TIGHT LOWER BOUNDS FOR THE ONLINE LABELING PROBLEM[*]

JAN BULÁNEK[†], MICHAL KOUCKÝ[‡], AND MICHAEL SAKS[§]

Abstract. We consider the *file maintenance problem* (also called the *online labeling problem*) in which n integer items from the set $\{1, \ldots, r\}$ are to be stored in an array of size $m \geq n$. The items are presented sequentially in an arbitrary order, and must be stored in the array in sorted order (but not necessarily in consecutive locations in the array). Each new item must be stored in the array before the next item is received. If $r \leq m$ then we can simply store item j in location j but if $r > m$ then we may have to shift the location of stored items to make space for a newly arrived item. The algorithm is charged each time an item is stored in the array, or moved to a new location. The goal is to minimize the total number of moves the algorithm has to do. This problem is nontrivial for $n \leq m < r$.

In the case that $m = Cn$ for some $C > 1$, algorithms are known that solve the problem with cost $O(n \log^2(n))$ (independent of r)[15, 19, 3]. For the case $m = n$, algorithms with cost $O(n \log^3(n))$ were given [20, 7]. In this paper we prove lower bounds that show that these algorithms are optimal, up to constant factors. Previously, a lower bound of $\Omega(n \log^2(n))$ was known for the restricted class of *smooth* algorithms [11, 20].

1. Introduction.

1.1. The file maintenance problem.

In the *file maintenance problem* n integer items from the set $\{1, \ldots, r\}$ are to be stored in an array of size $m \geq n$. The items are presented sequentially in an arbitrary order, and must be stored in the array in sorted order (but not necessarily in consecutive locations). Each new item must be stored in the array before the next item is received. If $r \leq m$ then we can simply store item j in location j but if $r > m$ then we may have to shift the location of stored items to make space for a newly arrived item. The algorithm is charged each time an item is stored in the array, or moved to a new location. The goal is to minimize the total number of such moves. This problem is nontrivial when $n \leq m < r$.

An alternate formulation is the *online labeling problem* in which arriving items must be assigned an integer label in the range $[1, m]$ so that the order on the labels agrees with the numerical ordering on the items. The algorithm pays one each time an item is labeled or relabeled. Typically in the literature the file maintenance problem refers to the *small space regime* in which $m = O(n)$. This case is the focus of this paper.

Itai et al. [15] were the first to design an algorithm that maintains an array of size $m = O(n)$ making only $O(n \log^2(n))$ moves in total, i.e., the amortized cost per item is $O(\log^2(n))$ moves. Willard [19] improved this to an algorithm with worst-case cost $O(\log^2(n))$ moves per item, and his result was simplified by Bender et al. [3]. Dietz [10] showed that for array size $m = n^{1+\varepsilon}$, $\varepsilon > 0$ constant the problem can be solved

[*]A preliminary version of this paper appeared in the Proceedings of the ACM Symposium on Theory of Computing, 2012.

[†]Department of Theoretical Computer Science and Mathematical Logic, Charles University, Prague. (jan.bulanek@gmail.com). Partially supported by GAUK project no. 344711 and grant P202/10/0854 of GA ČR.

[‡]Computer Science Institute of Charles University, Prague. (koucky@iuuk.mff.cuni.cz). Part of the work was done while visiting the University of Toronto, partially supported by NSERC, and while visiting Aarhus University, partially supported by the Sino-Danish Center CTIC. Partially supported by GA ČR P202/10/0854. The research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement n. 616787.

[§]Department of Mathematics, Rutgers University. (saks@math.rutgers.edu). Supported in part by NSF under grants CCF-0832787 and CCF-1218711, and by a sabbatical at Princeton Univesity.

with $O(n \log(n))$ moves. For the case that the array size m is exactly the number of items n, Zhang [20] gave an algorithm that achieves a (surprising) amortized upper bound of $O(\log^3(n))$ moves per item; this result was simplified in [7].

In recent years there has been renewed interest in this problem because of its applications in the design of cache-oblivious algorithms, e.g., design of cache-oblivious B-trees [4, 8] and cache-oblivious dynamic dictionaries [5].

1.2. Our results. In this paper we prove an $\Omega(n \log^2(n))$ lower bound on the number of moves for inserting n items into an array of size $m = O(n)$ for any online labeling algorithm, matching the known upper bound up to constant factors. For the case of array size $m \leq n + n^{1-\varepsilon}$ (where ε is a positive constant) we prove the asymptotically optimal lower bound $\Omega(n \log^3(n))$.

The above-mentioned upper bounds work for any input domain size (arbitrary r). As noted above, if $r \leq m$ then there is a trivial solution of cost n. A natural question (which has not, to our knowledge, been addressed previously in the literature) is whether it might be possible to improve the $O(n \log^2(n))$ upper bound if r is much larger than m but still restricted. Our lower bounds rule out such an improvement: we obtain an $\Omega(n \log^2(n))$ lower bound provided that r is at least a sufficiently large constant times m.

Our lower bounds extend to slightly superlinear array size. For example, in the case $m = O(n \log^{1-\varepsilon}(n))$ our results give an $\Omega(n \log^{1+\varepsilon/3}(n))$ lower bound (provided that the range size r is large enough).

Our bound is the first lower bound for general algorithms in the small space regime. Previously Dietz et al. [11, 20] proved a $\Omega(n \log^2(n))$ lower bound for the restricted case of so-called *smooth algorithms*.

Recently, Emek and Korman [14] showed that online labeling could be reduced to the Distributed Controller problem (introduced in [1]), in which nodes in an asynchronous distributed network receive requests from outside the network for units of a limited resource, and issue usage permits in response to the requests. The number of permits issued may not exceed the total resource supply, and the protocol must also ensure that no request is declined until the number of permits committed exceeds a $1 - \varepsilon$ fraction of the supply. Protocols with message complexity $O(n \log^2(n))$ on n-node networks are known (e.g. [1, 17]), and Emek and Korman noted that, using their reduction, a matching $\Omega(n \log^2(n))$ lower bound would follow from an $\Omega(n \log^2(n))$ lower bound on the online labeling problem. Our paper provides this lower bound so the known protocols are asymptotically optimal.

1.3. Previous and subsequent lower bounds. Lower bounds for this problem appeared previously in a paper of Dietz and Zhang [13] and papers of Dietz, Seiferas and Zhang [11, 12]. The first two ([13, 11], also available in Zhang's Ph.D. thesis [20]) prove lower bounds for a restricted class of algorithms, called *smooth algorithms*, which are limited to redistributing items in a uniform fashion. For smooth algorithms, they proved an $\Omega(n \log^2(n))$ lower bound for the small space regime (linear array size). The restriction to smooth algorithms is significant since Bender and Hu [6] give a non-smooth algorithm which performs asymptotically better on common input distribution while keeping the worst case time guarantees in amortized settings. The lower bound for the small space regime of smooth algorithms is obtained by considering the trivial adversary that inserts items in decreasing order. This lower bound clearly relies heavily on the smoothness of the algorithm; a non-smooth algorithm can easily handle the given adversary with constant amortized time per item. It was not known whether a non-smooth algorithm could give a significant advantage over a smooth algorithm on general inputs. Our lower bound rules this out.

There is some confusion in the literature about these lower bounds; the fact that they apply only to smooth algorithms is sometimes not mentioned (e.g., [7]), creating the impression that the general lower bound result was already established.

The present paper deals only with linear array size and does not address the polynomial array size regime where $m = n^C$ for some constant $C > 1$. For this range of space there is an $\Omega(n \log(n))$ upper bound, and Dietz et al. [12] proved a lower bound that matches this asymptotically. In recent work (using techniques quite different from the present paper), we (together with M. Babka and V. Čunát) clarified and simplified the bounds from [12] and extended their range of applicability to superpolynomial array size. In subsequent work, we showed that the $\Omega(n \log(n))$ lower bound applies in the polynomial array size regime even if randomization is allowed [9].

While there are some common ideas, the present paper on linear array size is quite separate from the work on polynomial array size. In particular, the main technical issues that need to be resolved in the papers are different (though it is certainly possible that someone may find a way to unify them).

The main open problem left open by this paper is whether our $\Omega(n \log^2(n))$ lower bound for linear array size can be extended to apply to randomized algorithms.

1.4. Proof techniques. We will describe our results in the language of the file maintenance problem, in which arriving integer items are placed in an array, rather than the online labeling problem. Our main lemma (Lemma 2.3) gives a lower bound on the cost of inserting n additional items into an array that is already partially full with $n_0 \geq n$ items. The $\Omega(n \log^2(n))$ lower bound for file maintenance for the case that $m = \Theta(n)$ is obtained by applying this lemma to bound the cost of inserting the second half of the items given that the first half of the items are initially in the array. The $\Omega(\log^3(n))$ lower bound for an array of size $m \leq n + n^{1-\varepsilon}$ is obtained by iterative application of the main lemma $\Theta(\log n)$ times where we initially start with half of the items in the array, and in each iteration we insert half of the remaining items. The main lemma shows that each iteration costs $\Theta(n \log^2(n))$. This parallels the structure of the iterative algorithm of [20] that gave a matching upper bound.

The general idea for proving the lemma (which builds heavily on the above-mentioned prior work [11, 20, 12]) is to build an adversary that forces the maintenance algorithm to repeatedly move many items. To do this, at each step the adversary identifies a densely populated (*crowded*) segment (consecutive sequence of locations) of the array and inserts a new item whose value lies between two items already stored in this crowded segment. Repeated insertions eventually force the algorithm to move many items.

Deriving a lower bound based on this idea has various complications. The natural measure of crowding of a segment is the ratio of stored items to the size of the segment. Whether a particular portion of the array is viewed as crowded may depend on the scale of segments being considered; there may be a relatively small segment that is very crowded, but larger segments containing it are uncrowded. To force the algorithm to work hard, we want to identify a segment that is crowded at many different scales. This suggests identifying a long nested sequence of segments (a *segment chain*) covering a wide range of scales, such that each is crowded. The hope is that inserting many items having value in the middle of the range of items stored in the smallest nested segment will eventually force the algorithm to do costly rearrangements at many different scales.

A straightforward way to select such segment chain is to start with the entire array, and successively select a subsegment having the highest density among subsegments of, say, half the size of the current segment. This results in a chain of increasing density, but this alone does not seem to be enough to give a good lower bound. The difficulty is that each successive selected subsegments may be chosen near the boundary of the predecessor segment. In such a situation, the algorithm may be able to relieve overcrowding by relatively inexpensive rearrangements that cross the boundary of many segments in the sequence into uncrowded segments.

To prevent the algorithm from "escaping" in this way, the adversary would like to select each subsegment in the sequence so that it has a significant buffer to its left and right within the predecessor segment, where each buffer contains a constant fraction of the items in the predecessor segment. The presence of such buffers will ensure that as a segment gets crowded, all of the items in either its left or right buffer will have to be moved.

However, when we insist on having these buffers we can no longer ensure that the density of the segments in the sequence do not decrease (because a given segment in the sequence may have its items concentrated near its boundary). So we have to allow some decrease in the segment density along the sequence.

Dietz et al. [12] showed how to construct segment chains in which each successive segment has a large left and right buffer. The drawback of their construction is that the density of successive segments in the sequence may decrease by as much as a constant factor, so that for a sequence of logarithmic length the density may decrease by a fraction $n^{\Omega(1)}$. This limits the quality of lower bounds that can be proved.

The key challenge we address is to give an adversary procedure for building segment chains that ensures large buffers, while at the same time guaranteeing that the density degrades very slowly. This is the main new contribution in our proof. Our approach begins with the observation that if for a given segment every subsegment having large buffers has density significantly smaller than the given segment, then there must be a large subsegment (located near the boundary of the given segment) having substantially higher density than the given segment. This allows us to build a chain of $\Theta(\log(n))$ segments, such that (1) a constant fraction of the segments have large buffers with respect to their predecessors, (2) the segments that don't have large buffers have significantly higher density than their predecessor segments, and (3) the degradation of density along the entire chain can be bounded by a constant factor. (To give a rough idea of the choice of parameters, when $m = \Theta(n)$, we allow decrease in density by a factor of at most $(1 - O(1/\log(n)))$ in a single step.)

The adversary we describe doesn't work exactly like this, but instead builds a chain of segments such that every segment in the chain (not just a constant fraction) has left and right buffers whose sizes are a constant fraction of the length of the segment. We do this by relaxing the requirement that the length of each segment in the chain is at least a constant fraction of the length of its predecessor. If we encounter a segment whose items are concentrated near the boundary then the next segment will be a small subsegment of high enough density whose distance from the boundary is large relative to its own size, even if this distance is small relative to the size of the predecessor segment. This raises a new problem: once we allow successive subsegments to shrink by more than a constant fraction we face the problem that the length of the segment chain d may not be $\Omega(\log(n))$. This is important because the lower bound that comes out of the analysis is proportional to d^2. So we need to ensure that the chains have length $\Omega(\log(n))$ even though we allow the length of segments to drop significantly. This is accomplished by a procedure (see Lemma 7.2) that allows us to construct a sequence of segments where each selected segment satisfies a strong uniformity property called *lower balance*: It has no subsegment of length at least $1/4$ of its length that has density significantly smaller than the given segment. (In the lower bound for smooth algorithms mentioned earlier, the ability to find such a sequence is essentially built into the smoothness restriction. Our construction allows us to dispense with this assumption.) To find the successor segment S' of a given lower balanced segment S we first restrict to the middle third T of the segment and choose S' to be a subsegment of T. The restriction to a subsegment of T ensures that S' has large buffers relative to S. Furthermore, the uniformity property of S ensures that the density of T is close to that of S. We want to choose S' inside T having density at least that of T, having size not much smaller than T, and having the desired uniformity property. To identify S' we maximize a certain quality function of the form $\rho(I)|I|^{\kappa}$ (where $\rho(I)$ is the density of items stored in I and κ is a small positive parameter). This balances the requirement that S' have high density and large size. Furthermore, choosing S' in this way guarantees that S' has the needed uniformity property (since the presence of a subsegment that violates the uniformity property would imply that there is a subsegment of S' that has a higher quality). Maximizing the quality function implicitly captures the process of successively choosing subsegments of significantly higher density until one arrives at a subsegment for which no such selection is possible.

After identifying a segment chain with the required properties at each step, the item selected by the adversary to insert is one whose value is between two items stored in the final segment of the chain. Whenever the maintenance algorithm rearranges some portion of the array the adversary rebuilds the affected portion of the segment chain. To obtain our lower bound $\Omega(n \log^2(n))$ we use a careful accounting argument (see Lemma 5.1) that encapsulates and extends the clever argument used in [11, 20] to obtain an $\Omega(n \log^2(n))$ bound for smooth algorithms.

