, 2023; Wang et al., 2025)) *For a bit B , there is an algorithm CHECK-BIT(B, δ) that can return the value of the bit with error probability $\leq \delta$ using*

$$(1 + o_{1/\delta}(1)) \frac{\log \frac{1}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}$$

noisy queries in expectation.

Lemma 41 (e.g., (Feller, 1970)) *Consider a biased random walk on \mathbb{Z} starting at 0. At each time step, the walk adds 1 to the current value with probability $p < 1/2$, and adds -1 to the current value with probability $1 - p$. Then the probability that the random walk ever reaches some integer $x \geq 0$ is $(p/(1 - p))^x$.*

Lemma 42 (e.g., (Feller, 1970)) *Consider a biased random walk on \mathbb{Z} starting at 0. At each time step, the walk adds 1 to the current value with probability $p < 1/2$, and adds -1 to the current value with probability $1 - p$. Then the expected number of steps needed to first reach some integer $-x \leq 0$ is $\frac{x}{1 - 2p}$.*

Our algorithm for k -Threshold and Counting uses the following asymmetric version of CHECK-BIT.

Lemma 43 *For a bit B , there is an algorithm ASYMMETRIC-CHECK-BIT(B, δ_0, δ_1) such that:*

- *If the actual value of B is 0, then the algorithm returns the value of the bit with error probability $\leq \delta_0$ using*

$$(1 + o_{1/\delta_1}(1)) \frac{\log \frac{1}{\delta_1}}{D_{\text{KL}}(p \parallel 1 - p)}$$

noisy queries in expectation.

- *If the actual value of B is 1, then the algorithm returns the value of the bit with error probability $\leq \delta_1$ using*

$$(1 + o_{1/\delta_0}(1)) \frac{\log \frac{1}{\delta_0}}{D_{\text{KL}}(p \parallel 1 - p)}$$

noisy queries in expectation.

Proof Let a and b be two integer parameters to be set later. The algorithm works as follows: we keep querying the input bit, and keep track of the number of queries that returns 1 (denoted by q_1) and the number of queries that returns 0 (denoted by q_0). We stop once $q_1 - q_0 = -a$, in which case we declare the bit to be 0, or $q_1 - q_0 = b$, in which case we declare the bit to be 1.

If the input bit is 1, then the probability that the algorithm returns 0 can be upper bounded by $\left(\frac{p}{1-p}\right)^a$ using Lemma 41, so by setting $a = \left\lceil \frac{\log(1/\delta_1)}{\log((1-p)/p)} \right\rceil$, this probability is upper bounded by δ_1 .

On the other hand, if the input bit is 0, then the probability that the algorithm returns 1 can be upper bounded by $\left(\frac{p}{1-p}\right)^b$ using Lemma 41, so by setting $b = \left\lceil \frac{\log(1/\delta_0)}{\log((1-p)/p)} \right\rceil$, this probability is upper bounded by δ_0 .

Now we consider the expected running time of the algorithm. First suppose the input bit is 1, then the algorithm can be viewed as a random walk on integers starting at 0, and each time it adds 1 with probability $1 - p$ and subtracts 1 with probability p . It stops once the random walk reaches $-a$ or b . This stopping time is upper bounded by the first time it reaches b , and the expected number of steps required for it to first reach b is $\frac{b}{1-2p} = (1 + o_{1/\delta_0}(1)) \frac{\log(1/\delta_0)}{D_{\text{KL}}(p||1-p)}$ by Lemma 42. Similarly, if the input bit is 0, then the expected number of queries used by the algorithm is upper bounded by $(1 + o_{1/\delta_1}(1)) \frac{\log(1/\delta_1)}{D_{\text{KL}}(p||1-p)}$. ■

D.2.2. The proof

We are now ready to state our proof of Theorem 39.

Our algorithm is extremely simple and is outlined in Algorithm 1. It repeatedly calls the subroutine ASYMMETRIC-CHECK-BIT($a_i, \delta/2n, \delta/2k$) from Lemma 43 for every $i \in [n]$, and uses the returned value as the guess for a_i . If at any point, the current number of a_i whose guess is 1 reaches k , the algorithm terminates early and return k . Otherwise, the algorithm returns the number of a_i whose guess is 1 at the end.

Algorithm 1

```

1: procedure THRESHOLD-COUNT( $\{a_1, \dots, a_n\}, k, \delta$ )
2:    $cnt = 0$ 
3:   for  $i = 1 \rightarrow n$  do
4:      $cnt = cnt + \text{ASYMMETRIC-CHECK-BIT}(a_i, \delta/2n, \delta/2k)$ 
5:     if  $cnt \geq k$  then return  $k$ 
6:   return  $cnt$ 

```

Error probability. We first analyze the error probability of the algorithm. First, suppose $\|a\|_1 \geq k$. Let $S \subseteq [n]$ be an arbitrary size- k set where $a_i = 1$ for $i \in S$. Then by the guarantee of ASYMMETRIC-CHECK-BIT, for every $i \in S$, the probability that the guess for a_i is not 1 is $\leq \delta/2k$. Therefore, by union bound, the guesses for a_i for all $i \in S$ are 1 with probability $\geq 1 - \delta/2$. Therefore, the count will be at least k so the algorithm will return k as the correct answer with error probability $\leq \delta/2 \leq \delta$.

