On Randomized Online Labeling with Polynomially Many Labels

Jan Bulánek[⋆1,2], Michal Koucký[⋆⋆2], and Michael Saks[⋆⋆⋆3]

[1] Department of Theoretical Computer Science and Mathematical Logic, Charles University, Prague

[2] Institute of Mathematics, Academy of Sciences CR, Prague

[3] Department of Mathematics, Rutgers University

Abstract. We prove an optimal lower bound on the complexity of *randomized* algorithms for the online labeling problem with polynomially many labels. All previous work on this problem (both upper and lower bounds) only applied to deterministic algorithms, so this is the first paper addressing the (im)possibility of faster randomized algorithms. Our lower bound $\Omega(n \log(n))$ matches the complexity of a known deterministic algorithm for this setting of parameters so it is asymptotically optimal.

In the online labeling problem with parameters n and m we are presented with a sequence of n *items* from a totally ordered universe U and must assign each arriving item a label from the label set $\{1, 2, \ldots, m\}$ so that the order of labels (strictly) respects the ordering on U. As new items arrive it may be necessary to change the labels of some items; such changes may be done at any time at unit cost for each change. The goal is to minimize the total cost. An alternative formulation of this problem is the *file maintenance problem*, in which the items, instead of being labeled, are maintained in sorted order in an array of length m, and we pay unit cost for moving an item.

1 Introduction

In the online labeling problem with parameters n, m, r, we are presented with a sequence of n *items* from a totally ordered universe U of size r and must assign each arriving item a label from the label set $\{1, 2, \ldots, m\}$ so that the order of labels (strictly) respects the ordering on U. As new items arrive it may be necessary to change the labels of some items; such changes may be done at any time at unit cost for each change. The goal is to minimize the total cost. An alternative formulation of this problem is the *file maintenance problem*, in which the items, instead of being labeled, are maintained in sorted order in an array of length m, and we pay unit cost for moving an item.

The problem, which was introduced by Itai, Konheim and Rodeh [13], is natural and intuitively appealing, and has had applications to the design of data structures (see for example the discussion in [10], and the more recent work on cache-oblivious data structures [4, 8, 5]). A connection between this problem and distributed resource allocation was recently shown by Emek and Korman [12].

⋆ Partially supported by student project GAUK 344711 and RVO: 67985840.

⋆⋆ Supported in part by grant IAA100190902 of GA AV ČR, Center of Excellence CE-ITI (P202/12/G061 of GA ČR) and RVO: 67985840.

⋆⋆⋆ The work of this author was done while on sabbatical at Princeton University and was also supported in part by NSF under grant CCF-0832787 and CCF-1218711.

The parameter m, the size of the *label space* must be at least the number of items n or else no valid labeling is possible. There are two natural ranges of parameters which have received the most attention. In the case of *linearly many labels* we have $m = cn$ for some $c > 1$, and in the case of *polynomially many labels* we have $m = \theta(n^C)$ for some constant $C > 1$. The size r of the universe U is also a parameter which is not discussed explicitly in most of the literature on the problem. If $r \leq m$, the problem can be solved with cost n, since then we can simply fix an order preserving bijection from U to $\{1, \ldots, m\}$ in advance. In this paper we assume $U = \{1, \ldots, 2^n\}$.

Itai et al. [13] gave an algorithm for the case of linearly many labels having worst case total cost $O(n \log(n)^2)$. Improvements and simplifications were given by Willard [15] and Bender et al. [3]. In the special case that $m = n$, algorithms with cost $O(\log(n)^3)$ per item are known [16, 6]. It is also well known that the algorithm of Itai et al. can be adapted to give total cost $O(n \log(n))$ in the case of polynomially many labels. All of these algorithms are deterministic.

In a previous paper [9], we proved a $\Omega(n \log(n)^2)$ lower bound in the case of linearly many labels, and $\Omega(n \log(n)^3)$ lower bound for the case $m = n$. In subsequent work with Babka and Čunát [2] we proved a lower bound $\Omega(n \log(n)/(\log\log(m) - \log\log(n)))$ when $n^{1+\varepsilon} \leq m \leq 2^{n^{\varepsilon}}$, In particular, this gives a $\Omega(n \log(n))$ bound for the case of m being polynomial in n. These lower bounds match the known upper bounds to within a constant factor. Both of these papers built heavily on previous partial results of Dietz, Seiferas and Zhang ([16, 11, 10]). These lower bounds apply only to *deterministic* algorithms, leaving open the possibility of better randomized algorithms.

In this paper we use a model in which the cost of a randomized labeling algorithm is the worst case over all input sequences of a given length n of the expected number of moves made by the algorithm. This corresponds to running the algorithm against an *oblivious adversary* (see [7]) who selects the input sequence having full knowledge of the algorithm, but not of the random bits flipped in the execution of the algorithm.

There are many online problems where randomized algorithms perform provably better than deterministic ones. For example, the best deterministic algorithm for the paging problem with k pages has competitive ratio k but there are randomized algorithms having competitive ratio $\Theta(\log(k))$ [7].

Our Results. In this paper we establish the first lower bound for *randomized* online labeling algorithms by showing that in the case of polynomially many labels any randomized online labeling algorithm will have expected cost $\Omega(n \log(n))$ (for the worst case input). This matches the known deterministic upper bounds up to constant factors, and thus randomization provides no more than a constant factor advantage over determinism. Our bound also implies an $\Omega(n \log(n))$ lower bound on the message complexity of randomized protocols for Distributed Controller Problem [12, 1].