There is one additional complication that arises because we want our lower bounds to apply even in the case that the range r of items is relatively small. In a given step, after selecting the chain, the adversary is supposed to choose the next item to insert to be an item that is between two items currently stored in the final segment of the chain. However, if the set of items stored in the final segment are a consecutive subset of the set of possible values, the adversary is unable to choose an item to insert. Of course this is not a problem if the range of values is, for example, all rationals in a given interval but it is a potential problem if the range is a bounded subset of the integers. For example, we noted earlier that if the range of possible items is small enough, $r \leq m$, then there is a trivial algorithm that incurs only unit cost per item, so our lower bound proof must fail. How large does r have to be so that the adversary described above can avoid this problem? It is not hard to show that $r \geq 2^n$ is sufficient, but in fact when $m = O(n)$ we only need r to be a (sufficiently large) constant multiple of n. To carry out the argument for such small r, we modify the definition of density of segments by weighting more recent items with a smaller (but still non-negligible) weight than older items. When our adversary selects a segment chain, the decreased weight on recent items will tilt the adversary to prefer segments that are crowded mainly with older items over segments crowded mainly with newer items (unless the latter is significantly more crowded than the former). The reason we want to do this is that if the adversary continues to place items in a segment that mainly has newer items there is a risk (because of the limited range size r) that the adversary will end up with a segment where the items are consecutive integers, and not have another item to insert. By giving more recent items a smaller (but not too small) weight we can avoid this possibility; see Lemma 4.1.

2. The model and main results.

2.1. A two player game. In this paper, interval notation is used for sets of consecutive integers, e.g., $[a, b]$ is the set $\{k \in \mathbb{Z} : a \leq k \leq b\}$. We consider an array with cells indexed by the set $[1, m]$ in which we store a set Y of integer-valued *items.* A *storage function* for Y is a map $f : Y \longrightarrow [1, m]$ that is strictly order preserving, i.e., for $x, y \in Y$ if $x < y$ then $f(x) < f(y)$. In particular f is one-to-one, so $|Y| \leq m$. Cells that are in the image of f are *occupied* and the others are *unoccupied.* A *configuration* is a pair (Y, f) where Y is a set of items and f is a storage function for Y.

To formalize the analysis of the file maintenance problem we define a game $G^n(m, r)$, where n, m, r are positive integer parameters, which is played by two players, the *adversary* and the *algorithm.* The game is played in a sequence of n steps. At step t, the *adversary* selects an item y^t from the set $\{1, \ldots, r\} - \{y^1, \ldots, y^{t-1}\}$, and the algorithm responds with a storage function f^t for the set $Y^t = \{y^1, \ldots, y^t\}$. We say that item y^t is *inserted* at step t. (Y^t, f^t) is called the *configuration at the end of step t* and also *the configuration at the beginning of step $t + 1$.*

An item y is *moved at step t* if $f^t(y) \neq f^{t-1}(y)$. In particular y^t is moved at step t. The set of moved items at step t is denoted $Move^t$. The *cost up to step t* is $\chi^t = \sum_{i=1}^{t} |Move^i|$. Clearly $\chi^t \geq t$ for every t. The objective of the algorithm is to minimize χ^n and the objective of the adversary is to maximize χ^n. We write $\chi^n(m, r)$ for the smallest cost that can be achieved by the algorithm against the best adversary.

$G^n(m, r)$ is not well defined if $n > m$ since there can be no storage function once the number of items exceeds the number of cells. Also, if $m \geq r$, there is a trivial algorithm that achieves optimal cost n by storing each item $y \in [r]$ in cell y. We therefore assume $n \leq m < r$.

2.2. The main results. In this section, we state our lower bound results for $\chi^n(m, r)$. We divide our results into two theorems, corresponding to the relation between the array size and the number of items. In formulating the theorems, we use N for the number of items rather, to avoid confusion with the parameter n appearing in the main lemma (Lemma 2.3 below). which is used in the proof of the theorems, and in which n stands for the number of items inserted during a portion of the game.

The first theorem applies whenever $2N \leq m$, and gives interesting results provided that m is not too large (slightly superlinear function of N). In the first part of the theorem, the range $[1, r]$ of possible items has size exponential in N. In the second part r is at most a constant times m. Despite this strong limitation, the lower bound is only slightly worse.

Theorem 2.1. *There is a (sufficiently large) constant C_1 so that the following holds. Let m, N be integers satisfying $C_1 \leq N$ and $2N \leq m$. Let $\delta = N/m$. Then*

1. *If $r \geq N2^{N-1}$ then $\chi^N(m, r) \geq N \log^2(N) \frac{\delta}{C_1(\log(1/\delta))^2}$.*
2. *If $r \geq C_1 m$ then $\chi^N(m, r) \geq N \log^2(N) \frac{\delta^2}{C_1(\log(1/\delta))^2}$.*

Here and elsewhere in the paper, the base of logarithms is 2.

In both parts, if $m = O(N)$ so that $\delta = O(1)$ then the lower bound obtained is $\Omega(N \log^2(N))$. The messy dependence on δ tells how our bound degrades in the case of $\delta = o(1)$, i.e., the array is much larger than the number of items. The first bound gives a nontrivial $\omega(n)$ bound for m up to $o(n \log^2(N)/ \log^2(\log(N)))$, while the second bound is nontrivial for $m = o(N \log(N)/ \log\log(N))$.

In the next result we consider array size satisfying $N < m < 2N$:

THEOREM 2.2. *There are (sufficiently large) constants C_2, C_3 so that the following holds. Let m, N be integers satisfying $C_3 \leq N < m < 2N$ and let $\delta = N/m$. Assume $r \geq (\frac{1}{1-\delta})^{C_2} N$. Then:*

$$\chi^N(m,r) \geq \frac{1}{C_3} N \log^2(N) \log\left(\frac{1}{1-\delta}\right). \tag{2.1}$$

In this theorem, the array size is assumed to be at most twice the number of items. As long as $m \geq n(1+c)$ for a fixed $c > 0$ we again get an $\Omega(N \log^2(N))$ bound. As $m - n$ gets smaller, the bound improves. In particular For $m \leq N + N^{1-\varepsilon}$ this gives a tight lower bound of $\Omega(N \log^3(N))$. For this lower bound we only need the range of items to be polynomial in m. (We believe that it is possible to refine the analysis to obtain an asymptotically similar lower bound when the range size is only $N + N^{1-O(\varepsilon)}$ but have not included this analysis so as not to lengthen an already lengthy paper.)

2.3. Reduction of the theorems to the main lemma. As the game has been defined, every cell is initially unoccupied. For the proofs of the main theorems, it will be convenient to consider a generalization of the game, in which the array is initially partially full. This version of the game is specified by the parameters n, m (but not r) and additionally takes a set Y^0 of items, whose size is denoted by n_0 and is required to be at least 2. The subset Y^0 is given to the algorithm who selects the initial storage function f^0 (at no cost). The game then proceeds as before, except that the adversary is restricted to inserting items in the range $(\min(Y^0), \max(Y^0))$ (where we assume $|Y^0| \geq 2$). Note that the parameter r does not appear in this formulation.

We denote the game by $G^n(m|Y^0)$ and write $\chi^n(m|Y^0)$ for the minimum cost that can be achieved by the algorithm against the best adversary. We assume that $m \geq n_0 + n$, otherwise there is not enough room to insert all of the items.

For a set Y of items, we define $\text{mingap}(Y)$ to be the minimum absolute difference between pairs of items in Y. The central result of this paper is:

LEMMA 2.3. *(*The Main Lemma*) There are positive constants C_0, C_4 so that the following holds. Let m, n, n_0 be integers and let $\delta_0 = n_0/m$ where:*

$$C_0 \leq n \leq n_0 \tag{2.2}$$
$$n + n_0 \leq m \tag{2.3}$$
$$\delta_0 \in (\log(n)^{-2}, 1 - n^{-1/5}). \tag{2.4}$$

Let Y^0 be any set of n_0 items and let $\mu_0 = \text{mingap}(Y^0)$.

1. *If $\mu_0 \geq 2^n$ then*

$$\chi^n(m|Y^0) \geq n(\log(n))^2 \frac{\delta_0(1-\delta_0)}{C_4 \log^2(1/\delta_0)}.$$

2. *If $\mu_0 \geq 1 + 12/\delta_0$, then*

$$\chi^n(m|Y^0) \geq n(\log(n))^2 \frac{\delta_0^2(1-\delta_0)}{C_4 \log^2(1/\delta_0)}.$$

We point out that the condition $n_0 \geq n$ means that the new items being added at most doubles the number of items in the array.

We now prove Theorems 2.1 and 2.2 using Lemma 2.3. In the proof of the first theorem we apply the main lemma once with $\delta_0 = 1/2$, and in the proof of the second theorem we'll apply the main lemma multiple times, with δ_0 getting closer and closer to 1 (which is why the dependence on δ_0 in the lower bound is important.) In every application of the main lemma, the array size parameter m of the lemma will be the same as the array size parameter m in the theorem being proved, but the number of items n in the lemma will vary and won't be the same as the number N in the theorem being proved.

Proof of Theorem 2.1. In the argument below we choose C_1 for the theorem large enough depending on C_4 in Lemma 2.3.

Given N, m, r for the theorem, let $n_0 = \lceil N/2 \rceil$ and $n = N - n_0$. Let B be the largest integer such that $n_0 B \leq r$. Let $Y^0 = \{Bt : t \in [1, n_0]\}$. Note that $\text{mingap}(Y^0) = B$. Consider the adversary for $G^N(m, r)$ that during the first n_0 steps inserts Y^0 and then follows the optimal adversary strategy for the game $G^n(m|Y^0)$.

For the first part of Theorem 2.1, the hypothesis that $r \geq N2^{N-1}$ implies $\text{mingap}(Y_0) \geq 2^n$ so the first part of Lemma 2.3 applies. Under the hypothesis of Theorem 2.1, $1 - \delta_0 \geq 1/2$, and so the conclusion of Lemma 2.3 yields the conclusion of the first part of Theorem 2.1.

For the second part of Theorem 2.1, the hypothesis $r \geq C_1 m$ and our freedom to choose C_1 to be a sufficiently large constant imply $B \geq \lfloor C_1 m/n_0 \rfloor \geq C_1/\delta_0 - 1 \geq 12/\delta_0 + 1$ and so part (2) of Lemma 2.3 gives the desired lower bound. □

We next turn to the proof of Theorem 2.2. The idea of the proof is simple: The array size m is not much larger than N. We bound the cost as the sum of the costs of $p = \Theta(\log(1/(1-\delta)))$ "subgames" where the first subgame starts when the array is (roughly) half full and consists of inserting the next (roughly) $m/4$ items. After subgame $i-1$ the number of empty spaces in the array is (roughly) $m/2^i$ and in the ith subgame we insert (roughly) $m/2^{i+1}$ additional items. Corollary 2.4 below (deduced from Lemma 2.3) says that each phase has cost $\Omega(m(\log m)^2)$; even though the number of items inserted per phase is decreasing by a factor of 2, this is counterbalanced by the increased crowding of the array.

The following consequence of Lemma 2.3 is what we need to analyze a single subgame.

Corollary 2.4. *There are positive constants C_0, C_5 so that the following holds. Let m, n be integers satisfying $C_0 \leq (m/2)^{5/6} \leq n \leq m/3$. Let Y^0 be any set of $m - 2n$ items such that $\mu_0 = \text{mingap}(Y^0) \geq 37$. Then:*

$$\chi^n(m|Y^0) \geq \frac{1}{C_5} m \log^2(m).$$

Furthermore, there is an adversary that achieves this bound and has the additional property that the mingap of the items in the array after inserting the n items is at least $\lfloor \mu_0/37 \rfloor$.

Proof. For the first conclusion, we apply the second part of the main lemma with $n_0 = m - 2n$ and $\delta_0 = n_0/m = 1 - \frac{2n}{m}$. We need to check the hypotheses of the lemma. Hypothesis (2.2) and (2.3) are immediate. Since $n \leq m/3$ and $m \leq 2n^{6/5}$ we have $1/3 \leq \delta_0 \leq 1 - n^{-1/5}$ as required for (2.3), and also that $\mu_0 \geq 37$ is at least $1 + 12/\delta_0$. Therefore the hypotheses of part 2 of the lemma hold, and from the conclusion we get:

$$\begin{aligned}\chi^n(m|Y^0) &\geq n\log^2(n)\frac{\delta_0^2(1-\delta_0)}{C_4\log^2(1/\delta_0)}\\ &\geq \frac{m(1-\delta_0)}{2}\log^2((m/2)^{5/6})\frac{1}{4^2\cdot C_4}\frac{1-\delta_0}{\log^2(1/\delta_0)}\\ &\geq m\log^2(m)\cdot\frac{1}{C_5},\end{aligned}$$

where the final inequality uses the (numerical) fact that for $\delta_0 \in [1/4, 1]$, $\log(1/\delta_0) \leq 3(1-\delta_0)$, and the fact that C_5 can be chosen to be a large enough constant.

For the second conclusion, for ease of notation we use B to represent the number 37. We first consider the case that $r = Bn_0$ and Y_0 is the set $\{B, 2B, \ldots, n_0B\}$. Thus $\mu_0 = B$. By part 2 of Lemma 2.3 here is an adversary strategy Γ for inserting the next n items that forces the claimed lower bound on $\chi^n(m|Y^0)$. The final mingap is at least 1, as required.

Now consider the general case that r is arbitrary and Y_0 is a set of size n_0 with mingap at least B. Let $v_1 < \cdots < v_{n_0}$ be the items in Y_0. Let $G = \lfloor \mu_0/B \rfloor$. For $1 \leq j < n_0$, let V_j be the set of items of the form $v_j + iG$ where $1 \leq i \leq B-1$ and let $V = V_1 \cup \cdots \cup V_{n_0-1}$. The smallest item of V_j is $v_j + G$ and the largest is at most at most $v_{j+1} - G$. Thus the set V contains exactly $B-1$ items between each pair v_j, v_{j+1} and so $V \cup Y_0$ is combinatorially equivalent to the above case that Y_0 is the set $\{B, 2B, \ldots, n_0B\}$. By the obvious adaptation of the strategy for that case, we can carry out the adversary strategy while only inserting items from V, so that at the conclusion of the game, the set of inserted items is a subset of $Y_0 \cup V$, and the mingap of this set is at least $G = \lfloor \mu_0/B \rfloor$. □

We now fill in the (routine) details of the above sketch of the proof of Theorem 2.2.
Proof of Theorem 2.2. Let m, N and δ be given as in the theorem. Suppose first that $\delta = N/m$ is bounded above by $1-c$ for some positive constant c of our choice. Then Theorem 2.2 follows from the second part of Theorem 2.1, since the quantity $\delta/\log(1/\delta)$ appearing in the conclusion of Theorem 2.1(2) can be bounded below by a positive constant and the quantity $\log(1/(1-\delta))$ appearing in the conclusion of Theorem 2.2(1) can be bounded above by a positive constant.