If $\|a\|_1 < k$, then by union bound, the probability that all guesses are correct is $\geq 1 - \frac{\delta}{2k} \cdot \|a\|_1 - \frac{\delta}{2n} \cdot (1 - \|a\|_1) \geq 1 - \delta$, so the returned count of the algorithm is also correct with probability $\geq 1 - \delta$.

Expected number of queries. We first consider the number of a_i with $a_i = 1$ that we pass to ASYMMETRIC-CHECK-BIT. One trivial upper bound is $\|a\|_1$. Also, the expected number of $a_i = 1$ we need to pass to ASYMMETRIC-CHECK-BIT before cnt is incremented by 1 is $\leq \frac{1}{1 - \delta/2k}$, so another upper bound is $\frac{k}{1 - \delta/2k} \leq (1 + o(1))k$ (as $\delta = o(1)$). On the other hand, the number of a_i where $a_i = 0$ that we pass to ASYMMETRIC-CHECK-BIT is upper bounded by $n - \|a\|_1$. Note that whether we pass some a_i to ASYMMETRIC-CHECK-BIT only depends on the queries we make to $a_{i'}$ for $i' < i$, so even given that we pass some a_i to ASYMMETRIC-CHECK-BIT, we can still use the bounds from Lemma 43 to bound the expected number of queries we make to a_i . Therefore, the

expected number of queries can be upper bounded by

$$(1 + o(1)) \left(\min \{ \|a\|_1, k \} \cdot \frac{\log \frac{n}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} + (n - \|a\|_1) \cdot \frac{\log \frac{k}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} \right).$$

Then we consider two cases depending on how large k is.

Case $k \geq n / \log n$. In this case, $\log(n) = (1 + o(1)) \log k$, so the expected number of queries can be upper bounded by

$$\begin{aligned} & (1 + o(1)) \left(\min \{ \|a\|_1, k \} \cdot \frac{\log \frac{k}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} + (n - \|a\|_1) \cdot \frac{\log \frac{k}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} \right) \\ &= (1 + o(1)) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}. \end{aligned}$$

Case $k \leq n / \log n$. In this case, $k \log \frac{n}{\delta} = o(n \log \frac{k}{\delta})$, so the expected number of queries can be upper bounded by

$$\begin{aligned} & (1 + o(1)) \left(k \cdot \frac{\log \frac{n}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} + n \cdot \frac{\log \frac{k}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} \right) \\ &= (1 + o(1)) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}. \end{aligned}$$

D.3. Bounds for Counting

Let us first prove a one-sided upper bound for Counting.

Theorem 44 *Given a sequence $a \in \{0, 1\}^n$, there is an algorithm that can output $\|a\|_1$ with error probability $\delta = o(1)$ using*

$$(1 + o(1)) \frac{n \log \frac{\|a\|_1 + 1}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}$$

noisy queries in expectation.

The difference between [Theorem 44](#) and the upper bound part of [Theorem 4](#) is that we have $\|a\|_1 + 1$ rather than $\min \{ \|a\|_1 + 1, n - \|a\|_1 + 1 \}$ inside the log term.

Proof

Outlined in [Algorithm 2](#), the algorithm for counting is an adaptation of [Algorithm 1](#). Notice that in [Algorithm 1](#), when calling ASYMMETRIC-CHECK-BIT, we have $\delta_0 = \delta/2n$ and $\delta_1 = \delta/2k$. The value δ_0 is fixed regardless of the value of k , and the value δ_1 depends on k . The algorithm for counting in some sense is simulating [Algorithm 1](#), but dynamically adjusting δ_1 based on the current estimate of k , which is the number of input bits that are believed to be 1's.

For simplicity, throughout the analysis, we use $k^* = \|a\|_1$ to denote the desired answer. Let $S_0 \subseteq [n]$ be indices i with $a_i = 0$ and let $S_1 = [n] \setminus S_0$.

Algorithm 2

```

1: procedure COUNTING( $\{a_1, \dots, a_n\}, \delta$ )
2:    $k \leftarrow 0$ 
3:    $c \leftarrow \{0\}^n$ 
4:    $Active \leftarrow [n]$ 
5:   while True do
6:      $i^* \leftarrow \arg \max_{i \in Active} c_i$ 
7:     if  $c_{i^*} \leq -\frac{\log(6(k+1)/\delta)}{\log((1-p)/p)}$  then return  $k$ 
8:     if QUERY( $a_{i^*}$ ) = 1 then
9:        $c_{i^*} = c_{i^*} + 1$ 
10:    if  $c_{i^*} \geq \frac{\log(6n/\delta)}{\log((1-p)/p)}$  then
11:       $Active \leftarrow Active \setminus \{i^*\}$ 
12:       $k \leftarrow k + 1$ 
13:    else
14:       $c_{i^*} = c_{i^*} - 1$ 

```

Error probability. For any i , we can view the value of c_i as a random walk, i.e., after every query to a_i , we either adds 1 to c_i or subtracts 1 from c_i . If $c_i = 1$, then the probability that we add 1 to c_i after every step is $1 - p$, and the probability that we subtract 1 is p . If $c_i = 0$, then the probability that we add 1 to c_i after every step is p , and the probability that we subtract 1 is $1 - p$. Conceptually, we can view c_i as an infinite random walk, and the algorithm only utilizes some prefix of it.