Unlike many other lower bounds for non-uniform computation models, our proof does not use Yao's principle. Yao's principle says (roughly) that to prove a lower bound on the expected cost of an arbitrary randomized algorithm it suffices to fix a distribution over inputs, and prove a lower bound on the expected cost of a deterministic algorithm against the chosen distribution. Rather than use Yao's principle, our proof takes an arbitrary randomized algorithm and selects a (deterministic) sequence that is hard for that algorithm.

The construction and analysis of the hard sequence follow the same overall strategy of the previous lower bound for deterministic algorithms in the case of polynomially many labels [10, 2] which involves relating online labeling to a family of one-player games called *bucketing games* (introduced in [10]) which involve the sequential placement of items into an ordered sequence of bins subject to certain rules and costs. We define a map (an *adversary*) which associates to a labeling algorithm **A** a hard sequence of items. We then show that the behavior of the algorithm on this hard sequence can be associated to a strategy for playing a particular bucketing game, such that the cost incurred by the algorithm on the hard sequence is bounded below by the cost of the associated bucketing game strategy. Finally we prove a lower bound on the cost of any strategy for the bucketing game, which therefore gives a lower bound on the cost of the algorithm on the hard input sequence.

In extending this argument from the case of deterministic algorithms to the randomized case, each part of the proof requires significant changes. The adversary which associates an algorithm to a hard sequence requires various careful modifications. The argument that relates the cost of **A** on the hard sequence to the cost of an associated bucketing strategy does not work for the original version of the bucketing game, and we can only establish the connection to a new variant of the bucketing game called tail-bucketing. Finally the lower bound proof on the cost of any strategy for tail-bucketing is quite different from the previous lower bound for the original version of bucketing.

Mapping a randomized algorithm to a hard input sequence. We now give an overview of the adversary which maps an algorithm to a hard input sequence $y_1, \ldots, y_n$. The adversary is deterministic. Its behavior will be determined by the expected behavior of the randomized algorithm. Even though we are choosing the sequence obliviously, without seeing the actual responses of the algorithm, we view the selection of the sequence in an online manner. We design the sequence item by item. Having selected the first $t-1$ items, we use the known randomized algorithm to determine a probability distribution over the sequence of labelings determined by the algorithm after each step. We then use this probability distribution to determine the next item, which we select so as to ensure that the expected cost incurred by the algorithm is large.

The adversary will maintain a hierarchy consisting of a nested sequence of subsets of the set of items inserted so far. The hierarchy serves a dual purpose: the hierarchy after step t is used by the adversary to select the item inserted at step $t+1$, and the sequence of hierarchies over time provides a way to lower bound the total (expected) cost incured by the algorithm. This hierarchy is denoted[4]

$$S_t(1) \supset T_t(2) \supset S_t(2) \supset T_t(3) \supset \cdots \supset T_t(d) \supset S_t(d).$$

The set $S_t(1)$ consists of all items inserted through step t. Each of the other sets is an interval relative to the items inserted so far, i.e., it consists of all inserted items in a given interval. The final subset $S_t(d)$ has between 2 and 6 elements. (Note that this differs from the previous work for deterministic algorithms where the hierarchy was a nested sequence of intervals of label values rather than items; this modification seems necessary to handle randomized algorithms). The next item to be inserted is selected to be an item that is between two items in the final set $S_t(d)$.

The hierarchy at step t is constructed as follows. The hierarchy for $t=0$ has $d_0 = 1$ and $S_0(1) = \{0, 2^n\}$. The hierarchy at step $t \geq 1$ is constructed based on the hierarchy at the previous step $t-1$ and the expected behavior of the algorithm on $y_1, \ldots, y_t$.

We build the sets for the hierarchy at step t in order of increasing level (i.e., decreasing size). Intervals are either *preserved* (carried over from the previous hierarchy, with the

[4] In this paper we work with various vectors and sequences that change over time. We use subscript to denote time and we use $(\cdot)$ notation to denote a particular coordinate of such a vector or sequence at that time.

addition of y_t) or *rebuilt*. To specify which intervals are preserved, we specify a *critical level for step* t, q_t which is at most the depth d_{t-1} of the previous hierarchy. We'll explain the choice of q_t below. At step t, the intervals $T_t(i)$ and $S_t(i)$ for $i \leq q_t$ are *preserved*, which means that it is obtained from the corresponding interval at step $t-1$ by simply adding y_t. The intervals $T_t(i)$ and $S_t(i)$ for $i \geq q_t$ are *rebuilt*. The rule for rebuilding the hierarchy for $i > q_t$ is defined by induction on i as follows: Given $S_t(i-1)$, $T_t(i)$ is defined to be either the first or second half of $S_t(i-1)$, depending on which of these sets is more likely to have a smaller range of labels (based on the distribution over labels determined by the given algorithm). More precisely, we look at the median item of $S_t(i-1)$ and check whether (based on the randomized labeling) it is more likely that its label is closer to the label of the minimum or to the maximum element of $S_t(i-1)$. If the median is more likely to have label close to the minimum we pick the first half as $T_t(i)$ otherwise the second half. Having chosen $T_t(i)$, we take $S_t(i)$ to be the middle third of items in $T_t(i)$. This process terminates when $|S_t(i)| < 7$ and the depth d_t of the hierarchy is set to this final i. The adversary selects the next requested item y_{t+1} to be between two items in $S_t(d)$.