We are left with the (main) case that $\delta > 1-c$ for a constant $c > 0$ of our choice; for convenience we assume $c \leq 1/16$ which implies, in particular that $\log(1/(1-\delta)) \geq 4$. Let $\Delta = m - N$ and let p be the largest integer such that $2^p\Delta \leq m/2$, so that $2^{p+1}\Delta > m/2$ and

$$p = \left\lfloor \log_2\left(\frac{m}{2\Delta}\right)\right\rfloor \geq \log\left(\frac{1}{1-\delta}\right) - 2 \geq \frac{1}{2}\log\left(\frac{1}{1-\delta}\right).$$

For the game of inserting N items into an empty array of size m, we identify p subgames where the subgames involve disjoint sets of steps. The first subgame starts after $z_0 = m - 2^p\Delta$ items have been inserted. Subgame 1 consists of inserting $n_1 = 2^{p-1}\Delta$ items. In general subgame i starts after $z_{i-1} = m - 2^{p-(i-1)}\Delta$ items have been inserted and consists of inserting the next $n_i = 2^{p-i}\Delta$ items. Note that after subgame p, all of the $N = m - \Delta$ items have been inserted.

We want to apply Corollary 2.4 to each of these subgames. To apply the Corollary to subgame i, the hypothesis of the corollary requires that $n_i \leq m/3$ and $n_i \geq$

$(m/2)^{5/6}$. The first condition holds since $n_i \leq n_1 \leq m/4$ (by the choice of p). The second condition holds provided that $i \leq \frac{p}{6} - 1$ since then $n_i \geq 2^{1+5p/6}\Delta \geq (2^{p+1}\Delta)^{5/6}\Delta^{1/6} \geq \left(\frac{m}{2}\right)^{5/6}$.

Thus for each of the first $\frac{p}{6} - 1$ subgames, the Corollary gives a cost lower bound of $\Theta(m \log^2(m)) = \Theta(N \log^2(N))$. The total cost can be bounded below by this times $\frac{p}{6} - 1 = \Omega(\log(1/(1-\delta)))$. $\square$

2.4. Organization of the rest of the paper. The remainder of the paper is devoted to proving Lemma 2.3. Throughout the rest of the paper, the input parameters to the lemma are fixed:

m	The array size.
Y^0	The set of initial items.
n_0	The size of Y^0.
μ_0	The mingap of Y^0.
δ_0	The initial density n_0/m.
n	The number of items to be inserted.

The rest of the paper is organized as follows.

- Section 3 gives some additional notation.
- Section 4 gives a full description of the adversary. The adversary is not too difficult to describe but the choices made may strike the reader as somewhat arbitrary. The subsequent discussion will (we hope) demystify the adversary.
- In section 5, we formulate seven (parameterized) properties of the adversary. We then state two main lemmas. Lemma 5.1 asserts that any adversary that satisfies these properties forces any algorithm to pay a high cost. Lemma 5.2 asserts that our adversary has these seven properties. We show how the main lemma follows easily from these two lemmas.
- In section 6 we prove Lemma 5.1, showing that an adversary satisfying these seven properties gives a good lower bound on any algorithm. We start with a sketch of the main idea of the proof and then follow with the full proof.
- In section 7 we establish that our adversary has these seven properties by proving Lemma 5.2. We begin the section with an informal discussion of how these properties led us to the chosen adversary.

Lemma 2.3 was formulated in sufficient generality (in terms of n, n_0, m and μ_0, so as to be able to prove both Theorems 2.2 and 2.1. These include both cases that the range of possible numbers is very large, or rather small and also very dense case where $\delta_0 = n_0/m$ is very close to 1. The reader may wish to focus on the following restrictions:

- $n_0 = n = \Theta(m)$. (We'll refer to this as the case of *small constant density.*)
- $r \geq 2^m$. (We'll refer to this as the case of *large initial mingap.*)

We refer to this restricted case as the *illustrative case.* This setting of parameters is enough to prove the first part of Theorem 2.2 in the case that $N = \Theta(m)$ and captures most of the difficulty of the proof. In organizing the proof, we considered first presenting the proof for this case, and only then doing the general proof, but decided against it because it would either require either duplicating much of the proof of the special case when doing the general proof, or leaving the reader to extrapolate the general proof based on the special case and a sketch of the differences. We'll provide some guideposts in the proof that will allow the reader to simplify details of the general proof in order to focus on the illustrative case.

3. Some notation and preliminaries for the proof of the main lemma.

We introduce some terminology:

- A *segment* is a subinterval of the set of cells $[1, m]$.
- A *step interval* is a subinterval of $[0, n]$ representing a sequence of consecutive steps of the game.
- An *item interval* is a subinterval of the set $[1, r]$ of items. If $Y \subseteq [1, r]$ is any set of items, a Y*-interval* is a set of the form $Y \cap I$ where I is an item interval.

Recall that at step t, $Move^t$ denotes the set of items moved at step t. For $y \in Move^t$ the *trail of* y *at step* t is the segment $Trail^t(y)$ between $f^{t-1}(y)$ and $f^t(y)$; for y^t it is just the location $f^t(y^t)$. The *busy region* at step t, denoted B^t is the union over $y \in Move^t$ of $Trail^t(y)$.

We say that an algorithm is *lazy* if B^t is a segment. The following proposition (assumed in Dietz et al. [12]) says that we may restrict attention to lazy algorithms.

Proposition 3.1. *Given any algorithm A there is a lazy algorithm A' such that for any initial item set Y^0 and any item sequence $y = (y^1, \ldots, y^n)$ the cost of A' on Y^0, y is at most the cost of A on Y^0, y.*

The idea is that if the busy region B^t is a union of two or more disconnected segments, then any relocation outside of the segment that contains $f^t(y^t)$ can be deferred until later. The following (straightforward) proof makes this precise.

Proof. The algorithm A' keeps track of the storage function g^t that would be produced by the algorithm A. Let f^t be the array actually produced by A'. (Keep in mind that since the algorithm only pays for actual moves, A' pays nothing for internally simulating A.)

Initially, prior to step 1, $f^0 = g^0$. At each step t, A' updates g^{t-1} to g^t based on algorithm A. The storage function f^t is then determined as follows.

Given an array segment T, we say that two storage functions h and h' are *similar on* T if the set of items stored by each inside T is the same (though the items need not be stored in the same locations under both.) Let j_L be the largest index such that g^t and f^{t-1} are similar on $[1, j_L]$ (where possibly $j_L = 0$ so that $[1, j_L]$ is empty). Let j_R be the least index such that g^t and f^{t-1} are similar on $[j_R, m]$ (where possibly $j_R = m + 1$). Note that we must have $j_L < g^t(y^t) < j_R$. Let $M = [j_L + 1, j_R - 1]$. and define f^t so that it agrees with g^t on M and agrees with f^{t-1} everywhere else.

We claim that the busy region B^t of A' is equal to the segment M. Clearly $B^t \subseteq M$, since A' moves no items outside of M. For the reverse inclusion, suppose for contradiction that $j \in M \setminus B^t$ and assume without loss of generality that $j_L < j < g^t(y^t)$. Note that the definition of f^t implies that f^t and g^t are similar on $[1, j]$. Note that the condition that $j \notin B^t$ implies that f^t and f^{t-1} are similar on $[1, j]$. But then g^t and f^{t-1} are similar on $[1, j]$ contradicting the definition of j_L.

Finally, we need to bound the cost of relocations by A' from above by the cost of relocations by A. For this, consider all relocations of a fixed item y. An easy induction shows that up through the end of any step t the number of steps that y was moved by A' is less than or equal to the number of steps that y was moved by A, with a strict inequality if $f^t(y) \neq g^t(y)$. □

What this proposition is that when we design an adversary, we may assume that the algorithm it plays against is lazy. If the algorithm A against which the adversary plays is not lazy, the adversary can conceptually simulate the lazy algorithm A' given by the proposition, and choose his insertions in response to the simulated A'. The proposition implies that the cost incurred by the actual algorithm A is at least the cost incurred by the virtual algorithm A'.

Therefore, for the rest of the paper we assume that the algorithm is lazy, and refer to B^t as the *busy segment at step t*. It is easy to see that for lazy algorithms B^t is the smallest segment that contains location $f^t(y^t)$ and the starting and ending location of all items that were moved at step t.

4. A description of the adversary for Lemma 2.3.

In this section we present our adversary strategy. The first subsection discusses the preliminary notion of a gap, which is a pair of items that have been inserted such that no item between them has been inserted yet. The second subsection describes the class of *segment chain strategies* which includes our adversary strategy. The third subsection specifies our strategy within this class.

4.1. Gaps and suitable gaps.

During each step t the adversary must choose an item y^t to insert into the array. For a set Y of items, a *Y-gap* is a pair $y_L < y_R$ of items belonging to Y such that no item of Y has value in the item interval (y_L, y_R). The *gap length* is $y_R - y_L$. We emphasize that a gap refers to the set of possible item values between y_L and y_R and not to the region of the array in which the items are stored.

Provided that a gap has length at least 2, there is always an item between y_L and y_R that is available to be inserted. We call such a gap *suitable*. A *suitable segment* is one that contains a suitable gap. The condition of being a suitable segment is equivalent to: the set of items currently stored in the segment is not a consecutive sequence of integers.

Our adversary will choose a suitable segment, identify the longest suitable gap (y_L, y_R) stored in the segment and select the item $\lfloor (y_L+y_R)/2 \rfloor$, which is the midpoint of the gap rounded down to the nearest integer. The segment (resp., gap) chosen by the adversary at step t is referred to as the *chosen segment (resp., gap) at step t*.

When the adversary selects the segment S, we must ensure that S contains a suitable gap. For $\text{mingap}(Y_0) \geq 2^n$ (as in the illustrative setting mentioned at the end of Section 2), an easy induction shows that $\text{mingap}(Y_t) \geq 2^{n-t}$ for every $t < n$. Therefore any segment S that contains at least two items is suitable. We refer to this case as *large initial mingap*. In the case of *small initial mingap*, $\text{mingap}(Y_0) < 2^n$, we need to be more careful to ensure that the selected segment is suitable. This will complicate things a bit, but not in any significant way. The reader may wish to focus on the case of large initial mingap; we will identify the places in the argument where the arguments diverge.

The choice of the suitable segment at step t will depend on the configuration (Y^{t-1}, f^{t-1}). Intuitively, the adversary will select a suitable segment that is currently located in an area of the array that is relatively "crowded". A natural notion of crowding for a segment S is the ratio of the number of items stored in the segment to the length of the segment. This notion of crowding is sufficient for the case of large initial mingap.

To handle both the case of large and small initial mingap, we need a notion of crowding that depends on a *weight parameter* $\lambda \in (0, 1]$. Each item in the array is assigned a weight which is 1 if the item belongs to the initial set Y^0 and is λ if it was inserted by the adversary. Given a configuration (Y, f), we define the following functions on segments $S \subseteq [1, m]$:

- The weight $w(S) = w(S, f)$ is the sum of the weights of all items stored in S under f.
- The density $\rho(S) = \rho(S, f)$ is $w(S)/|S|$. The density function provides a natural measure of crowding of S.

In the case of large initial mingap ($\mathrm{mingap}(Y_0) \geq 2^n$) we set the weight parameter λ to 1. Thus the weight of a segment is just the number of items stored in it and the density is the fraction of occupied cells. This is the only difference between this case and the case of small initial mingap.

The following lemma shows that by choosing λ appropriately we can get a sufficient condition for a segment to contain a suitable gap. Readers who wish to concentrate only on the case of large mingap can skip to the next subsection.

LEMMA 4.1. *(Suitable Gap Lemma) Suppose that $\lambda \in (0, 1/2)$. Let $S \subseteq [1, m]$ be a segment and Y_S be the set of integer stored in S with a specified set Y_S^0 of* initial *items. Define the weight $w(S)$ to be $|Y_S^0| + \lambda|Y_S \setminus Y_S^0|$ and its density $\rho(S)$ to be $w(S)/|S|$. If:*

1. *The mingap of Y_S^0 is at least $1 + \frac{2}{\lambda}$*
2. *$w(S) \geq 2$*
3. *$\rho(S) \geq 2\lambda$.*

then S is suitable, i.e. the set of items stored in S is not consecutive.

The key hypothesis is $\rho(S) \geq 2\lambda$. Since non-initial items are given weight only λ, an interval S of density at least 2λ must have a substantial fraction of initial items. Since the mingap of Y_S^0 is not too small, we will be able to show that there is a pair $x < y$ of items stored in S that form a gap in Y_S^0 that have fewer than $\mu(Y_S^0)$ items stored between them in S, which implies that there is at least one item between them that has not been inserted yet, so S is suitable.

Proof. Let $a = |Y_S^0|$ and $b = |Y_S \setminus Y_S^0|$. We first note that $a \geq b\lambda$. This follows from $w(S) = a + b\lambda = |S|\rho(S) \geq 2(a+b)\lambda$ so $(1-2\lambda)a > b\lambda$ which implies $a > b\lambda$. In particular, this implies $a \geq 2$, since a is an integer larger than $(a+b\lambda)/2 = w(S)/2 \geq 1$.

Let $\min_0$ and $\max_0$ be the smallest and largest items in Y_S^0. Suppose for contradiction that there is no suitable gap between $\min_0$ and $\max_0$. Then all of the $\max_0 - \min_0 + 1$ items in the range $[\min_0, \max_0]$ must have been inserted already. There are $a-1$ gaps between items of Y_S^0, each of size at least $\mu(Y_S^0)$ so by the hypothesis on λ, $b \geq (a-1)(\mu_0 - 1) \geq 2(a-1)/\lambda \geq a/\lambda$ (since $a \geq 2$) which contradicts $a > b\lambda$. $\square$

When we design our adversary strategy to prove Lemma 2.3 we will ensure that (for the case of small mingap) the segment selected by the algorithm satisfies the hypotheses of the suitable gap lemma. We haven't explained our strategy yet so we can't show this, but we sketch how this is done. We'll choose λ to be $\delta_0/6$. Then the hypothesis $\mu(Y^0) \geq 1 + 12/\delta_0$ from part 2 of Lemma 2.3 gives the needed lower bound on $\mu(Y^0)$ in this lemma. Also our strategy will always choose a segment of weight at least 2, and density at least $\delta_0/3 = 2\lambda$, so the hypotheses of the suitable gap lemma will be satisfied.

