Optimal Time-Backlog Tradeoffs for the Variable-Processor Cup Game

William Kuszmaul*, Shyam Narayanan†

July 26, 2022

Abstract

The *p-processor cup game* is a classic and widely studied scheduling problem that captures the setting in which a p-processor machine must assign tasks to processors over time in order to ensure that no individual task ever falls too far behind. The problem is formalized as a multi-round game in which two players, a filler (who assigns work to tasks) and an emptier (who schedules tasks) compete. The emptier's goal is to minimize backlog, which is the maximum amount of outstanding work for any task.

Recently, Kuszmaul and Westover (ITCS, 2021) proposed the *variable-processor cup game*, which considers the same problem, except that the amount of resources available to the players (i.e., the number p of processors) fluctuates between rounds of the game. They showed that this seemingly small modification fundamentally changes the dynamics of the game: whereas the optimal backlog in the fixed p-processor game is $\Theta(\log n)$, independent of p, the optimal backlog in the variable-processor game is $\Theta(n)$. The latter result was only known to apply to games with *exponentially many* rounds, however, and it has remained an open question what the optimal tradeoff between time and backlog is for shorter games.

This paper establishes a tight trade-off curve between time and backlog in the variable-processor cup game. We show that, for a game consisting of t rounds, the optimal backlog is $\Theta(b(t))$ where

$$b(t) = \begin{cases} t & \text{if } t \leq \log n \\ t^{1/3} \log^{2/3}\left(\frac{n^3}{t} + 1\right) & \text{if } \log n < t \leq n^3 \\ n & \text{if } n^3 < t. \end{cases}$$

An important consequence is that the optimal backlog is $\Theta(n)$ if and only if $t \geq \Omega(n^3)$. Our techniques also allow for us to resolve several other open questions concerning how the variable-processor cup game behaves in beyond-worst-case-analysis settings.

1 Introduction

The classical p-processor cup game. The p processor cup game captures the general problem in which there are some number n of tasks competing for a smaller number p of processors [7, 21, 8, 34, 32, 38, 6, 24, 35, 36, 17, 10, 27, 1, 16, 33, 25]. A scheduler must assign tasks to processors over time in order to ensure that no individual task ever falls too far behind.

---

*Massachusetts Institute of Technology. Email: `kuszmaul@mit.edu`
†Massachusetts Institute of Technology. Email: `shyamsn@mit.edu`.

Formally, this is captured as a game with n cups, each capable of holding an arbitrarily large amount of water, and two competing players, a filler and an emptier. In each round of the game, the filler distributes p new units of water into the cups, placing at most 1 unit of water into any particular cup. The emptier then selects p distinct cups and removes up to 1 unit of water from each of them. Note that, whereas the filler may place their p units of water in fractional amounts across arbitrarily many cups, the emptier can only choose p cups per step to empty from. The emptier's goal is to minimize the backlog of the system, which is the amount of water in the fullest cup.

If one views the cup game as a scheduling problem, then the cups represent tasks, the water represents work, the filler represents an adversary that determines when work arrives, and the emptier represents a scheduler that can select p tasks to run on a given time step (we will use the terms "round" and "time step" interchangeably). Although we will primarily be interested in the cup game as a scheduling problem [7, 21, 8, 34, 32, 38, 6, 24, 35, 36, 1, 33, 17], it has also found applications to many other problems (e.g. deamortization of data structures [2, 17, 16, 3, 39, 23, 18, 26, 9], network-switch buffer management [22, 4, 40, 20], quality-of-service guarantees [7, 1, 33], etc.).

Beginning in the late 1960s, much of the early work on the p-processor cup game focused on the fixed-rate version of the game, in which the filler's behavior is the same at every round [7, 21, 8, 34, 32, 38, 6, 24, 35, 36]. In this version of the game, it is possible for the emptier to achieve a backlog of $O(1)$, both in the single-processor cup game (i.e., $p = 1$) [35, 36] and in the multi-processor cup game (i.e., $p > 1$) [7]. In recent decades, much of the research has shifted to focus on the non-fixed-rate version of the game, in which the filler is an adaptive adversary that can change their behavior from step to step [1, 16, 27, 33, 5, 19, 15]. In this setting, it is possible for the emptier to achieve backlog $O(\log n)$ [1, 16, 27], and this is known to be asymptotically optimal for all $p \leq n - \sqrt{n}$ [27]. There is also a long history of researchers applying techniques from beyond-worst-case analysis to cup games, e.g., resource augmentation [10, 27, 33, 17], smoothed analysis [27, 10], adversary restrictions [10, 27, 14, 28, 17], semi-clairvoyance [33], etc.

A repeating theme in these directions of work has been the relative difficulty of analyzing the multi-processor case in comparison to the single-processor case. As Liu discussed in his seminal 1969 paper [35], and as many later authors have subsequently reiterated [33, 7, 24, 10, 27], the difficulty of the multi-processor case stems from the fact that the emptier must remove water from p *distinct* cups, even if the vast majority of the water is in a smaller number of cups. For both the fixed-rate and the non-fixed-rate games, the optimal backlog in the multi-processor version of the game [7, 27] was proven decades after the corresponding result for the single-processor game was first shown [35, 16].

The variable-processor cup game. Recent work by Kuszmaul and Westover [30] has considered the question of what happens if the parameter p is permitted to change over time, with the filler adaptively determining both what value of p will be used at each round and where the p new units of water are placed. The resulting game, which is known as the variable-processor cup game, captures settings in which the amounts of resources available to the players fluctuate over time.[1]

The problem of what to do when computing resources fluctuate has received increasing attention

---

[1]As discussed in [30], there is no fundamental reason why the amount p of resources should fluctuate in the same way for the filler as it does for the emptier over time. However, by assuming that p is always the same for both players, one ensures that there is a fair playing field: neither player has an advantage over the other in terms of their resources.

in recent years due to the proliferation of shared-computing systems in which multiple users and virtual operating systems simultaneously run on a single physical multi-core machine; the fact that the machine is shared means that the amount of resources (eg., cache, processors, memory bandwidth, etc.) available to each user is constantly changing, depending on the current demands of other users. This phenomenon has led researchers to revisit problems in which computing resources have traditionally been viewed as static [37, 11, 12, 13, 31, 30].

Intuitively, the variable-processor cup game would seem to be relatively similar to its classical p-processor counterpart. Indeed, the backlog in the p-processor cup game is $O(\log n)$ regardless of the value of the parameter p, suggesting that the same should be true if p is permitted to vary. The central result of [30] is that this intuition turns out to be completely wrong: given sufficiently many time steps, the filler can actually force a backlog of $\Omega(n)$ in the variable-processor cup game, and this backlog is asymptotically optimal.

In order to achieve the backlog of $\Omega(n)$, the authors [30] construct a strategy for the filler in which the number of processors p follows a recursive "fractal-like" pattern. The recursive structure requires a relatively large number of time steps to complete—to achieve the full backlog of $\Omega(n)$, the construction requires exponentially many time steps.

The unexpectedly large backlog prompts several questions. The main open question is the problem of determining the optimal trade-off between backlog and time in the variable-processor cup game, and, in particular, what the optimal backlog is in games of polynomial lengths. In this setting, it is not even known whether the filler can achieve polynomial backlog—the best known filler strategy in this case [30] achieves backlog $2^{O(\sqrt{\log n})}$. Understanding the optimal trade-off between time and backlog in polynomial-length games is especially important since, for instance, one may have a bound on the number of rounds in a given scheduling application, which may allow for a much better guarantee on the backlog.

The authors of [30] also raise the question of whether smaller bounds on backlog can be achieved via beyond-worst-case analysis. Based on their results, they propose two directions, in particular, that seem promising. One is to place an additional restriction on the filler that p can only change at a certain rate; this would thwart the recursive structure of their lower bound construction which changes p dramatically between levels of recursion. The other is to consider the use of resource augmentation, meaning that the emptier is allowed to remove slightly more water in each time step than the filler is permitted to place into the cups. This direction seems promising due to the large amount of time required by the filling strategy of [30], since over such a large amount of time, the resource augmentation would potentially offer a large advantage to the emptier.

Our results. The main result of this paper is a tight trade-off curve between time and backlog in the variable-processor cup game. We show that, for a game consisting of t rounds, the optimal backlog is $\Theta(b(t))$ where

$$b(t) = \begin{cases} t & \text{if } t \leq \log n \\ t^{1/3} \log^{2/3}\left(\frac{n^3}{t} + 1\right) & \text{if } \log n < t \leq n^3 \\ n & \text{if } n^3 < t. \end{cases} \quad (1)$$

By optimal, we mean that there exists an emptying strategy such that no filler can achieve backlog greater than $\Omega(b(t))$ after t rounds, and there exists a filling strategy such that no emptier can achieve backlog less than $O(b(t))$ after t rounds. In the case of the emptying strategy, we show that this tradeoff curve is achieved by the greedy emptying algorithm (i.e., always empty from the fullest cups).

Equation (1) comes with several interesting takeaways. The first is that in short games, of length $o(n^3)$, the emptier *can* achieve sub-linear backlog—prior to this result, it remained open whether the emptier could even ensure $o(n)$ backlog for only n rounds. The second is that in games of size $\Omega(n^3)$, the optimal backlog is $\Theta(n)$—this resolves the open question of [30] as to whether linear backlog can be achieved in polynomial time. The third is that the optimal tradeoff between backlog and time has a somewhat unexpected polylogarithmic low-order term, which disappears only when t grows to be $\Omega(n^3)$.

By examining the inverse of the function $b(t)$, another way to think about (1) is that, for any quantity $b \le n$, the amount of time $t(b)$ needed for an optimal filler to force a backlog of $\Omega(b)$ against an optimal emptier is

$$t(b) = \Theta\left(b + \frac{b^3}{\log^2 \frac{n}{b}}\right), \tag{2}$$

and that backlogs $b = \omega(n)$ are not achievable by the filler (the latter fact, of course, is already known due to [30]).

The second contribution of this paper is to analyze the variable-processor cup game under two forms of beyond-worst-case analysis, each of which resolves an open question posed by [30]. We begin by considering the setting in which the rate at which the filler can change p is severely limited: p is permitted to change by most ± 1 per time step, and the filler can only change p every $\mathrm{poly}(n)$ time steps. Remarkably, this has no effect on the optimal backlog, and the filler can still force a backlog of $n/2$ in polynomial time. Next, we consider the setting in which the emptier has $\varepsilon > 0$ resource augmentation, meaning that, in each time step, the emptier is permitted to remove up to $1 + \varepsilon$ (rather than 1) units of water from each of p cups. This has a dramatic effect on the optimal backlog, reducing it to $O(\frac{1}{\varepsilon} \log n)$, which asymptotically matches the optimal backlog of the standard p-processor cup game when $\varepsilon = \Omega(1)$.

Techniques and paper outline. We formally describe the necessary preliminaries in Section 2, and provide a detailed overview of the technical ideas in the paper in Section 3.

In Section 4, we prove a result that ends up being useful in many of the later sections, namely that the greedy emptying algorithm is actually the *exact* optimal online algorithm, and that this holds no matter what the starting state of the cups is. Interestingly, our proof of greedy emptying being the optimal emptying strategy also applies to the fixed p-processor cup game, for which the result was also not previously known: in this setting, the greedy emptying algorithm was previously only known to be *asymptotically* optimal [27], and this was only known for the starting state with all empty fills. We remark that, although the fact that greedy emptying is optimal is certainly intuitive, it actually isn't true for every variant of the cup game; notably, the greedy emptying algorithm isn't even asymptotically optimal for the fixed-rate version of the game [1].

In Section 5, we construct an asymptotically optimal strategy for the filler. The strategy achieves the bound in (1) no matter what strategy the emptier follows, and the strategy also easily generalizes the case where the filler is restricted to change p at a slow rate. This implies that the optimal backlog $b(t)$ is *at least* that in (1). While the filling algorithm works against any emptier, we focus on the filler working against the greedy emptier, which we know is optimal. We start with a warm-up (subsection 5.1) proving that one can establish $\Omega(n)$ backlog in $O(n^3)$ time, and we then generalize our techniques to apply to games of arbitrary lengths.

The remainder of the paper has been deferred to the Appendix. In Section 6, we turn our attention to proving a tight upper bound for the maximum backlog against a greedy emptier, that

4

is, we show that $b(t)$ is *at most* that in (1). We begin by creating a variation of the variable-processor cup game, which we call the **stone game**, in which the filler's behavior is limited in a certain combinatorially natural way. We analyze the maximum backlog of any filling strategy for the stone game by devising two potential functions and comparing their growth rates; this allows us to establish that $\Omega(n^3)$ time steps are needed to achieve backlog $\Omega(n)$ in the stone game. We then tighten the bound on backlog when there are fewer time steps by partitioning the cups into levels and arguing that a constant fraction of the levels interact especially nicely with the potential functions; this yields (1) for the stone game. Finally, we show that, if the emptier behaves greedily, then this stone game encapsulates the main problem, that is, is always advantageous to the filler in the variable-processor cup game to act as though they are in the stone game. Thus we can transfer out bounds on the stone game into bounds on the variable-processor cup game.

Section 6 is, in our opinion, the most technically involved section in our work. Perhaps the most interesting mathematical contribution in this section is to analyze the furthest backlog in the stone game by constructing two different potential functions and comparing their relative growth rates. While individual potential functions have been used to analyze cup games [10], this is the first time that comparing two potential functions has been applied to cup games.

Finally, in Section 7, we give a very simple analysis of the variable-processor cup game in the presence of resource augmentation. Our argument is non-constructive, employing the probabilistic method in order to show that there *exists* an emptying algorithm that achieves backlog $O(\frac{1}{\varepsilon} \log n)$ (interestingly, the same argument also gives a nontrivial bound of $O(\sqrt{t \log n})$ in the resource-augmentation-free setting). Since the greedy emptying algorithm is optimal, it must also achieve the same bound. To the best of our knowledge, this is the first example of the probabilistic method being used to analyze cup games.

2 Preliminaries

We first formally define the **variable-processor cup game**. In this game, there are n cups of real-valued fills $x_1, \ldots, x_n$, all starting at 0, and two *adaptive* players, a filler and an emptier. At each round, the filler chooses an integer $1 \leq p \leq n$, chooses real numbers $a_1, \ldots, a_n$ such that $0 \leq a_i \leq 1$ for all i and $\sum_{i=1}^n a_i = p$, and replaces x_i with $x_i' = x_i + a_i$ for all $1 \leq i \leq n$. The emptier then chooses a set $S \subset [n]$ of size p, and for each $i \in S$, replaces x_i' with $\max(0, x_i' - 1)$ but does not change x_i' for $i \notin S$. A single **round** (which we also call **time step**) consists of both the filler's and emptier's moves. We define the **state** of the cups at a fixed round to be the current sequence $\{x_1, x_2, \ldots, x_n\}$ of the values of cups. Since the filler and emptier are adaptive, we note that two states are equivalent if the sequences are equal up to permutation. We will also say that the **fills** are $x_1, x_2, \ldots, x_n$ (we think of x_i as the fill of the ith cup). Finally, for any state $X = \{x_1, \ldots, x_n\}$, we define the **backlog** of X as the maximum fill, or $\max_{1 \leq i \leq n} x_i$. The goal of the filler is to maximize the backlog after t rounds for some fixed t, whereas the goal of the emptier is to minimize it. We also define the **variable-processor cup game with ε resource augmentation** as the same as the variable-processor cup game, except that the emptier, for each $i \in S$, replaces x_i' with $\max(0, x_i' - (1 + \varepsilon))$.

Next, we define the **negative-fill variable-processor cup game** as the same as the variable-processor cup game, except that the emptier, for each $i \in S$, replaces x_i' with $x_i' - 1$. This may mean that some of the fills in a state become negative, which is allowed. We analogously define round, state, fills, and backlog (note that the backlog is $\max_{1 \leq i \leq n} x_i$, not $\max_{1 \leq i \leq n} |x_i|$).

5

Unless explicitly stated otherwise, the *standard* game refers to the variable-processor cup game, and the *negative-fill* game refers to the negative-fill variable-processor cup game. We will explicitly state whenever we talk about the fixed p-processor cup game (where $p = \sum_{i=1}^{n} a_i$ is fixed for every round), or the variable-processor cup game with $\varepsilon > 0$ resource augmentation.

We conclude this section by briefly commenting on the relationship between the standard game and the negative-fill game. It is easy to see that the optimal backlog in the standard game is at least as large as the optimal backlog in the negative-fill game. What is less obvious, but worth noting, is that the optimal backlogs in the two games are actually asymptotically *equal*. We provide a proof of this in Appendix A. Thus, in general, either version of the game is equally valid (although the only place where we will take advantage of this in any nontrivial way will be Section 6.3).

3 Technical Overview

In this section, we provide a technical overview for both the lower and upper bound for the variable-processor cup game, as well as the upper bound for the variable-processor cup game with resource augmentation.

3.1 Overview of the Lower Bound on Backlog

In Section 5, we provide a lower bound on the backlog that the filler can achieve over t rounds. For now, let us assume that we are playing the negative-fill game and that the emptier is greedy, i.e., at each time step, if the filler fills p units of water, then the emptier empties 1 unit from the p fullest cups. We shall remove these assumptions at the end of the subsection.

Achieving backlog $\Omega(n)$ in n^3 steps. Suppose at some time step, the fills are $x_1 \geq x_2 \geq \cdots \geq x_n$. Consider any subsequence $x_i, \ldots, x_j$ of even length, such that $x_i = x_{i+1} = \cdots = x_j = k$ for some value k. We claim that it is possible for the filler to place water into the cups so that, once the greedy emptier has performed their move, the net change to the fills will simply be that: half of the cups $i, \ldots, j$ now have fills $k + \frac{1}{2}$ and half of the cups $i, \ldots, j$ now have fills $k - \frac{1}{2}$. Indeed, to achieve this, the filler can simply set $p = (i-1) + (j-i+1)/2$, place 1 unit of water into each of cups $1, 2, \ldots, i-1$ (this water will promptly be removed by the emptier), and place $1/2$ units of water into each of cups $i, \ldots, j$ (the emptier will empty from half of these cups).

Why is this type of move good for the filler? Consider the potential function measuring the sum of squares of the fills, i.e., $\Phi(x_1, \ldots, x_n) := \sum_{i=1}^{n} x_i^2$. If we let $q = \frac{j-i+1}{2}$, so that q of the fills go up by $1/2$ and q of the fills go down by $1/2$, then it is not hard to show that Φ increases by $\frac{q}{2} \geq \frac{1}{2}$. If the filler can force this to happen for n^3 consecutive time steps, then Φ will increase to $\Omega(n^3)$, which means that at least one of the $|x_i|$'s must be $\Omega(n)$. If the filler is careful, then it turns out they can further ensure that the cups are symmetric (i.e., for every cup with fill s there is another cup with fill $-s$). Thus if $|x_i| \geq \Omega(n)$ for some n, then a backlog of $\Omega(n)$ has been achieved.