This construction of the hierarchy is similar to that used in [2] in the deterministic case. An important difference comes in the definition of the critical level q_t. In the deterministic case the critical level is the smallest index i such that neither endpoint of $T_{t-1}(i)$ was moved by the algorithm when inserting y_t. In the randomized case we need a probabilistic version of this: the critical level is the smallest index i such that the probability that either endpoint of $T(i)$ was moved since the last step it was rebuilt is less than $1/4$.

One of the crucial requirements in designing the adversary is that the hierarchy never grows too deep. Note that when we rebuild $T_t(i)$ it's size is at most $|S_t(i-1)|/2$ and when we rebuild $S_t(i)$ its size is at most $|T_t(i)|/3$. This suggests that as we proceed through the hierarchy each set is at most $1/2$ the size of the previous and so the depth is at most $\log(n)$. This reasoning is invalid because during a sequence of steps in which a set in the hierarchy is not rebuilt its size grows by 1 at each step and so the condition that the set is at most half the size of its predecessor may not be preserved. Nevertheless we can show that the depth never grows to more than $4\log(m+1)$ levels.

A lower bound on the expected cost of the algorithm on the hard sequence. In [9] it is noted that we can assume without loss of generality that the algorithm is *lazy*, in the sense that for each step the set of relabeled items is a sub-interval of the inserted items that contains the most recently inserted item. (Intuitively, a non-lazy algorithm can be modified so that any relabeling that violates laziness is deferred until later). This observation extends to randomized algorithms.

In the deterministic case, this assumption and the definition of the critical level q_t can be used to show that when the algorithm responds to the item y_t it moved at least a constant fraction of the items belonging to $S_{t-1}(q_t+1)$ and so the total cost of the algorithm is at least $DLB = \Omega(\sum_t |S_{t-1}(q_t+1)|)$. In the randomized case we get a related bound that the expected total number of moves is $RLB = \Omega(\sum_t |S_{t-1}(q_t) \setminus S_{t-1}(q_t+1)|)$. So the cost incured by the algorithm is related to the extent of changes in the hierarchy.

Bucketing games. The next step in the analysis is to define bucketing games, and to show that the lower bound on the cost of the algorithm given in the previous paragraph is an upper bound on the cost of an appropriate bucketing game.

The prefix bucketing game with n items and k buckets is a one player game. The game starts with k empty buckets indexed $1, \ldots, k$. At each step the player places an item in some bucket p. All the items from buckets $1, \ldots, p-1$ are then moved into bucket

p as well, and the cost is the number of items in buckets $1, \ldots, p$ before the merge, which is the number of items in bucket p after the merge. The goal is to select the sequence of indices so as to minimize the total cost. The total cost is the sum of the costs of each step. The goal is to select the sequence of indexes p so that we would minimize the total cost. In [2] (following [10]) it is shown that any deterministic labeling algorithm could be associated to a bucketing strategy such that the cost of the labeling algorithm against our adversary is at least a constant times the cost of the bucketing strategy. This result is deduced using the lower bound of $\Omega(\sum_t |S_{t-1}(q_t+1)|)$ for the cost of the algorithm mentioned earlier. It was also shown in [10] (see also [2]) that the minimal cost of any bucketing strategy (for more than $2\log(n)$ buckets) is $\Omega(n\log(n)/(\log(k) - \log\log(n))$. These results together gave the lower bound on deterministic labeling.

We use the same basic idea for the randomized case, but require several significant changes to the game. The first difficulty is that the lower bound on the cost of the randomized algorithm stated earlier, RLB, is not the same as the lower bound DLB that was known for deterministic algorithms. While DLB was shown to be at least the minimal cost of the prefix bucketing, this is not true for RLB. To relate RLB to bucketing, we must replace the cost function in bucketing by a smaller cost function, which is the number of items in the bucket p *before* the merge, not after. In general, this cost function is less expensive (often much less expensive) than the original cost function and we call it the *cheap* cost function. The argument relating the cost of a randomized algorithm to a bucketing strategy requires that the number of buckets be at least $4\log(m)$ buckets. If we could prove a lower bound on the cost of bucketing under the cheap function similar to the bound mentioned above for the original function this would be enough to deduce the desired lower bound on randomized labeling. However with this cheap cost function this lower bound fails: if the number of buckets is at least $1 + \log(n)$, there is a bucketing strategy that costs 0 with the cheap cost function! (For example a strategy which always picks p to be the smallest index of an empty bucket has cost zero; it emulates incrementing a binary counter.) So this will not give any lower bound on the cost of a randomized labeling algorithm

We overcome this problem by observing that we may make a further modification of the rules for bucketing and still preserve the connection between the cost of a randomized algorithm against our adversary and the cheap cost of a bucketing. This modification is called *tail bucketing*. In a tail bucketing, after merging all the items into the bucket p, we redistribute these items back among buckets $1, \ldots, p$, so that bucket p keeps $1-\beta$ fraction of the items and passes the rest to the bucket $p-1$, bucket $p-1$ does the same, and the process continues down until bucket 1 which keeps the remaining items. It turns our that our adversary can be related to tail bucketing for $\beta = 1/6$. We can prove that the minimal *cheap* cost of tail bucketing is $\Omega(n\log(n))$ when $k = O(\log(n))$. This lower bound is asymptotically optimal and yields a similar bound for randomized online labeling.