For $i \in S_1$, by Lemma 41, the probability that c_i ever reaches $-\left\lceil \frac{\log(6/\delta)}{\log((1-p)/p)} \right\rceil$ is at most

$$\left(\frac{p}{1-p} \right)^{\frac{\log(6/\delta)}{\log((1-p)/p)}} = \delta/6.$$

Furthermore, if some c_i never reaches $-\left\lceil \frac{\log(6/\delta)}{\log((1-p)/p)} \right\rceil$, then i will eventually be removed from $Active$ and contributes towards k . Since the random walks c_i 's are independent for different i 's, the probability that there are $\lceil k^*/2 \rceil + 1$ many $i \in S_1$ where c_i reaches $-\left\lceil \frac{\log(6/\delta)}{\log((1-p)/p)} \right\rceil$ can be upper bounded by

$$\binom{k^*}{\lceil k^*/2 \rceil + 1} \cdot (\delta/6)^{\lceil k^*/2 \rceil + 1} \leq 2^{k^*} (\delta/6)^{k^*/2 + 1} \leq (1.5\delta)^{k^*/2} \cdot (\delta/6) \leq \delta/6,$$

where the last step holds because $\delta = o(1)$ and hence we can assume $1.5\delta \leq 1$. Thus, up to $\delta/6$ error probability, the number of $i \in S_1$ that contributes towards k is at least $k^* - (\lceil k^*/2 \rceil + 1)$, which means the final value of k is $\geq \max\{0, k^* - (\lceil k^*/2 \rceil + 1)\} = \max\{0, \lfloor k^*/2 \rfloor - 1\}$. This further implies that $k + 1 \geq k^*/3$. Let E_1 be the event $k + 1 \geq k^*/3$ and as analyzed above, $\mathbb{P}(\neg E_1) \leq \delta/6$.

Next, we consider the event E_2 where all c_i for $i \in S_1$ never reaches $-\left\lceil \frac{\log(6(k^*/3)/\delta)}{\log((1-p)/p)} \right\rceil$. By Lemma 41, the probability that each c_i for $i \in S_1$ reaches $-\left\lceil \frac{\log(6(k^*/3)/\delta)}{\log((1-p)/p)} \right\rceil$ is upper bounded by

$$\left(\frac{p}{1-p} \right)^{\frac{\log(6(k^*/3)/\delta)}{\log((1-p)/p)}} = \frac{\delta}{2k^*}.$$

Therefore, by union bound, $\mathbb{P}(\neg E_2) \leq \delta/2$.

Next, let E_3 be the event where all c_i for $i \in S_0$ never reaches $\left\lceil \frac{\log(6n/\delta)}{\log((1-p)/p)} \right\rceil$. By Lemma 41, the probability that each c_i for $i \in S_0$ reaches $\left\lceil \frac{\log(6n/\delta)}{\log((1-p)/p)} \right\rceil$ is upper bounded by

$$\left(\frac{p}{1-p} \right)^{\frac{\log(6n/\delta)}{\log((1-p)/p)}} = \frac{\delta}{6n}.$$

Therefore, by union bound, $\mathbb{P}(\neg E_3) \leq \delta/6$.

Assume E_1, E_2, E_3 all happen. Then we know that $k+1 \geq k^*/3$, and so all $i \in S_1$ will eventually be removed from *Active* and contribute towards k . Also, all $i \in S_0$ will not contribute towards k . Therefore, the returned value of k will be equal to k^* . Hence, the error probability of the algorithm is upper bounded by $\mathbb{P}(\neg E_1 \vee \neg E_2 \vee \neg E_3) \leq \delta/6 + \delta/2 + \delta/6 \leq \delta$.

Expected number of queries. Next, we analyze the expected number of queries used by the algorithm.

First, for every $i \in S_1$, the expected number of times we query a_i is upper bounded by the expected number of steps it takes for the random walk c_i takes to reach $\left\lceil \frac{\log(6n/\delta)}{\log((1-p)/p)} \right\rceil$. By Lemma 42, it is bounded by

$$\frac{\left\lceil \frac{\log(6n/\delta)}{\log((1-p)/p)} \right\rceil}{1-2p} = (1+o(1)) \frac{\log \frac{n}{\delta}}{D_{\text{KL}}(p \parallel 1-p)}.$$