The only way that the filler might be prevented from performing this type of move for n^3 consecutive time steps is if, at some time step, no two cups have the same fills as each other. Note, however, that the filler always adds half-integer values to cups and the emptier always subtracts integer values, which means that $x_1, \ldots, x_n$ are always half-integers. So, if the x_i's are all distinct, then $\max_{1 \leq i \leq n} |x_i| \geq \frac{n-1}{4} = \Omega(n)$. Recalling that the filler can ensure symmetry of the cups, it follows that in this case the filler has also achieved an $\Omega(n)$ backlog in only n^3 time steps.

Considering smaller backlogs. What if we only want to reach some backlog $o(n)$? We will now describe how the filler can achieve backlog $\Omega(t \log(n/t))$ in $O(t^3 \log(n/t))$ time steps. Combining this with some edge cases (which we defer to Section 5) results an optimal filling strategy for any backlog.

First, we claim that within t^3 steps, the filler can cause $\Theta(n)$ cups to have fills $\Omega(t)$. To see this, note that if at least half of the cups have fills less than t, then by the pigeonhole principle, there must exist some half-integer s with $|s| \leq t$ and $\Omega(n/t)$ cups of fill exactly s. If the filler causes half of these cups to go up in fill by $1/2$ and half of them to go down in fill by $1/2$, then the net effect on Φ will be that it increases by $\Omega(n/t)$. The filler can repeatedly force Φ to increase by $\Omega(n/t)$, while keeping the maximum backlog at t, until either t^3 steps have passed and $\Phi = \Omega(n \cdot t^2)$ or until at least half of the cups have reached backlog $\pm t$. Either way, at least $\Theta(n)$ of the cups must have fills more than $\Omega(t)$ or less than $-\Omega(t)$, and recalling again that the filler can also ensure a symmetry of the cups, it follows that a constant fraction of the cups have fills $\Omega(t)$.

Once we have cn cups of fill $\Omega(t)$, for some constant $c > 0$, the filler can focus on these cups, and force $c^2 n$ of these cups to fill $2 \cdot \Omega(t)$, then $c^3 \cdot n$ to fill $3 \cdot \Omega(t)$, and so on for a logarithmic number of phases. Overall, one can achieve backlog $\Omega(t \log(n/t))$ in $O(t^3 \log(n/t))$ time steps.[2]

The final piece: establishing that greedy emptying is optimal. To conclude our overview of the lowerbound, let us revisit the assumptions that (a) we are playing the negative-fill game, and (b) the emptier is playing greedily. The first assumption is trivial to remove, since the filler in the standard game is strictly better off than the filler in the negative-fill game. The second assumption, that the emptier is playing greedily, requires us to prove that the greedy emptying algorithm is always optimal (this ends up being useful to have for later results as well).

To prove the greedy emptying is optimal, in Section 4, we construct a specially designed poset on the possible states X of the system. Say that a state $X = \{x_1, \ldots, x_n\}$ **weakly monopolizes** a state $Y = \{y_1, \ldots, y_n\}$ if it is possible to order the cups such that either:

- $x_i \geq y_i$ for all i;

- or we have (a) that $x_i = y_i$ for all $i > 2$, that (b) $x_1 > y_2$, and that (b) we can get from X to Y by removing exactly 1 unit of water from cup 1 and placing some quantity $0 \leq c \leq 1$ of water into cup 2.

The transitive closure of weak monopolization induces a partial ordering on the set of all possible system states, where $A \geq B$ if there is a sequence $A = A_1, A_2, \ldots, A_j = B$ such that each A_i weakly monopolizes each A_{i+1}.

We prove that, given the choice between two states A, B with $A \geq B$, the emptier should always prefer state B. Furthermore, starting from any state X, greedy emptying always results in a state B that satisfies $B \leq A$ for every other state A that the emptier could have reached from X. Thus the greedy emptying algorithm is optimal.

3.2 Overview of the Upper Bound on Backlog

In Section 6, we show that the greedy emptying algorithm achieves an upper bound on backlog matching the lower bound of Section 5. We first consider a filler that is restricted to only making

---

[2]Note that one can only do $\log(n/t)$ phases, rather than the more intuitive $\log n$ phases, since we need at least t cups to reach fill $a \cdot t$ in order for anything to reach fill $(a+1) \cdot t$.

moves of the form described in Subsection 3.1, that is, so that the net effect of each round is that some number $2q$ of cups at some height k are replaced with q cups at height $k + 1/2$ and q cups at height $k - 1/2$. We will later show that these types of moves are (essentially) always optimal for the filler, which means this restriction is actually without loss of generality. Considering this restricted filler along with a greedy emptier leads to the following simple combinatorial problem, which we call the **stone game** (we call it the **stone-variant cup game** in Section 6):

> *Suppose you have n stones on a number line, all starting at 0. At each time step t, you may pick any point k on the number line with 2 or more stones, choose any integer $q \geq 1$ such that there are at least $2 \cdot q$ stones at k, and move q of the stones at position k to position $k - 1$ and q of the stones at position k to position $k + 1$. If you repeat this for T time steps, what is the furthest that any of the n stones may be from the origin?*

Analyzing the time to get a stone to position $\Omega(n)$. To analyze the stone game, we start by considering how long it takes for some stone to reach $b := n/10$ in absolute value. Our goal is to show that one needs at least $T = \Omega(n^3)$ time steps. To highlight the relationship between the cup game and the stone game, we shall sometimes refer to the distance of the furthest stone from the origin simply as the backlog.

Let the positions of the stones be $x_1, \ldots, x_n$. We start by recalling the potential function $\Phi = \sum_{i=1}^{n} x_i^2$. One would naturally hope that Φ could help us establish a bound of $T = \Omega(n^3)$ (just as it helped us with the $T = O(n^3)$ bound in Subsection 3.1). For example, one way to prove the $\Omega(n^3)$ time bound would be to show that Φ increases by at most $O(1)$ in each step, and that Φ takes a value of at least $\Omega(n^3)$ after the final step (i.e., when backlog $b = n/10$ is achieved). Unfortunately, we run into two problems. The first problem is that even if $\max |x_i| = b = n/10$, we may still have that $\sum_{i=1}^{n} x_i^2$ is just $O(n^2)$ after the final step (as opposed to the desired $\Omega(n^3)$). The second, more difficult problem is that the number of stones we move in each direction could be large at each time step. If we move q stones right and q stones left, Φ increases by $2q$ (rather than the desired $O(1)$), which could be as large as n. Together, these two problems mean that, a priori, we can only get a trivial $T = \Omega(n)$ bound, since the change in the potential function Φ is up to n at each time step and the final potential may be as small as $\Theta(n^2)$.

The first problem can be resolved with a more careful analysis: in particular, one can show that a backlog of $\Omega(n)$ actually does imply $\Phi = \Omega(n^3)$. The key is to prove that there can never be large gaps between consecutive stones. Namely, one can show that if there exists a stone at some position $k > 0$, there must be at least one stone at position $k - 1$ or $k - 2$, and likewise, if there exists a stone at $k < 0$, there must be at least one stone at position $k + 1$ or $k + 2$. As a result, if there is a stone at $b = n/10$, there must be a stone at position either $b - 1$ or $b - 2$, at either $b - 3$ or $b - 4$, and so on, so one can show that there are $\Omega(b)$ stones at positions $b/2$ or greater. Thus, if there is a stone at position b, then $\Phi \geq \Omega(b) \cdot (b/2)^2 \geq \Omega(n^3)$, as desired.

The more difficult piece of the analysis is to show that, even though Φ can grow significantly in a single step, Φ only increases *on average* by $O(1)$ per step. An important insight here is to create a second potential function, Ψ, and compare the growth rates of Ψ and Φ. We define this new potential function $\Psi := \sum_{i<j} |x_i - x_j|$, where $x_1, \ldots, x_n$ are the locations of the n stones. In a given step of the stone game, if we move q stones up from k to $k + 1$ and q stones down from k to $k - 1$, the first potential function Φ increases by $2q$. However, one can show that Ψ must increase by at least $2q^2$ – the core reason for this is that the q stones that moved up now each have distance 2 from the q stones that moved down, whereas before they had distance 0. Now suppose

8

for contradiction that Φ grows on average by some amount $\overline{q} = \omega(1)$ per step. This would mean that the average growth of Ψ per step is at least $\Omega(\overline{q}^2)$, so the final values of Ψ and Φ must satisfy $\Psi = \omega(\Phi)$. However, we know that $\Phi = \Omega(n^3)$ at the end of the game, and it is not hard to see that Ψ must be $O(n^3)$, since all of the x_i's are bounded in the range $[-O(n), O(n)]$. Thus we have a contradiction, and the average growth of Φ per step is actually $O(1)$. The fact that Φ is $\Omega(n^3)$ at the end of the game and grows by $O(1)$ on average per step is sufficient to show that the time T needed to achieve backlog $b = n/10$ in the stone game is at least $\Omega(n^3)$ time steps.

Considering smaller backlogs in the stone game. But what happens if we wish to analyze the time needed to achieve backlog k in the stone game, for $k \ll n$? If, at the end of the game, we knew that at least a constant fraction of the stones were in positions $\Omega(k)$, then we could use the same argument as the one above to show that the average growth rate of Φ is $O(1)$. Unfortunately, the number of stones at positions $\Omega(k)$ might be as small as $O(k)$. In this case, Φ could be as small as $O(k^3)$, whereas Ψ must be at least $\Omega(k^2 n)$. If the average growth of Φ were $\Theta(n/k)$ and the average growth of Ψ were $\Theta((n/k)^2)$, these values of Φ and Ψ could be obtained in $O(k^4/n)$ rounds, which is much smaller than our desired bound of $\Omega(k^3/\log^2(n/k))$.

To obtain the optimal bound of $\Omega(k^3/\log^2(n/k))$, we introduce another variant of the stone game that has what we call checkpoints: namely, we choose an integer ℓ and play the same game but now, once a stone has reached a position $a \cdot \ell$ for any $a \geq 0$, it can never go below it. In other words, if we move q stones from $a \cdot \ell$ to $a \cdot \ell + 1$, rather than moving q stones from $a \cdot \ell$ down to $a \cdot \ell - 1$, we keep them as is. A key insight is that, if a player wishes to get a stone to position $10 \log n \cdot \ell$, the player must have the property that, for the majority of the checkpoints, the player gets at least half of the stones that reach checkpoint $a \cdot \ell$ all the way up to checkpoint $(a + 1) \cdot \ell$.

We can think of the steps of the stone game (with checkpoints) as being split into subgames, where each subgame takes place between two checkpoints $a \cdot \ell$ and $(a + 1) \cdot \ell$. We analyze each subgame individually by creating potential functions Φ_a, Ψ_a between each pair of checkpoints at $a \cdot \ell$ and $(a + 1) \cdot \ell$. At least half of the subgames have the property that at least half of their stones make it to the next checkpoint, and this property makes each such subgame amenable to being analyzed using Φ_a and Ψ_a. By analyzing the subgames individually, we can show that the total length of the full stone game (which is the sum of the lengths of the subgames) is at least $\Omega(k^3/\log^2(n/k))$.

Finally, one can show that adding the checkpoints to the original stone game only makes it easier for the player to achieve a large backlog. This involves proving that if states X and Y have $x_i \geq y_i$ for all i, then given any stone-game operation on Y to obtain a state Y', we can generate a corresponding (but perhaps different) stone-game operation on X to obtain X', and preserve the ordering $x'_i \geq y'_i$ for all i. Thus, the $T = \Omega(k^3/\log^2(n/k))$ bound also applies to the original stone game.

Transferring bounds from the stone game to the cup game. Overall, the potential-function arguments above get optimal bounds for the *stone game* but they do not directly give bounds for the more general variable-processor cup game. The main issue is that in certain time steps in the cup game, it might be possible for the filler to make "backward" moves where both Φ and Ψ decrease considerably: in the worst case, Ψ can even decrease by up to n^2. Our analysis of the stone game relied heavily on the fact that the change in Ψ was comparable to the square of the change in Φ (in a given step), but if Ψ can decrease significantly in a single step, then our comparison of growth rates no longer works. Our fix for this is not to modify the potential functions, but rather to show that these backward moves, or any other "non-stone game" moves,

are never advantageous to the filler, even in the long run.

To show that non-stone-game moves are never advantageous for the filler, we analyze the relationship between the stone game and the variable-processor cup game. Say that a sequence $X = \{x_1, \ldots, x_n\}$ of real numbers **_majorizes_** another sequence $Y = \{y_1, \ldots, y_n\}$ (where $x_1 \geq \cdots \geq x_n$ and $y_1 \geq \cdots \geq y_n$) if $x_1 + \cdots + x_i \geq y_1 + \cdots + y_i$ for all $1 \leq i \leq n$, and if $\sum_i x_i = \sum_i y_i$. We prove that, for any sequence of cup game rounds, one can create a corresponding sequence of stone game rounds such that after every round, the sequence of stone positions majorizes the sequence of cup fills. Since X majorizing Y implies that $x_1 \geq y_1$, we get that the maximum backlog after T steps of the cup game is at most the maximum backlog after T steps of the stone game. Therefore, even if we cannot utilize the potential functions on the general cup game, it suffices to look at the stone game as it will have a greater maximum backlog.

The main technical claim needed to show this majorization result is to show that if a sequence X majorizes a sequence Y, and if Y can be converted to a sequence Y' in one round of filling/greedy-emptying in the variable-processor cup game, then it is possible to convert X into some X' (also with a single round of filling/greedy emptying) such that X' majorizes Y'. This claim is quite casework-heavy and crucially uses the fact that we are in the *variable-processor* cup game. Indeed, the choice of p may have to differ between the round performed on X and the round performed on Y, and perhaps surprisingly, the claim is actually *false* in the fixed p-processor cup game.

3.3 Overview of Resource-Augmentation Analysis

In Section 7, we analyze the variable-processor cup game with ε resource augmentation (and non-negative fills), meaning that in each time step, the emptier is permitted to remove up to $1 + \varepsilon$ units of water from each of p cups (rather than just 1 unit of water).

We prove that even a very small amount of resource augmentation significantly decreases backlog of the game: the greedy emptying algorithm achieves backlog $O(\varepsilon^{-1} \log n)$.

The proof uses the probabilistic method. Rather than analyzing the greedy emptying algorithm directly, we instead construct a randomized emptying algorithm that, at any given moment, achieves backlog $O(\varepsilon^{-1} \log n)$ with *non-zero* probability. (Importantly, the randomized algorithm is against an *adaptive* filler, not an oblivious one.) The existence of such a randomized algorithm non-constructively implies the existence of a deterministic emptying algorithm with the same guarantee. But we already know that the best deterministic emptying algorithm is the greedy one. Thus greedy emptying achieves backlog $O(\varepsilon^{-1} \log n)$ (deterministically).

To simplify our discussion here, let us consider only games of polynomial length (in Section 7 we consider arbitrary game lengths). In this case, our randomized emptying algorithm can simply take an approach that we call **_proportional emptying_**: in each time step of the game, if the filler places some amount q_j of water into cup j, then the emptier empties from cup j with probability exactly q_j.

To analyze proportional emptying, we show that, at any given moment, each cup has fill $O(\varepsilon^{-1} \log n)$. Roughly speaking, the amount of water in each cup can then be modeled as a biased random walk: in each time step, the expected amount of water that the emptier removes is a factor of $1 + \varepsilon$ larger than the amount of water that the filler inserts. The bias in the random walk prevents it from ever reaching a large fill. The result is that a simple Chernoff-style analysis (modified using a variation on Azuma's martingale inequality to handle the fact that the filler is an adaptive adversary) can be used to bound the fill in each cup by $O(\varepsilon^{-1} \log n)$ (with high probability).

Interestingly, the above argument also immediately yields a nontrivial bound in the resource-augmentation-free setting. Now the amount of water in each cup follows an *unbiased* random walk. At any given time step t, one can bound the height of such random walk by $O(\sqrt{t \log n})$ with high probability. Using the fact that greedy emptying is as good as any randomized emptying strategy, it follows that greedy emptying achieves backlog $O(\sqrt{t \log n})$ in a t time step game.

4   The Greedy Emptier is Always Optimal

Intuitively, the greedy emptying algorithm (i.e., always empty from the fullest cups) should be the optimal deterministic emptying algorithm (for both the p-processor cup game and the variable-processor cup game). This intuition is known to be true for the single-processor cup game starting in a state with all empty cups (in particular, the lower and upper bounds on backlog match in this case [1]), but whether or not the intuition is correct in general has remained an open question. (We remark that there are variants of the game, for example the fixed-rate cup game, where greedy emptying is *not* optimal, even asymptotically [1].) In this section, we prove that greedy emptying is, in fact, optimal for both of the p-processor cup game and the variable-processor cup game.. That is, for any starting state of the cups, and for any game length, greedy emptying is the best possible algorithm for minimizing backlog against an adaptive filler.

For any state S and any length t, define $\text{OPT}(S,t)$ to be the supremum backlog that a filler can achieve at the end of a t-step game starting at state S assuming the emptier plays optimally. That is, $\text{OPT}(S,0)$ is just the amount of water in the fullest cup of S, and $\text{OPT}(S,t)$ for $t > 0$ is defined by induction as

$$\sup_{S' \text{ reachable from } S \text{ by filler}} \quad \min_{S'' \text{ reachable from } S' \text{ by emptier}} \text{OPT}(S'', t-1).$$

Note that this also allows for us to talk about the ***optimal emptier***, which is the emptier that achieves backlog at most $\text{OPT}(S,t)$ in any t-step game starting at any state S.

Define $\text{GREEDY}(S,t)$ to be the supremum backlog that a filler can achieve at the end of a t-step game starting at state S, assuming the emptier plays greedily. We wish to prove that $\text{OPT}(S,t) = \text{GREEDY}(S,t)$ for all S,t. Throughout the section we shall prove all of our results for both the versions of the games with non-negative fills and the versions of the games with negative fills.

We say that a state A ***monopolizes*** a state B if it is possible to assign labels $1, 2, \ldots, n$ to the cups in A and B such that:

- cups $3, 4, \ldots, n$ contain the same amounts of water in both states;

- cup 1 in A contains more water than cup 2 in B;

- cup 1 in A contains one more unit of water than cup 1 in B;

- cup 2 in B contains c more units of water than cup 2 in A for some $0 \le c \le 1$.

In other words, you can get from A to B by removing 1 unit of water from cup 1 and and placing $c \le 1$ units into cup 2, and cup 1 in A contains more water than cup 2 in B.