The lower bound proof for the cheap cost of tail bucketing has some interesting twists. The proof consists of several reductions between different versions of bucketing. The reductions show that we can lower bound the cheap cost of tail bucketing with $C\log(n)$ buckets (for any C) by the cheap cost of ordinary prefix bucketing with $k = \frac{1}{4}\log(n)$ buckets. Even though the cheap cost of ordinary bucketing dropped to 0 once $k = \log(n) + 1$, we are able to show that for $k = \frac{1}{4}\log(n)$ there is a $\theta(n\log(n))$ bound for ordinary bucketing with the cheap cost.

2 The Online Labeling Problem

We first define the deterministic version of online labeling. We have parameters $n \leq m < r$, and are given a sequence of n numbers from the set $U = [1, r]$ and must assign to each of them a label in the range $[1, m]$. (Here, and throughout the paper, interval notation is used for consecutive sets of integers). A *deterministic* online labeling algorithm A with parameters (n, m, r) is an algorithm that on input sequence $(y_1, y_2, \ldots, y_t)$ with $t \leq n$ of distinct elements from U outputs a *labeling* $f_A : \{y_1, y_2, \ldots, y_t\} \to [m]$ that respects the natural ordering of $y_1, \ldots, y_t$, that is for any $x, y \in \{y_1, y_2, \ldots, y_t\}$, $f_A(x) < f_A(y)$ if and only if $x < y$. We refer to $y_1, y_2, \ldots, y_t$ as *items*.

Fix an algorithm A. Any item sequence $y_1, \ldots, y_n$ determines a sequence $f_{A,0}, f_{A,1}, \ldots, f_{A,n}$ of labelings where $f_{A,t}$ is the labeling of $(y_1, \ldots, y_t)$ determined by A immediately after y^t was presented. When the algorithm A is fixed we omit the subscript A. We say that *A relabels $y \in \{y_1, y_2, \ldots, y_t\}$ at step t* if $f_{t-1}(y) \neq f_t(y)$. In particular, y_t is relabeled at step t. $Rel_t = Rel_{A,t}$ denotes the set of items relabeled at step t. The cost of A on $y_1, y_2, \ldots, y_n$ is $\chi_A(y_1, \ldots, y_n) = \sum_{t=1}^{n} |Rel_t|$.

A *randomized* online labeling algorithm $\mathbf{A}$ is a probability distribution on deterministic online labeling algorithms. Given an item sequence $y_1, \ldots, y_n$, the algorithm $\mathbf{A}$ determines a probability distribution over sequences of labelings $f_0, \ldots, f_n$. The set Rel_t is a random variable whose value is a subset of $y_1, \ldots, y_t$. The cost of $\mathbf{A}$ on $y_1, y_2, \ldots, y_n \in U$ is the expected cost $\chi_{\mathbf{A}}(y_1, \ldots, y_n) = \mathbf{E}[\chi_A(y_1, \ldots, y_n)]$. The maximum cost $\chi_{\mathbf{A}}(y_1, \ldots, y_n)$ over all sequences $y_1, \ldots, y_n$ is denoted $\chi_{\mathbf{A}}(n)$. We write $\chi_m(n)$ for the smallest cost $\chi_{\mathbf{A}}(n)$ that can be achieved by any algorithm $\mathbf{A}$ with range m.

We state our main theorem.

Theorem 1. *For any constant C_0, there are positive constants C_1 and C_2 so that the following holds. Let $\mathbf{A}$ be a randomized algorithm with parameters (n, m, r), where $n \geq C_1$, $r \geq 2^n$ and $m \leq n^{C_0}$. Then $\chi_{\mathbf{A}}(n) \geq C_2 n \log(n)$.*

To prove the theorem we will need some additional definitions. Let $S \subseteq Y \subseteq U$. We write $\min(S)$ and $\max(S)$ for the least and greatest elements, respectively. We say that S is a *Y-interval* if $S = Y \cap [\min(S), \max(S)]$. We write $\text{med}(S)$ for the *median* of S which we take to be the $\lceil |S|/2 \rceil$-th largest element of S. We define **left-half**$(S) = \{y \in S | y \leq \text{med}(S)\}$ and **right-half**$(S) = \{y \in S | y \geq \text{med}(S)\}$ (note that $\text{med}(S)$ is contained in both). Also define **left-third**(S) to be the smallest $\lfloor |S|/3 \rfloor$ elements, **right-third**(S) to be the largest $\lfloor |S|/3 \rfloor$ elements and **middle-third**$(S) = S -$ **left-third**$(S) -$ **right-third**(S).

Given a labeling f of Y and a Y-interval S, we say that the Y-interval S is *left-leaning* with respect to f if $\text{med}(S)$ has a label that is closer to the label of $\min(S)$ than it is to the label of $\max(S)$, i.e. $f(\text{med}(S)) - f(\min(S)) \leq f(\max(S)) - f(\text{med}(S))$. It is *right-leaning* otherwise.

A deterministic labeling algorithm is *lazy* if at each step t, the set of relabeled items is a Y_t-interval (which necessarily contains y_t), and a randomized algorithm is lazy if it is a distribution over lazy deterministic algorithms. In [9], it was shown that there is an optimal deterministic algorithm that is lazy, and the same proof works to show that there is an optimal lazy randomized algorithm. (Intuitively this is the case because if the relabeled items at step t do not form a Y_t-interval and W is the largest Y_t-interval of relabeled items containing y_t then we can defer relabeling the items in $Y_t \setminus W$ until later.)