Then we consider the expected number of times we query a_i for $i \in S_0$. Fix any $i \in S_0$. Let F_j be the event where for exactly j distinct $i' \in S_0 \setminus \{i\}$, the infinite random walk c_j ever reaches $\left\lceil \frac{\log(6n/\delta)}{\log((1-p)/p)} \right\rceil$. By analysis in the error probability part, $\mathbb{P}(F_j) \leq \binom{n}{j} \cdot (\delta/6n)^j \leq \delta^j$. Note that F_j is independent with the random walk c_i . If F_j holds, then the number of times we query a_i is upper bounded by the expected number of steps it takes for the random walk c_i first hits $\left\lceil \frac{\log(6(k^*+j+1)/\delta)}{\log((1-p)/p)} \right\rceil$ (because once c_i hits this value, k can never be larger than $k^* + j$ under F_j , so we will not query a_i again). By Lemma 42, this expectation is

$$\frac{\left\lceil \frac{\log(6(k^*+j+1)/\delta)}{\log((1-p)/p)} \right\rceil}{1-2p} = (1+o(1)) \frac{\log \frac{k^*+j+1}{\delta}}{D_{\text{KL}}(p \parallel 1-p)}.$$

Let Q_i be the number of times we query a_i . Then we have

$$\begin{aligned}
\mathbb{E}(Q_i) &= \sum_{j \geq 0} \mathbb{E}(Q_i | F_j) \cdot \mathbb{P}(F_j) \\
&\leq \frac{1 + o(1)}{D_{\text{KL}}(p \parallel 1 - p)} \sum_{j \geq 0} \delta^j \cdot \log \left(\frac{k^* + j + 1}{\delta} \right) \\
&\leq \frac{1 + o(1)}{D_{\text{KL}}(p \parallel 1 - p)} \sum_{j \geq 0} \delta^j \cdot \log \left(\frac{(k^* + 1)(j + 1)}{\delta} \right) \\
&= \frac{1 + o(1)}{D_{\text{KL}}(p \parallel 1 - p)} \left(\sum_{j \geq 0} \delta^j \cdot \log \left(\frac{k^* + 1}{\delta} \right) + \sum_{j \geq 0} \delta^j \cdot \log(j + 1) \right) \\
&\leq \frac{1 + o(1)}{D_{\text{KL}}(p \parallel 1 - p)} \left(\frac{1}{1 - \delta} \cdot \log \left(\frac{k^* + 1}{\delta} \right) + O(1) \right) \\
&\leq (1 + o(1)) \frac{\log \frac{k^* + 1}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}.
\end{aligned}$$

Summing up everything, the overall expected number of queries is

$$(1 + o(1)) \left(k^* \cdot \frac{\log \frac{n}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} + (n - k^*) \cdot \frac{\log \frac{k^* + 1}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} \right).$$

Similar to the proof of [Theorem 39](#), we consider two cases depending on how large k^* is.

Case $k^* \geq n/\log n$. In this case, $\log(n) = (1 + o(1)) \log(k^* + 1)$, so the expected number of queries can be bounded by

$$\begin{aligned}
&(1 + o(1)) \left(k^* \cdot \frac{\log \frac{k^* + 1}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} + (n - k^*) \cdot \frac{\log \frac{k^* + 1}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} \right) \\
&= (1 + o(1)) \frac{n \log \frac{k^* + 1}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}.
\end{aligned}$$

Case $k^* \leq n/\log n$. In this case, $k^* \log \frac{n}{\delta} = o(n \log \frac{k^* + 1}{\delta})$, so the expected number of queries can be bounded by

$$\begin{aligned}
&(1 + o(1)) \left(k^* \cdot \frac{\log \frac{n}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} + n \cdot \frac{\log \frac{k^* + 1}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)} \right) \\
&= (1 + o(1)) \frac{n \log \frac{k^* + 1}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}.
\end{aligned}$$

■

Finally, we prove [Theorem 4](#), which we recall below:

Theorem 4 (Noisy query complexity of Counting) *Given a sequence $a \in \{0, 1\}^n$, computing $\|a\|_1$ with error probability $\delta = o(1)$ needs and only needs*

$$(1 \pm o(1)) \frac{n \log \frac{\min\{\|a\|_1, n - \|a\|_1\} + 1}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}$$

noisy queries in expectation.

Proof

Suppose there is an algorithm for Counting with

$$(1 - \epsilon) \frac{n \log \frac{\min\{\|a\|_1, n - \|a\|_1\} + 1}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)}$$

noisy queries in expectation, for some absolute constant $c > 0$. First, suppose $\|a\|_1 \leq n/2$. Recall that in the lower bound for k -Threshold, the hard distribution for k -Threshold is to distinguish whether the input contains k or $k - 1$ 1's. Thus, by running the assumed algorithm for Counting, the running time would be

$$(1 - \epsilon) \frac{n \log \frac{k+1}{\delta}}{D_{\text{KL}}(p \parallel 1 - p)},$$

which contradicts the lower bound for k -Threshold.

If $\|a\|_1 > n/2$, we can simply first flip all the bits in the k -Threshold instance, and then use the same argument.

For the upper bound, given [Theorem 44](#), it suffices to first estimate whether $\|a\|_1$ is bigger than $n/2$ or smaller. When it is smaller, we can directly run [Theorem 44](#); otherwise, we first flip all input bits and then run [Theorem 44](#).

More precisely, we randomly sample $n^{0.99}$ input elements with replacements, use [Lemma 40](#) to estimate them with error probability $1/n^{100}$. If the fraction of elements in the sample whose estimates are 1 is $\leq \frac{1}{2}$, we directly call [Theorem 44](#) with error bound δ and return the result; otherwise, we flip all input bits, call [Theorem 44](#) with error bound δ , and return n minus the result.