We say that A ***dominates*** B if it is possible to label the cups such that each cup i in A contains at least as much water as cup i in B. Finally, we say that A ***weakly monopolizes*** B if either A monopolizes B, or A dominates B.

11

We now prove several properties of weak monopolization in the p-processor cup game (for any p). Our first lemma says that, if A weakly monopolizes B, then for any filler move on B, there is some filler move on A that preserves the weak monopolization of A over B.

Lemma 4.1. *Suppose A weakly monopolizes B. Suppose that there is a p-processor filler move that transforms B into some B'. Then there exists a p-processor filler move that transforms A into some A' such that A' weakly monopolizes B'.*

Proof. If A dominates B, then the lemma is trivial. Thus we can assume that A monopolizes B. Let q be the amount by which cup 1 in A contains more water than cup 2 in B.

Define X to be a state that we reach from A if we perform the same filler move that transforms B into B'. If cup 1 in X contains more water than cup 2 in B' then we can set $A' = X$ and be done.

Suppose cup 1 in X does not contain more water than cup 2 in B'. Then, in the transformation from B to B', the filler must have placed at least q more water into cup 2 than into cup 1. Let r be the amount of water that the filler placed into cup 1, and let $r + q + s$ (for some $s \geq 0$) the amount of water that the filler placed into cup 2.

Now define A' to be the state that one would reach from A by performing the following p-processor filler move: place $r + s$ units of water into cup 1 and $q + r$ units of water into cup 2 (and then place water into cups $3, 4, \ldots, n$ in the same way as to transform B to B'). Cup 1 in A' contains the same amount of water as cup 2 in B', and cup 2 and A' contains the same amount of water as cup 1 in B'. Thus A' and B' are equivalent states, meaning that A' weakly monopolizes B'. $\square$

Our next lemma says that, if A weakly monopolizes B, and if A is then greedily emptied from, then it is possible to empty from B in such a way that the monopolization relationship is preserved.

Lemma 4.2. *Suppose A weakly monopolizes B. Let A' be the state reached if a p-processor emptier greedily empties from A. Then there exists a valid p-processor emptier move on B that achieves some state B' such that A' weakly monopolizes B'.*

Proof. If A dominates B, then the lemma is trivial. Thus we can assume that A monopolizes B.

Let a_1, a_2 denote the fills of cups 1 and 2 in A, and let b_1, b_2 denote the fills of cups 1 and 2 in B. So $a_1 = b_1 + 1$, $b_2 = a_2 + c$ for some $0 \leq c \leq 1$, and $a_1 > b_2$. Furthermore, let a'_1, a'_2 the fills of cups 1 and 2 in A', and let b'_1, b'_2 denote the fills of cups 1 and 2 in B' (once we've defined it).

If the greedy emptier on A empties from neither cup 1 nor cup 2, then the same set of empties on B results in a B' that is weakly monopolized by A'.

Next consider the case where the greedy emptier on A empties from both cups 1 and 2. Then we empty from the same set of cups in B to arrive at a state B'. In the negative-fill game, is immediate to see that A' monopolizes B'. In the standard game, we must be careful about the fact that one of b_1 or a_2 might be less than 1. If $b_1 < 1$, then $b'_1 = 0 \leq a'_2$, and since $a_1 > b_2$, we also have $a'_1 \geq b'_2$; thus A' dominates B' in this case. If $a_2 < 1$ but $b_1 \geq 1$, then we have $a'_1 = b'_1 + 1$ and $b'_2 = a'_2 + c'$ for some $0 \leq c \leq 1$; we also have $a'_1 \geq b'_2$ (since $a_1 \geq b_2$), so A' monopolizes B' in this case.

Finally, consider the case where the greedy emptier on A empties from only one of the cups 1 or 2. Since $a_1 > b_2 \geq a_2$, the emptier must empty from cup 1. Suppose we construct B' by performing

12

the same set of empties on B as were performed on A, but we remove water from cup 2 instead of cup 1. Then $a_1' = b_1'$ and $a_2' \geq b_2'$, so A' dominates B'. This completes the proof. □

By combining the previous two lemmas, we can arrive at the following.

Lemma 4.3. *Consider either the p-processor cup game, for some p, or the variable processor cup game (and consider either the negative-fill case or the non-negative-fill case). Suppose A weakly monopolizes B. Then for any t,*

$$\text{GREEDY}(A, t) \geq \text{OPT}(B, t).$$

Proof. We prove the lemma by induction on t. In the base case of $t = 0$, the lemma reduces to showing that A has backlog at least as large as B; this follows from the fact that A weakly monopolizes B.

Now consider some $t > 0$, and suppose the result holds for $t - 1$. For each state B' that the filler can reach from B, Lemma 4.1 tells us that there must exist a state A' that the filler can reach from A in such that A' weakly monopolizes B'. Moreover, Lemma 4.2 tells us that, if greedy emptying from A' results in some state A'', then there must be an emptying move on B' that results in a state B'' such that A'' weakly monopolizes B''.

Note that, in the previous paragraph, the filler's move from B to B' (i.e., the choice of B') fully determines A' (by the construction in Lemma 4.1), A'' (by greedy emptying on A'), and B'' (by the construction in Lemma 4.2). Thus, we will think of A', A'', B'' as being functions of B'.

Expanding out $\text{GREEDY}(A, t)$, we have

$$\text{GREEDY}(A, t) \geq \sup_{B'} \text{GREEDY}(A'', t - 1).$$

By the inductive hypothesis, since each A'' weakly monopolizes B'', we have $\text{GREEDY}(A'', t-1) \geq \text{OPT}(B'', t - 1)$. Thus

$$\text{GREEDY}(A, t) \geq \sup_{B'} \text{OPT}(B'', t - 1).$$

Since $\sup_{B'} \text{OPT}(B'', t - 1) \geq \text{OPT}(B, t)$, the lemma follows. □

Recall that our goal is to upperbound $\text{GREEDY}(A, t)$ by $\text{OPT}(A, t)$ for all A, t. Thus Lemma 4.3 may at first seem to be backward progress, since the lemma establishes a *lower bound* on $\text{GREEDY}(A, t)$. The trick to using Lemma 4.3 is that we will only ever apply the lemma to states A and game lengths t for which we have already proven by induction that $\text{OPT}(A, t) = \text{GREEDY}(A, t)$; in this setting, the lemma establishes that $\text{OPT}(A, t) \geq \text{OPT}(B, t)$ which, as we shall see, ends up being critical to the proof.

Theorem 4.4. *Consider either the p-processor cup game, for some p, or the variable-processor cup game (and consider either the negative-fill case or the non-negative-fill case). For all states A and game lengths t,*

$$\text{GREEDY}(A, t) = \text{OPT}(A, t).$$

Proof. We prove the theorem by induction on t with a trivial base case of $t = 0$. Consider $t > 0$, and suppose the theorem holds for $t - 1$.

Consider any state A' that the filler can reach from A. Let A'' be the state reached from A' if the emptier empties greedily, and let B be the state reached from A' if the emptier empties according

13

to OPT. Since the transformation from A' to A'' empties from cups with at least as much water as the transformation from A' to B, there must be a sequence of states $X_0, X_1, \ldots, X_k$, with $X_0 = B$ and $X_k = A''$, such that each X_i weakly monopolizes X_{i-1}. In particular, one can define the X_i's such the only difference between each X_i and X_{i+1} is that, to get from A' to X_{i+1} instead of from A' to X_i, the emptier removes water from a cup j instead of a cup k, where cup j contains less water than cup k in state A'. It is straightforward to verify that this results in each X_{i+1} weakly monopolizing each X_i: if cup k (in state A') has fill at least 1, then X_{i+1} monopolizes X_i; and otherwise, both cups k and j (in state A') have fills less than 1, and thus X_{i+1} dominates X_i.

By the inductive hypothesis, we know that $\text{OPT}(X_i, t-1) = \text{GREEDY}(X_i, t-1)$ for each X_i. Thus Lemma 4.3 tells us that $\text{OPT}(X_i, t-1) \geq \text{OPT}(X_{i-1}, t-1)$. By transitivity, it follows that $\text{OPT}(X_k, t-1) \geq \text{OPT}(X_0, t-1)$, and thus $\text{OPT}(B, t-1) \geq \text{OPT}(A'', t-1)$. Again applying the inductive hypothesis to deduce that $\text{OPT}(A'', t-1) = \text{GREEDY}(A'', t-1)$, we have that

$$\text{OPT}(B, t-1) \geq \text{GREEDY}(A'', t-1).$$

Thus

$$\text{GREEDY}(A, t) = \sup_{A''} \text{GREEDY}(A'', t-1) \leq \sup_B \text{OPT}(B, t-1) = \text{OPT}(A, t).$$

Finally, as $\text{GREEDY}(A, t) \geq \text{OPT}(A, t)$ trivially, we have $\text{GREEDY}(A, t) = \text{OPT}(A, t)$. $\qquad\square$

Corollary 4.5. *Theorem 4.4 continues to hold for the game with non-negative fills even if the emptier is given $1 + \varepsilon$ resource augmentation.*

Proof. The proof of Theorem 4.4 continues to hold without modification. The only difference is that, now, we say that a state A monopolizes a state B if there is a ordering of the cups for which two properties hold: (1) we can take $1 + \varepsilon$ unit of water from some cup 1 in A, place some amount $0 \leq c \leq 1 + \varepsilon$ of water into some other cup 2 in A, and in doing so arrive at B; and (2) cup 1 in A contains more water than cup 2 in B. $\qquad\square$

5 Lower Bounding Backlog

In this section, we prove the optimal lower bound on the backlog. In other words, we show that for any integer $t \geq 1$, there exists a filling strategy that can guarantee a backlog of $\Omega(b(t))$ against *any* emptying strategy, after t rounds of the variable-processor cup game, where $b(t)$ is defined as in Equation (1) in the introduction (Section 1). This result can be captured compactly in the following theorem:

Theorem 5.1. *For some absolute constant $c > 0$ and any $k \leq c \cdot n$, the filler in the variable-processor cup game can force a maximum backlog of $\Omega(k)$ using only $O\left(k + \frac{k^3}{\log^2(n/k)}\right)$ time steps.*

We remark that due to the optimality of greedy emptying, which we proved in the previous section, we focus in this section on designing filling strategies that are effective against the greedy emptier.

In Subsection 5.1, we give a simple proof for obtaining $\Omega(n)$ backlog in $O(n^3)$ rounds. In Subsection 5.2, we generalize Subsection 5.1 and develop the full proof of Theorem 5.1.

14

5.1 Warmup: Achieving a $\Theta(n)$ Backlog Quickly Against a Greedy Emptier

In this subsection, we show that in only n^3 rounds, a filler can achieve a backlog of $\frac{n-1}{2}$ against a greedy emptier. While the following argument is simpler than the more refined lower bound in Subsection 5.2, it conveys much of the intuition behind the general lower bound. Here, we only consider the standard variable-processor cup game, though in subsection 5.2 we will consider both the standard and negative-fill games.

Theorem 5.2. *Against a greedy emptier in the standard variable-processor cup game, the filler can achieve a backlog of $\frac{n-1}{2}$ in only n^3 rounds.*

Proof. The filler follows the following simple strategy, which will guarantee that the fills at the end of each round are half integers (i.e., integer multiples of $\frac{1}{2}$). Suppose the cups after some round, in sorted order, are $x_1 \geq x_2 \geq \cdots \geq x_n$, which are all half-integers.

If $x_i > x_{i+1}$ for all $1 \leq i \leq n-1$, then we must have that $x_i \geq x_{i+1} + \frac{1}{2}$ for all $i \leq n-1$, so $x_1 \geq \frac{n-1}{2}$. In this case, we have already achieved a backlog of $\frac{n-1}{2}$.

Otherwise, we consider the potential function $\Phi = \sum_{i=1}^n x_i^2$, i.e., the sum of the squares of the backlogs. Consider the smallest p such that $x_p = x_{p+1}$. We set the number of processors for the round to be p, we place 1 unit of water into each of cups $x_1, \ldots, x_{p-1}$ and we place $1/2$ unit of water into each of cups x_p, x_{p+1}. Then, the greedy emptier will empty 1 unit from each of the first $p-1$ cups and will choose to empty 1 unit from either the pth cup or the $(p+1)^{\text{th}}$ cup (WLOG, they choose the $(p+1)$-th cup). This means that, over the course of the entire round, the first $p-1$ cups are unchanged, the pth cup goes up by $\frac{1}{2}$, and $(p+1)^{\text{th}}$ cup goes down by $\frac{1}{2}$. The only exception is if $x_p = x_{p+1} = 0$ at the beginning, in which case one of these two cups will go up to $1/2$ and the other remains at 0. Clearly, all of the fills remain half-integers at the end of each round. If we replace x_p and $x_{p+1} = x_p$ with $x_p + \frac{1}{2}$ and $x_p - \frac{1}{2}$, then the sum of the squares of the backlogs increases by $1/2$. Otherwise, if we replace $x_p = x_{p+1} = 0$ with 0 and $1/2$ in some order, then the sum of the squares of the backlogs increases by $1/4$. Thus, at the end of each round, unless one of the backlogs was already $\frac{n-1}{2}$ or greater, Φ increases by at least $\frac{1}{4}$.

Therefore, after n^3 rounds, either at some point we will have achieved $x_1 > x_2 > \cdots > x_n$ and thus a backlog of $\frac{n-1}{2}$, or we have that $\sum_{i=1}^n x_i^2 \geq \frac{n^3}{4}$, which would mean that $\max x_i \geq \frac{n}{2}$, since all of the x_i's are nonnegative. As a result, there exists a filling strategy that can guarantee a $\frac{n-1}{2}$ backlog against a greedy emptier in at most n^3 rounds. $\qquad\blacksquare$

5.2 The General Lower Bound

In this subsection, we prove the lower bound on the backlog for the variable-processor cup game after t rounds, showing that the optimal backlog is $\Omega(b(t))$ for $b(t)$ defined as in Equation (1). Equivalently, we show that for any integer $k \leq n$, a filler can achieve backlog $\Omega(k)$ in $O\left(k + \frac{k^3}{\log^2(n/k)}\right)$ rounds against any emptying strategy. We remark that if we are playing t rounds of the game and the filler has achieved backlog b by round $t' < t$, the filler can ensure the backlog stays at b by setting $p = n$ and filling every cup for steps $t'+1, \ldots, t$. Thus, we just need to show the filler can obtain this backlog *within* this many rounds.

Throughout this subsection, we assume that we are playing the negative-fill game, and that the emptier is always greedy. We remove both of these assumptions at the end in Theorem 5.1.

Lemma 5.3. *Assume that the emptier is always greedy, and that at the beginning of some round, the fills are all (possibly negative) half-integers. For some integer $k \in \mathbb{Z}$ and positive integer $q \in \mathbb{N}$, suppose there are at least $2q$ cups having fill exactly $k/2$. Then, the filler can ensure that at the end of the round (i.e., after both the filler and emptier move), exactly q of the cups of fill exactly $k/2$ have increased to $(k+1)/2$, exactly q of the cups of fill exactly $k/2$ have decreased to $(k-1)/2$, and all remaining cups are unchanged.*

Proof. Suppose the cups are sorted in order $x_1 \geq x_2 \geq \cdots \geq x_n$, such that $x_i = x_{i+1} = \cdots = x_{i+2q-1} = k/2$. Also, suppose i is the smallest integer such that $x_i = k/2$, so either $x_{i-1} > k/2$ or $i = 1$. Then, the filler will set $p = i - 1 + q$ and fill the first $i - 1$ cups with 1 unit of water and the cups $x_i, \ldots, x_{i+2q-1}$ each with $1/2$ unit of water. Then, if $i > 1$, the emptier is forced to empty 1 unit of water from each of the first $i-1$ cups. In addition, we have that the cups $i, i+1, \ldots, i+2q-1$ all have fills $(k+1)/2$, which is at least $1/2$ unit of water more than all later cups, which means that the emptier will remove 1 unit of water from exactly q of the cups $i, i+1, \ldots, i+2q-1$. Therefore, all cups $1, \ldots, i-1$ are unchanged (since 1 unit of water is added and then removed) and all cups $i+2q, \ldots, n$ are unchanged (since water is never added nor removed). Finally, among the cups $i, i+1, \ldots, i+2q-1$, exactly q of them will end up at $(k-1)/2$ and exactly q of them will end up at $(k+1)/2$. $\qquad\square$

Lemma 5.4. *Suppose that k, m are positive integers such that $4k | m$, and suppose there are at least m cups that currently have fill 0 (where $n \geq m$ is the total number of cups). Then, against a greedy emptier, the filler can ensure that at least $m/4$ cups will have fill exactly $k/2$ after $O(k^3)$ rounds.*

Proof. WLOG assume that the first m cups have fills $x_1, x_2, \ldots, x_m$, and at the start, $x_1 = x_2 = \cdots = x_m = 0$. (We do not assume the cups are sorted in order of fill.) We only ever modify the first m cups, and after each step, we will maintain the following invariants:

1. All of the fills are half-integers.

2. For each integer j, the number of cups of fill $j/2$ equals the number of cups of fill $-j/2$.

3. For each integer j, the number of cups of fill $j/2$ is always a multiple of $\frac{m}{4k}$.

4. No cup has fill greater than $k/2$ or less than $-k/2$.

Trivially, all 4 of these invariants are true at the beginning, since all the fills are 0 and $\frac{m}{4k} | m$. Our procedure is the following. Suppose that there exists some integer j such that $-k < j < k$ and the number of cups of fill $j/2$ is at least $\frac{m}{2k}$. Then, we use Lemma 5.3 to move $\frac{m}{4k}$ of these cups to fill $(j+1)/2$ and move $\frac{m}{4k}$ of these cups to fill $(j-1)/2$. In addition, if $j \neq 0$, then we know the number of cups of fill $-j/2$ is at least $\frac{m}{2k}$. So, in the next step, we move $\frac{m}{4k}$ of these cups to fill $(-j+1)/2$ and move $\frac{m}{4k}$ of these cups to fill $(-j-1)/2$. When $j \neq 0$, we perform these two steps consecutively as a pair. We keep repeating these types of steps (and pairs of steps) until we can no longer do so. It is clear that the first three invariants are preserved, and the last is preserved since we only modify cups that have fill between $-(k-1)/2$ and $(k-1)/2$, and we change their fills by at most $1/2$ per round.

Now, we note that at each step, the potential function $\Phi = \sum_{i=1}^m x_i^2$ increases by $\frac{m}{8k}$, since

$$\frac{m}{4k} \cdot \left(\frac{j+1}{2}\right)^2 + \frac{m}{4k} \cdot \left(\frac{j-1}{2}\right)^2 - \frac{m}{2k} \cdot \left(\frac{j}{2}\right)^2 = \frac{m}{8k}.$$

16

This also means that if we do a pair of steps (i.e., modifying cups of fill $j/2$ and $-j/2$), the potential function Φ increases by $\frac{m}{4k}$.