2.1 The adversary

We now specify an adversary $\mathbf{Adversary}(\mathbf{A}, n, m)$ which given an online labeling algorithm $\mathbf{A}$, a length n, and label space size m, constructs a item sequence $y_1, y_2, \ldots, y_n$ from the universe $U = \{1, \ldots, 2^n - 1\}$. Our adversary and notation borrow from past work in the deterministic case [10, 9].

We think of the adversary as selecting $y_1, \ldots, y_n$ online, but after each step the adversary only knows a probability distribution over the configurations of the algorithm. It is important to keep in mind that the adversary knows the randomized algorithm $\mathbf{A}$ but does not know the random coins of the algorithm.

During the construction of the adversary sequence $y_1, \ldots, y_n$, the adversary will maintain a nested sequence of subsets of $\{y_1, \ldots, y_t\}$:

$$S_t(1) \supset T_t(2) \supset S_t(2) \supset T_t(3) \supset \cdots \supset T_t(d_t) \supset S_t(d_t)$$

called the *hierarchy at step* t. Each of the sets will be of size at least 2 and it will form $\{y_1, \ldots, y_t\}$-interval. The depth d_t of the hierarchy may vary with t. The sets $S_t(i)$ and $T_t(i)$ are said to be at *level* i in the hierarchy. This hierarchy serves a dual purpose: it allows us to select the next item y_{t+1} and it also allows us to estimate the cost of the algorithm.

To avoid having to deal with special cases in the description of the adversary, it is convenient to imagine that the set of items is augmented by items 0 and 2^n which are given (permanent) labels 0 and $m+1$ respectively. We write Y_t for the set $\{y_1, \ldots, y_t\} \cup \{0, 2^n\}$ of labeled items after step t. At the beginning of step t, having chosen items Y_{t-1} and having constructed the hierarchy $S_{t-1}(1) \supset \cdots \supset S_{t-1}(d_{t-1})$, the adversary will select y_t to be $\min(S_{t-1}(d_{t-1})) + 2^{n-t}$. It is easy to see by induction on t that the items belonging to Y_t are multiples of 2^{n-t}, and it follows that y_t is strictly between the smallest and second smallest elements of $S_{t-1}(d_{t-1})$. Therefore all of the chosen items are distinct.

We need to specify how we determine the hierarchy at step t. The pseudo-code for the adversary is given in Figure 2.1.

The hierarchy for $t = 0$ has $d_0 = 1$ and $S_0(1) = \{0, 2^n\}$. The hierarchy at step $t \geq 1$ is constructed based on the hierarchy at the previous step $t-1$ and the expected behavior of the algorithm on $y_1, \ldots, y_t$ as reflected by the joint probability distribution over the sequence of functions $f_{\mathbf{A},1}, \ldots, f_{\mathbf{A},t}$.

We build the sets for the hierarchy at step t in order of increasing level (i.e., decreasing size). Intervals are either *preserved* (carried over from the previous hierarchy, with the addition of y_t) or *rebuilt.* To specify which intervals are preserved, we specify a *critical level* q_t *for step* t, which is at least 1 and at most the depth d_{t-1} of the previous hierarchy. We'll explain the choice of q_t below. At step t, the intervals $S_t(i)$ for $i \leq q_t$ are *preserved*, which means that $S_t(i)$ is obtained simply by adding y_t to $S_{t-1}(i)$, and the rest are *rebuilt.* In particular, for $t \geq 7$, $S_t(1)$ is always preserved, and is equal to Y_t. The rule for rebuilding the hierarchy for $i > q_t$ is defined by induction on i as follows: If $|S_t(i-1)| < 7$ then the hierarchy is terminated with $d_t = i - 1$. Otherwise, consider the labeling of $S_t(i-1)$ by f_t (which is randomly distributed depending on $\mathbf{A}$). If the probability that $S_t(i-1)$ is left-leaning with respect to f_t is at least $1/2$, then set $T_t(i) = \mathbf{left\text{-}half}(S_t(i-1))$ otherwise $T_t(i) = \mathbf{right\text{-}half}(S_t(i-1))$. Set $S_t(i) = \mathbf{middle\text{-}third}(T_t(i))$. Observe that since $|S_t(i-1)| \geq 7$, we have $|T_t(i)| \geq 4$ and $|S_t(i)| \geq 2$.

It remains to explain how the critical level q_t is selected. When constructing each set $S_t(i)$ of the hierarchy for $i \geq 2$, the adversary defines a parameter $\mathbf{birth}_t(i)$ which is set equal to t if $S_t(i)$ is rebuilt, and is otherwise set to $\mathbf{birth}_{t-1}(i)$. It is easy to see (by induction on t), that $\mathbf{birth}_t(i)$ is equal to the largest step $u \leq t$ such that $S_u(i)$ was rebuilt. It follows that for each $u \in [\mathbf{birth}_t(i), t]$, $\min(T_u(i)) = \min(T_t(i))$ and $\max(T_u(i)) = \max(T_t(i))$.

Say that item y has *stable label* during interval $[a, b]$ if the label $f_u(y)$ is the same for all u in $[a, b]$, and has *unstable label* on $[a, b]$ otherwise. We define the event $\mathbf{stable}_t(i)$ to be the event (depending on $\mathbf{A}$) that both $\min(T_t(i))$ and $\max(T_t(i))$ have stable labels during interval $[\mathbf{birth}_{t-1}(i), t]$.