Regardless of whether we flip the input bits, the output is always correct assuming the returned result of [Theorem 44](#) is correct. Therefore, the error probability of the algorithm is at most δ .

Next, we analyze the expected running time of the algorithm. Let E_1 be the event that the fraction of sampled elements that are 1 is within $n^{-0.01}$ of the fraction of all elements that are 1. By Chernoff bound,

$$\mathbb{P}(\neg E_1) \leq 2e^{-2(n^{-0.01})^2 \cdot n^{0.99}} \leq O\left(\frac{1}{n^{99}}\right).$$

Let E_2 be the event that the returned result of [Lemma 40](#) for all sampled elements are correct. By union bound,

$$\mathbb{P}(\neg E_2) \leq \frac{1}{n^{99}}.$$

If E_1 and E_2 both hold, the expected number of queries of [Theorem 44](#) can be bounded as

$$(1 + o(1)) \cdot \frac{n}{D_{\text{KL}}(p \parallel 1 - p)} \cdot \begin{cases} \log \frac{\|a\|_1 + 1}{\delta}, & \text{if } \|a\|_1 \leq n/2 - n^{0.99}, \\ \log \frac{n+1}{\delta}, & \text{if } n/2 - n^{0.99} < \|a\|_1 \leq n/2 + n^{0.99}, \\ \log \frac{n - \|a\|_1 + 1}{\delta}, & \text{if } \|a\|_1 > n/2 + n^{0.99}. \end{cases}$$

Regardless of which case it is, the expected number of queries is always $(1+o(1)) \frac{n \log \frac{\min\{\|a\|_1, n-\|a\|_1\}+1}{\delta}}{D_{\text{KL}}(p \| 1-p)}$. Even if E_1 and E_2 do not both hold, the expected number of queries of [Theorem 44](#) can be bounded as

$$(1+o(1)) \cdot \frac{n \log \frac{n+1}{\delta}}{D_{\text{KL}}(p \| 1-p)}.$$

Therefore, the overall expected number of queries can be bounded as

$$\begin{aligned} & n^{0.99} \cdot (1+o(1)) \cdot \frac{\log \frac{1}{n^{100}}}{D_{\text{KL}}(p \| 1-p)} + (1+o(1)) \frac{n \log \frac{\min\{\|a\|_1, n-\|a\|_1\}+1}{\delta}}{D_{\text{KL}}(p \| 1-p)} \\ & + \mathbb{P}(\neg E_1 \vee \neg E_2)(1+o(1)) \frac{n \log \frac{n+1}{\delta}}{D_{\text{KL}}(p \| 1-p)} \\ & = (1+o(1)) \frac{n \log \frac{\min\{\|a\|_1, n-\|a\|_1\}+1}{\delta}}{D_{\text{KL}}(p \| 1-p)} \end{aligned}$$

as desired. ■

Appendix E. A simpler proof of hardness of Threshold for small k

In this section we provide a simpler proof of [Theorem 36](#). Let us recall the theorem statement.

Theorem 36 *For $k = o(n)$, solving TH_k^n with $\delta = o(1)$ error probability requires*

$$(1-o(1)) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$$

noisy queries in expectation, even when the input is uniformly chosen from $\binom{[n]}{k}$ with probability $1/2$ and uniformly chosen from $\binom{[n]}{k-1}$ with probability $1/2$.

Our proof uses the three-phase framework which we also used in the proof of [Theorem 2](#). It is a refinement of ([Feige et al., 1994](#))’s two-phase framework, which were used to prove that computing TH_k^n requires $\Omega(n \log \frac{k}{\delta})$ noisy queries. Here we add a phase where the oracle can send extra information to the algorithm, allowing for a more precise analysis obtaining the exact constant.

Comparison with ([Wang et al., 2025](#))’s proof. Let us briefly compare our proof with ([Wang et al., 2025](#))’s proof. Both proofs are based on ([Feige et al., 1994](#))’s two-phase framework. ([Wang et al., 2025](#))’s proof is divided into three cases: $\log(1/\delta) \log \log(1/\delta) < \log k \leq \frac{\log(1/\delta)}{\log \log(1/\delta)}$, $\log k > \log(1/\delta) \log \log(1/\delta)$, $\log k \leq \frac{\log(1/\delta)}{\log \log(1/\delta)}$. The first two cases are handled using the two-phase framework, and the last case is proved using Le Cam’s two point method. In comparison, our proof is simpler and handles all $k = o(n)$ and $\delta = o(1)$ uniformly. We achieve this simplification by carefully designing the information that the oracle reveals to the algorithm for free.

E.1. Three-phase problem

Let us now describe the three-phase problem. Let $\epsilon_1, \epsilon_2 > 0$ be two absolute constants (i.e., they do not grow with n).

1. In Phase 1, the algorithm makes $m_1 = (1 - \epsilon_1) \frac{\log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$ queries to every element.
2. In Phase 2, the oracle reveals some elements to the algorithm.
3. In Phase 3, the algorithm makes $m_2 = (1 - \epsilon_2)n$ adaptive exact queries.

The goal of the algorithm is to distinguish whether there are at least $k + 1$ ones among the elements.