Now, after $4k^3$ steps (and pairs of steps), Φ is at least $4k^3 \cdot \frac{m}{8k} = \frac{1}{2} \cdot mk^2$. But this is impossible, since all of the fills of cups 1 through m are between $k/2$ and $-k/2$, so the maximum possible value of Φ is $\frac{1}{4} \cdot mk^2$. This means that before reaching $4k^3$ of steps or pairs of steps, we must not be able to do any more steps. That is, for all j such that $-k < j < k$, there are less than $\frac{m}{2k}$ cups of fill exactly $j/2$, and because of our third invariant, this means there are at most $\frac{m}{4k}$ cups of fill exactly $j/2$. Therefore, the number of cups of fill between $-k+1$ and $k-1$ is at most $(2k-1) \cdot \frac{m}{4k} \leq \frac{m}{2}$. So, at least $\frac{m}{2}$ cups must have fill $\pm\frac{k}{2}$, and by the second invariant, this means at least $\frac{m}{4}$ cups must have fill $\frac{k}{2}$. Reaching this stage required at most $4k^3$ steps and pairs of steps, which means the filler can succeed in at most $8k^3$ steps. $\qquad\square$

Corollary 5.5. *Suppose k, m are positive integers such that $4k|m$, and suppose there are at least m cups that currently have fill exactly t for some real number t. Then, after $O(k^3)$ time steps, the filler can force at least $m/4$ of these cups to have fill exactly $t + \frac{k}{2}$, against a greedy emptier.*

Proof. We can apply same argument as in Lemma 5.4, where all of the fills are shifted up by the same number t. So, after at most $8k^3$ time steps, at least $m/4$ of the cups will have fill exactly $t + \frac{k}{2}$. $\qquad\square$

From here, we are able to conclude with our main result of this section.

Theorem 5.1. *For some absolute constant $c > 0$ and any $k \leq c \cdot n$, the filler in the variable-processor cup game can force a maximum backlog of $\Omega(k)$ using only $O\left(k + \frac{k^3}{\log^2(n/k)}\right)$ time steps.*

Proof. For now we will assume that the emptier is greedy, and that we are playing the negative-fill cup game. We will remove both assumptions at the end of the proof.

First, suppose $k \geq c \log n$. In this case, let $k' := \left\lceil \frac{k}{c \log(n/k)} \right\rceil \geq 2$. Also, let n' be such that $n \geq n' > \frac{n}{4}$ and $n' = k' \cdot 4^r$ for some integer $r \geq 1$.

Since there are at least $n' = k' \cdot 4^r$ cups of fill 0 at the beginning of the game, we can apply Corollary 5.5 with $t = 0$ and $m = n'$ to make sure there are at least $k' \cdot 4^{r-1}$ cups of fill k' after $O(k'^3)$ time steps, since $4k'|n'$. Then, we can again apply Corollary 5.5 with $t = k'$ and $m = n'/4$ to make sure there are at least $k' \cdot 4^{r-2}$ cups of fill $2k'$ after an additional $O(k'^3)$ time steps. We can repeat this r times until we have at least k' cups of fill $r \cdot k'$ after a total of $O(r \cdot k'^3)$ time steps. But since $r = \log_4(n'/k') = \Theta(\log(n/k \cdot \log(n/k))) = \Theta(\log(n/k))$, this means that the filler can achieve a maximum backlog of $r \cdot k' = \Theta(k)$ using only $O(r \cdot k'^3) = O(k^3/\log^2(n/k))$ time steps.

Next, suppose $k < c \log n$. In this case, set $n' = 2^k$: if $c < 1$ then $n' < n$. By Lemma 5.3, we can send 2^{k-1} cups to fill 1 during the first round, then 2^{k-2} of those cups to fill 2 during the second round, and so on until we have 1 cup at fill k after k rounds.

Up until now we have assumed that the filler was operating against a greedy emptying algorithm in the negative-fill cup game. However, by Theorem 4.4, if the emptier uses a different emptying algorithm, the filler could always modify their strategy to get equal or greater backlog in the same amount of time. So, regardless of the emptying strategy, the filler can force a maximum backlog of $\Theta(k)$ using $O\left(k + \frac{k^3}{\log^2(n/k)}\right)$ time steps, assuming we are playing the negative-fill cup game. Finally, recall that the maximum backlog that the filler can ensure in the negative-fill game is at most the maximum backlog that the filler can ensure in the standard game for any fixed number

of time steps. Thus, if the filler can force a maximum backlog of $\Theta(k)$ using $O\left(k + \frac{k^3}{\log^2(n/k)}\right)$ time steps in the negative-fill game, then the filler can force the same backlog using the same number of time steps in the standard game, which completes the theorem. $\square$

Finally, to conclude the section, we use the optimality of greedy emptying to resolve an open question of [30] concerning whether a filler who is limited in the speed at which they can change p can still force a large backlog.

Proposition 5.6. *Consider the variable-processor cup game, starting with all empty cups (and with negative fills either allowed for disallowed). Suppose the filler is restricted to only change p once every n^c steps, for some constant $c \geq 0$, and to only change p by ± 1 at a time. Then the filler can still force a backlog of $\Omega(n)$ in polynomial time.*

Proof. Call this version of the game the *change-limited variable-processor cup game*. Notice that the proof of optimality for greedy emptying never changes the value of p used in any given step. Thus the same proof implies that greedy emptying is optimal in the change-limited variable-processor cup game. Henceforth we will assume that the emptier is greedy.

Next, observe that, no matter the value of p, the filler can always "skip" a step in the following way: the filler simply places 1 unit of water into each of the p fullest cups, forcing the greedy emptier to remove water from those cups, so that the step has no net effect.

We have already proven that, in the standard variable-processor cup game, the filler can cause backlog $\Omega(n)$ in polynomial time. The filler can use the same strategy here, but with the following modification: between any two consecutive steps that use p and q processors, respectively, the filler spends $|p - q|n^c$ steps changing the number of processors from p to q. That is, for each $i \in [p, q]$, the filler spends n^c steps with i processors, and renders each of those steps to have no net effect. The result is that, in polynomial time, the filler can force backlog $\Omega(n)$, as desired. $\square$

6 Upper Bounding Backlog

In this section, we prove that for any integer $t \geq 1$, the greedy emptying strategy can guarantee that the backlog does not exceed $O(b(t))$ against *any* filling strategy, after t rounds of the variable-processor cup game, where $b(t)$ is defined as in Equation (1) in the introduction (Section 1). The result can be phrased compactly as follows:

Theorem 6.1. *Let $1 \leq k \leq O(n)$. Then, assuming the emptier follows a greedy emptying strategy in the variable-processor cup game, the filler needs $\Omega\left(k + \frac{k^3}{\log^2(n/k)}\right)$ rounds to achieve backlog k.*

Our approach to proving the theorem will be as follows. In Subsection 6.1, we introduce a combinatorial variant of the cup game, which we analyze via potential function arguments. In Subsection 6.2, we introduce and prove important results on majorization and domination. Finally, in Subsection 6.3, we use the results from 6.2 to justify why this combinatorial variant captures the standard game, and in doing so we prove Theorem 6.1.

6.1 The ℓ-Variant Cup Game

Before analyzing the performance of greedy on the standard variable-processor cup game, we consider the following variants of the cup game in which the filler can only make a certain highly structured type of move.

The stone-variant of the cup game. We define the *stone-variant cup game* as follows (note that this game is equivalent to the "stone game" in Section 3, but is phrased as a game on cups rather than stones).

The stone-variant cup game is a variation on the negative-fill variable-processor cup game. At the start of each round, the fills are nonnegative integers $x_1 \geq \cdots \geq x_n$, and initially, the fills are all 0. During every round, the player of the game chooses an integer $k \geq 0$ and an integer $q \geq 1$ such that at least $2q$ of the x_i's are exactly k (if no such k and q exist, then the game is over). Then, among these x_i's, we raise exactly q of them by 1 and lower exactly q of them by 1. Finally, at the end of each round (for notational convenience), we re-sort the x_is to be in descending order. The player's goal is to maximize backlog, that is, the value of x_1.

Let us take a moment to remark on how the stone-variant cup game relates to the negative-fill variable-processor cup game. Recall the proof of Lemma 5.3 that, in the negative-fill variable-processor cup game, it is possible for the filler to perform a move with the following net effect: some number $2q$ of cups that all have the same fills k are replaced with q cups that have fills $k + 1/2$ and q cups that have fills $k - 1/2$. This is equivalent to performing a move in the stone-variant game, except that fills in the stone-variant game are re-normalized to be twice as large. Thus, the stone-variant game can be viewed as a variation on the negative-fill variable-processor cup game, where the filler's behavior is limited to a specific type of move.

The ℓ-variant of the cup game. We define the ℓ-variant of the cup game to be similar to the stone-variant, but in addition, there are infinitely equally spaced checkpoints $0, \ell, 2\ell, \cdots$, where ℓ is some positive integer. The ℓ-variant of the game adds the constraint that: once a cup reaches a checkpoint, it never goes below that checkpoint. In other words, if during some round $k = a \cdot \ell$ for some integer a, then we raise q of the x_i's from k to $k + 1$ but rather than lowering q other x_i's from k to $k - 1$, we keep them at k. (Note that that the checkpoint at 0 ensures that all cups have non-negative fills.)

How we will use the games. Although the filler in the two above games has been restricted (compared to the filler in the original game), we will show in subsequent subsections how to reduce the standard variable-processor cup game to the stone-variant, and then how to reduce the stone-variant to the ℓ-variant. In other words, we show that restricting the filler to make ℓ-variant-style moves does not make the game (asymptotically) harder for the filler.

In this subsection, we focus on the task of bounding the backlog ℓ-variant of the game. As described in Section 3, our main insight in analyzing this game is to create two potential functions and compare their growth. In the following propositions, we define two slightly simplified potential functions Φ, Ψ, and analyze how certain operations similar to those in the ℓ-variant cup game affect Φ and Ψ.

Proposition 6.2. *Suppose that $x_1 \geq x_2 \geq \cdots \geq x_n$ are integers, and $q \geq 1$ is an integer such that for some subset $S \subset [n]$ of size $2q$, all of the x_i's for $i \in S$ are all equal. Suppose that for q values of $i \in S$ we increase x_i by 1, and for the other q values of $i \in S$ we decrease x_i by 1. Then, the potential function $\Phi = \sum_{i=1}^{n} x_i^2$ increases by exactly $2q$ and the potential function $\Psi = n \cdot \sum_i |x_i| + \sum_{i<j} |x_i - x_j|$ increases by at least $2q^2$.*

Proof. The fact that $\Phi = \sum x_i^2$ increases by $2q$ is straightforward, since for any x, $q \cdot (x + 1)^2 + q \cdot (x - 1)^2 - 2q \cdot x^2 = 2q$, and since the x_i's we modified are all initially at the same value.

Now, we prove that Ψ increases by at least $2q^2$. First, we note that since the x_i's are sorted in descending order, we can assume WLOG that $S = [r : s]$ for some $1 \leq r \leq s \leq n$ with $s - r + 1 = 2q$.

19

Moreover, we can assume WLOG that we decrease $x_r, \ldots, x_{r+q-1}$ by 1 and increase $x_{r+q}, \ldots, x_s$ by 1. Then, we note that if i, j are both not in the set $[r : s]$, that $|x_i - x_j|$ remains constant. Also, if i is not in the set $[r : s]$, then $\sum_{j \in [r:s]} |x_i - x_j|$ does not change if either $x_i \geq x_r + 1$ or $x_i \leq x_r - 1$, and otherwise increases by $2q$ if $x_i = x_r$. Hence, this sum does not decrease for any i. Finally, if we just restrict to $i < j$ with $i, j \in [r : s]$, originally each $|x_i - x_j|$ is 0, but after increasing q of the x_i's by 1 and decreasing q of the x_i's by 1, the sum of these $|x_i - x_j|$'s becomes exactly $2q^2$. In total, this means that

$$\sum_{i<j} |x_i - x_j| = \sum_{\substack{i<j \\ i,j \notin [r:s]}} |x_i - x_j| + \sum_{i \notin [r:s]} \sum_{j \in [r:s]} |x_i - x_j| + \sum_{\substack{i<j \\ i,j \in [r:s]}} |x_i - x_j|$$

increases by at least $0 + 0 + 2q^2 = 2q^2$. Finally, we have that $\sum_i |x_i|$ does not decrease, since for any integer x, $q \cdot |x+1| + q \cdot |x-1| \geq 2q \cdot |x|$. This completes the proof. $\square$

Proposition 6.3. *Suppose that $x_1 \geq x_2 \geq \cdots \geq x_n \geq 0$ are integers, such that for some subset $S \subset [n]$ of size q, all of the x_i's for $i \in S$ are all 0. Suppose that we increase x_i by 1 for all $i \in S$. Then, the potential function $\Phi = \sum_{i=1}^{n} x_i^2$ increases by exactly q and the potential function $\Psi = n \cdot \sum_i |x_i| + \sum_{i<j} |x_i - x_j|$ increases by at least q^2.*

Proof. Showing that Φ increases by q is straightforward, since each x_i for $i \in S$ increases from 0 to 1.

Now, we show that Ψ increases by at least q^2. First, note that since the x_i's are sorted in descending order, we can assume WLOG that $S = [r : n]$ for $r = n - q + 1$. Now, note that $n \cdot \sum_i |x_i|$ increases by exactly $n \cdot q$ since each of $x_r, \ldots, x_n$ increase from 0 to 1. In addition, for any $i < j$, $|x_i - x_j|$ remains constant if $i, j < r$ or $i, j \geq r$, and does not decrease by more than 1 if $i < r \leq j$. There are exactly $(r-1) \cdot (n-r+1) = (n-q) \cdot q$ pairs (i, j) satisfying $i < r \leq j$, so Ψ increases by at least $n \cdot q - (n-q) \cdot q \geq q^2$. $\square$

Now, we return to the ℓ-variant cup game. We think of $\mathbf{x}^{(t)} = (x_1^{(t)}, \ldots, x_n^{(t)})$ as the set of fills after time step t, where $\mathbf{x}^{(0)} = (0, 0, \ldots, 0)$. Suppose that there exists a sequence $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}$ such that after T time steps, there are precisely n_a cups that have reached the checkpoint $a \cdot \ell$ for each $a \geq 0$. (In other words, $x_i^{(T)} \geq a \cdot \ell$ if and only if $i \leq n_a$.) Note that $n_0 = n$, and since a cup never goes below a checkpoint after reaching it, if $i > n_a$ then $x_i^{(t)} < a \cdot \ell$ for all $0 \leq t \leq T$.

Now, we will consider the following potential functions that are modifications of the potential functions Φ, Ψ that we used in Propositions 6.2 and 6.3. For any nonnegative integer a, we define $f_{a,\ell}(x) := \max(0, \min(x - a \cdot \ell, \ell))$. In other words,

$$f_{a,\ell}(x) = \begin{cases} 0 & \text{if } x < a \cdot \ell \\ x - a \cdot \ell & \text{if } a \cdot \ell \leq x \leq (a+1) \cdot \ell \\ \ell & \text{if } x > (a+1) \cdot \ell. \end{cases}$$

For any integers $x_1, \ldots, x_n$ and $0 \leq n_a \leq n$, define $S := S_{n_a}(x_1, \ldots, x_n)$ to be the set of the largest n_a indices of the x_i's (breaking ties arbitrarily). In other words, $S = S_{n_a}(x_1, \ldots, x_n) \subset [n]$ is a set of size n_a, such that for all $i \in S$ and $j \notin S$, $x_i \geq x_j$. For $S = S_{n_a}(x_1, \ldots, x_n)$, let the potential function Φ_a be

$$\Phi_a := \Phi_{a,\ell,n_a}(x_1, \ldots, x_n) = \sum_{i \in S} f_{a,\ell}(x_i)^2$$

20

and let the potential function Ψ_a be

$$\Psi_a := \Psi_{a,\ell,n_a}(x_1,\ldots,x_n) = n_a \cdot \sum_{i \in S} |f_{a,\ell}(x_i)| + \sum_{\substack{i<j \\ i,j \in S}} |f_{a,\ell}(x_i) - f_{a,\ell}(x_j)|.$$

Note that Φ_a and Ψ_a are defined for each nonnegative integer a, and that the tiebreaking order to determine S does not affect the values Φ_a or Ψ_a. In addition, note that the numbers n_a only depend on the final state $\mathbf{x}^{(T)}$, whereas the potential functions are defined for each $\mathbf{x}^{(t)}$ for all $0 \le t \le T$.

Lemma 6.4. *Let $k \le n$, $h = \Theta\left(\log\left(\frac{n}{k}+1\right)\right)$ such that $h \ge 2$ and $h|k$, and $\ell = \frac{k}{h}$. Suppose that in the ℓ-variant cup game, it is possible that after T time steps, there are exactly n_a cups that have reached the ath checkpoint or above, for integers $n_0, n_1, n_2, \ldots$ Moreover, suppose that $n_h \ge k$. Then, $T \ge \Omega(h \cdot \ell^3)$.*

Proof. By definition of the n_a's, we have that $n = n_0 \ge n_1 \ge \cdots \ge n_h$, and we are assuming $n_h \ge k$. Because $\frac{n_0}{n_h} \le \frac{n}{k} \le e^{C \cdot h}$ for some fixed constant $C > 0$, we must have that for at least $\frac{h}{2}$ choices of $0 \le a \le h-1$, that $\frac{n_a}{n_{a+1}} \le e^{2C}$, or equivalently, $\frac{n_{a+1}}{n_a} \ge e^{-2C} =: c$ for some fixed $c > 0$.

Let T_a represent the set of time steps $t \le T$ during which cups with fill between $a \cdot \ell$ and $(a+1) \cdot \ell - 1$, inclusive, at the beginning of the time step are modified. Then, for any time step $t \in T_a$, the potential function Φ_a increases by some q_t and the potential function Ψ_a increases by at least $q_t^2/2$, by Propositions 6.2 and 6.3. Otherwise, if $t \notin T_a$, then Φ_a and Ψ_a are unaffected: to see why, we justify that for all i, $f_{a,\ell}(x_i)$ is unaffected at time step t if $t \notin T_a$. Note that we are only modifying cups of fill either at least $(a+1) \cdot \ell$ or at most $a \cdot \ell - 1$. If we modify a cup of fill $x_i \ge (a+1) \cdot \ell$, the cup will not go below the $(a+1)^{\text{th}}$ checkpoint, so $f_{a,\ell}(x_i)$ will remain ℓ. Likewise, if we modify a cup of fill $x_i \le a \cdot \ell - 1$, then $x_i \le a \cdot \ell$ even if we increase the fill by 1, so $f_{a,\ell}(x_i) = 0$ still. Finally, note that sorting the cups based on x_i has no effect on Φ_a or Ψ_a. We also remark that each time $t \le T$ is in exactly one set T_a for $a \ge 0$, since every cup we modify at any time step t had the same fill at the start of time step t.