4.2. Segment chain strategies. This section describes a general class of adversary strategies from which we will select our adversary.

At each step t, our adversary will identify a segment satisfying the conditions of Lemma 4.1 (and other conditions as well.) The adversary will actually specify a chain of segments $S_1^t \supset S_2^t \supset \cdots \supset S_d^t$, where the parameter d will be fixed (later) to be $\Theta(\log(n))$. (Note that, in general the first segment S_1^t need not equal $[1, m]$.) The segment S_d^t will satisfy the conditions of Lemma 4.1 and will be used as the selected suitable segment at step t.

This type of strategy for the adversary, where we close in on a suitable segment by taking a chain of segments, is called a *segment chain strategy*. This general approach (though not the terminology) comes from Dietz et al. [12] who used it to prove a

$\Omega(n\log(n))$ lower bound in the case that m is at most polynomially larger than n. Their method of choosing this sequence can be applied to the case that m is at most linear in n but the lower bound does not improve. Our main contribution is to describe and analyze an alternative way to select the chain of segments which gives an $\Omega(n\log^2(n))$ lower bound.

It is natural that the segment chain selected at step t should relate to the sequence selected at the previous step $t-1$ and the moves made by the algorithm. Recall that B^t is the busy region at step t, which is determined by the moves made by the algorithm in response to y^t. Also recall that by Proposition 3.1 we assume that the algorithm is lazy so that B^t is always a segment. Define the *critical index* j^{t-1} after step $t-1$ to be the smallest index $j \in [1, d+1]$ for which B^{t-1} is not a subset of S_j^{t-1}. In particular, $j^{t-1} = d+1$ if B^{t-1} is a subset of S_d^{t-1}, and $j^0 = 1$. Our adversary will satisfy the following natural rule (which was also satisfied by the adversary of Dietz et al.):

Conservative Selection Rule. For every $t \geq 2$: The sequence $S_1^t, \ldots, S_d^t$ is chosen so that $S_j^t = S_j^{t-1}$ for $j < j^{t-1}$.

This rule puts no restriction on the selection of S_j^t for $j \geq j^{t-1}$; in particular the adversary can use any information about the present or past configurations.

Assuming the use of a segment chain strategy with conservative selection, we now summarize the sequence of events that occur during step t of the game. Note that the configuration (Y^{t-1}, f^{t-1}), the busy segment B^{t-1}, and the critical index j^{t-1} were determined during step $t-1$.

- The adversary selects the sequence $S_1^t \supset \cdots \supset S_d^t$. (We will specify how this is done below.) The selection will be done subject to the conservative selection rule and will be done in a way that ensures that S_d^t satisfies the hypothesis of Lemma 4.1 (with respect to (Y^{t-1}, f^{t-1})), and therefore contains a suitable gap.
- The adversary chooses the longest gap in S_d^t and lets y^t be the approximate midpoint. Y^t is set to be $Y^{t-1} \cup \{y^t\}$.
- The algorithm selects the storage function f^t for Y^t.
- The choice of f^t together with the previous storage function f^{t-1} determines the busy segment B^t.
- The critical index j^t is determined by B^t and $S_1^t, \ldots, S_d^t$.

4.3. Specifying the segment chain. It remains to describe how the adversary selects the segments S_j^t for $j \geq j^{t-1}$. Our strategy is not hard to describe but it is not so easy to motivate the (seemingly arbitrary) choices made. We'll present it first with a minimum of discussion. Hopefully the discussion in Section 7.1 and the proofs of correctness will demystify it.

Our strategy uses an auxiliary sequence of sets $T_1^t, \ldots, T_d^t$ which is interleaved with $S_1^t, \ldots, S_d^t$:

$$T_1^t \supset S_1^t \supset T_2^t \supset S_2^t \supset \cdots \supset T_d^t \supset S_d^t.$$

Recall that j^{t-1} is the critical index after step $t-1$. For $1 \leq j < j^{t-1}$, we satisfy the conservative selection rule by setting $T_j^t = T_j^{t-1}$ and $S_j^t = S_j^{t-1}$.

The significant part of the specification consists of three parts:

- The choice of $T_{j^{t-1}}^t$ for the critical level j^{t-1}.
- For $i \in [j^{t-1}, d]$, the choice of S_i^t given T_i^t.

- For $i \in [1 + j^{t-1}, d]$, the choice of T_i^t given S_{i-1}^t.

To specify these, we will need some definitions. For a segment U:

- **middle**(U) is the subsegment of U defined as follows: Break U into three segments from left to right, L, M, R where $|L| = |R| = \lfloor |U|/3 \rfloor$. **middle**$(U)$ is the segment M (which is roughly the middle third of U).

Let (Y, f) be an arbitrary configuration with associated density ρ. Let $\kappa > 0$. For an arbitrary segment U:

- The *quality of* U *with respect to* (Y, f) is the real number $\phi(U) = \phi_\kappa(U) = \rho(U)^{1/\kappa}|U|$.
- **densify**(U) is the subsegment V of U that maximizes $\phi(V)$ (breaking ties arbitrarily).
- **balance**(U) is the subsegment V of U given by **middle**(**densify**(U)).
- we write ρ^{t-1}, ϕ^{t-1}, **densify**$^{t-1}$ and **balance**$^{t-1}$ to be the functions based on the configuration (Y^{t-1}, f^{t-1})

Our adversary depends on three parameters: the length d of the segment chain will be fixed to be $\Theta(\log(n))$ by (5.4) and the parameter κ used in the quality function (which will be fixed to be $\Theta(1/\log(n))$ in (5.3)) and the weight parameter λ which appears implicitly because it determines the functions w^t,ρ^t, and ϕ^t and therefore also the functions **densify** and **balance** defined above. The parameter λ is 1 in the case of large mingap and will be set to $\delta_0/6$ otherwise. We'll explain these choices later but for now we leave d, κ and λ as parameters. Recall, $j^0 = 1$.

The specification of adversary $\textsc{Adv}(d, \kappa, \lambda)$

Adv1 Specification of T_j^t for the critical level j^{t-1}:

Adv1.a If $j^{t-1} = 1$ then $T_{j^{t-1}}^t$ is the segment of length between $n/2$ and n of highest density (breaking ties by an arbitrary rule).

Adv1.b If $j^{t-1} > 1$ (and so necessarily $t > 1$) set $T_{j^{t-1}}^t = T_{j^{t-1}}^{t-1} \cup B^{t-1}$.

Adv2 Specification of S_i^t given T_i^t for $i > j^{t-1}$: $S_i^t = \textbf{balance}^{t-1}(T_i^t)$.

Adv3 Specification of T_i^t given S_{i-1}^t for $i > j^{t-1}$: $T_i^t = \textbf{middle}(S_{i-1}^t)$.

It is not clear that the above adversary is well-defined, because we need that the final segment S_d^t in each segment chain be suitable as defined in Section 4.1; if it isn't then the adversary is unable to proceed with inserting an item according to the requirements. Indeed if d is chosen too large then the chain of segments will eventually degenerate to a segment of length 1 which will just be repeated until the chain ends. When we fix the parameters, one of the things we'll have to prove is that each of the segments S_d^t is indeed suitable (which is formulated as Property (P4) below).

Also, in our adversary we require that for all t and j, T_j^t and S_j^t are segments. This is not immediately apparent from the description of the adversary (because of Adv1.b) but can be proved by induction on t and for fixed t by induction on j. The only case that requires some discussion is the definition of T_j^t when j is critical and $j > 1$. By induction, T_j^{t-1} is a segment, and by laziness of the algorithm B^{t-1} is a segment that must intersect T_j^{t-1}, so their union is a segment.

Notice, in the case of Adv1.a, the whole segment $[1, m]$ can be partitioned into non-overlapping pieces each of size between $n/2$ and n. At least one of them must have density at least δ_0 so the segment T_1^t selected in Adv1.a has density at least δ_0.

5. Important properties of the adversary. The rest of this paper is devoted to proving that the above adversary, with suitably chosen parameters, gives the bounds of Lemma 2.3. In this subsection we state seven properties (P1)-(P7) that encapsulate what we need from our adversary. In subsequent sections we'll show that these properties imply a cost lower bound, and that our adversary satisfies these properties.

The first three properties are fairly mild technical "boundary" conditions on the sizes and densities of the segments appearing in a segment chain. We introduce parameters σ and δ^* to represent these conditions.

(P1) For each segment chain the first segment in the chain (and therefore all others) have size at most $n/2$.

(P2(σ)) For each segment chain, the final segment (and therefore all others) have size at least σ. (The reader should think of σ as a function of n that grows slowly, but not too slowly. Later, we'll choose it to be $n^{1/4}$, but the exponent $1/4$ is fairly arbitrary.)

(P3(δ^*)) Every segment in any of the segment chains has density at least δ^*. (Later we set $\delta^* = \delta_0 2^{\delta_0 - 1}$ which is between $\delta_0/2$ and δ_0. In the illustrative case in which $\delta_0 < 1/2$ the reader should think of δ^* as $c\delta^*$ for some constant c around $1/2$ where the constant is unimportant. (In the dense case that δ_0 is close to 1, the desired lower bound of $\Theta(n \log^2 n/(1-\delta_0))$ gets larger as δ_0 gets closer to 1. In this case it is not enough to take the δ^* to be a constant fraction of δ^0, instead it is roughly $(\delta_0)^2$.)

The next property is crucial since without it the adversary is not well-defined. However, it will be easy to satisfy.

(P4) For each t, S^t_d has a suitable gap. In the case of large mingap we only need $w^{t-1}(S^t_d) \geq 2$ (which will follow from (P2(σ)) and (P3(δ^*))), but for the case of small mingap, we'll need to make sure that S^t_d satisfies the hypotheses of Lemma 4.1 (which will not be hard to do.)

The next simple property plays a significant role in the lower bound.

(P5) For each t and $i \geq 2$, $|S^t_i| \leq |S^t_{i-1}|/2$. (Segment sizes decrease by at least a factor of 2 along a segment chain.)

The next property is especially important to the argument. As discussed in the introduction, when we choose the segment chain we would like that (among other properties) each successive segment should have density at least that of the previous segment. We can't necessarily do this but it's enough that the density not decrease by too much. This is quantified by the following property, which depends on a *density degradation parameter* α:

(P6(α)) $\rho^{t-1}(S^t_1) \geq \frac{1}{2^\alpha}\delta_0$ and for $i \geq 2$, $\rho^{t-1}(S^t_i) \geq \frac{1}{2^\alpha}\rho^{t-1}(S^t_{i-1})$.

The value of α we're able to achieve plays a crucial role in our lower bound. The heart of the lower bound argument (Lemma 5.1) will give a lower bound on $\chi^n(m|Y^0)$ of roughly $\Omega(nd^2/(\alpha d+1))$. We will have $d = \Theta(\log(n))$ (it can't be larger because of (P5)) so this argument has the potential of giving a lower bound of $\Omega(n \log^2 n)$ if we can make α small enough. The argument of Dietz et al. [12] constructs the segment chain with $\alpha = O(1)$ which gives an $\Omega(n \log(n))$ lower bound; we'll be able to achieve $\alpha = O(1/\log(n))$ which will give the optimal $\Omega(n \log^2 n)$ lower bound.

The final property (P7) is the most technical, and is crucial for the analysis. It concerns the way that we'll account for the cost of the algorithm, and describing this property requires some additional terminology.

Each of the segment chains $S^t_1, \ldots, S^t_d$ has length d. We say that the segment S^t_j is at *level* j. For each $j \in [1, d]$, we define $\Sigma_j \subseteq [1, n]$ to be the set of steps $t \in [n]$ such that B^t is not a subset of S^t_j. Therefore, according to the definition of the adversary,

Σ_j is the set of steps t such that S_j^{t+1} is not determined by the conservative selection rule, but rather is rebuilt. We use the set Σ_j to define a partition Π_j of $[1, n]$ into intervals where the first interval starts at 1 and ends at the smallest element of Σ_j and each successive interval ends at the next smallest element of $\Sigma_j \cup \{n\}$. These intervals are called the *epochs* at level j. We also define the partition Π_0 to be the trivial partition consisting of the single epoch $[1, n]$, and Π_{d+1} to be the partition into n singleton sets. Epochs at different levels are considered to be distinct objects even though they may consist of the same set of steps. We will often compare their associated sets of steps by inclusion. We make a few observations:

- By the conservative selection rule, for each epoch E, all of the segments S_j^t are the same for all $t \in E$. We therefore define S^E to be this unique segment. For the epoch $[1, n]$ at level 0 we define $S^{[1,n]} = [1, m]$, and for the singleton epochs at level $d + 1$, S^E is not defined.
- The definition of Σ_j implies that $\Sigma_j \subseteq \Sigma_{j+1}$ and therefore the partition of $[1, n]$ into epochs at level $j + 1$ refines the partition into epochs at level j.

Each epoch E is an interval $[s^E, c^E]$ where s^E is the *start time* of the epoch and c^E is the *closing time*. The closing times of the epochs at level j are the elements of $\Sigma_j \cup \{n\}$ and the start times are each 1 more than the closing time of the previous epoch. We write E_j^t for the epoch at level j that contains step t.

An epoch is said to be *terminal* if it contains n, so it is the final epoch at its level. An epoch is *non-terminal* otherwise.

We now build a rooted tree, called the *epoch tree* whose nodes are the epochs at all levels together with one leaf for each $t \in [1, n]$:

- The root is the level 0 epoch $[1, n]$.
- For an epoch at level $j \geq 1$, its parent is the unique epoch at level $j - 1$ that contains it.

Thus the tree has depth $d + 1$ and it has n leaves corresponding to singleton subsets. We visualize the tree as ordered so the leaves are in order from left to right.

Observe that for each $t \in [1, n]$ the path from the root $[1, n]$ to the leaf $\{t\}$ traverses the sequence $E_1^t, \ldots E_d^t$ of epochs and this sequence corresponds precisely to the sequence $S_1^t, \ldots, S_d^t$ of segments selected by the algorithm at step t.

The epoch tree will provide a convenient way to account for the cost of relocations done by a given algorithm. Fix a segment chain strategy and an algorithm. Let χ denote the cost of the algorithm against that strategy.