Lemma 45 *If no algorithm can solve the three-phase problem with error probability $\delta > 0$, then no algorithm can solve TH_k^n with error probability δ using at most $(1 - \epsilon_1)(1 - \epsilon_2) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$ noisy queries.*

The proof is similar to [Lemma 6](#) and omitted.

By [Lemma 45](#), to prove [Theorem 36](#), it suffices to prove hardness of the three-phase problem.

Proposition 46 *For any absolute constants $\epsilon_1, \epsilon_2 > 0$, no algorithm can solve the three-phase problem for TH_k^n with error probability δ for $k = o(n)$ and $\delta = o(1)$, where the input is uniformly chosen from $\binom{[n]}{k}$ with probability $1/2$ and uniformly chosen from $\binom{[n]}{k-1}$ with probability $1/2$.*

[Theorem 36](#) follows by combining [Lemma 45](#) and [Proposition 46](#).

The rest of the section is devoted to the proof of [Proposition 46](#).

E.2. Phase 1

Define $A = \{i \in [n] : a_i = 1\}$ where a is the input bit string. Then $A \in \binom{[n]}{k} \cup \binom{[n]}{k-1}$, and the goal is to distinguish whether $|A| = k$ or $k - 1$.

In Phase 1, the algorithm makes $m_1 = (1 - \epsilon_1) \frac{\log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$ queries to every element $i \in [n]$. Let a_i denote the number of times where a query to i returns 1. Then for $i \in A$, $a_i \sim \text{Bin}(m_1, 1 - p)$; for $i \notin A$, $a_i \sim \text{Bin}(m_1, p)$. For $0 \leq j \leq m_1$, define

$$p_j = \mathbb{P}(\text{Bin}(m_1, 1 - p) = j) = \binom{m_1}{j} (1 - p)^j p^{m_1 - j}.$$

Let $I = [pm_1 - m_1^{0.6}, pm_1 + m_1^{0.6}]$.

Lemma 47 *Let $x \sim \text{Bin}(m_1, 1 - p)$, $y \sim \text{Bin}(m_1, p)$. Then*

$$\mathbb{P}(x \in I) = (\delta/k)^{1 - \epsilon_1 \pm o(1)}, \tag{25}$$

$$\mathbb{P}(y \in I) = 1 - o(1). \tag{26}$$

Proof [Eq. \(25\)](#) is by [Lemma 5](#). [Eq. \(26\)](#) is by Chernoff bound. ■

Let $\mathbb{P}^{(0)}$ denote the prior distribution of A and $\mathbb{P}^{(1)}$ denote the posterior distribution of A conditioned on observations in Phase 1. Let $\mathcal{C}^{(0)}$ (resp. $\mathcal{C}^{(1)}$) denote the support of $\mathbb{P}^{(0)}$ (resp. $\mathbb{P}^{(1)}$). Then $\mathcal{C}^{(1)} = \mathcal{C}^{(0)} = \binom{[n]}{k-1} \cup \binom{[n]}{k}$ and for any set $B \in \mathcal{C}^{(0)}$ we have

$$\begin{aligned} \mathbb{P}^{(1)}(B) &\propto \mathbb{P}((a_i)_{i \in [n]} | B) \mathbb{P}^{(0)}(B) \\ &= \left(\prod_{i \in B} p_{a_i} \right) \left(\prod_{i \in B^c} p_{m_1 - a_i} \right) \mathbb{P}^{(0)}(B). \end{aligned}$$

E.3. Phase 2

In Phase 2, the oracle reveals some elements in A and not in A as follows.

- 2a. In Step 2a, the oracle reveals elements i with $a_i \notin I$.
- 2b. In Step 2b, the oracle reveals every $i \in A$ independently with probability q_{a_i} . We choose $q_j = 1 - \frac{p_{m_1-j} p_{j_l}}{p_j p_{m_1-j_l}}$ for $j \in I$ where $j_l = pm_1 - m_1^{0.6}$.
- 2c. In Step 2c, the oracle reveals $k-1$ elements of A as follows. If $|A| = k-1$, reveal all elements of A . Otherwise, $|A| = k$. If A contains an element that is not revealed yet, uniformly randomly choose an element i^* from all such elements and reveal all elements in $A \setminus i^*$. If all elements of A have been revealed, report failure.

Step 2a and Step 2b. By the same analysis as in [Sections B.3.1](#) and [B.3.2](#), observations up to Step 2b have the same effect as the following procedure:

- (1) Observe every element $i \in A$ independently with probability

$$\begin{aligned} p_+ &= 1 - \sum_{j \in I} p_j (1 - q_j) = 1 - \frac{p_{j_l}}{p_{m_1-j_l}} \cdot \sum_{j \in I} p_{m_1-j} \\ &= 1 - (1 \pm o(1)) \frac{p_{j_l}}{p_{m_1-j_l}} = 1 - (\delta/k)^{1-\epsilon_1 \pm o(1)}. \end{aligned}$$

- (2) Observe every element $i \in A^c$ independently with probability

$$p_- = \mathbb{P}(\text{Bin}(m_1, p) \notin I) = o(1).$$

Let $A_+^{(2a)}$ (resp. $A_-^{(2a)}$) denote the set of elements in I (resp. not in I) revealed in Step 2a. Let $A_+^{(2b)}$ be the set of elements in I revealed in Step 2b, and $A_+^{(\leq 2b)} = A_+^{(2a)} \cup A_+^{(2b)}$. Then the oracle reports failure in Step 2c if and only if $|A_+^{(\leq 2b)}| = k$.