Now, fix some a such that $\frac{n_{a+1}}{n_a} \ge c$. At the beginning of the game, we have that $f_{a,\ell}(x_i) = 0$ for all i, which means $\Phi_a = \Psi_a = 0$. Moreover, at the end of the T time steps, $f_{a,\ell}(x_i) \in [0, \ell]$ for all of the top n_a values of x_i, with at least $n_{a+1} \ge c \cdot n_a$ of them being ℓ. Therefore, at the end of the game, $\Phi_a \ge n_{a+1} \cdot \ell^2 \ge cn_a \cdot \ell^2$ but $\Psi_a \le n_a^2 \cdot \ell + \binom{n_a}{2} \cdot \ell \le 2n_a^2 \cdot \ell$. In addition, recall that for each $t \in T_a$, Φ_a increases by some q_t, but Ψ_a increases by at least $q_t^2/2$. Therefore,

$$\sum_{t \in T_a} q_t = \Phi_a \ge cn_a \cdot \ell^2$$

but

$$\sum_{t \in T_a} q_t^2 \le 2\Psi_a \le 4n_a^2 \cdot \ell.$$

Now, using the Cauchy-Schwarz inequality, we have that

$$|T_a| \ge \frac{\left(\sum_{t \in T_a} q_t\right)^2}{\sum_{t \in T_a} q_t^2} \ge \frac{(cn_a \cdot \ell^2)^2}{4n_a^2 \cdot \ell} = \frac{c^2}{4} \cdot \ell^3.$$

Finally, since the T_a's are disjoint sets, and since this inequality is true for at least $\frac{h}{2}$ values of a, we have that $T \ge \sum_a |T_a| \ge \frac{c^2}{8} \cdot h \cdot \ell^3$, as desired. $\square$

21

Note that Lemma 6.4 had an assumption that $n_h \geq k$, or that at least k cups have reached the checkpoint at $h \cdot \ell$. The following proposition will be useful in removing that assumption.

Proposition 6.5. *In the ℓ-variant cup game, if there exists a cup of fill k for some $k \geq 3$, then there must exist a cup of fill either $k-1$ or $k-2$.*

Proof. Suppose not, so there was a first time t such that for some integer k, there was a cup of fill k but no cups of fill $k-1, k-2$. Suppose at the previous time step there was a cup of fill $k-2$. Then, at the current time step, either some of these cups still have fill $k-2$, or at least one of the cups increased to have fill $k-1$. Suppose at the previous time step, some of the cups had fill $k-1$. Then, at the current time step, either some of the cups still have fill $k-1$, or we increased half of these cups to fill k and decreased the other half to fill $k-2$. It is possible that $k-1$ was a checkpoint, in which case the decreased cups would remain at fill $k-1$. Regardless, if there was a cup of fill either $k-1$ or $k-2$ at the previous time step, there will remain at least one such cup.

Therefore, in the previous time step there were no cups of fill $k-1$ or $k-2$. On the other hand, there must have been at least one cup of fill k or greater, since in the current time step there is a cup with fill k. Thus, in time step $t-1$ there is some fill $k' \geq k$ such that there was a cup with fill k' but no cups with fill $k'-1$ or $k'-2$. This contradicts the assumption that t is the smallest time such that for some integer k, there is a cup of fill k but no cups of fill $k-1, k-2$. $\square$

Combining Proposition 6.5 with Lemma 6.4, we have the following result, which essentially provides us with tight bounds for the ℓ-variant cup game.

Theorem 6.6. *Let $k \leq n$, $h = \Theta\left(\log\left(\frac{n}{k}+1\right)\right)$ such that $2 \leq h \leq k$, and $\ell = \left\lfloor \frac{k}{h} \right\rfloor$. Suppose that in the ℓ-variant cup game, it is possible that after T time steps, there is a cup of fill $4k$ or greater. Then, $T \geq \Omega(h \cdot \ell^3) = \Omega(k^3/h^2)$.*

Proof. By Proposition 6.5, if there is a cup of fill $4k$ or greater, then there must be at least k cups of fill k or greater, meaning that there are at least k cups which have reached the hth checkpoint or higher, since $k \geq h \cdot \ell$. Therefore, we can apply Lemma 6.4 to conclude our proof. $\square$

6.2 Majorization and Domination

In this subsection, we develop the necessary toolkit that will allow us to relate the ℓ-variant cup game to the standard variable-processor cup game. First, we define majorization and domination, which relate to comparing two real-valued sequences, and we state some basic results about them. We then prove more complicated results which will be important in relating the two games.

For any sequence of fills $x_1, \ldots, x_n \in \mathbb{R}$ of the n cups, we abbreviate the state of the cups as $\{x_i\}$. We will explicitly specify whenever we assume $x_1 \geq x_2 \geq \cdots \geq x_n$, in which case we say that the sequence $\{x_i\}$ is ***sorted***. We also do not require $x_1, \ldots, x_n$ to be nonnegative in this subsection.

Definition 6.7. Suppose that we have two sequences $x_1 \geq x_2 \geq \cdots \geq x_n$ and $y_1 \geq y_2 \geq \cdots \geq y_n$, where $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$. Then, we say that $\{x_i\}$ ***majorizes*** $\{y_i\}$ if for every $m \leq n$, $\sum_{i=1}^{m} x_i \geq \sum_{i=1}^{m} y_i$. In general, if the sequences $\{x_i\}$ and $\{y_i\}$ are not necessarily sorted, we say that $\{x_i\}$ majorizes $\{y_i\}$ if this is true after sorting $\{x_i\}$ and $\{y_i\}$.

Remark. Note that majorization satisfies transitivity, so for any real number r, majorization creates a partial order on the set of length n sequences (up to permutation of terms) that add to r.

We note two basic results about majorization.

Proposition 6.8. *Suppose that $\{a_1, \ldots, a_k\}$ majorizes $\{b_1, \ldots, b_k\}$ (where $\sum_{i=1}^{k} a_i = \sum_{i=1}^{k} b_i$). Then, for any sequence $\{c_1, \ldots, c_r\}$, $\{a_1, \ldots, a_k, c_1, \ldots, c_r\}$ majorizes $\{b_1, \ldots, b_k, c_1, \ldots, c_r\}$.*

Proposition 6.9. *If $\{x_i\}$ and $\{y_i\}$ are sorted, and $\{x_i\}$ majorizes $\{y_i\}$, then we can convert $\{y_i\}$ into $\{x_i\}$ via a finite series of perturbations of increasing some y_j by ε and decreasing y_k by ε for some $0 < \varepsilon < 1$ and $j > k$. (The perturbations do not all need to use the same value of ε.) Moreover, after each such perturbation, the sequence is still sorted.*

Definition 6.10. Given two sequences $x_1 \geq x_2 \geq \cdots \geq x_n$ and $y_1 \geq y_2 \geq \cdots \geq y_n$, we say that $\{x_i\}$ **dominates** $\{y_i\}$ if $x_i \geq y_i$ for all i. In general, we say that $\{x_i\}$ dominates $\{y_i\}$ if this is true after sorting $\{x_i\}$ and $\{y_i\}$.

Remark. Unlike majorization, where we only say that $\{x_i\}$ majorizes $\{y_i\}$ if $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$, we only have that $\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i$ and $\{x_i\}$ dominates $\{y_i\}$ if the sequences $\{x_i\}$ and $\{y_i\}$ are the same (after sorting).

We note the following well-known proposition about domination.

Proposition 6.11. *Suppose that $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ are real-valued, not necessarily sorted, sequences, but where $x_i \geq y_i$ for all i. Then, after sorting, $\{x_i\}$ dominates $\{y_i\}$.*

For completeness, we prove Propositions 6.8, 6.9, and 6.11 in Appendix B.

Next, we prove the following proposition relating to adding fills.

Proposition 6.12. *If the cups have fills $x_1 \geq x_2 \geq \cdots \geq x_n$ before the start of a time step, and if the filler fills a_i to each x_i during the time step, where $0 \leq a_i \leq 1$, then we can always assume without loss of generality that $a_1 + x_1 \geq a_2 + x_2 \geq \cdots \geq a_n + x_n$.*

Proof. Suppose that $a_i + x_i \leq a_{i+1} + x_{i+1}$. Then, since $x_i \geq x_{i+1}$, this means that $x_{i+1} \leq x_i \leq a_i + x_i \leq a_{i+1} + x_{i+1} \leq 1 + x_{i+1}$ since fills are always between 0 and 1. Instead, we can replace a_{i+1} with $(a_i + x_i) - x_{i+1}$ and a_i with $(a_{i+1} + x_{i+1}) - x_i$, which are both in the range $[0, 1]$, and the fills of the ith and $(i+1)^{\text{th}}$ cups will just be switched. We can repeat this bubble-sorting procedure until we actually have $a_1 + x_1 \geq a_2 + x_2 \geq \cdots \geq a_n + x_n$, and we do not affect the fills of the cups up to permutation. Moreover, all fills are still between 0 and 1 for that round. $\square$

Hence, we assume that for any round, both before and after the filler moves, the cups are sorted in non-increasing order. Next, the (greedy) emptier removes 1 unit of water from each of the first p cups, and then sorts the cups in non-increasing order.

Our main technical result to reduce from the standard variable-processor cup game to the stone-variant cup game is the following lemma. The lemma roughly implies that if some sequence $\{x_i\}$ majorizes another sequence $\{y_i\}$, then the filler would always be at an advantage in the future (in the negative-fill cup game) if the fills were $\{x_i\}$ instead of $\{y_i\}$.

Lemma 6.13. *Suppose that $x_1 \geq x_2 \geq \cdots \geq x_n$ and $y_1 \geq y_2 \geq \cdots \geq y_n$ are reals, where $\{x_i\}$ majorizes $\{y_i\}$, and that there exists some filling procedure on $\{y_i\}$ such that after greedy emptying (allowing for negative fills), $\{y_i\}$ becomes some sequence $\{y_i'\}$ ($\{y_i'\}$ is not necessarily sorted). Then, there exists a filling procedure on $\{x_i\}$ (with a possibly different value of p) such that after greedy emptying (allowing for negative fills), $\{x_i\}$ becomes $\{x_i'\}$ and $\{x_i'\}$ majorizes $\{y_i'\}$.*

23

Proof. By Proposition 6.9 and the transitivity of majorization, we can assume that the only difference between $\{x_i\}$ and $\{y_i\}$ is that there exist some $j < k$ such that $x_j = y_j + \varepsilon$ and $x_k = y_k - \varepsilon$, and for all other i, $x_j = y_j$, and moreover, the sequences $\{x_i\}$ and $\{y_i\}$ are sorted.

By Proposition 6.12, we may assume in the filling procedure for $\{y_i\}$ we add a_i to each y_i, so that $\sum a_i = p$ (where $1 \le p \le n$ is an integer) and $a_1 + y_1 \ge \cdots \ge a_n + y_n$, and then remove 1 from each of the first p of the $(a_i + y_i)$'s (as they are the largest).

First, if $p = n$, it is clear that the filler fills every cup with 1 unit, and then the emptier empties every cup, so $\{y_i'\} = \{y_i\}$. So, we can just do the same filling procedure on $\{x_i\}$, and after the emptier moves, we have $\{x_i'\} = \{x_i\}$ which majorizes $\{y_i\}$. So, from now on we assume $1 \le p \le n - 1$. We have 4 main cases for what can happen.

Case 1: $a_j > 0$ and $a_k < 1$. In this case, let $\varepsilon' = \min(a_j, 1 - a_k, \varepsilon)$. If we had $x_j = y_j + \varepsilon'$ and $x_k = y_k - \varepsilon'$ instead, then we could just perform the following filling procedure on $\{x_i\}$ to get $\{x_i'\}$: fill the jth cup with $a_j - \varepsilon'$, the kth cup with $a_k + \varepsilon'$, and the remaining cups with a_i for each i (note that, by the definition of ε', this is guaranteed to be a valid filling procedure, placing between 0 and 1 units of water in each cup). Then, in fact we would have $\{x_i'\} = \{y_i'\}$, since even before the greedy emptying (but after the filling), the fills would be the same. If $\varepsilon = \varepsilon'$, this completes the case. Otherwise, either $a_j = \varepsilon'$ or $1 - a_k = \varepsilon'$, in which case we let $\{z_i\}$ be the same as $\{y_i\}$, except $z_j = y_j + \varepsilon'$ and $z_k = y_k - \varepsilon'$. We have shown there is a filling procedure on $\{z_i\}$ that obtains the same results as the filling procedure on $\{y_i\}$, so it suffices to prove the lemma for $\{x_i\}$ and $\{z_i\}$ (with the filler placing water into $\{z_i\}$ so that after greedy emptying we get $\{y_i'\}$) instead of for $\{x_i\}$ and $\{y_i\}$. But in this case, since $\varepsilon' \in \{a_j, 1 - a_k\}$, the filler on $\{z_i\}$ either does not place any fill in the jth cup (if $\varepsilon' = a_j$) or places a full 1 unit of fill into the kth cup (if $\varepsilon' = 1 - a_k$), so we have successfully reduced to the case where either $a_j = 0$ or $a_k = 1$.

Case 2: $a_j = 0$ and $p < j$. In this case, for the $\{x_i\}$'s we do the same filling procedure of adding a_i to each x_i. Therefore, since $a_1 + y_1 \ge a_2 + y_2 \ge \cdots \ge a_n + y_n$ and since $x_j = y_j + \varepsilon, x_k = y_k - \varepsilon$, the largest p values of $\{x_i + a_i\}$ are either the first p values, or the first $p - 1$ values and the jth value. But since $a_j = 0$, we have that $x_j + a_j = x_j$, and since both the $\{y_i\}$'s and $\{x_i\}$'s are sorted in nonincreasing order, this means that $x_j + a_j = x_j \le x_p \le x_p + a_p$, so the largest p values of $\{x_i + a_i\}$ are just the first p values. So, the greedy emptier empties the first p cups, and $y_i' = x_i'$ for all i, except $x_j' = y_j + a_j + \varepsilon$ and $x_k' = y_k + a_k - \varepsilon$. Since $y_j + a_j \ge y_k + a_k$, this implies that $\{x_j', x_k'\}$ majorizes $\{y_j', y_j'\}$, so by Proposition 6.8, $\{x_i'\}$ majorizes $\{y_i'\}$.

Case 3: $a_j = 0$ and $j \le p$. In this case, for the $\{x_i\}$'s we fill 1 unit for cup j and a_i units for all $i \ne j$ (so we now have filled $p + 1$ total units). But now, we empty the fullest $p + 1$ cups (instead of the fullest p cups). Note that, whereas the transformation of $\{y_i\}$ to $\{y_i'\}$ uses a p-processor time step of the cup game, the transformation from $\{x_i\}$ to $\{x_i'\}$ uses a $p + 1$-processor time step of the cup game. We have 5 possible subcases:

a) $j \le p < k - 1$. In this case, after filling the $\{x_i\}$ cups, the $p + 1$ fullest cups are just the first $p + 1$ cups. So, the only cups that are of different fills when changing from $\{y_i'\}$ to $\{x_i'\}$ are

$$
\begin{aligned}
y_j' &= y_j - 1 && \text{becomes } x_j' &&= y_j + \varepsilon && \text{(since } a_j = 0 \text{ and } x_j = y_j + \varepsilon) \\
y_{p+1}' &= y_{p+1} + a_{p+1} && \text{becomes } x_{p+1}' &&= y_{p+1} + a_{p+1} - 1 \\
y_k' &= y_k + a_k && \text{becomes } x_k' &&= y_k + a_k - \varepsilon && \text{(since } x_k = y_k - \varepsilon)
\end{aligned}
$$

Note that $y_j' + y_{p+1}' + y_k' = x_j' + x_{p+1}' + x_k'$, and since $y_j \ge y_{p+1} + a_{p+1} \ge y_k + a_k$, we have that $\max(y_j', y_{p+1}', y_k') \le y_j \le x_j' \le \max(x_j', x_{p+1}', x_k')$, and $\min(y_j', y_{p+1}', y_k') = \min(y_j - 1, y_k + a_k) \ge$

$\min(y_{p+1} + a_{p+1} - 1, y_k + a_k - \varepsilon) \geq \min(x'_j, x'_{p+1}, x'_k)$. Therefore, $\{x'_j, x'_{p+1}, x'_k\}$ majorizes $\{y'_j, y'_{p+1}, y'_k\}$, so by Proposition 6.8, $\{x'_i\}$ majorizes $\{y'_i\}$.

b) $p = k - 1$ and either $y_k + a_k - \varepsilon \geq y_{k+1} + a_{k+1}$ or $k = n$. In this case, after filling the $\{x_i\}$ cups, the $p+1$ fullest cups are still the first $p+1$ cups. So, the only cups that are of different fills when changing from $\{y'_i\}$ to $\{x'_i\}$ are

$$
\begin{aligned}
y'_j &= y_j - 1 \quad \text{becomes} \quad x'_j = y_j + \varepsilon && \text{(since } a_j = 0\text{)} \\
y'_k &= y_k + a_k \quad \text{becomes} \quad x'_k = y_k + a_k - \varepsilon - 1 && \text{(since } k = p + 1\text{)}.
\end{aligned}
$$

We have that $y'_j + y'_k = x'_j + x'_k$, but $x'_j \geq y_j = y_j + a_j \geq \max(y'_j, y_k + a_k) = \max(y'_j, y'_k)$, so $\{x'_j, x'_k\}$ majorizes $\{y'_j, y'_k\}$. Therefore, by Proposition 6.8, $\{x'_i\}$ majorizes $\{y'_i\}$.

c) $p = k - 1$ and $y_k + a_k - \varepsilon < y_{k+1} + a_{k+1}$. In this case, after filling the $\{x_i\}$ cups, the $p+1$ fullest cups are the first p cups and the $(k+1)^{\text{st}}$ cup. So, the only cups that are of different fills when changing from $\{y'_i\}$ to $\{x'_i\}$ are

$$
\begin{aligned}
y'_j &= y_j - 1 && \text{becomes} \quad x'_j &&= y_j + \varepsilon && \text{(since } a_j = 0\text{)} \\
y'_k &= y_k + a_k && \text{becomes} \quad x'_k &&= y_k + a_k - \varepsilon \\
y'_{k+1} &= y_{k+1} + a_{k+1} && \text{becomes} \quad x'_{k+1} &&= y_{k+1} + a_{k+1} - 1
\end{aligned}
$$