We define a *move* to be a pair (y, t) where y is an item and t a step such that the algorithm moves item y at step t. The cost χ incurred by the algorithm is the total number of moves.

For accounting purposes, we assign each move (y, t) to the smallest epoch E such that $t \in E$ and $f^{t-1}(y) \in S^E$. In terms of the epoch tree, we travel from the leaf t to the root and assign (y, t) to the first epoch encountered on the path for which S^E contains $f^{t-1}(y)$. (By definition (y, t) is not assigned to a leaf.) Thus if (y, t) is assigned to epoch E at level i then $f^{t-1}(y) \in S_i^t \setminus S_{i+1}^t$. Denoting the cost of all moves assigned to E by q^E we have:

$$\chi = \sum_E q^E. \tag{5.1}$$

We are now ready to state the final property of the adversary.

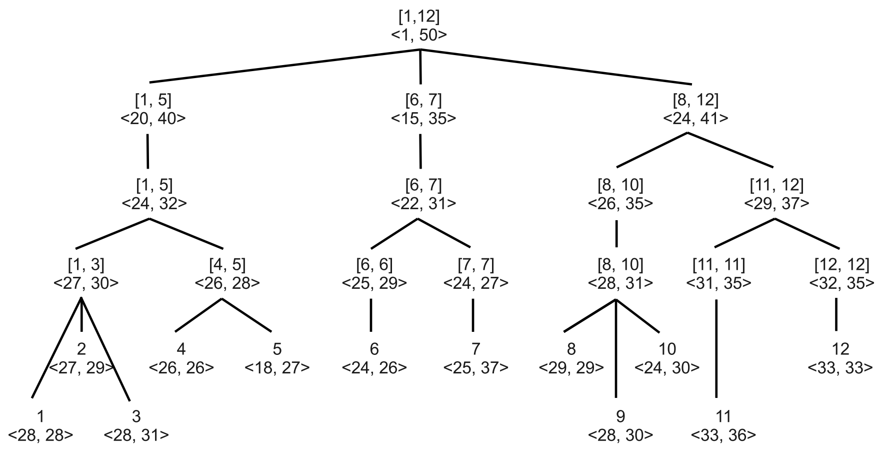

Fig. 5.1: An epoch tree

This figure depicts an epoch tree for a 12 step game played on an array with 50 positions. Each non-leaf in the tree is an epoch, which is an interval of steps. Epochs are denoted by $[a, b]$. The leaves correspond to the individual steps. Each epoch E is associated to a segment of the array S^E. Segments of the array are denoted by $\langle r, s\rangle$. The segments corresponding to non-leaves are nested as one moves away from the root. For each step t, the sequence of segments labeling the non-root, non-leaf nodes along the path to t gives the segment chain at time t.
Each leaf is labeled by the busy segment B^t. Note that an epoch E containing t ends at step t if and only if B^t is not a subset of S^E.

(P7) For any non-terminal epoch E with start time s we have $q^E \geq \frac{1}{8}w^{s-1}(S^E)$, that is, the total cost of moves assigned to E is at least a $1/8$ fraction of the weight of the associated segment S^E at the start of the epoch.

We now formulate two lemmas concerning these properties. The first lemma gives a lower bound on the cost incurred by any algorithm against a segment chain strategy that satisfies the above properties, in terms of the parameters σ, α and λ in the properties. This lemma encapsulates and extends the main accounting argument of Dietz et al. [11, 20], which they used to prove an $\Omega(\log^2(n))$ amortized lower bound for the special case of *smooth* algorithms.

LEMMA 5.1. *(*Properties imply lower bound*) Let m, n, n_0, δ_0 and Y^0 be as in Lemma 2.3. Let $\sigma \geq 1$ and α, λ be positive parameters. If a segment chain strategy produces segment chains with d levels satisfying (P1)-(P7) then the cost incurred by the algorithm satisfies*

$$\chi^n(m|Y^0) \geq \frac{\lambda\delta^* nd^2}{128(\alpha d + \frac{\lambda}{\sigma} + 1 - \delta_0)}. \tag{5.2}$$

We next turn to the second lemma, Lemma 5.2, which shows that with suitable parameters our adversary satisfies (P1)-(P7). Before giving the specific parameter choices, we give some intuition for these choices. The choice of parameters is directly motivated by the expression in the lower bound. We would like the numerator to be large, while keeping the denominator small.

The main parameter in the numerator is the depth of the segment chains d. Property (P5), which requires segment sizes to at least halve each time, and property (P4) which requires that each S_d^t be suitable, limit d to be $O(\log(n))$. We will indeed be able to choose d to be $\Theta(\log(n))$ (here and elsewhere in this overview the $\Theta(\cdot)$ may have an implicit dependence on δ_0). The numerator also involves λ and δ^* which are both at most δ_0 and we'll pick them to be $\Omega(\delta_0)$. Thus the numerator will just be $\Theta(n \log^2(n))$ (where we again hide the dependence on δ_0).

The denominator must be at least $1 - \delta_0$ and we'll be able to achieve a bound of $\Theta(1 - \delta_0)$. To do this we'll need that the parameter α which bounds the density degradation along each chain by $O((1-\delta_0)/d)$ so (ignoring the dependence on δ_0) we need α to be $O(\frac{1}{\log(n)})$. We'll also need that $\lambda/\sigma = O(1-\delta_0)$ but ensuring it is a minor detail.

The density degradation parameter α is the crucial part of the denominator. The parameter α is closely related to the input parameter κ of the adversary (which determines the quality function). As we'll see later in Lemma 7.3, α is in fact $O(\kappa)$. Since we want α to be $O(\frac{1}{\log(n)})$ we'll want to choose κ to be $O(\frac{1}{\log(n)})$.

On the other hand, if we make κ too small then this has the effect of making the quality function depend mostly on the density and very little on the size. This can be bad because, as the adversary builds a segment chain the segment sizes may shrink rapidly along the chain, which could force us to make the length d of the segment chains smaller than $\Theta(\log(n))$ in order to be sure that the final segment in each chain is suitable (as required by property (P4)).

Fortunately, as we'll show, we will be able to choose κ (and hence also α) to be $\Theta(\frac{1}{\log(n)})$ and d to be $\Theta(\log(n))$. Thus we'll get the $\Theta(n \log^2 n)$ lower bound.

With all of this in mind, we now specify the parameters.

We choose the input parameters to our adversary as follows:

$$\kappa = 2\log(1/\delta_0)\frac{1}{\log(n)} \tag{5.3}$$

$$d = \left\lfloor \frac{1-\delta_0}{8C_6\log(1/\delta_0)}\log(n)\right\rfloor, \tag{5.4}$$

$$\lambda = \frac{1}{6}\delta_0. \tag{5.5}$$

where

$$C_6 = 60. \tag{5.6}$$

We'll also need a lower bound on n:

$$C_0 = 2^{1000C_6} \tag{5.7}$$

(Both C_6 and C_0 are chosen large enough to satisfy Lemma 7.3 below.)

The auxiliary parameters needed to specify properties (P2), (P3) and (P6) are set as follows:

$$\sigma = n^{1/4} \qquad \text{(segment size lower bound)} \tag{5.8}$$
$$\delta^* = \delta_0 2^{\delta_0 - 1} \qquad \text{(segment density lower bound)} \tag{5.9}$$
$$\alpha = 2C_6\kappa = 4C_6 \log(1/\delta_0)\frac{1}{\log(n)} \quad \text{(density degradation parameter)} \tag{5.10}$$

Here is the promised lemma:

LEMMA 5.2. *Let $m, n, n_0, Y^0, \delta_0, \mu_0$ be as in Lemma 2.3. Let the parameters be set according to (5.3)-(5.10). Then* ADV(d, κ, λ) *satisfies (P1)-(P7).*

5.1. Proof of lemma 2.3. Before proving Lemmas 5.2 and 5.1, we show how they combine to prove the main lemma.

Let m, n, n_0, Y^0, δ_0 be as in Lemma 2.3. Lemma 5.2 implies that, with the given setting of parameters, our adversary satisfies (P1)-(P7).

Lemma 5.1 gives a lower bound on $\chi^n(m|Y^0)$. The denominator of (5.2) is $\Theta(\alpha d + \frac{\lambda}{\sigma} + 1 - \delta_0)$. The settings given by (5.10) and (5.4) give $\alpha d \leq (1-\delta_0)/2$. The setting $\sigma = n^{1/4}$ and $\lambda \leq 1$ and the hypothesis of the main lemma that $\delta_0 \leq 1 - n^{-1/5}$ give $\frac{\lambda}{\sigma} \leq 1 - \delta_0$. So the denominator of (5.2) is $\Theta(1-\delta_0)$

For the numerator, the setting of d gives $d^2 = \Theta(\log^2(n))(1-\delta_0)^2/(\log(1/\delta_0))^2$. For large mingap $\lambda = 1$ and the fraction in (5.2) simplifies to:

$$\chi = \Theta\left(n\log^2(n)\frac{\delta_0(1-\delta_0)}{(\log(1/\delta_0))^2}\right),$$

while for small mingap $\lambda = \frac{\delta_0}{6}$,

$$\chi = \Theta\left(n\log^2(n)\frac{\delta_0^2(1-\delta_0)}{(\log(1/\delta_0))^2}\right),$$

as required to prove Lemma 2.3.

It remains to prove Lemmas 5.1 and 5.2. Each of these lemmas is proved in its own section, and the two sections are completely independent so can be read in either order.

6. Proof of Lemma 5.1.

The proof that the properties give a good lower bound is a careful accounting argument that is a reworking of the idea used in [11, 20] to obtain an $\Omega(n\log^2(n))$ bound for smooth algorithms.

6.1. Some preliminaries and an overview. For simplicity we write χ for $\chi^n(m|Y^0)$. By (5.1), $\chi \geq \sum_E q^E$. The critical property (P7) says that, for E non-terminal, q^E is bounded below by $\frac{1}{8}$ of the weight of the associated segment S^E at the beginning of the epoch. Using this together with the lower bound on the density of any segment from (P3(δ^*)) gives:

$$q^E \geq \frac{1}{8}w^{s^E-1}(S^E) \geq \frac{\delta^*}{8}|S^E|.$$

Letting $\mathcal{N}$ be the set of non-terminal epochs at level between 1 and d, we have:

$$\chi \geq \sum_{E\in\mathcal{N}} q^E \geq \frac{\delta^*}{8}\sum_{E\in\mathcal{N}}|S^E|. \tag{6.1}$$

So we are reduced to proving a lower bound on $\sum_{E \in \mathcal{N}} |S^E|$. To this end we start with two easy observations.

PROPOSITION 6.1.

1. *For any epoch E, $|E| \leq |S^E|$.*
2. *For each level $i \in [1,d]$ the sum of the lengths of all non-terminal epochs at level i is at least $n/2$.*

The first observation holds because during epoch E, $|E|$ new items are stored within the segment $|S^E|$. The second holds since the sum of the lengths of all epochs at level i is n and by (P1) the terminal epoch has length at most $n/2$.

Combining these observations with (6.1),

$$\chi \geq \frac{\delta^*}{8} \sum_{E \in \mathcal{N}} |S^E| \geq \frac{\delta^*}{8} \sum_{E \in \mathcal{N}} |E| \geq \frac{\delta^* nd}{16}.$$

This is a non-trivial lower bound, but we want a lower bound of $\Omega(nd^2)$. To improve the bound from $\Omega(nd)$ to $\Omega(nd^2)$ we try to show that the bound $|S^E| \geq |E|$ used above can typically be improved to $|S^E| = \Omega(|E|d)$. This is not universally true for all epochs, but the following weaker statement will suffice: for a constant fraction of steps t, $|S^E| = \Omega(|E|d)$ holds for a constant fraction of the epochs containing t. This would follow if we could show that for a constant fraction of steps t, $\sum_{E:t \in E} \frac{|E|}{|S^E|} = O(1)$.

The following proposition shows that something like this holds if we replace $|E|$ in the numerator by a related quantity. For epoch E at level at least 1, let $\pi(E)$ denote the parent of epoch E in the tree, which is the epoch containing E whose level is one less than that of E. Let:

$$\Delta(E) = s^E - s^{\pi(E)},$$

which is the time from the start of $\pi(E)$ until the start of E.

PROPOSITION 6.2. *For any time t,*

$$\sum_{E \neq [1,n]: t \in E} \frac{\Delta(E)}{|S^{\pi(E)}|} \leq \frac{1}{\lambda}(1 - \delta_0 + d\alpha).$$

Proof. For $i \in [0,d]$ we make the following definitions:

- E_i denotes the epoch at level i containing t.
- s_i is the start time of E_i. Observe that $s_0 = 1 \leq s_1 \leq \cdots \leq s_{d+1} = t$.
- S_i is the segment associated E_i.
- $\rho_i = \rho^{s_i - 1}(S_i)$, which is the density of the segment S_i just prior to the start of epoch E_i. Note that $\rho_0 = \rho^0([1,m]) = \delta_0$. This definition doesn't work for ρ_{d+1} (since there is no segment S_{d+1}) so we define $\rho_{d+1} = 1$ for convenience.
- $\Delta_i = \Delta(E_i)$ which is equal to $s_i - s_{i-1}$.

For any $i \in [1,d]$, the step interval $[s_{i-1}, s_i - 1]$ has size Δ_i and is a subset of E_{i-1}. During this interval of steps all inserted items are placed in S_{i-1} and no item leaves S_{i-1}. Hence the weight of S_{i-1} increases by $\lambda\Delta_i$ and so:

$$\rho^{s_i - 1}(S_{i-1}) - \rho_{i-1} = \rho^{s_i - 1}(S_{i-1}) - \rho^{s_{i-1}-1}(S_{i-1}) = \lambda \frac{\Delta_i}{|S_{i-1}|}.$$

We also have:

$$\rho_i \geq \rho^{s_i-1}(S_{i-1})2^{-\alpha} \geq \rho^{s_i-1}(S_{i-1}) - \alpha.$$

For $i \leq d$ the first inequality holds by property (P6(α)) (which bounds the rate at which density degrades along a segment chain) and for $i = d+1$ it holds from the choice we made that $\rho_{d+1} = 1$. The second inequality holds because ρ_i and α are in $[0,1]$.