We are finally ready to define q_t. If there is at least one level $i \geq 2$ for which $\mathbf{Pr}[\mathbf{stable}_t(i)] \leq 3/4$, let $i_{\min}$ be the least such level, and choose $q_t = i_{\min} - 1$. Otherwise set $q_t = d_{t-1}$.

```
**Adversary**($\mathbf{A}, n, m$)
$t = 0$: $S_0(1) \longleftarrow \{0, 2^n\}$ and $d_0 = 1$.
For $t = 1, \ldots, n$ do
  – $y_t \longleftarrow \min(S_{t-1}(d_{t-1})) + 2^{n-t}$.
  – (Choose critical level) Consider the sequence of (dependent) random functions $f_1, \ldots, f_t$
    produced by $\mathbf{A}$ in response to $y_1, \ldots, y_t$. If there is an index $i \geq 2$ for which $\mathbf{Pr}[\mathbf{stable}_t(i)] \leq$
    $3/4$, let $i_{\min}$ be the least such index and let $q_t = i_{\min} - 1$. Otherwise set $q_t = d_{t-1}$.
  – $S_t(1) \longleftarrow S_{t-1}(1) \cup \{y_t\}$.
  – $i \longleftarrow 1$.
  – (Preserve intervals up to the critical level) While $i < q_t$ do:
      • $i \longleftarrow i + 1$.
      • $T_t(i) \longleftarrow T_{t-1}(i) \cup \{y_t\}$.
      • $S_t(i) \longleftarrow S_{t-1}(i) \cup \{y_t\}$.
      • $\mathbf{birth}_t(i) \longleftarrow \mathbf{birth}_{t-1}(i)$.
  – (Build intervals after the critical level) While $|S_t(i)| \geq 7$ do:
      • $i \longleftarrow i + 1$.
      • If $S_t(i-1)$ is left-leaning with respect to $f_t$ with probability at least $1/2$ then $T_t(i) \longleftarrow$
        **left-half**$(S_t(i-1))$ otherwise $T_t(i) \longleftarrow$ **right-half**$(S_t(i-1))$.
      • $S_t(i) \longleftarrow$ **middle-third**$(T_t(i))$.
      • $\mathbf{birth}_t(i) \longleftarrow t$. [Record that $S_t(i)$ and $T_t(i)$ were rebuilt]
  – $d_t \longleftarrow i$.
*Output:* $y_1, y_2, \ldots, y_n$.
```

Fig. 1. Pseudocode for the adversary

We will prove the following lemma about the adversary, which immediately implies Theorem 1.

Lemma 1. *Let $c \geq 1$ be an arbitrary constant and n, m be large enough integers such that $m < (n+1)^c$. Let $\mathbf{A}$ be a lazy randomized online labeling algorithm with the range m. Let $y_1, y_2, \ldots, y_n$ be the output of* **Adversary**$(\mathbf{A}, n, m)$. *Then the cost satisfies:*

$$\chi_{\mathbf{A}}(y_1, y_2, \ldots, y_n) \geq \frac{1}{96}\left(\frac{1}{6}\right)^{512c^2}(n+1)\log(n+1) - \frac{n}{4}.$$

The proof of this lemma has two main steps. The first step is to bound the cost $\chi_{\mathbf{A}}(y_1, \ldots, y_n)$ from below by the minimum cost of a variant of the prefix-bucketing game. The prefix-bucketing game was introduced and studied before to get lower bounds for deterministic online labeling. The variant we consider is called *tail-bucketing*. The second step is to give a lower bound on the cost of tail-bucketing.

To prove the first step we will need two properties of **Adversary**$(\mathbf{A}, n, m)$. **Adversary**$(\mathbf{A}, n, m)$ determines $y_1, \ldots, y_n$ and the critical levels $q_1, \ldots, q_n$.

Lemma 2. *For any $t \in [1, n]$, $d_t \leq 4\log(m+1)$.*

Lemma 3. *The cost of* $\mathbf{A}$ *on* $y_1, \ldots, y_n$ *satisfies:*

$$\chi_{\mathbf{A}}(y_1, y_2, \ldots, y_n) \geq \frac{1}{40}\sum_t |S_{t-1}(q_t) \setminus S_{t-1}(1+q_t)|,$$

where the sum ranges over steps $t \in [1, n]$ for which $q_t < d_{t-1}$.

For the proofs of these two lemmas we need certain random variables associated with the execution of $\mathbf{A}$ on $y_1, \ldots, y_n$. Since all of the randomness comes from the distribution over $\mathbf{A}$, the value of each random variable is determined by the random selection of $\mathbf{A}$, and we sometimes subscript random variables by $\mathbf{A}$ to emphasize this dependence. Furthermore, we replace $\mathbf{A}$ by a deterministic algorithm A in the subscript to indicate the value of the random variable when $\mathbf{A} = A$. We make the following definitions.

– For any subset S of Y_t, $\mathbf{length}_{\mathbf{A},t}(S) = f_{\mathbf{A},t}(\max(S)) - f_{\mathbf{A},t}(\min(S))$.
– For a (i, t) such that $i < d_t$, $\mathbf{shrink}_{\mathbf{A},t}(i)$ is the 0-1 indicator of the event that

$$\mathbf{length}_{\mathbf{A},t}(S_t(i+1)) \leq \mathbf{length}_{\mathbf{A},t}(S_t(i))/2.$$

– Define $\mathbf{shrink}_{\mathbf{A},t} = \sum_{i=1}^{d_t-1} \mathbf{shrink}_{\mathbf{A},t}(i)$.