Note that $y'_j + y'_k + y'_{k+1} = x'_j + x'_k + x'_{k+1}$. Also, $\max(y'_j, y'_k, y'_{k+1}) \leq y_j + a_j = y_j \leq x'_j \leq \max(x'_j, x'_k, x'_{k+1})$, and $\min(y'_j, y'_k, y'_{k+1}) \geq \min(y_j - 1, y_{k+1} + a_{k+1}) \geq x'_{k+1} \geq \min(x'_j, x'_k, x'_{k+1})$. Therefore, $\{x'_j, x'_k, x'_{k+1}\}$ majorizes $\{y'_j, y'_k, y'_{k+1}\}$, so by Proposition 6.8, $\{x'_i\}$ majorizes $\{y'_i\}$.

d) $p \geq k$ and either $y_k + a_k - \varepsilon \geq y_{p+2} + a_{p+2}$ or $p = n - 1$. In this case, after filling the $\{x_i\}$ cups, the $p+1$ fullest cups are the first $p+1$ cups. So, the only cups that are of different fills when changing from $\{y'_i\}$ to $\{x'_i\}$ are

$$
\begin{aligned}
y'_j &= y_j - 1 && \text{becomes} \quad x'_j &&= y_j + \varepsilon && \text{(since } a_j = 0\text{)} \\
y'_k &= y_k + a_k - 1 && \text{becomes} \quad x'_k &&= y_k + a_k - 1 - \varepsilon \\
y'_{p+1} &= y_{p+1} + a_{p+1} && \text{becomes} \quad x'_{p+1} &&= y_{p+1} + a_{p+1} - 1
\end{aligned}
$$

Note that $y'_j + y'_k + y'_{p+1} = x'_j + x'_k + x'_{p+1}$. Also, since $y_j \geq y_k + a_k \geq y_{p+1} + a_{p+1}$, we have that $\max(x'_j, x'_k, x'_{p+1}) = y_j + \varepsilon > \max(y'_j, y'_k, y'_{p+1})$. Also, since $x'_k \leq y'_k \leq y'_j$ and $x'_{p+1} \leq y'_{p+1}$, we have that $\min(x'_j, x'_k, x'_{p+1}) \leq \min(y'_j, y'_k, y'_{p+1})$. Therefore, $\{x'_j, x'_k, x'_{p+1}\}$ majorizes $\{y'_j, y'_k, y'_{p+1}\}$, so by Proposition 6.8, $\{x'_i\}$ majorizes $\{y'_i\}$.

e) $p \geq k$ and $y_k + a_k - \varepsilon < y_{p+2} + a_{p+2}$. In this case, after filling the $\{x_i\}$ cups, the $p+1$ fullest cups are the first $p+2$ cups, except the kth cup. So, the only cups that are of different fills when changing from $\{y'_i\}$ to $\{x'_i\}$ are

$$
\begin{aligned}
y'_j &= y_j - 1 && \text{becomes} \quad x'_j &&= y_j + \varepsilon && \text{(since } a_j = 0\text{)} \\
y'_k &= y_k + a_k - 1 && \text{becomes} \quad x'_k &&= y_k + a_k - \varepsilon \\
y'_{p+1} &= y_{p+1} + a_{p+1} && \text{becomes} \quad x'_{p+1} &&= y_{p+1} + a_{p+1} - 1 \\
y'_{p+2} &= y_{p+2} + a_{p+2} && \text{becomes} \quad x'_{p+2} &&= y_{p+2} + a_{p+2} - 1
\end{aligned}
$$

We have that $y'_j + y'_k + y'_{p+1} + y'_{p+2} = x'_j + x'_k + x'_{p+1} + x'_{p+2}$. Also, since $y_j \geq y_k + a_k \geq y_{p+1} + a_{p+1} \geq y_{p+2} + a_{p+2}$, we have that $x'_j \geq x'_k \geq x'_{p+1} \geq x'_{p+2}$. However, it is simple to see that $x'_j \geq \max(y'_j, y'_k, y'_{p+1}, y'_{p+2})$, that $x'_{p+2} \leq \min(y'_j, y'_k, y'_{p+1}, y'_{p+2})$, and that $x'_j + x'_k = y_j + (y_k + a_k)$ which is greater than or equal to the sum of any two of $y'_j, y'_k, y'_{p+1}, y'_{p+2}$. Therefore, $\{x'_j, x'_k, x'_{p+1}, x'_{p+2}\}$ majorizes $\{y'_j, y'_k, y'_{p+1}, y'_{p+2}\}$, so by Proposition 6.8, $\{x'_i\}$ majorizes $\{y'_i\}$.

Case 4: $a_k = 1$. We show that this case can be derived from Cases 2 and 3. The basic idea is that we proceed in three steps: first we make use of symmetry within the negative-fill variable-processor cup game in order to show that if we began with fills $\{-y_1, -y_2, \ldots, -y_n\}$, it would be possible to perform a round in the game that results in states $\{-y'_1, -y'_2, \ldots, -y'_n\}$; we then apply Cases 2 and 3 to the states $\{-x_1, -x_2, \ldots, -x_n\}$ and $\{-y_1, -y_2, \ldots, -y_n\}$ in order to deduce that it is possible to transform $\{-x_1, -x_2, \ldots, -x_n\}$ in a single round into some $\{-x'_1, -x'_2, \ldots, -x'_n\}$ that majorizes $\{-y'_1, -y'_2, \ldots, -y'_n\}$; finally, we again make use of symmetry to argue that it is also possible to transform $\{x_1, x_2, \ldots, x_n\}$ into $\{x'_1, x'_2, \ldots, x'_n\}$ in a single round, and that $\{x'_1, x'_2, \ldots, x'_n\}$ majorizes $\{y'_1, y'_2, \ldots, y'_n\}$.

First, note that by the definition of majorization, $\{x_1, \ldots, x_n\}$ majorizes $\{y_1, \ldots, y_n\}$ if and only if $\{-x_n, \ldots, -x_1\}$ majorizes $\{-y_n, \ldots, -y_1\}$. Therefore, since $\{x_i\}$ and $\{y_i\}$ are sorted, we have that $\{-x_n, \ldots, -x_1\}$ majorizes $\{-y_n, \ldots, -y_1\}$ with $-x_k = -y_k + \varepsilon$ and $-x_j = -y_j - \varepsilon$.

Now, instead of adding a_i to each y_i (where $a_k = 1$), we consider what happens if we add $1 - a_i$ to $-y_i$ for each i. This results in fills $1 - (y_n + a_n) \geq 1 - (y_{n-1} + a_{n-1}) \geq \cdots \geq 1 - (y_1 + a_1)$. Then, since $\sum_{i=1}^{n}(1 - a_i) = n - p$, the greedy emptier subtracts 1 from the $n - p$ fullest cups to get the sequence $\{-(y_n - a_n), \ldots, -(y_{p+1} - a_{p+1}), 1 - (y_p + a_p), \ldots, 1 - (y_1 + a_1)\}$. This sequence precisely equals $\{-y'_n, -y'_{n-1}, \ldots, -y'_1\}$, where $y'_i = y_i + a_i - 1$ for $1 \leq i \leq p$ and $y'_i = y_i + a_i$ otherwise. (Note that $\{y'_1, \ldots, y'_n\}$ is not necessarily sorted.) In other words, it is possible to perform a round on the $\{-y_i\}$s that brings us directly to the $\{-y'_i\}$s.

Since $a_k = 1$, we have that $1 - a_k = 0$, and moreover, $-y_k$ comes before $-y_j$ in the non-increasing sorted order. Thus, we may use either Case 2 or Case 3 on the sequences $\{-x_n, \ldots, -x_1\}$ and $\{-y_n, \ldots, -y_1\}$ to obtain that there exists a sequence of fills $1 - b_n, \ldots, 1 - b_1$ for $0 \leq b_1, \ldots, b_n \leq 1$, such that $\sum_{i=1}^{n}(1 - b_i) = n - p'$ for some p' possibly different from p, and such that when $1 - b_i$ is added to $-x_i$ for all i and then 1 is subtracted from the fullest $n - p + 1$ cups, the resulting sequence majorizes $\{-y'_n, \ldots, -y'_1\}$. Notice, however, that this resulting sequence is just $\{1 - (b_n + x_n), \ldots, 1 - (b_1 + x_1)\}$ but with the largest $n - p'$ values lowered by 1; or equivalently, it is $\{-(b_n + x_n), \ldots, -(b_1 + x_1)\}$ but with the smallest p' values increased by 1. So, returning to the original sequence $\{x_1, \ldots, x_n\}$, we have shown that there exist $0 \leq b_1, \ldots, b_n \leq 1$, where $\sum_{i=1}^{n} b_i = p'$, such that, if the filler added b_i to each x_i, and the emptier subtracted 1 from the largest p' values, we would get a sequence $\{x'_1, \ldots, x'_n\}$, not necessarily sorted, such $\{-x'_n, \ldots, -x'_1\}$ majorizes $\{-y'_n, \ldots, -y'_1\}$ after sorting. Since majorization is preserved by negation, it follows that $\{x'_1, \ldots, x'_n\}$ majorizes $\{y'_1, \ldots, y'_n\}$, as desired. $\square$

We next prove a similar result to Lemma 6.13, though for domination and specifically for a time step of the stone-variant cup game.

Lemma 6.14. *Suppose that $\{x_i\}$ and $\{y_i\}$ are sorted and integer valued, and $\{x_i\}$ dominates $\{y_i\}$. Suppose we perform a round of the stone-variant cup game on $\{y_i\}$ to get some sequence $\{y'_i\}$. Then, one can perform a round of the stone-variant cup game on $\{x_i\}$ to get a sequence $\{x'_i\}$ that dominates $\{y'_i\}$.*

Proof. Suppose that $y_1 \geq y_2 \geq \cdots \geq y_n$, where for some integer k, $[p : q] = \{p, p + 1, \ldots, q\}$ is the set of indices i such that $y_i = k$. Assume that for the stone-variant cup game on $\{y_i\}$, we increase y_i for $i \in [p : p']$ by 1, and decrease y_i for $i \in [q' : q]$ by 1, where $p' < q'$ and $p' - p = q - q' \geq 0$. We can always make the assumption WLOG that any given time step of the stone-variant cup game proceeds in this fashion. In addition, note that the resulting sequence $\{y_i'\}$ is already sorted.

Now, we have that $x_1 \geq x_2 \geq \cdots \geq x_n$, and $x_i \geq y_i$ for all i. Now, let r be the first index such that $x_r \leq k$ (if $x_n > k$, then set $r = n + 1$). Now, if $r > p'$, then $x_i \geq k + 1 \geq y_i'$ for all $i \in [p : p']$, and $x_i \geq y_i \geq y_i'$ for all $i \notin [p : p']$, so we can do nothing to the sequence $\{x_i\}$ and we get $\{x_i'\}$ that dominates $\{y_i'\}$. Alternatively, we have that $p \leq r \leq p'$. By domination, we must have that $x_i = k$ for all $i \in [r : q]$. So, if we increase x_i for all $i \in [r : p']$ by 1 and decrease x_i for all $i \in [s : q]$ by 1, where s is chosen so that $p' - r = q - s$, then we claim $\{x_i'\}$ dominates $\{y_i'\}$. First, since $p' - p = q - q'$ and $p' \geq r \geq p$, this means that $q \geq s \geq q'$, so $[r : p']$ and $[s : q]$ are disjoint nonempty sets. Next, note that $x_i = x_i' \geq y_i = y_i'$ for all $i \notin [p : q]$. Next, for $p \leq i < r$, we have $x_i = x_i' \geq k + 1 = y_i'$, and for $r \leq i \leq p'$, we have $x_i' = x_i + 1 = k + 1 = y_i'$. For $p' < i < s$, we have $x_i' = k \geq y_i'$, and for $s \leq i \leq q$, we have $x_i' = x_i - 1 = k - 1 = y_i'$. Therefore, $\{x_i'\}$ dominates $\{y_i'\}$. $\square$

The final result in this section roughly tells us that to ensure majorization, a stone-variant-like time step is often better than anything the filler can hope to achieve in the standard variable-processor cup game.

Lemma 6.15. *Suppose that $x_1 \geq x_2 \geq \cdots \geq x_n$ are all even integers, and let $\{x_i'\}$ be the fills after some filling and greedy emptying procedure. Then, there exists some integer $k \in \{x_1, \ldots, x_n\}$ such that we can increase some (possibly 0) of the x_i's that equal k to $k + 2$, decrease an equal number of the x_i's that equal k to $k - 2$, and the new sequence majorizes the sorted order of $\{x_i'\}$.*

Proof. Consider any filling procedure with fills $a_1, \ldots, a_n$, followed by greedy emptying, on $\{x_i\}$. Let $k = x_p$. First, note that after filling and greedy emptying, $x_i' = x_i + a_i - 1$ for $i \leq p$ and $x_i' = x_i + a_i$ for $i > p$. Therefore, $|x_i' - x_i| \leq 1$ for all i. In addition, if $x_i < k$, then $i > p$ so $x_i' \geq x_i$, and if $x_i > k$, then $i < p$ so $x_i' \leq x_i$.

Now, suppose that the set of i such that $x_i = k$ is $[q : r] = \{q, q+1, \ldots, r\}$ (note that $q \leq p \leq r$). Now, let $q' = \lfloor \frac{q+r-1}{2} \rfloor$ and $r' = \lceil \frac{q+r+1}{2} \rceil$. Note that $[q : q']$ and $[r' : r]$ have the same size. Also, if $[q : r]$ has even size, then $q \not\equiv r \bmod 2$, so $r' = q' + 1$, and if $[q : r]$ has odd size, then $r' = q' + 2$.

Now, let x_i'' be the sequence where $x_i'' = x_i$ if $x_i \not\equiv k$, and otherwise, $x_i'' = x_i + 2$ for $q \leq i \leq q'$, $x_i'' = x_i - 2$ for $r' \leq i \leq r$, and $x_i'' = x_i$ for $q' < i < r'$. Note that $\{x_i''\}$ is sorted. In addition, since $\{x_i\}$ are all even integers, this means that even if $\{x_i'\}$ is not sorted, we have that $x_1', \ldots, x_{q-1}' \geq x_q', \ldots, x_r' \geq x_{r+1}', \ldots, x_n'$. Now, let z_i be the sorted (in non-increasing order) of x_i'. Since $x_i \geq x_i'$ for all $1 \leq i \leq q - 1$, we have that $\{x_i\}_{i=1}^{q-1}$ dominates $\{x_i'\}_{i=1}^{q-1}$ by Proposition 6.11, so $x_i \geq z_i$ for all $1 \leq i \leq q - 1$. Likewise, $\{x_i'\}_{r+1}^n$ dominates $\{x_i\}_{r+1}^n$, so $x_i \leq z_i$ for all $r + 1 \leq i \leq n$. Finally, we have that $z_i \in [k - 1, k + 1] = [x_i - 1, x_i + 1]$ for all $q \leq i \leq r$.

We wish to show that $\{x_i''\}$ majorizes $\{z_i\}$. To do so, first we note that since $|[q : q']| = |[r' : r]|$, $\sum_{i=1}^n x_i'' = \sum_{i=1}^n x_i$, and we already know that $\sum_{i=1}^n x_i = \sum_{i=1}^n x_i'$. Therefore, $\sum_{i=1}^n x_i'' = \sum_{i=1}^n z_i$.

It suffices to prove that for all $1 \leq s \leq n - 1$, we have $\sum_{i=1}^s x_i'' \geq \sum_{i=1}^s z_i$. If $s < q$, then

$$\sum_{i=1}^s x_i'' = \sum_{i=1}^s x_i \geq \sum_{i=1}^s z_i.$$

If $s \geq r$, then

$$\sum_{i=s+1}^{n} x_i'' = \sum_{i=s+1}^{n} x_i \leq \sum_{i=s+1}^{n} z_i,$$

so $\sum_{i=1}^{s} x_i'' \geq \sum_{i=1}^{s} z_i$. If $q \leq s \leq q'$, then

$$\sum_{i=1}^{s} x_i'' = \sum_{i=1}^{q-1} x_i + \sum_{i=q}^{s} (x_i + 2) \geq \sum_{i=1}^{q-1} z_i + \sum_{i=q}^{s} z_i = \sum_{i=1}^{s} z_i.$$

If $r' - 1 \leq s \leq r - 1$, then

$$\sum_{i=s+1}^{n} x_i'' = \sum_{i=s+1}^{r} x_i'' + \sum_{i=r+1}^{n} x_i'' = \sum_{i=s+1}^{r} (x_i - 2) + \sum_{i=r+1}^{n} x_i \leq \sum_{i=s+1}^{r} z_i \sum_{i=r+1}^{n} z_i = \sum_{i=s+1}^{n} z_i,$$

so $\sum_{i=1}^{s} x_i'' \geq \sum_{i=1}^{s} x_i'$. Since either $r' - 1 = q'$ or $r' - 1 = q' + 1$, we have covered all cases of s. Therefore, $\{x_i''\}$ majorizes $\{z_i\}$. $\qquad\square$

6.3 Finishing the Proof

In this section, we prove the main upper bound by combining our results on the ℓ-variant cup game from Subsection 6.1 with our results on majorization and domination from Subsection 6.2 that allow us to relate the optimal backlog in the ℓ-variant cup game to the optimal backlog in the negative-fill variable-processor cup game.

While it is well-known that the standard game is harder than the negative-fill game for the filler, it turns out that against a greedy emptier, the converse is also (asymptotically) true. Namely, we have the following lemma, the proof of which is deferred to Appendix A:

Lemma 6.16. *For any $t \geq 1$, suppose the filler can guarantee that the backlog exceeds some $b \geq 0$ after t rounds of the standard game, where the emptier is promised to be greedy. Then, the filler can guarantee that the backlog exceeds $b/2$ after t rounds of the negative-fill game, where again the emptier is promised to be greedy.*

Since the emptier is assumed to be greedy, from now on we may assume that the filler and emptier are competing in the negative-fill cup game. Thus, to prove Theorem 6.1, it suffices to instead prove the following theorem about the negative-fill game.

Theorem 6.17. *Let $1 \leq k \leq O(n)$. Then, assuming the emptier follows a greedy emptying strategy in the negative-fill variable-processor cup game, the filler needs $\Omega\left(k + \frac{k^3}{\log^2(n/k)}\right)$ rounds to achieve backlog k.*

Further note that the filler trivially needs at least k rounds to achieve backlog k. Thus our task reduces to proving a lower bound of $\Omega\left(\frac{k^3}{\log^2(n/k)}\right)$ on the number of rounds that the filler needs.