Using the second inequality with the first and rearranging we get:

$$\frac{\Delta_i}{|S_{i-1}|} \leq \frac{1}{\lambda}\left(\rho_i - \rho_{i-1} + \alpha\right).$$

Summing this inequality for each E_i, the sum on the right telescopes to give:

$$\sum_{E \neq [1,n]: t \in E} \frac{\Delta(E)}{|S^{\pi(E)}|} \leq \frac{1}{\lambda}(\rho_{d+1} - \rho_0 + d\alpha) \leq \frac{1}{\lambda}(1 - \delta_0 + d\alpha),$$

as required. □

Intuitively, this proposition is useful because we expect that for a "typical" epoch E, $\Delta(E) = \Omega(|\pi(E)|)$, i.e. the number of steps from the start of $\pi(E)$ to the start of E is typically a constant fraction of the length of $|\pi(E)|$. The technical work done in the proof makes this intuition precise.

6.2. The proof. Following the discussion of the previous subsection, we return to (6.1) and try to show that $\sum_{E\in\mathcal{N}} |S^E|$ is a $\Theta(d)$ factor larger than $W = \sum_{E\in\mathcal{N}} |E|$ To facilitate this comparison we define $\beta(E) = |E|/W$. Using the arithmetic-harmonic mean inequality (which says that if X is a positive valued random variable then $\mathbb{E}[X] \geq 1/\mathbb{E}[1/X]$) we get:

$$\sum_{E\in\mathcal{N}} |S^E| = W \sum_{E\in\mathcal{N}} \beta(E)\frac{|S^E|}{|E|} \geq \frac{W}{\sum_{E\in\mathcal{N}} \beta(E)\frac{|E|}{|S^E|}} = \frac{W^2}{\sum_{E\in\mathcal{N}} \frac{|E|^2}{|S^E|}} \geq \frac{n^2 d^2}{4\sum_{E\in\mathcal{N}} \frac{|E|^2}{|S^E|}}.$$

If we can show that the denominator is $O(n)$ (where the big O does not depend on d) then we'll get the desired lower bound. This is indeed true, and the conclusion of the lemma follows immediately by combining the previous inequality with (6.1) and the following lemma:

Lemma 6.3.

$$\sum_{E\in\mathcal{N}} \frac{|E|^2}{|S^E|} \leq \frac{4n}{\lambda}\left(\frac{\lambda}{\sigma} + 1 - \delta_0 + d\alpha\right).$$

The reader should take note that we have switched from proving a lower bound on a sum to proving an upper bound on a related sum.

Proof. We prove the upper bound with the sum extended to the set $\mathcal{E}$ of all epochs at levels from 1 to d (not just non-terminal epochs).

We will prove the bound by rewriting the sum using some elementary accounting tricks. Here's the first one. Recall that for a positive integer k, k^2 is the sum of the first k odd numbers. Thus, letting s^E be the start time of epoch E we can write:

$$|E|^2 = \sum_{t \in E} 1 + 2(t - s^E).$$

We therefore have:

$$\sum_{E \in \mathcal{E}} \frac{|E|^2}{|S^E|} = \sum_{t=1}^{n} \sum_{E \in \mathcal{E}: t \in E} \frac{1 + 2(t - s^E)}{|S^E|}. \tag{6.2}$$

We hold t fixed, and bound the inner sum (and then multiply it by n). Let $\mathcal{E}(t)$ be the set of epochs at level 1 to d containing t. Let $\mathcal{F}(t)$ be the epochs at level 1 to $d+1$ containing t (i.e. including the leaf $\{t\}$).

Recalling the definition of $\Delta(F)$ for an epoch F from the previous subsection, we have that $t - s^E = \sum_{F \in \mathcal{F}(t): F \subset E} \Delta(F)$, where $F \subset E$ is strict containment. Therefore:

$$\begin{aligned}
\sum_{E \in \mathcal{E}: t \in E} \frac{1 + 2(t - s^E)}{|S^E|} &= \sum_{E \in \mathcal{E}(t)} \frac{1}{|S^E|} + 2 \sum_{E \in \mathcal{E}(t)} \sum_{F \in \mathcal{F}(t): F \subset E} \frac{\Delta(F)}{|S^E|} \\
&= \sum_{E \in \mathcal{E}(t)} \frac{1}{|S^E|} + 2 \sum_{F \in \mathcal{F}(t)} \Delta(F) \sum_{E \in \mathcal{E}(t): F \subset E} \frac{1}{|S^E|} \qquad (6.3)
\end{aligned}$$

Now we note that property (P5) (that the segment sizes decrease by at least a factor of 2 down the epoch tree) implies that for any epoch $D \in \mathcal{E}$:

$$\sum_{E: D \subset E} \frac{1}{|S^E|} \leq \frac{2}{|S^{\pi(D)}|} \leq \frac{1}{|S^D|}. \tag{6.4}$$

We can use this to bound the first sum in (6.3) by taking D to be the epoch at level d. By property (P2(σ)), $|S^D| \geq \sigma$. For the second sum in (6.3) we take D to be F and use the first inequality in (6.4) to get:

$$\sum_{E \in \mathcal{E}: t \in E} \frac{1 + 2(t - s^E)}{|S^E|} \leq \frac{2}{\sigma} + 4 \left(\sum_{F \in \mathcal{F}(t)} \frac{\Delta(F)}{|S^{\pi(F)}|} \right)$$

We can now apply Proposition 6.2 to obtain

$$\begin{aligned}
\sum_{E \in \mathcal{E}: t \in E} \frac{1 + 2(t - s^E)}{|S^E|} &\leq 4 \left(\frac{1}{\sigma} + \frac{1 - \delta_0 + d\alpha}{\lambda} \right) \\
&= \frac{4}{\lambda} \left(\frac{\lambda}{\sigma} + 1 - \delta_0 + d\alpha \right).
\end{aligned}$$

Summing over all $t \in [n]$ and using (6.2) gives the desired bound. □

This completes the proof of Lemma 5.1.

7. Proof of Lemma 5.2. In this final section we show that the given strategy satisfies properties (P1)-(P7). We start with a detailed informal overview of the ideas of the adversary construction.

7.1. Motivating our adversary strategy. Our adversary has a reasonably short description, but the specific choices made in the definition may seem arbitrary:

- Why do we introduce the auxiliary sets T_i^t?
- Why do we choose the particular quality function?
- If $i > j$ then it follows that $S_i^t = \mathbf{middle}(\mathbf{densify}^{t-1}(\mathbf{middle}(S_{i-1}^t)))$ from the above. Why do we need the two applications of **middle**?

These issues were introduced at a high level in Section 1.4. Here we consider them again in more technical detail, and show how we arrived at the specific choices. The proof in the later subsections can in principle be read without reading this section, but the technicalities involved in the proof hide the main ideas, which we discuss in this subsection. To simplify the discussion in this section we restrict to the case of large initial mingap. For this case the weight of a segment is the number of items, and the density is the fraction of occupied locations, and a segment is suitable (as defined earlier) if and only if there are at least two items stored in it.

As remarked following the statement of Lemma 5.1, when we apply the lemma to get a lower bound of $\Omega(n\log^2(n))$, we want the depth of the hierarchy to be $d = \Theta(\log(n))$, and the density degradation parameter $\alpha = O(1/\log(n))$. The reader should keep these parameters in mind.

7.1.1. Satisfying (P1)-(P6). Achieving all of the desired properties except (P7) with the desired parameter values is straightforward. In discussing these properties, we'll reason inductively. We start by proving the properties for the first segment chain. Then for step $t > 1$ we prove the properties for the tth segment chain assuming that they hold for the $t-1$st chain. When considering the tth chain we reason about the segments by increasing level. A natural attempt for selecting the first step is to take S_1^1 to be the segment of size $\lfloor n/2 \rfloor$ having highest density, and for $i \geq 1$, define S_i^1 to be the subsegment of S_{i-1}^1 of size $\lfloor |S_{i-1}^1|/2 \rfloor$ having highest density. This does not quite work because it might not satisfy (P6(α)); for a segment S, all subintervals of some fixed size k might have density less than, say $0.9\rho(S)$. This difficulty disappears if we relax the requirement that S have size exactly k: it is easy to show that for a segment S of size at least 8 and $k \leq |S|/2$, there must be a subsegment of size between $k/2$ and k having density at least that of S.

So for a given configuration and segment S consider the subsegment of size between $|S|/4$ and $|S|/2$ having maximum density (breaking ties arbitrarily). We'll refer to this as the *densest large subsegment* of S.

This suggests we choose S_1^1 to be the densest large subsegment of $[1, m]$ and for $i \geq 2$ choose S_i^1 to be the densest large subsegment of S_{i-1}^1. The resulting sequence has nondecreasing density (so certainly satisfies (P3(δ^*) and (P6(α))) and as the segment sizes decrease by at most a factor of 4, we can continue it for $\Theta(\log(n))$ levels and have conditions (P1)-(P6) hold for the first segment chain.

Next let's consider the case $t > 1$. Assume that we've already selected the segment chain for step $t-1$ so that conditions (P1)-(P6) hold. We want to build the chain for step t so that these continue to hold. Let j^{t-1} be the critical level for step $t-1$, as defined earlier, i.e., the first level for which B^{t-1} is not a subset of S_j^{t-1} (or $d+1$ if there is no such level). For $i < j^{t-1}$ the segment chain restriction requires $S_i^t = S_i^{t-1}$ and it is easy to check that properties (P1)-(P6) are preserved, because after the move by the algorithm the set of items stored in S_i^{t-1} changes only by the addition of the newly inserted item y^{t-1}. For levels $i \geq j^{t-1}$, it is not in general sufficient to take $S_i^t = S_i^{t-1}$ because the algorithm may have moved many items out of S_i^{t-1} and so the density restrictions (P3(δ^*)) and (P6(α)) need not hold. Instead, we rebuild the segments S_i^t for $i \geq j^{t-1}$, iteratively taking S_i^t to be the densest large subsegment of S_{i-1}^t. This adversary satisfies (P1)-(P6). We will refer to this adversary as the *naive adversary*.

7.1.2. Requiring left and right buffers. We now turn our attention to condition (P7). Let E be an epoch at level i with associated segment S and start time s, and let Z be the set of items stored in S immediately prior to step s. We want to ensure that the number of moves charged to E is a constant fraction of $|Z|$. Consider the set Z' of items of Z that move at least once during E and for each $y \in Z'$, let t_y be the first step during the epoch that it moves. Under our convention for charging moves to epochs, we charge the move (y, t_y) to E provided that at step t_y, y is not stored within the successor interval S_{i+1}^t. So q^E is at least the number of items $y \in Z$ that move during epoch E such that the first step it is moved it is not in the successor interval of S. To ensure that this is large, at the start time s we split S into three segments L, M, R where L is on the left and R is on the right. These are referred to, respectively, as the *left buffer* and *right buffer* of S. Let Y_L, Y_M and Y_R be the sets of items stored in each of these sets. We want our strategy to satisfy two conditions:

1. L, M, R are chosen so that Y_L and Y_R each contain a constant fraction of items from Z.
2. At every step $t \in E$, we restrict the choice of S_{i+1}^t so that it does not contain any items from $Y_L \cup Y_R$ that have not yet moved during epoch E. More precisely, let $B^{E,t-1}$ be the union of the busy segments from the beginning of E until step $t-1$; this is the region where the algorithm has shifted items during epoch E. Let $M^t = M \cup B^{E,t-1}$. We require that $S_{i+1}^t \subseteq M^t$. In particular, when t is the start time of E we require $S_{i+1}^t \subseteq M$.

If the selection strategy chooses S_{i+1}^t for each $t \in E$ to satisfy these two conditions, then (P7) holds for epoch E because the second condition implies that q^E is at least the number of items $y \in Y_L \cup Y_R$ that move at least once during the epoch and by the (assumed) laziness of the algorithm, during the epoch either all of the items in Y_L move, or all of the items in Y_R move.

7.1.3. Failure of the naive adversary. Unfortunately, the naive adversary does not meet these two conditions. We now discuss why, and modify the strategy so that it satisfies the above two conditions (and therefore (P7)) while preserving (P1)-(P6). To focus attention on the two conditions, we fix an epoch E at level i with start time s. The segment S associated to this epoch is equal to S_i^t for all $t \in E$. The epoch E is divided into one or more epochs at level $i+1$. Let H be the set of start times for level $i+1$ epochs during E. The two conditions restrict the choice of S_{i+1}^t for each $t \in H$.

For each step $t \in H$, the naive adversary selects S_{i+1}^t to be the densest large subsegment. Condition 2 above requires that $S_{i+1}^t \subseteq M^t$ (and not just $S_{i+1}^t \subseteq S$) so it is natural to modify the definition of S_{i+1}^t to be the subsegment of M^t (rather than the subsegment of S) of maximum density and size between $|M^t|/4$ and $|M^t|/2$.

The modified algorithm will still satisfy (P1), (P2(σ)) and (P5). However, where before we could be sure that the density of S_{i+1}^t is at least the density of S, now we only have that it is at least the density of M^t, but the density of M^t might be much lower than that of S. Thus conditions (P3(δ^*)), (P4) and (P6(α)) are no longer guaranteed.

7.1.4. Enforcing κ-lower balance. To solve this problem we'll insist that every segment chosen to start a new epoch satisfy a condition that will ensure that the density of M^t is either greater than that of S or only slightly less. More precisely, we introduce a property of segments called *κ-lower balance*, where $\kappa > 0$ is a parameter that we will eventually take to be $\Theta(1/\log(n))$. We will say S is κ-lower balanced with respect to a given configuration if every subsegment of size at least $|S|/4$ has density at least the density of S times $\left(\frac{1}{4}\right)^{\kappa}$, which for small κ is $(1 - O(\kappa))$. If we knew that each selected segment satisfied this additional condition then for every step $t \in H$, M^t (which has size at least $|S|/4$) will have density at least $1 - O(\kappa)$ times that of S. This will be enough to get (P1)-(P7).

So we modify the construction of the segments S_u^j once again so that whenever a new epoch is started, the segment chosen for that epoch is κ-lower balanced. This is done as follows. Fix an epoch E at level i with start time s and associated segment S as before, and assume inductively that S is κ-lower balanced at the beginning of the epoch. We split S into three parts L, M, R where M is the middle third of S. By κ-lower balance we know that L,M,R all have density at least $\left(\frac{1}{4}\right)^{\kappa}$ times the density of S, and consequently L and R each contain about $1/3$ of the items stored in S.