Let $\mathbb{P}^{(2b)}$ denote the posterior distribution of A after Step 2b and $\mathcal{C}^{(2b)}$ be its support. By the same analysis as in [Sections B.3.1](#) and [B.3.2](#), the posterior probability after Step 2b satisfies

$$\mathbb{P}^{(2b)}(B) \propto \left(\frac{p_{j_l}}{p_{m_1-j_l}} \right)^{|B|-(k-1)} \mathbb{P}^{(0)}(B)$$

for $B \in \mathcal{C}^{(2b)}$. In particular, $(A_+^{(\leq 2b)}, A_-^{(2a)}, |A|)$ is a sufficient statistic for A at the end of Step 2b.

We note that $|A_+^{(\leq 2b)}| < k$ with probability $\geq \delta^{1-\epsilon_1 \pm o(1)}$ because

$$\mathbb{P}(\text{Bin}(k, 1 - p_+) > 0) = 1 - p_+^k \geq 1 - \exp(k(1 - p_+)) \geq \delta^{1-\epsilon_1 \pm o(1)}. \quad (27)$$

Step 2c. Let $A_+^{(2c)}$ be the set of elements in I revealed in Step 2c but not in previous steps, and $A_+^{(\leq 2c)} = A_+^{(\leq 2b)} \cup A_+^{(2c)}$. Let $\mathbb{P}^{(2c)}$ be the posterior distribution of A and $\mathcal{C}^{(2c)}$ be the support of $\mathbb{P}^{(2c)}$. Then

$$\begin{aligned}\mathcal{C}^{(2c)} &= \left\{ A : A \in \binom{[n]}{k-1} \cup \binom{[n]}{k}, A_+^{(\leq 2c)} \subseteq A \subseteq [n] \setminus A_-^{(2a)} \right\} \\ &= \left\{ A_+^{(\leq 2c)} \right\} \cup \left\{ A_+^{(\leq 2c)} \cup \{i\} : i \in [n] \setminus \left(A_+^{(\leq 2c)} \cup A_-^{(2a)} \right) \right\}.\end{aligned}$$

For simplicity of notation, define $A_0 = A_+^{(\leq 2c)}$ and $A_i = A_+^{(\leq 2c)} \cup \{i\}$ for $i \in [n] \setminus \left(A_+^{(\leq 2c)} \cup A_-^{(2a)} \right)$. Then

$$\begin{aligned}\mathbb{P}^{(2c)}(A_0) &\propto \mathbb{P}^{(0)}(A_0), \\ \mathbb{P}^{(2c)}(A_i) &\propto \frac{1}{k - |A_+^{(\leq 2b)}|} \cdot \frac{p_{j_l}}{p_{m_1-j_l}} \cdot \mathbb{P}^{(0)}(A_i).\end{aligned}$$

Recall that $\mathbb{P}^{(0)}(B) = \frac{1}{2} \cdot \frac{1}{\binom{n}{|B|}}$ for $|B| \in \{k-1, k\}$. Let $\mathbb{P}_k^{(2c)}$ (resp. $\mathbb{P}_{k-1}^{(2c)}$) denote the probability measure of observations at the end of Step 2c conditioned on $|A| = k$ (resp. $|A| = k-1$). Summing over i , we have

$$\frac{d\mathbb{P}_k^{(2c)}}{d\mathbb{P}_{k-1}^{(2c)}} = \left| [n] \setminus \left(A_+^{(\leq 2c)} \cup A_-^{(2a)} \right) \right| \cdot \frac{1}{k - |A_+^{(\leq 2b)}|} \cdot \frac{p_{j_l}}{p_{m_1-j_l}} \cdot \frac{\binom{n}{k-1}}{\binom{n}{k}}.$$

Lemma 48

$$\mathbb{P}_k^{(2c)} \left(\frac{d\mathbb{P}_k^{(2c)}}{d\mathbb{P}_{k-1}^{(2c)}} \leq \delta^{-\epsilon_1/2} \right) \geq \delta^{1-\epsilon_1 \pm o(1)}.$$

Proof Because $p_- = o(1)$, with probability $1 - o(1)$, we have $|A_-^{(2a)}| \leq np_-^{1/2}$. Then

$$\left| [n] \setminus \left(A_+^{(\leq 2c)} \cup A_-^{(2a)} \right) \right| = n - (k-1) - |A_-^{(2a)}| = (1 \pm o(1))n$$

and

$$\begin{aligned}\frac{d\mathbb{P}_k^{(2c)}}{d\mathbb{P}_{k-1}^{(2c)}} &= (1 \pm o(1))n \cdot \frac{1}{k - |A_+^{(\leq 2b)}|} \cdot \frac{p_{j_l}}{p_{m_1-j_l}} \cdot \frac{k}{n} \\ &= (1 \pm o(1)) \frac{k}{k - |A_+^{(\leq 2b)}|} \cdot \frac{p_{j_l}}{p_{m_1-j_l}}.\end{aligned}$$