We now perform a series of arguments to establish Theorem 6.17.

Relating the stone-variant to the negative-fill game. Let $\{y^{(0)}\}, \{y^{(1)}\}, \{y^{(2)}\}$ represent sequences of fills where $\{y^{(i)}\}$ is the (sorted) sequence of fills after i rounds of some negative-fill cup game. In other words, the fills start out as $\{y^{(0)}\} = \{0, 0, \ldots, 0\}$. At time step $i+1$, the filler converts

the sequence $\{y^{(i)}\} = \{y_1^{(i)}, \ldots, y_n^{(i)}\}$, where $y_1^{(i)} \geq y_2^{(i)} \geq \cdots \geq y_n^{(i)}$, to $\{z^{(i)}\} = \{z_1^{(i)}, \ldots, z_n^{(i)}\}$, where $z_1^{(i)} \geq z_2^{(i)} \geq \cdots \geq z_n^{(i)}$, by adding a total of p_i fill, and finally, the emptier removes 1 from each of the first p_i cups and sorts the cups to obtain $\{y^{(i+1)}\}$. We claim that we can simulate a stone-variant cup game that starts with $\{x^{(0)}\} = \{0, 0, \ldots, 0\}$, but for each i, if the cups after i time steps have fills $\{x^{(i)}\} = \{x_1^{(i)}, \ldots, x_n^{(i)}\}$, then $2 \cdot \{x^{(i)}\}$ majorizes $\{y^{(i)}\}$, and $\{x^{(i)}\}$ is entirely integer-valued. We prove this by induction on i.

Clearly, for $i = 0$, we have $2 \cdot \{x^{(i)}\} = (0, 0, \ldots, 0) = \{y^{(i)}\}$, so $2 \cdot \{x^{(0)}\}$ majorizes $\{y^{(0)}\}$. Now, suppose that we have simulated the stone-variant cup game so that $2 \cdot \{x^{(i)}\}$ majorizes $\{y^{(i)}\}$ and $x_1^{(i)}, \ldots, x_n^{(i)}$ are all integers. Then, by Lemma 6.13, for any $\{y^{(i+1)}\}$ achievable from a round of the negative-fill cup game on $\{y^{(i)}\}$, there exists a sequence of fills on $2 \cdot \{x^{(i)}\}$ that, after greedy emptying, produces some $\{x'\}$ which majorizes $\{y^{(i+1)}\}$. Moreover, by Lemma 6.15, since all terms in $2 \cdot \{x^{(i)}\}$ are even integers, there exists a round of the stone-variant cup game (scaled by 2) applied to $2 \cdot \{x^{(i)}\}$ which majorizes $\{x'\}$. Therefore, there exists a round of the stone-variant cup game applied to $\{x^{(i)}\}$ to produce some $\{x^{(i+1)}\}$ such that $2 \cdot \{x^{(i+1)}\}$ majorizes $\{y^{(i+1)}\}$. This completes the induction.

The above argument, along with the fact that $2 \cdot \{x^{(i+1)}\}$ majorizing $\{y^{(i+1)}\}$ implies that $2 \cdot x_1^{(i+1)} \geq y_1^{(i+1)}$, directly implies the following lemma.

Lemma 6.18. *Suppose that after T time steps of the negative-fill cup game against a greedy emptier, it is possible to have a fill of k or more in some cup. Then, after T time steps of the stone-variant cup game, it is possible to have a fill of $k/2$ or more in some cup.*

Relating the ℓ-variant to the stone-variant. Now, we compare the stone-variant cup game to the ℓ-variant cup game for a suitable integer $\ell \geq 1$. Specifically, for any sequence $\{x^{(i)}\}$ of the stone-variant cup game obtainable in i time steps, and any $\ell \geq 1$, we show there is a corresponding ℓ-variant cup game that produces a sequence $\{w^{(i)}\}$ in i time steps, where $\{w^{(i)}\}$ dominates $\{x^{(i)}\}$. Again, we prove this by induction, where the base case of $i = 0$ is clear since $\{w^{(i)}\} = (0, 0, \ldots, 0) = \{x^{(i)}\}$. Now, if time step i of the stone-variant cup game converts $\{x^{(i)}\}$ to $\{x^{(i+1)}\}$, then by Lemma 6.14, there exists a time step of the stone-variant cup game that converts $\{w^{(i)}\}$ to some $\{w'\}$ that dominates $\{x^{(i+1)}\}$. Therefore, if we did the same time step but in the ℓ-variant cup game, each fill would either remain the same or go up by an additional 1 to ensure that checkpoints are preserved, so we would reach a state $\{w^{(i+1)}\}$ that dominates $\{x^{(i+1)}\}$. This completes the induction, and along with Lemma 6.18, directly implies the following lemma.

Lemma 6.19. *Suppose that after T time steps of the negative-fill cup game against a greedy emptier, it is possible to have a fill of k or more in some cup. Then, after T time steps of the ℓ-variant cup game, it is possible to have a fill of $k/2$ or more in some cup.*

To finish the proof of Theorem 6.17, first suppose that $k \geq 16 \log n$. Let $k' = k/8$, $h = \lceil \log \left(\frac{n}{k'} + 1 \right) \rceil$, and k'' be the largest multiple of h that is at most k'. Note that $k' \geq 2 \log n$ but $h \leq \lceil \log n \rceil < k'$, so $k'' = \Theta(k)$. Also, we have that $h = \Theta \left(\log \left(\frac{n}{k''} + 1 \right) \right)$, that $h \geq 2$, and that $h | k''$. Set $\ell = k''/h$. By Lemma 6.19, if the filler in the negative-fill cup game can achieve a fill of k after T time steps (against a greedy emptier), then it is also possible to achieve a fill of $k/2 \geq 4k''$ after T time steps in the ℓ-variant cup game. We can therefore apply Theorem 6.6 to deduce that $T \geq \Omega((k'')^3 / h^2) = \Omega \left(\frac{k^3}{\log(n/k)^2} \right)$. Finally, since the backlog cannot increase by more than 1 in a step, $T \geq k$, even if $k < 16 \log n$. This completes the proof of Theorem 6.17.

7 Analyzing Greedy with Resource Augmentation

In this section, we analyze the variable-processor cup game with ε resource augmentation, meaning that in each step, the emptier is permitted to remove up to $1 + \varepsilon$ units of water from each of p cups (rather than just 1 unit of water). Of course, so that the resource augmentation does not cause the average fill of cups to become negative, the emptier is prohibited from reducing the fill of any cup below 0.

We prove that even a very small amount of resource augmentation significantly decreases backlog of the game:

Theorem 7.1. *In the variable-processor cup game with resource augmentation $\varepsilon \in [\omega(1/n), 1]$, the greedy emptying algorithm achieves backlog $O(\varepsilon^{-1} \log n)$.*

We prove the theorem using the probabilistic method. Rather than analyzing the greedy emptying algorithm directly, we instead construct a randomized emptying algorithm that, at any given moment, achieves backlog $O(\varepsilon^{-1} \log n)$ with non-zero probability. (Importantly, the randomized algorithm is against an *adaptive* filler, not an oblivious one.) The existence of such a randomized algorithm implies the existence of a deterministic emptying algorithm with the same guarantee. But by Corollary 4.5, we already know that the greedy emptying algorithm is the optimal deterministic emptying algorithm, and thus it must also achieve backlog $O(\varepsilon^{-1} \log n)$.

Suppose $\omega(1/n) \le \varepsilon \le 1/2$. Throughout the rest of the section, we make an important WLOG assumption about the filler's behavior: we will assume that the filler always places at least $1/n^2$ units of water into each cup on each step. To see why this assumption is WLOG, up to constant factor changes in ε and in the backlog, let us describe how we can obtain the assumption as the composition of two more obviously WLOG assumptions. We can assume WLOG that the filler is only able to place $(1-\varepsilon)p$ units of water into cups in a given step (rather than p units of water), with at most $(1 - \varepsilon)$ units of water going to any given cup (rather than 1 unit of water). In particular, this is equivalent to re-normalizing what 1 unit of water corresponds to by a factor of $(1 - \varepsilon)$, and then changing ε by a constant factor. We can then further modify the filler to place an additional $\varepsilon p/n$ units of water into each cup (on top of the water that the filler was originally going to place); this modification is also WLOG since it only helps the filler. The new filler places at most p units of water into the system at each step, with at most 1 unit of water going to any given cup, and with at least $\varepsilon p/n \ge 1/n^2$ units of water going to each cup.

The assumption that the filler always places at least $1/n^2$ units of water into each cup on each step makes it so that a very simple randomized emptying algorithm achieves small backlog, even in arbitrarily long games. The randomized emptying algorithm takes an approach that we call **proportional emptying**: in each step of the game, if the filler places some amount q_j of water into cup j, then the emptier empties from cup j with probability exactly q_j. (If the algorithm empties from a cup with less than $1 + \varepsilon$ water, then the amount of water in the cup becomes 0).

Note that, if we did not have our WLOG assumption about the filler, then one problem with proportional emptying would be that, in very long games, the filler could use the following strategy to achieve large backlog. The filler simply waits until they get lucky and have a cup j with some large backlog, and then the filler never places any more water into that cup j. This means that the emptier will never again empty from cup j, and the backlog will stay large forever. This is why, in the analysis of proportional emptying, we need the assumption that the filler always places at least $1/n^2$ units of water into each cup on each step; of course, if the assumption weren't there, we could

just modify proportional emptying to simulate the assumption (since the assumption is WLOG), but it is cleaner to just assume it.

The following lemma shows that it is possible to implement proportional emptying on any given step.

Lemma 7.2. *Let $p \in \mathbb{N}$ and let $q_1, q_2, \ldots, q_n \in [0, 1]$ satisfy $\sum_j q_j = p$. It is possible to select distinct $x_1, x_2, \ldots, x_p \in [n]$ in a random fashion such that for each $j \in [n]$,*

$$\Pr[j \in \{x_i\}] = q_j.$$

Proof. Partition the real interval $I = [0, p)$ into half-open intervals $Q_1, Q_2, \ldots, Q_n$ where $|Q_j| = q_j$ for all j. Select $u_1 \in [0, p)$ uniformly at random, and define $u_2 = u_1 + 1, u_3 = u_1 + 2, \ldots, u_n = u_1 + n - 1$, where each u_i is taken modulo p. Finally, define x_i to be the index j such that $u_i \in Q_j$.

Since each interval Q_j has size 1 or smaller, it can contain at most one u_i. For each $i \in [n]$, the value u_i is uniformly random in $[0, p)$, and thus $\Pr[u_i \in Q_j] = q_j/p$. Since the events $\{u_i \in Q_j\}_{i=1}^p$ are disjoint, it follows that

$$\Pr[u_i \in Q_j \text{ for any } i \in [p]] = \sum_{i=1}^{p} \Pr[u_i \in Q_j] = q_j.$$

Thus $\Pr[j \in \{x_i\}] = q_j$ and the lemma is proven. $\qquad\square$

To analyze proportional emptying, we show that, at any given moment, each cup has fill $O(\varepsilon^{-1} \log n)$ with high probability. Roughly speaking, the amount of water in each cup can then be modeled as a biased random walk: in each step, the expected amount of water that the emptier removes from the cup is a factor of $1 + \varepsilon$ larger than the amount of water that the filler inserts. The bias in the random walk prevents it from reaching a large fill at any given moment, and as we shall now see, a simple Chernoff-style analysis (modified to handle the fact that the filler is an adaptive adversary) can be used to bound the fill by $O(\varepsilon^{-1} \log n)$.

In our analysis of proportional emptying, we will need the following standard Chernoff-style bound, which can be viewed as a variant of Azuma's martingale inequality:

Proposition 7.3. *Fix $\mu, t \in \mathbb{N}$. Suppose that Alice constructs a sequence of random variables $X_1, \ldots, X_t$, with $X_i \in \{0, 1\}$, using the following iterative process. Once the outcomes of $X_1, \ldots, X_{i-1}$ are determined, Alice then selects a probability r_i and draws X_i from $\{0, 1\}$ with $\Pr[X_i = 1] = r_i$. Alice is an adaptive adversary, meaning that r_i can be a function of $X_1, \ldots, X_{i-1}$. The only constraint on Alice is that $\sum_i r_i \geq \mu$.*

If $X = \sum_i X_i$, then for any $j > 0$,

$$\Pr[X \leq \mu - j] \leq \exp\left(j^2/(2\mu)\right).$$

If Alice were an oblivious adversary, unable to see the outcomes of the X_is, then Proposition 7.3 would follow by a standard lower-tail Chernoff bound. Fortunately, Chernoff bounds still hold against an adaptive adversary like Alice. For a proof of Proposition 7.3, see [29]; Corollary 11 of [29] gives the upper-tail version of Proposition 7.3 and by modifying the argument in the standard way to get a lower-tail bound, one arrives at Proposition 7.3.

We complete the section by analyzing proportional emptying.

Lemma 7.4. *Suppose $\varepsilon \in (\omega(1/n), 1]$. If the emptier uses proportional emptying, and if the filler always places at least $1/n^2$ units of water into each cup on each step, then at any time t, the backlog is $O(\varepsilon^{-1} \log n)$ with high probability in n.*

Proof. We will show that, for any given cup j and time t_0, with high probability in n, the amount of water in cup j is at most $O(\varepsilon^{-1} \log n)$. By taking a union bound over $j \in [n]$, it follows that the backlog at time t_0 is $O(\varepsilon^{-1} \log n)$ with high probability in n.

Since the cups are defined symmetrically, let us just analyze the amount of water in cup 1 over time. Let c_i be the amount of water that the filler places into cup 1 in step i. Recall that the filler is an adaptive adversary, so c_i may be determined partially based on the emptier's behavior in the first $i - 1$ steps.

Let r be the largest $r \leq t_0$ such that cup 1 is empty at the end of step r, and define $t = t_0 - r$. Let μ be the amount of water that the filler has placed into cup 1 during steps $r + 1, \ldots, t_0$, and let X be the number of times that the emptier has emptied from cup 1 during those same steps. The amount of water in cup 1 after step t is exactly

$$\mu - (1 + \varepsilon)X.$$

To complete the analysis, it suffices to show that

$$\Pr\left[X \leq \frac{1}{1 + \varepsilon} \cdot \lfloor \mu \rfloor - \gamma \varepsilon^{-1} \log n\right] \leq \frac{1}{\text{poly}(n)}, \tag{3}$$

where γ is taken to be a sufficiently large constant.

For any integer $\mu \geq 1$ and any $1 \leq t \leq t_0$, if the end of round $r = t_0 - t$ is the last time that the fill of cup 1 is 0, and the integer part (floor) of the amount of water placed into cup 1 during steps $r + 1, \ldots, t_0$ equals μ, define $X_{\mu,t}$ to be the number of times that the emptier has removed water from cup 1 during steps $t_0, t + 1, \ldots, t$. Otherwise, define $X_{\mu,t}$ to be ∞. To bound (3), it suffices to bound the probability that there exists any t and any integer $\mu \geq 1$ such that $X_{\mu,t} \leq \frac{1}{1+\varepsilon}\mu - \gamma \varepsilon^{-1} \log n$. For any given μ, t, we have by Proposition 7.3 that

$$\Pr\left[X_{\mu,t} \leq \frac{1}{1 + \varepsilon}\mu - \gamma \varepsilon^{-1} \log n\right] \leq \exp\left(-\left(\frac{\varepsilon\mu + \gamma \varepsilon^{-1} \log n}{\sqrt{2\mu}}\right)^2\right).$$

In addition to assuming that $\mu \geq 1$ and t are integers, we can also assume that $\mu \geq t/n^2$ since, by assumption, each step places at least $1/n^2$ water into cup 1.

Taking a union bound over all possible $\mu \geq 1$ and $t \geq 0$ satisfying $\mu \geq t/n^2$, we have

$$\Pr[X \leq \frac{1}{1+\varepsilon}\mu - \gamma\varepsilon^{-1}\log n]$$

$$\leq \sum_{t=1}^{n} \sum_{\mu=\lceil t/n^2 \rceil}^{tn} \Pr[X_{\mu,t} \leq \frac{1}{1+\varepsilon}\mu - \gamma\varepsilon^{-1}\log n]$$

$$\leq \sum_{\mu=1}^{\infty} \sum_{t=1}^{n^2\mu} \Pr[X_{\mu,t} \leq \frac{1}{1+\varepsilon}\mu - \gamma\varepsilon^{-1}\log n]$$

$$= \sum_{\mu=1}^{\infty} n^2\mu \Pr[X_{\mu,t} \leq \frac{1}{1+\varepsilon}\mu - \gamma\varepsilon^{-1}\log n]$$

$$\leq \sum_{\mu=1}^{\infty} n^2\mu \exp\left(-\left(\frac{\varepsilon\mu + \gamma\varepsilon^{-1}\log n}{\sqrt{2\mu}}\right)^2\right)$$

$$\leq \sum_{\mu=1}^{\gamma\varepsilon^{-2}\log n} n^2\mu \exp\left(-\left(\frac{\gamma\varepsilon^{-1}\log n}{\sqrt{2\mu}}\right)^2\right) + \sum_{\mu > \gamma\varepsilon^{-2}\log n} n^2\mu \exp\left(-\left(\frac{\varepsilon\mu}{\sqrt{2\mu}}\right)^2\right)$$

$$\leq \sum_{\mu=1}^{\gamma\varepsilon^{-2}\log n} n^2\mu \exp\left(-\frac{\gamma\log n}{2}\right) + \sum_{\mu > \gamma\varepsilon^{-2}\log n} n^2\mu \exp\left(-\varepsilon^2\mu/2\right)$$

$$\leq \frac{1}{\text{poly}(n)},$$

where the final inequality uses that γ is a sufficiently large constant. $\square$

Proof of Theorem 7.1. Let c be a sufficiently large constant. Consider the two-player game of length t where the filler is declared to win if he achieves backlog at least $c\varepsilon^{-1}\log n$ and the emptier wins otherwise. This is a perfect-information game with no draws, so one of the players must be able to deterministically force a win. Lemma 7.4 (along with the fact that the filler WLOG places at least $1/n^2$ water into each cup on each step) tells us that, if c is sufficiently large, then the filler cannot always force a win. Therefore the emptier *can* always force a win, and there is a deterministic emptying algorithm that achieves backlog $c\varepsilon^{-1}\log n$. By Corollary 4.5, it follows that greedy emptying also achieves backlog $c\varepsilon^{-1}\log n$, completing the proof. $\square$

Interestingly, the argument above can also be used to obtain a nontrivial bound in the resource-augmentation-free setting. Now the amount of water in each cup follows an *unbiased* random walk. At any given step t, one can bound the height of such random walk by $O(\sqrt{t\log n})$ with high probability. Using the fact that greedy emptying is as good as any randomized emptying strategy, it follows that greedy emptying achieves backlog $O(\sqrt{t\log n})$ in a t step game.