We consider a step $t \in E$ that starts a new $i + 1$ epoch (this includes the case $t = s$) and describe how we'll select S_{i+1}^t. As previously defined, let $M^t = M \cup B^{E,t-1}$. By κ-lower balance, the density of M^t at the start of E is at least $1 - O(\kappa)$ times the density of S. This is still true at step t since both S and M^t contain the same items they had at the beginning of the epoch plus the items added during the epoch. We need to select $S_{i+1}^t \subseteq M^t$ that is κ-lower balanced. If M^t were κ-lower balanced we could just take S_{i+1}^t to be M, but in general M^t need not be κ-lower balanced, so we'll need to search for a large κ-lower balanced subsegment.

It turns out to be easier to find a subsegment with a closely related property called κ-upper balance. S is *κ-upper balanced* if every subsegment of size at least $|S|/4$ has density *at most* the density of S times 4^{κ}. While κ-upper balance does not imply κ-lower balance, it is not hard to show that if U is $\kappa/24$-upper balanced then the middle third of U is κ-lower balanced. So we'll find a large subset of M^t that is $\kappa/24$-upper balanced and has density at least that of M^t and then take S_{i+1}^t to be the middle third.

The following simple iterative process can be used to find a $\kappa/24$-upper balanced subset of M^t. Initialize the set A to be M^t and perform the following iteration: If A is $\kappa/24$-upper balanced then stop. Otherwise, since $\kappa/24$-upper balance is violated there is a subset A' of size at least $|A|/4$ that has density more than 4^{κ} times that of A. Now replace A by A' and repeat. Since during each iteration the density of A increases by a factor at least 4^{κ} at each step this must terminate.

This gives a well defined process for selecting the segment S_{i+1}^t at the start of a level i epoch: Start from M^t, apply the iterative process until it terminates and then take the middle third.

7.1.5. Controlling segment lengths. There is one remaining issue: the iterative process must terminate but might require many iterations, and $|S_i^t|$ may be much smaller than $|S_{i-1}^t|$. Therefore it is no longer immediate that we can continue this for $\Omega(\log(n))$ levels (while ensuring that the final segment has at least 2 items). Nevertheless, we can indeed continue this for $\Omega(\log(n))$ levels. In the troublesome case that $|S_i^t|$ is much smaller than $|S_{i-1}^t|$ (because the selection of S_i^t took many iterations), the density of S_i^t will be significantly higher than that of S_{i-1}^t. Since the density can not rise above 1, this can be used to show that the segment sizes can't shrink too quickly. A convenient way to simultaneously account for size and density is to measure the quality of a segment by a function that combines both the density and size. A natural form for such a function is $\phi(U) = \rho(U)^{\alpha}|U|^{\beta}$. It's easy to check that if $\beta/\alpha \geq \kappa$ then this function increases during each of the above iterations. So we fix $\alpha = 1$ and $\beta = \kappa$, and for this choice of ϕ its not hard to show that in passing from S_i^t to S_{i+1}^t, the function ϕ decreases by at most a factor $(1 - O(\kappa))$. This implies that we can take the depth to be $\Theta(1/\kappa)$. For $\kappa = \Theta(1/\log(n))$ this gives what we want.

The introduction of the quality function ϕ allows for a slight simplification of the selection of S_i^t. The proof that ϕ is nondecreasing in each iteration actually shows that if U is a segment whose ϕ value is at least that of any subsegment of U then U is κ-upper balanced. So to obtain a κ-upper balanced subset of M^t we can (and do) replace the iterative process by the function **densify** of the previous section which selects the subset of maximum ϕ value.

To summarize, for the start time s of E, having chosen $S = S_{i-1}^s$ the selection of S_i^t involves three steps: first we partition S into L, M, R and restrict to the middle third M of S (this corresponds to the set T_i^s in the definition of the adversary). The process of constructing S_{i+1}^t for $t \in E$ with $t > s$ is similar except we take T_i^t to be M^t instead of M.

7.2. The proof that the adversary satisfies (P1)-(P7).

The remainder of this section gives the proof of Lemma 5.2 that our adversary with the parameters chosen according to (5.6)-(5.10) satisfies (P1)-(P7).

We start by noting that property (P1) and (P5) which require that the initial segment of every chain have size at most $n/2$ and the segments decrease in size by at least a factor of 2 are obvious from the definition of the adversary. It remains to verify the remaining five properties.

7.2.1. Properties of balance and ϕ.

Property (P6(α)) asserts that in each selected segment chain, the density of a selected segment can not be much smaller than its predecessor segment. The proof sketch of the previous subsection explained qualitatively how the function **balance** accomplishes this; here we provide the technical details. This will also be needed to establish (P3(δ^*)) and (P2(σ)) which give lower bounds on the density and weight of any segment occurring in any chain, and (P4) which establishes that there is always a suitable gap.

Let us fix a configuration (Y, f) and let ρ be the associated density function. Let $\kappa > 0$. We start by formally defining κ-upper balance and κ-lower balance with respect to the configuration (Y, f), which were mentioned in the previous subsection.

- S is *κ-upper balanced* (with respect to ρ) if every subsegment of size at least $|S|/4$ has density at most $\rho(S)4^{\kappa}$.
- S is *κ-lower balanced* if every subsegment of size at least $|S|/4$ has density at least $\rho(S)(1/4)^{\kappa}$.

Recall the following definitions:

- The *quality of U with respect to* (Y, f) is the real number $\phi(U) = \phi_{\kappa}(U) = \rho(U)^{1/\kappa}|U|$.
- **densify**(U) is the subsegment of V of U that maximizes $\phi(V)$ (breaking ties arbitrarily).
- **balance**(U) is the subsegment V of U given by **middle**(**densify**(U)).

PROPOSITION 7.1. *Let (Y, f) be an arbitrary configuration and let ρ be the associated density function. Let T be a segment and $D =$* **densify**(T).

1. *$\phi(T) \leq |T|$, i.e. the quality of a segment is at most its length. [This holds since $\rho(T) \leq 1$.]*

2. *$\rho(D) \geq \rho(T)$, i.e. applying* **densify** *to T produces a segment that is at least as dense.*
3. *D is κ-upper balanced. [Proof: If U is a subsegment of D of length at least $|D|/4$, the choice of D implies $\phi(U) \leq \phi(D)$, which implies $\phi(U)^\kappa \leq \phi(D)^\kappa$ which implies $\rho(U)(\frac{1}{4})^\kappa \leq \rho(D)$, which is the condition of κ-upper balance.]*

LEMMA 7.2. *Fix a configuration (Y, f). Let T be an arbitrary segment, let $D =$* **densify**(T) *and $S =$* **balance**$(T) =$ **middle**(D). *Assume that:*

$|S| \geq 4$ *AND* $\kappa \leq 1/24\ln(4)$.

Then:

1. *For any subsegment U of S having size at least $|S|/4$,*

$$\rho(U)/\rho(T) \geq \rho(U)/\rho(D) \geq (\frac{1}{4})^{24\kappa}.$$

2. *S is 25κ-lower balanced with respect to ρ.*
3. *$\phi(S)/\phi(T) \geq \frac{1}{3}(\frac{1}{4})^{24}$.*

Proof. We first show that parts 2 and 3 follow easily from part 1. For part 2, let U be a subsegment of S of size at least $|S|/4$. We have

$$\frac{\rho(U)}{\rho(S)} = \frac{\rho(U)}{\rho(D)}\frac{\rho(D)}{\rho(S)} \geq \left(\frac{1}{4}\right)^{24\kappa} \cdot \left(\frac{1}{4}\right)^{\kappa} = \left(\frac{1}{4}\right)^{25\kappa},$$

where the inequality uses part 1 and the fact that D is κ-upper balanced (by Proposition 7.1) and that $|S| \geq |D|/4$.

For part 3, note that

$$\frac{\phi(S)}{\phi(T)} \geq \frac{\phi(S)}{\phi(D)} = \frac{|S|}{|D|}\left(\frac{\rho(S)}{\rho(D)}\right)^{1/\kappa} \geq \frac{1}{3}\left(\frac{1}{4}\right)^{24},$$

by applying part 1 with $U = S$.

It remains to prove the first part. From Proposition 7.1, D is κ-upper balanced. $D \backslash U$ consists of 2 segments L (on the left) and R (on the right). The κ-upper balance of D implies that $\rho(L)$ and $\rho(R)$ can't be much higher high than $\rho(D)$, and we'll show that this implies that the density of U can't be much lower lower $\rho(D)$. We have:

$$|D|\rho(D) = w(D) = w(L) + w(R) + w(U) = |L|\rho(L) + |R|\rho(R) + |U|\rho(U),$$

which implies:

$$\frac{\rho(U)}{\rho(D)} = \frac{1}{|U|}\left(|D| - |L|\frac{\rho(L)}{\rho(D)} - |R|\frac{\rho(R)}{\rho(D)}\right).$$

Since $|S| \geq 4$ and $S =$ **middle**(D) it follows that $|D| \geq 8$ and $|S| \leq |D|/2$. Since $U \subseteq S =$ **middle**(D), we have $|L|, |R| \geq |D|/4$ and since D is κ-upper balanced, it follows that $\rho(L)/\rho(D)$ and $\rho(R)/\rho(D)$ are each at most 4^κ. So:

$$\begin{aligned}\frac{\rho(U)}{\rho(D)} &\geq \frac{1}{|U|}(|D| - (|L| + |R|)4^{\kappa}) \\ &\geq \frac{1}{|U|}(|D|4^{-\kappa} - |L| - |R|) \\ &= \frac{1}{|U|}(|D|4^{-\kappa} - |D| + |U|) \\ &= \frac{1}{|U|}(|U| - |D|(1 - e^{-\ln(4)\kappa})) \\ &\geq \left(1 - \frac{|D|}{|U|}\ln(4)\kappa)\right) \\ &\geq (1 - 12\ln(4)\kappa) \geq \left(\frac{1}{4}\right)^{24\kappa},\end{aligned}$$

where the final inequality uses the hypothesis that $\kappa \leq 1/(24\ln(4))$ and the inequality $(1-x) \geq e^{-2x}$ for $x \leq 1/2$. □

7.2.2. Some technical inequalities involving the parameters. Our aim is to establish that the adversary satisfies (P1)-(P7) with the parameter choices of Equations (5.3)-(5.10).

We complete the proof in the next two subsections. The main part of the argument is an induction which provides a lower bound on the density and quality of all of the segments S_i^t and T_i^t produced by the adversary. Obviously the selected values of the parameters will play a role, and this role shows up as certain arithmetic inequalities that are required for the argument. Each of these inequalities can be verified by a routine calculation by plugging in the specific values of the parameters.

In preparation for this, we now collect all of the technical inequalities that are needed for the coming argument. We mention the role each inequality plays in the argument and highlight the most important ones. This section is optional for the reader; the reader can go directly to the next section and when encountering one of these needed inequalities in the next section, the reader can either do the (usually easy) verification himself or take our word for it.

Readers who choose to read this section may find the brief comments after each one hard to follow completely, because they refer forward to specific details in the subsequent proofs. The hope is that giving the reader an impression of the role that these technical properties play will facilitate reading the proof.

Here are the hypotheses carried over from Lemma 2.3:

(A1) $n \geq C_0$

(A2) $\delta_0 \in (\log(n))^{-2}, 1 - n^{-1/5})$.

The following properties of the chosen parameters can be deduced from (A1)-(A2) and the definitions of the parameters (5.3)-(5.10).

(R1) $\delta^*\sigma \geq 2$. This is a technical condition that ensures that S_d^t satisfies the hypotheses of Lemma 4.1 and thus has a suitable gap. It holds with a lot of room to spare.

(R2) $\sigma \geq 4$. For (P2), every segment S_i^t has size at least 4. This is a minor technical hypothesis of Lemma 7.3, which is needed so that we can apply Lemma 7.2. It holds trivially.

(R3) $(1/\delta_0)^{1/\kappa} \leq \sqrt{n}$. This is a significant inequality. In the basis of our induction, we want to show that (at the beginning of step t) the chosen segment T_1^t has quality value at least $\sqrt{n}/2$. This quality function value is easily bounded below by $\frac{n}{2}\delta_0^{1/\kappa}$. Since we want this to be at least $\sqrt{n}/2$ this restricts κ to be large enough. On the other hand we want κ to be $O(1/\log(n))$ so that the density degradation parameter α is that small. Fortunately these two restrictions can both be satisfied by the selected value of κ.

(R4) $\sigma \leq 2^{-2C_6 d}\sqrt{n}$. We want every segment to have length at least σ. The length of a segment is bounded below (at each step) by the quality function value, and the analysis will give us a lower bound on that. Our induction argument will lead to equations (7.11) and (7.9) which give lower bounds on the quality function value of all of the selected segments, and these lower bounds are all at least $2^{-2C_6 d}\sqrt{n}$. By adjusting the multiplicative constant for d we can make this as close to $\sqrt{n}$ as we want. We arbitrarily chose σ to be $n^{1/4}$ and adjusted d appropriately.

(R5) $2^{-2dC_6\kappa}\delta_0 \geq \delta^*$. For (P3($\delta^*$)) we need that every segment in every segment chain has density at least δ^*. This will follow immediately from the present inequality, and (7.10) which says that every segment in a segment chain has density at least $2^{-2dC_6\kappa}\delta_0$.

(R6) $\kappa \leq \frac{1}{24\ln(4)}$ and $\kappa \leq 1/50$. These are minor (and easily verifiable) technical upper bounds needed, respectively, for the hypothesis of Lemma 7.2, and the end of the proof of (P7)).

(R7) $\alpha = 2C_6\kappa$. This is a restatement of the definition (5.10) of α, which relates it to the adversary parameter κ. We prove (7.4) and (7.5) which give a lower bound on the ratio of densities of successive S_i^t in each chain as $2^{-2C_6\kappa}$, so setting $\alpha = 2C_6\kappa$ gives us property Property (P6(α)) which requires that the ratio of densities of successive subsegments is at most $2^{-\alpha}$.

7.2.3. The behavior of ϕ and ρ along a segment chain. The next step in establishing the remaining properties is to prove a lemma that for each step t, gives a lower bound on ρ^{t-1} and ϕ^{t-1} of the first segment S_1^t and also shows that as we move from a segment to its successor, ρ^{t-1} and ϕ^{t-1} can't decrease by much.