Proof of Lemma 2. For $t = 1$ the claim is trivial so assume $t > 1$. For any algorithm A, $\mathbf{length}_{A,t}(S_t(1)) = m+1$ and $\mathbf{length}_{A,t}(S_t(d_t)) \geq 2$, and $\mathbf{length}_{A,t}(S_t(i)) > \mathbf{length}_{A,t}(S_t(i+1))$ for $i \in [1, d_t - 1]$. Therefore $\mathbf{shrink}_{A,t}(i)$ can be 1 for at most $\log(m+1) - 1$ values of i. Thus $\mathbf{shrink}_{A,t} \leq \log(m+1) - 1$.

Next we claim and prove below that for $i \in [1, d_t - 1]$, $\mathbf{Pr}[\mathbf{shrink}_{\mathbf{A},t}(i) = 1] \geq 1/4$. This claim implies $\mathbf{E}[\mathbf{shrink}_{\mathbf{A},t}] \geq (d_t - 1)/4$ which then gives $d_t \leq 4\log(m+1)$ to complete the proof of the lemma.

So it remains to prove the claim. Consider first the case that $i + 1 > q_t$. Sets $T_t(i+1)$ and $S_t(i+1)$ are rebuilt at step t. By definition of the adversary $T_t(i+1)$ is either **left-half**$(S_t(i))$ or **right-half**$(S_t(i))$. Furthermore this choice is made so that $\mathbf{length}_t(T_t(i+1)) \leq \mathbf{length}_t(S_t(i))/2$ with probability at least 1/2 and since $S_t(i+1) \subseteq T_t(i+1)$, $\mathbf{Pr}[\mathbf{shrink}_t(i) = 1] \geq 1/2$.

Next consider the case that $i+1 \leq q_t$ so that $T_t(i+1)$ and $S_t(i+1)$ are preserved at step t. These intervals were most recently rebuilt at step $s = \mathbf{birth}_t(i+1) = \mathbf{birth}_{t-1}(i+1)$ and the endpoints of $T_u(i+1)$ are the same for all $u \in [s, t]$. Since $i + 1 > 1$, $s > 1$. Since $i + 1 > q_s$, $\mathbf{Pr}[\mathbf{shrink}_s(i) = 1] \geq 1/2$. We now claim and prove below that if both $\mathbf{shrink}_s(i)$ and $\mathbf{stable}_t(i+1)$ happen then $\mathbf{shrink}_t(i)$ happens. From this claim, and the assumption that $i+1 \leq q_t$ we deduce: $\mathbf{Pr}[\mathbf{shrink}_t(i)] \geq \mathbf{Pr}[\mathbf{shrink}_s(i) \cap \mathbf{stable}_t(i+1)] \geq \mathbf{Pr}[\mathbf{shrink}_s(i)] + \mathbf{Pr}[\mathbf{stable}_t(i+1)] - 1 \geq 1/2 + 3/4 - 1 = 1/4$, as required.

To see the final claim, assume that the event $\mathbf{stable}_t(i+1)$ occurred. For each endpoint of $T_t(i+1)$, its label remained the same under each of the functions $f_s, \ldots, f_t$, and by the laziness of the algorithm, it also happened that for each endpoint of $S_t(i)$, its label remained the same under each of the functions $f_s, \dots, f_t$. Thus if, in addition, $\mathbf{shrink}_s(i)$ happens then so does $\mathbf{shrink}_t(i)$. $\square$

Proof of Lemma 3. An *item-step pair* (y, u) is a pair where $y \in Y_u$. For each step t such that $q_t < d_{t-1}$ we will define a set W_t of item-step pairs. The sets W_t will be disjoint for different steps t and will consist of some set of item-step pairs (y, u) with $u \leq t$. Say that the item-step pair (y, u) is a *relabel event* if $f_u(y) \neq f_{u-1}(y)$. Define $\mathbf{relabs}_t$ be the (random) number of relabel events in W_t. It follows that the cost of the algorithm is at least $\sum_{t:q_t<d_{t-1}} \mathbf{E}[\mathbf{relabs}_t]$. We will show that $\mathbf{E}[\mathbf{relabs}_t] \geq \frac{1}{40}|S_t(q_t) \setminus S_t(1+q_t)|$, which will suffice to prove the lemma.

We now define W_t for each t such that $q_t < d_{t-1}$. Let $b_t = \mathbf{birth}_{t-1}(1+q_t)$. For all steps $u \in [b_t + 1, t)$ the sets $T_u(1+q_t)$ are preserved and also the sets $S_u(1+q_t)$ are preserved and so from step $u-1$ to u they each change only by the addition of y_u. Defining for all steps s and levels i, $\Delta_s(i) = T_s(i) \setminus S_s(i)$, we have that the sets $\Delta_u(1+q_t)$ are all the same for each $u \in [b_t, t)$. We define W_t to be the set of pairs (y, u) with $y \in \Delta_{u-1}(1+q_t)$ and $u \in (b_t, t]$, i.e., $W_t = \Delta_{t-1}(1+q_t) \times [b_t+1, t]$.