Note that $k - |A_+^{(\leq 2b)}| \sim \text{Bin}(k, 1 - p_+)$, and $|A_+^{(\leq 2b)}|$ is independent with $|A_-^{(2a)}|$ conditioned on $|A|$. Recall that

$$1 - p_+ = (1 \pm o(1)) \frac{p_{j_l}}{p_{m_1-j_l}} = (\delta/k)^{1-\epsilon_1 \pm o(1)}.$$

If $k^{\epsilon_1} \delta^{1-\epsilon_1} \geq \delta^{-\epsilon_1/2}$, then $k(1-p_+) = \omega(1)$ and by concentration

$$\mathbb{P} \left(\text{Bin}(k, 1-p_+) \geq \frac{1}{2}k(1-p_+) \right) = 1 - o(1)$$

Under $\mathbb{P}_k^{(2c)}$, conditioned on $k - |A_+^{(\leq 2b)}| \geq \frac{1}{2}k(1-p_+)$, we have

$$\frac{d\mathbb{P}_k^{(2c)}}{d\mathbb{P}_{k-1}^{(2c)}} = (1 \pm o(1)) \frac{k}{k - |A_+^{(\leq 2b)}|} \cdot \frac{p_{jl}}{p_{m_1-j_l}} = O(1).$$

If $k^{\epsilon_1} \delta^{1-\epsilon_1} \leq \delta^{-\epsilon_1/2}$, then conditioned on $|A_+^{(\leq 2b)}| < k$ (which happens with probability at least $\delta^{1-\epsilon_1 \pm o(1)}$ by Eq. (27)), we have

$$\frac{d\mathbb{P}_k^{(2c)}}{d\mathbb{P}_{k-1}^{(2c)}} = (1 \pm o(1)) \frac{k}{k - |A_+^{(\leq 2b)}|} \cdot \frac{p_{jl}}{p_{m_1-j_l}} \leq \delta^{-\epsilon_1/2 \pm o(1)}.$$

Combining both cases we finish the proof. ■

From Lemma 48 we can conclude that no algorithm can determine $|A|$ with error probability $\leq \delta$ at the end of Phase 2. Suppose for the sake of contradiction that such an algorithm exists. Let \mathcal{E} denote the event that $\frac{d\mathbb{P}_k^{(2c)}}{d\mathbb{P}_{k-1}^{(2c)}} \leq \delta^{-\epsilon_1/2}$. Under $\mathbb{P}_k^{(2c)}$, conditioned on \mathcal{E} , the algorithm outputs “ $|A| = k$ ” with probability at least $\frac{1}{2}$ (otherwise the overall error probability under $\mathbb{P}_k^{(2c)}$ is at least $\frac{1}{2} \cdot \delta^{1-\epsilon \pm o(1)}$). However, this implies that under $\mathbb{P}_{k-1}^{(2c)}$, the algorithm outputs $|A| = k$ with probability at least $\frac{1}{2} \delta^{\epsilon_1/2}$ by definition of \mathcal{E} .

E.4. Phase 3

In Phase 3, the algorithm makes at most $(1-\epsilon_2)n$ adaptive exact queries. If $\left| [n] \setminus \left(A_+^{(\leq 2c)} \cup A_-^{(2a)} \right) \right| = (1 \pm o(1))n$, then with probability $(1 \pm o(1))\epsilon_2$, these exact queries do not hit any elements in A . Let $\mathbb{P}_k^{(3)}$ (resp. $\mathbb{P}_{k-1}^{(3)}$) denote the probability measure of observations at the end of Phase 3 conditioned on $|A| = k$ (resp. $|A| = k-1$). Conditioned on that the exact queries do not hit any elements in A , we have

$$\frac{d\mathbb{P}_k^{(2c)}}{d\mathbb{P}_{k-1}^{(2c)}} = \left| [n] \setminus \left(A_+^{(\leq 2c)} \cup A_-^{(2a)} \cup A^{(3)} \right) \right| \cdot \frac{1}{k - |A_+^{(\leq 2b)}|} \cdot \frac{p_{jl}}{p_{m_1-j_l}} \cdot \frac{\binom{n}{k-1}}{\binom{n}{k}}$$

where $A^{(3)}$ denotes the set of elements queried in Phase 3. By a similar proof as Lemma 48, we can prove that

$$\mathbb{P}_k^{(3)} \left(\frac{d\mathbb{P}_k^{(3)}}{d\mathbb{P}_{k-1}^{(3)}} \leq \epsilon_2^{-1} \delta^{-\epsilon_1/2} \right) \geq \epsilon_2 \delta^{1-\epsilon_1 \pm o(1)}.$$

However, ϵ_2 is a constant, so the discussion in the end of Section E.3 still applies. This concludes that no algorithm can solve the three-phase problem with error probability $\leq \delta$ using at most $(1 - \epsilon) \frac{n \log \frac{k}{\delta}}{D_{\text{KL}}(p \| 1-p)}$ noisy queries.