LEMMA 7.3. *Let C_6 and C_0 be as in (5.6) and (5.7). Suppose that the parameters d,κ,α,σ and δ^* satisfy (R1)-(R7) and that $n \geq C_0$. Let S_i^t and T_i^t be the segments chosen by the adversary and assume that all of them have size at least 4. For each $t \in [1, n]$ we have:*

$$\rho^{t-1}(T_1^t) \geq \delta_0, \tag{7.1}$$
$$\phi^{t-1}(T_1^t) \geq \sqrt{n}/2, \tag{7.2}$$

and for $i \in [1, d]$ we have:

$$\text{if } t \text{ starts an } i\text{-epoch then } S_i^t \text{ is } 25\kappa \text{ lower-balanced with respect to } \rho^{t-1}. \tag{7.3}$$

$$\frac{\rho^{t-1}(S_i^t)}{\rho^{t-1}(T_i^t)} \geq 2^{-C_6\kappa} = 2^{-\alpha/2}. \tag{7.4}$$

$$\frac{\rho^{t-1}(T_{i+1}^t)}{\rho^{t-1}(S_i^t)} \geq 2^{-C_6\kappa} = 2^{-\alpha/2} \qquad \textit{if } i \leq d-1. \tag{7.5}$$

$$\frac{\phi^{t-1}(S_i^t)}{\phi^{t-1}(T_i^t)} \geq 2^{-C_6}. \tag{7.6}$$

$$\frac{\phi^{t-1}(T_{i+1}^t)}{\phi^{t-1}(S_i^t)} \geq 2^{-C_6} \qquad \textit{if } i \leq d-1. \tag{7.7}$$

Proof. We fix an epoch E and prove the result for all $t \in E$. Let i be the level of the epoch. We divide into cases depending on whether or not t is the start time s^E.

Case 1. $t = s^E$. For (7.1) and (7.2), recall that T_1^t is chosen to have highest density among segments of length between $n/2$ and n. Since the m locations of the array can be partitioned into segments of size between $n/2$ and n, and one of those must have density at least the density of the entire array, we conclude that $\rho^{t-1}(T_1^t) \geq \delta_0$. From the definition of ϕ^{t-1}, $\phi^{t-1}(T_1^t) \geq (n/2)\delta_0^{1/\kappa}$, which is at least $\sqrt{n}/2$ (see (R3)).

For the proofs of the remaining parts in this case, we apply Lemma 7.2 with $T = T_i^t$ and $S = S_i^t$. The hypotheses of Lemma 7.2 require that κ be smaller than $1/24\ln(4)$ which is (R6) and that $|S_i^t| \geq 4$, which is a hypothesis of the present lemma. Since $S_i^t = \mathbf{balance}(T_i^t)$, part 2 of Lemma 7.2 implies that S_i^t is 25κ-lower balanced. Inequality (7.4) follows from the first part of Lemma 7.2 with $U = S$, using $C_6 = 60 \geq 48$. Inequality (7.6) follows from the third part of Lemma 7.2 with $U = S$. Since S_i^t is 25κ-lower balanced and $|T_{i+1}^t| \geq |S_i^t|/4$ we have (7.5) and by the definition of ϕ^{t-1} also (7.7).

Case 2. $t > s^E$. We may assume by the first case that the four inequalities hold with t replaced by s^E. We will show that they continue to hold throughout the epoch E. We will repeatedly use the following easy fact:

Proposition 7.4. *Let $S \subseteq S'$ be segments and $s < t$ be steps. Suppose that for all steps $r \in [s, t-1]$, the busy segment B^r is a subset of S. Then $\rho^r(S)$, $\phi^r(S)$, $\rho^r(S)/\rho^r(S')$ and $\phi^r(S)/\phi^r(S')$ are all nondecreasing as a function of $r \in [s, t-1]$*

Proof. This follows from: At each step in $[s, t-1]$, both $w^r(S)$ and $w^r(S')$ increase by λ and $w^r(S) \leq w^r(S')$. □

Let $s < t$ be the start time of the epoch. Note that during epoch E, the sets $S_i^t = S_i^s$ and $T_i^t = T_i^s$. Therefore (7.1) and (7.2) and (7.4) and (7.6) follow from the corresponding inequalities for step s together with Proposition 7.4.

For (7.5) and (7.7), we cannot apply Proposition 7.4 directly because, while $S_i^t = S_i^s$, it may not be true that $T_{i+1}^t = T_{i+1}^s$ because there may have been one or more new $i+1$-epochs started between s and t and so T_{i+1}^u may change during the epoch. Note that since the step interval $[s, t]$ is a subinterval of the epoch E, at any time $u \in [s, t]$ that T_{i+1}^u changes $i+1$ must be the critical level j^{u-1}. We will need to make careful use of the adversary construction (Adv1.b) of the T-set at the critical level.

Proposition 7.5. *Let $i \in [1, d]$ be a level and t a step that belongs to the level i epoch E, whose start time is s. Then $T_{i+1}^t = T_{i+1}^s \cup \bigcup_u B^u$ where u ranges over all steps $u \in [s, t-1]$.*

Proof. We prove this by induction on $t \geq s$; it holds trivially for $t = s$. Assume $t > s$; by induction it suffices to show that $T_{i+1}^t = T_{i+1}^{t-1} \cup B^{t-1}$. If the critical level j^{t-1} is greater than $i+1$ then this is trivial since then $B^{t-1} \subseteq S_{i+1}^{t-1} \subseteq T_{i+1}^{t-1}$. Otherwise $j^{t-1} = i+1$ and the conclusion follows directly from the definition of the adversary. □

We are trying to show that the density of T^t_{i+1} can not be much smaller than the density of S^t_{i+1}. According to Proposition 7.5, the interval T^t_{i+1} is equal to T^s_{i+1} combined with some busy sets. These busy sets are determined by the algorithms behavior, so it is conceivable that the algorithm might be able to choose the busy sets cleverly so as to drive the density of T^t_{i+1} down significantly. We'll show this can't happen using the first part of Lemma 7.2. What we'll do is consider the density of the segment T^t_{i+1} with respect to the density function ρ^{s-1} used during step s. Since T^t_{i+1} is a subsegment of $S^t_i = S^s_i$ that contains T^s_{i+1} and is therefore of size at least $|S^t_i|/4$, we can apply the second part of Lemma 7.2 to get that $\rho^{s-1}(T^t_{i+1})/\rho^{s-1}(S^s_i) \geq \left(\frac{1}{4}\right)^{25\kappa}$ which is at least $2^{-C_6\kappa}$. Proposition 7.4 implies that the same inequality holds for ρ^{t-1} (keeping in mind that $S^t_i = S^s_i$). Since $|T^t_{i+1}|/|S^t_i| \geq 1/3$ we also get (7.7). □

7.2.4. Completing the proof of Lemma 5.2. Using Lemma 7.3 repeatedly we have by induction on $i = 1, \ldots, d$ for fixed $t \in [1, n]$, that:

$$\rho^{t-1}(T^t_i) \geq \delta_0 2^{(2-2i)C_6\kappa} \tag{7.8}$$

$$\phi^{t-1}(T^t_i) \geq \frac{\sqrt{n}}{2} 2^{(2-2i)C_6} \geq \sigma \tag{7.9}$$

$$\rho^{t-1}(S^t_i) \geq \delta_0 2^{(1-2i)C_6\kappa} \geq \delta^* \tag{7.10}$$

$$\phi^{t-1}(S^t_i) \geq \frac{\sqrt{n}}{2} 2^{(1-2i)C_6} \geq \sigma. \tag{7.11}$$

The final inequality of (7.10) follows from (R5). The final inequalities of (7.9) and (7.11) follow from (R4). Note that the final inequality of (7.11) and (R2) imply that as we proceed to level i in the induction, the hypothesis $|S^t_i| \geq 4$ of Lemma 7.3 holds at each step.

This gives us what we need to establish the remaining properties.

Proof of Property (P2(σ)). This says that all segments selected for segment chains have length at least σ. Since $|S^t_d| \geq \phi^{t-1}(S^t_d)$ this follows from (7.11). □

Proof of Property (P3(δ^*)). This says that all segments selected for segment chains have density at least δ^* and is included in (7.10). □

Proof of Property (P4). We need to establish that the hypotheses of Lemma 4.1 (the suitable gap lemma) are satisfied. For the lower bound on the mingap of Y^0 we have from the hypothesis of Lemma 2.3 and the choice $\lambda = \delta_0/6$ that $\text{mingap}(Y^0) \geq 1 + 12/\delta_0 \geq 1 + 2/\lambda$ as required. The hypothesis $\rho^{t-1}(S^t_i) \geq 2\lambda$ follows from (P3(δ^*)) and the fact that $\delta^* \geq \delta/3 \geq 2\lambda$. The hypothesis $w^{t-1}(S^t_i) \geq 2$ follows from $w^{t-1}(S^t_i) = |S^t_i|\rho(S^t_i) \geq \sigma\delta^*$ by properties (P2(σ)) and (P3(δ^*)) which is at least 2 (e.g. see (R1)). □

Proof of Property (P6(α)). This requires a lower bound on $\rho^{t-1}(S^{t-1}_1)$ and on the ratios $\rho^{t-1}(S^t_i)/\rho^{t-1}(S^t_{i-1})$. For the first bound we use (7.4) with $i = 1$ and (7.1). For the second we combine (7.4) and (7.5). In both cases we need the fact that $\alpha = 2C_6\kappa$. □

Proof of Property (P7). Let E be an epoch at level i. We want to bound q^E from below, where q^E is the number of moves done during epoch E that were assigned to epoch E, which means moves that occured during steps $t \in E$ of items that were stored in $S^t_i \setminus T^t_{i+1}$ prior to the move. Let s denote the start time, and c denote the closing time, of epoch E. The busy segment B^c includes a location outside of S^E_i (this is the reason that the epoch closed at step c.) Without loss of generality let us say that B^c includes a location that is to the left of S_i^E. Let L be the left segment of $S_i^s \setminus T_{i+1}^s$.

The number of moves charged to epoch E is q^E. We'll show that (1) every item stored in L at the start time s of E moves during E and (2) the first such move is charged to epoch E, and (3) the number of such items is at least $|S_i^s|\rho^{s-1}(S_i^s)/8$. This immediately gives (P7).

For (1), note that B^c must include a location in $S_{i+1}^c \subseteq T_{i+1}^c$ and so $B^c \cup T_{i+1}^c$ is a segment that must contain all of L. By Proposition 7.5, L must be a subset of the union of the busy segments $B^s \cup \cdots \cup B^c$ which means that every item stored in L under f^{s-1} must move during E. For (2) consider an item y that was stored in location $\ell \in L$ under f^{s-1}. Let t be the earliest step in E that $\ell \in B^t$. Thus $\ell \notin T_{i+1}^t$ so $\ell \notin S_{i+1}^t$ and so the move of y at step t is charged to epoch E. For (3) L is a subsegment of $S_i^t = S_i^s$ of size at least $|S_i^s|/4$. As S_i^s is 25κ-lower balanced with respect to ρ^{s-1} by (7.3), $\rho(L) \geq \rho(S_i^s)(1/4)^{25\kappa} \geq \rho(S_i^s)/2$, since $\kappa \leq 1/50$ by (R6). □

This completes the proof of Lemma 5.2, and thereby also the proof of Lemma 2.3.

Ackknowledgements.. We are grateful to two anonymous referees whose extensive and valuable comments helped us to improve the exposition of the paper.

REFERENCES

[1] Yehuda Afek, Baruch Awerbuch, Serge A. Plotkin, and Michael E. Saks. *Local management of a global resource in a communication network.* J. ACM, 43(1):1–19, 1996.

[2] Martin Babka, Jan Bulánek, Vladimír Čunát, Michal Koucký, and Michael E. Saks. *On online labeling with polynomially many labels.* In Proceedings of the 20th Annual European Symposium on Algorithms, ESA '12, pages 121–132, 2012.

[3] Michael A. Bender, Richard Cole, Erik D. Demaine, Martin Farach-Colton, and Jack Zito. *Two simplified algorithms for maintaining order in a list.* In Proceedings of the 10th Annual European Symposium on Algorithms, ESA '02, pages 152–164, 2002.

[4] Michael A. Bender, Erik D. Demaine, and Martin Farach-Colton. *Cache-oblivious B-trees.* SIAM J. Comput., 35:341–358, 2005.

[5] Michael A. Bender, Ziyang Duan, John Iacono, and Jing Wu. *A locality-preserving cache-oblivious dynamic dictionary.* Journal of Algorithms, 53(2):115 – 136, 2004.

[6] Michael A. Bender, and Haodong Hu. *An adaptive packed-memory array.* ACM Transactions on Database Systems, 32(4):26, 2007.

[7] Richard S. Bird and Stefan Sadnicki. *Minimal on-line labelling.* Information Processing Letters, 101:41–45, 2007.

[8] Gerth Stølting Brodal, Rolf Fagerberg, and Riko Jacob. *Cache oblivious search trees via binary trees of small height.* In Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms, SODA '02, pages 39–48. Society for Industrial and Applied Mathematics, 2002.

[9] Jan Bulánek, Michal Koucký, and Michael Saks. *On Randomized Online Labeling with Polynomially Many Labels.* In Proceedings of the 40th International Colloquium on Automata, Languages and Programming, ICALP '13, pages 291-302, 2013.

[10] Paul F. Dietz. *Maintaining Order in a Linked List*, in Proceedings of the 14th Annual ACM Symposium on Theory of Computing, STOC 1982, pages 122–127.

[11] Paul F. Dietz, Joel I. Seiferas, and Ju Zhang. *Lower bounds for smooth list labeling.* Manuscript, 2005. (Listed in the references of [12]).

[12] Paul F. Dietz, Joel I. Seiferas, and Ju Zhang. *A tight lower bound for online monotonic list labeling.* SIAM J. Discret. Math., 18:626–637, 2005.

[13] Paul F. Dietz and Ju Zhang. *Lower bounds for monotonic list labeling.* In SWAT, pages 173–180, 1990.

[14] Yuval Emek and Amos Korman. *New bounds for the controller problem.* Distributed Computing, 24(3-4):177–186, 2011.

[15] Alon Itai, Alan G. Konheim, and Michael Rodeh. *A sparse table implementation of priority queues.* In Proceedings of the 8th Colloquium on Automata, Languages and Programming, pages 417–431, 1981.

[16] Tsvi Kopelowitz. *On-Line Indexing for General Alphabets via Predecessor Queries on Subsets of an Ordered List.* In Proceedings 53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS), pages 283–292, 2012.

[17] Amos Korman and Shay Kutten. *Controller and estimator for dynamic networks.* In PODC, pages 175–184, 2007.

[18] Joel Seiferas. Personal communication, 2011.

[19] Dan E. Willard. *A density control algorithm for doing insertions and deletions in a sequentially ordered file in a good worst-case time.* Inf. Comput., 97:150–204, April 1992.

[20] Ju Zhang. *Density Control and On-Line Labeling Problems.* PhD thesis, University of Rochester, Rochester, NY, USA, 1993.