Acknowledgments

References

[1] Micah Adler, Petra Berenbrink, Tom Friedetzky, Leslie Ann Goldberg, Paul Goldberg, and Mike Paterson. A proportionate fair scheduling rule with good worst-case performance. In *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 101–108, 2003. URL: `http://doi.acm.org/10.1145/777412.777430`, `doi:10.1145/777412.777430`.

[2] Amihood Amir, Martin Farach, Ramana M. Idury, Johannes A. La Poutré, and Alejandro A. Schäffer. Improved dynamic dictionary matching. *Inf. Comput.*, 119(2):258–282, 1995. `doi:10.1006/inco.1995.1090`.

[3] Amihood Amir, Gianni Franceschini, Roberto Grossi, Tsvi Kopelowitz, Moshe Lewenstein, and Noa Lewenstein. Managing unbounded-length keys in comparison-driven data structures with applications to online indexing. *SIAM Journal on Computing*, 43(4):1396–1416, 2014.

[4] Yossi Azar and Arik Litichevskey. Maximizing throughput in multi-queue switches. *Algorithmica*, 45(1):69–90, 2006.

[5] Amotz Bar-Noy, Ari Freund, Shimon Landa, and Joseph (Seffi) Naor. Competitive on-line switching policies. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 525–534, 2002. URL: `http://dl.acm.org/citation.cfm?id=545381.545452`.

[6] Amotz Bar-Noy, Aviv Nisgav, and Boaz Patt-Shamir. Nearly optimal perfectly periodic schedules. *Distributed Computing*, 15(4):207–220, 2002.

[7] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel. Proportionate progress: A notion of fairness in resource allocation. *Algorithmica*, 15(6):600–625, Jun 1996. `doi:10.1007/BF01940883`.

[8] Sanjoy K Baruah, Johannes E Gehrke, and C Greg Plaxton. Fast scheduling of periodic tasks on multiple resources. In *Proceedings of the 9th International Parallel Processing Symposium*, pages 280–288, 1995.

[9] Michael Bender, Rathish Das, Martín Farach-Colton, Rob Johnson, and William Kuszmaul. Flushing without cascades. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2020.

[10] Michael Bender, Martín Farach-Colton, and William Kuszmaul. Achieving optimal backlog in multi-processor cup games. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, 2019.

[11] Michael A Bender, Rezaul A Chowdhury, Rathish Das, Rob Johnson, William Kuszmaul, Andrea Lincoln, Quanquan C Liu, Jayson Lynch, and Helen Xu. Closing the gap between cache-oblivious and cache-adaptive analysis. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 63–73, 2020.

[12] Michael A Bender, Rezaul A Chowdhury, Rathish Das, Rob Johnson, William Kuszmaul, Andrea Lincoln, Quanquan C Liu, Jayson Lynch, and Helen Xu. Closing the gap between cache-oblivious and cache-adaptive analysis. In *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 63–73, 2020.

[13] Michael A Bender, Roozbeh Ebrahimi, Jeremy T Fineman, Golnaz Ghasemiesfeh, Rob Johnson, and Samuel McCauley. Cache-adaptive algorithms. In *Proceedings of the twenty-fifth annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 958–971. SIAM, 2014.

[14] Michael A Bender and William Kuszmaul. Randomized cup game algorithms against strong adversaries. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2059–2077. SIAM, 2021.

[15] Peter Damaschke and Zhen Zhou. On queuing lengths in on-line switching. *Theoretical computer science*, 339(2-3):333–343, 2005.

[16] Paul Dietz and Daniel Sleator. Two algorithms for maintaining order in a list. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing (STOC)*, pages 365–372, 1987. URL: `http://doi.acm.org/10.1145/28395.28434`, `doi:10.1145/28395.28434`.

[17] Paul F. Dietz and Rajeev Raman. Persistence, amortization and randomization. In *Proceedings of the Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 78–88, 1991. URL: `http://dl.acm.org/citation.cfm?id=127787.127809`.

[18] Johannes Fischer and Paweł Gawrychowski. Alphabet-dependent string searching with wexponential search trees. In *Annual Symposium on Combinatorial Pattern Matching (CPM)*, pages 160–171, 2015.

[19] Rudolf Fleischer and Hisashi Koga. Balanced scheduling toward loss-free packet queuing and delay fairness. *Algorithmica*, 38(2):363–376, Feb 2004. `doi:10.1007/s00453-003-1064-z`.

[20] H Richard Gail, G Grover, Roch Guérin, Sidney L Hantler, Zvi Rosberg, and Moshe Sidi. Buffer size requirements under longest queue first. *Performance Evaluation*, 18(2):133–140, 1993.

[21] Leszek Gasieniec, Ralf Klasing, Christos Levcopoulos, Andrzej Lingas, Jie Min, and Tomasz Radzik. Bamboo garden trimming problem (perpetual maintenance of machines with different attendance urgency factors). In *International Conference on Current Trends in Theory and Practice of Informatics*, pages 229–240. Springer, 2017.

[22] Michael H. Goldwasser. A survey of buffer management policies for packet switches. *SIGACT News*, 41(1):100–128, 2010. URL: `http://doi.acm.org/10.1145/1753171.1753195`, `doi:10.1145/1753171.1753195`.

[23] Michael T Goodrich and Paweł Pszona. Streamed graph drawing and the file maintenance problem. In *International Symposium on Graph Drawing*, pages 256–267. Springer, 2013.

[24] Nan Guan and Wang Yi. Fixed-priority multiprocessor scheduling: Critical instant, response time and utilization bound. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*, pages 2470–2473. IEEE, 2012.

[25] Sungjin Im, Benjamin Moseley, and Rudy Zhou. The matroid cup game. *Operations Research Letters*, 49(3):405–411, 2021.

[26] Tsvi Kopelowitz. On-line indexing for general alphabets via predecessor queries on subsets of an ordered list. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 283–292, 2012.

[27] William Kuszmaul. Achieving optimal backlog in the vanilla multi-processor cup game. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2020.

[28] William Kuszmaul. How asymmetry helps buffer management: achieving optimal tail size in cup games. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1248–1261, 2021.

[29] William Kuszmaul and Qi Qi. The multiplicative version of azuma's inequality, with an application to contention analysis. *arXiv preprint arXiv:2102.05077*, 2021.

[30] William Kuszmaul and Alek Westover. The variable-processor cup game. In *12th Innovations in Theoretical Computer Science Conference (ITCS)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[31] Andrea Lincoln, Quanquan C Liu, Jayson Lynch, and Helen Xu. Cache-adaptive exploration: Experimental results and scan-hiding for adaptivity. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 213–222, 2018.

[32] Ami Litman and Shiri Moran-Schein. On distributed smooth scheduling. In *Proceedings of the Seventeenth Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 76–85, 2005.

[33] Ami Litman and Shiri Moran-Schein. Smooth scheduling under variable rates or the analog-digital confinement game. *Theor. Comp. Sys.*, 45(2):325–354, June 2009. URL: `http://dx.doi.org/10.1007/s00224-008-9134-x`, `doi:10.1007/s00224-008-9134-x`.

[34] Ami Litman and Shiri Moran-Schein. On centralized smooth scheduling. *Algorithmica*, 60(2):464–480, 2011.

[35] Chung Laung Liu. Scheduling algorithms for multiprocessors in a hard real-time environment. *JPL Space Programs Summary, 1969*, 1969.

[36] Chung Laung Liu and James W Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)*, 20(1):46–61, 1973.

[37] Richard T Mills, Andreas Stathopoulos, and Dimitrios S Nikolopoulos. Adapting to memory pressure from within scientific applications on multiprogrammed cows. In *Proc. 8th International Parallel and Distributed Processing Symposium (IPDPS)*, page 71, 2004.

[38] Mark Moir and Srikanth Ramamurthy. Pfair scheduling of fixed and migrating periodic tasks on multiple resources. In *Proceedings of the 20th IEEE Real-Time Systems Symposium*, pages 294–303, 1999.

[39] Christian Worm Mortensen. Fully-dynamic two dimensional orthogonal range and line segment intersection reporting in logarithmic time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 618–627, 2003.

[40] Michael Rosenblum, Michel X Goemans, and Vahid Tarokh. Universal bounds on buffer size for packetizing fluid policies in input queued, crossbar switches. In *Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 2, pages 1126–1134, 2004.

A Asymptotic Equivalence between Negative-Fill and Standard Game

We recall that in the negative-fill cup game, we allow cups to have arbitrary real number (possibly negative) fills, whereas in the standard variable-processor cup game, if an emptier removes water from a cup of fill between 0 and 1, the fill goes down to 0 rather than a negative number. In both games, if the cups have fills $x_1, \ldots, x_n$ the goal of the filler is to maximize the backlog, where the backlog is $\max_{1 \leq i \leq n} x_i$ as opposed to $\max_{1 \leq i \leq n} |x_i|$, and the goal of the emptier is to minimize the backlog.

It is easy to see that the negative-fill cup game is harder for the filler than in the standard variable-processor cup game. That is, if the emptier can guarantee the backlog does not exceed some $b \geq 0$ after t rounds of the standard variable-processor game, then they can make the same guarantee in the negative-fill game (by simply simulating it to be as though they never empty any cup below 0).

In this section, we show the converse is also true, up to a factor of 2 for backlog. First, we show the converse is true if we are promised the emptier is greedy. We note that this is all that we need for our main theorems, though we will also show a corollary that removes the greedy-emptier assumption.

First, we establish the following proposition.

Proposition A.1. *For any integers $n, t \geq 1$ and real number $b \geq 0$, suppose the filler is playing the negative-fill variable-processor cup game over n cups against a greedy emptier, for t rounds. Then, the following are equivalent:*

1. *The filler can guarantee at least one cup has fill $\geq b$.*

2. *The filler can guarantee at least one cup has fill $\leq -b$.*

3. *The filler can either guarantee that at least one cup has fill $\geq b$ or at least one cup has fill $\leq -b$ (but does not have to know which one).*

Proof. We first establish the equivalence of the first two items. Since the emptier's strategy is fixed, the filler can know exactly what the state is after each round. Suppose that the filler can play a strategy where after the ith round, the state of the cups is $X^{(i)} = (x_1^{(i)}, \ldots, x_n^{(i)})$. Note that $X^{(0)} = (0, 0, \ldots, 0)$. Then, we claim that the filler is also capable of ensuring that after the ith round, the state of the cups is $-X^{(i)} = (-x_1^{(i)}, \ldots, -x_n^{(i)})$.

To do so, suppose the filler can go from $X^{(i)}$ to $X^{(i+1)}$ by adding $a_j^{(i)}$ to the jth cup and then having the emptier remove 1 unit from the $p = \sum_{j=1}^n a_j^{(i)}$ fullest cups. Then, to go from $-X^{(i)}$ to $-X^{(i+1)}$, the filler will add $1 - a_j^{(i)}$ to the jth cup, so that the jth cup has fill $1 - (X_j^{(i)} + a_j^{(i)})$. Then, $\sum_{j=1}^n (1 - a_j^{(i)}) = n - p$, so the emptier will subtract 1 from the $n - p$ largest values of $1 - (X_j^{(i)} + a_j^{(i)})$. This is equivalent to subtracting 1 from all of the values $1 - (X_j^{(i)} + a_j^{(i)})$ to get $-(X_j^{(i)} + a_j^{(i)})$, and then adding 1 to the p smallest values of $-(X_j^{(i)} + a_j^{(i)})$. Overall, we will precisely get $-X_j^{(i+1)}$, since the p smallest values of $-(X_j^{(i)} + a_j^{(i)})$ are just the negatives of the p largest values of $X_j^{(i)} + a_j^{(i)}$. Since we only care about cups up to permutations, we do not have to worry about tiebreaking issues.

This establishes that the first two items are equivalent, since if after t rounds the filler can guarantee the fills are $(X_1^{(t)}, \ldots, X_n^{(t)})$ with $\max_{1 \leq j \leq n} X_j^{(t)} \geq b$, then the filler can guarantee the fills are $(-X_1^{(t)}, \ldots, -X_n^{(t)})$, where $\min_{1 \leq j \leq n} -X_j^{(t)} \leq -b$, and vice versa.

Finally, to see why the third item is equivalent, it is straightforward that either the first two items implies the third item. But since the filler is really playing a 1-player game as the emptier's strategy is known beforehand, if the filler can guarantee one of the two events can occur (but doesn't know which one), then the filler can either guarantee the first event will occur or can guarantee that the second event occurs. But since these two are equivalent, this establishes the equivalence between all three items. $\square$

Now, we are able to establish the main lemma of this section, assuming the emptier is greedy.

Lemma A.2. *For any $n, t \geq 1$, suppose the filler can guarantee that the backlog exceeds some $b \geq 0$ after t rounds of the standard variable-processor game over n cups, where the emptier is promised to be greedy. Then, the filler can guarantee that the backlog exceeds $b/2$ after t rounds of the negative-fill game, where again the emptier is promised to be greedy.*

Proof. Since adding fill always benefits the filler, we know that if we were to start in a state where all cups have fills $b/2 > 0$, then the filler would still be able to guarantee that the backlog exceeds b after t rounds in the standard variable-processor game against a greedy emptier. In order for a

cup to reach fill b starting from this state, either some cup must reach 0 fill at or before time step t, or some cup must reach fill b without any cup reaching 0 fill during the t time steps.

The former case means that the same filling procedure on the negative-fill game starting with all cups with fill 0 will result in some cup having fill less than or equal to $-b/2$ at some point during the first t time steps. If this takes fewer than t time steps, then the filler can preserve the state of the cups for the remaining steps (by always setting $p = n$ and putting 1 unit of water into every cup on each step) so that after t time steps, the minimum fill is still less than or equal to $-b/2$. The latter case, on the other hand, means that in the negative-fill cup game starting with all cups with fill 0 will result in some cup having fill $b/2$ at t time steps. Therefore, the filler can guarantee that either the minimum fill is $-b/2$ after t rounds, or the maximum fill is $b/2$ after t rounds. Thus, by Proposition A.1, the filler can guarantee a backlog of $b/2$ after t rounds in the negative-fill cup game. □

By noting the fact that the greedy emptier is optimal in both the negative-fill and standard variable-processor cup games, we have the following corollary.

Corollary A.3. *For any $n, t \geq 1$, suppose the filler can guarantee that the backlog exceeds some $b \geq 0$ after t rounds of the standard variable-processor game over n cups, against any emptier. Then, the filler can guarantee that the backlog exceeds $b/2$ after t rounds of the negative-fill game, against any emptier.*

B   Propositions on Majorization and Domination

In this section, we prove Propositions 6.8, 6.9, and 6.11.

Proof of Proposition 6.8. We wish to show that for any $1 \leq t \leq k + r$, the sum of the largest t elements of $\{a_1, \ldots, a_k, c_1, \ldots, c_r\}$ is at least the sum of the largest t elements of $\{b_1, \ldots, b_k, c_1, \ldots, c_r\}$. To prove this, assume WLOG that $a_1 \geq \cdots \geq a_k$, $b_1 \geq \cdots \geq b_k$, and $c_1 \geq \cdots \geq c_r$. Suppose the largest t elements of $\{b_1, \ldots, b_k, c_1, \ldots, c_r\}$ are $b_1, \ldots, b_g, c_1, \ldots, c_{r-g}$, where $0 \leq g \leq t$. Then, since $\{a_1, \ldots, a_k\}$ majorizes $\{b_1, \ldots, b_k\}$, we have that $a_1 + \cdots + a_g \geq b_1 + \cdots + b_g$, so $a_1 + \cdots + b_g + c_1 + \cdots + c_{r-g} \geq b_1 + \cdots + b_g + c_1 + \cdots + c_{r-g}$. So, the sum of the largest t elements of $\{a_1, \ldots, a_k, c_1, \ldots, c_r\}$ must be at least as large as $b_1 + \cdots + b_g + c_1 + \cdots + c_{r-g}$. □

Proof of Proposition 6.9. We additionally show that in each step, that after each perturbation, none of the terms will be x_1 or less than x_n. We prove this by induction on n, the length of the sequences $\{x_i\}$ and $\{y_i\}$. If $n = 1$, then we must have $x_1 = y_1$ for majorization, so the proof is trivial.

For $n \geq 2$, assume WLOG that $\{x_i\}$ and $\{y_i\}$ are sorted in nondecreasing order. Then, if $\{x_i\}$ dominates $\{y_i\}$, $x_1 \geq y_1 \geq y_n \geq x_n$. If $x_1 = y_1$, then we use the inductive hypothesis on $\{x_2, \ldots, x_n\}$ and $\{y_2, \ldots, y_n\}$, and keep $x_1 = y_1$ as is. Since each perturbation does not increase anything to greater than $x_2 \leq x_1$ or decrease anything to less than x_n, we have that the order of the full sequence (including the first term which remains at y_1) is preserved and no term will ever increase to greater than x_1 or less than x_n. A symmetric argument takes care of the case when $x_n = y_n$, where we use the inductive hypothesis on $\{x_1, \ldots, x_{n-1}\}$ and $\{y_1, \ldots, y_{n-1}\}$.

Otherwise, let $\alpha = \min(x_1 - y_1, y_n - x_n)$, and let $\varepsilon = \frac{\alpha}{1 + \lfloor \alpha \rfloor}$. For each of $1 + \lfloor \alpha \rfloor$ steps, increase y_1 by ε and decrease y_n by ε. Since $y_1 \geq y_2 \geq \cdots \geq y_n$ at the beginning, we clearly preserve this at each step, and since $x_1 \geq y_1$ and $x_n \geq y_n$ at the beginning, we also clearly preserve this

throughout. Finally, we have either increased y_1 to x_1 or decreased y_n to x_n, so we can apply the previous paragraph. $\square$

Proof of Proposition 6.11. Suppose the contrary, so there exists $1 \leq j \leq n$ such that jth largest element of x is less than the jth largest element of y. Then, if $x^{(i)}$ represents the ith largest element of $\{x_i\}$ and $y^{(i)}$ represents the ith largest element of $\{y_i\}$, then $y^{(1)} \geq \cdots \geq y^{(j)} > x^{(j)} \geq \cdots \geq x^{(n)}$. Note, however, that $y^{(1)}, \ldots, y^{(j)}$ comprises j numbers and $x^{(j)}, \ldots, x^{(n)}$ comprises $n-j+1$ numbers. Therefore, by the Pigeonhole principle, there exists $1 \leq k \leq j$ and $j \leq \ell \leq n$ such that $y^{(k)} = y_m$ and $x^{(\ell)} = x_m$ for some $m \in [n]$. But then this implies that $y_m > x_m$, which is a contradiction. $\square$