We now show that the sets W_t and $W_{t'}$ are disjoint for all pairs of steps $t < t'$. Suppose for contradiction that $W_t \cap W_{t'} \neq \emptyset$. Then $(b_t, t] \cap (b_{t'}, t'] \neq \emptyset$ and so $\mathbf{birth}_{t'-1}(1+q_{t'}) = b_{t'} < t$. This means that level $1+q_{t'}$ is not rebuilt at step t but level $1+q_t$ is rebuilt at step t, so $q_t > q_{t'}$. But then this contradicts $\Delta_{t-1}(1+q_t) \cap \Delta_{t'-1}(1+q_{t'}) \neq \emptyset$ since $\Delta_{t-1}(1+q_t) \subset T_{t-1}(1+q_t) \subset S_{t-1}(1+q_{t'}) \subset S_{t'-1}(1+q_{t'})$ while $\Delta_{t'-1}(1+q_{t'}) \cap S_{t'-1}(1+q_{t'}) = \emptyset$.

Finally, let us bound $\mathbf{E}[\mathbf{relabs}_t]$ from below. By the definition of the adversary $\Delta_{t-1}(1+q_t)$ is the union of the two equal-sized sets $\textbf{left-third}(T_{b_t}(1+q_t)) \cup \textbf{right-third}(T_{b_t}(1+q_t))$. By the definition of q_t, the probability that both $\min(T_{b_t}(1+q_t))$ and $\max(T_{b_t}(1+q_t))$ have stable label during $[b_t, t]$ is at most $3/4$. By the laziness of the algorithm, on any run in which the left (resp. right) endpoint of $T_{t-1}(1+q_t)$ has unstable label during $[b_t, t]$ all items in $\textbf{left-third}(T_{b_t}(1+q_t))$ (resp. $\textbf{right-third}(T_{b_t}(1+q_t))$) have unstable label during $[b_t, t]$ and so at least half the items of $\Delta_{t-1}(1+q_t)$ have unstable label during $[b_t, t]$. Since this occurs with probability at least $1/4$, thus the expected number of relabel events is at least $|\Delta_{t-1}(1+q_t)|/8$.

To complete the proof of the lemma, we show that $|\Delta_{t-1}(1+q_t)| \geq \frac{1}{5}|S_{t-1}(q_t) \setminus S_{t-1}(1+q_t)|$. The sets $S_u(q_t) \setminus S_u(1+q_t)$ are the same for all $u \in [b_t, t)$ and the same is true for the sets $\Delta_u(1+q_t)$. We compare these two sets for $u = b_t$. Letting $c = |S_{b_t}(q_t)|$ we have $c \geq 7$ since q_t is not the last level at step b_t. Since $T_{b_t}(1+q_t)$ and $S_{b_t}(1+q_t)$ are rebuilt, $|T_{b_t}(1+q_t)| \geq \lceil c/2 \rceil$ and $|\Delta_{b_t}(1+q_t)| \geq 2\lfloor(\lceil c/2 \rceil)/3\rfloor \geq c/5$ (where the final inequality uses $c \geq 7$, and is tight for $c = 10$). $\square$

3 Prefix bucketing and Tail bucketing

We will need several variants of the prefix bucketing game introduced by Dietz, Seiferas and Zhang [10]. We have k buckets numbered $1, \dots, k$ in which items are placed. A *bucket configuration* is an arrangement of items in the buckets; formally it is a mapping $C : \{1, \dots, k\}$ to the nonnegative integers, where $C(i)$ is the number of items in bucket i. It will sometimes be convenient to allow the range of the function C to be the nonnegative real numbers, which corresponds to allowing a bucket to contain a fraction of an item.

A *bucketing game* is a one player game in which the player is given a sequence of groups of items of sizes $n_1, \dots, n_\ell$ and must sequentially place each group of items into a bucket. The case that $n_1 = \cdots = n_\ell = 1$ is called *simple bucketing*. The placement is done in ℓ steps, and the player selects a sequence $p_1, \ldots, p_\ell \in [1,k]^\ell$, called an (ℓ,k)-*placement sequence* which specifies the bucket into which each group is placed.

Bucketing games vary depending on two ingredients, the *rearrangement rule* and the *cost functions.*

When a group of m items is placed into bucket p, the items in the configuration are rearranged according to a specified rearrangement rule, which is not under the control of the player. Formally, a rearrangement rule is a function R that takes as input the current configuration C, the number m of new items being placed and the bucket p into which they are placed, and determines a new configuration $R(C,m,p)$ with the same total number of items.

The *prefix rearrangement rule* is as follows: all items currently in buckets below p are moved to bucket p. We say that items are *merged into bucket* p. Formally, the new configuration $C' = R(C,m,p)$ satisfies $C'(i) = 0$ for $i < p$, $C'(p) = C(1) + \cdots + C(p) + m$ and $C'(i) = C(i)$ for $i > p$. Most of the bucketing games we'll discuss use the prefix rearrangement function, but in Section 3.1 we'll need another rearrangement rule.

The *cost function* specifies a cost each time a placement is made. For the cost functions we consider the cost of placing a group depends on the current configuration C and the selected bucket p but not on the number m of items being placed. We consider four cost functions

– In *cheap* bucketing, the cost is the number of items in bucket p before the placement:

$$\mathbf{cost}_{\mathbf{cheap}}(C,p) = C(p).$$

– In *expensive* bucketing, the cost is the number of items in buckets p or higher before the placement:

$$\mathbf{cost}_{\mathbf{exp}}(C,p) = \sum_{i=p}^{k} C(i).$$

– For $\gamma \in [0,1]$, in the γ-*discounted* bucketing, the cost is:

$$\mathbf{cost}_{\gamma-\mathbf{disc}}(C,p) = \sum_{i=p}^{k} C(i)\gamma^i.$$

(Note that $\mathbf{cost}_{1-\mathbf{disc}} = \mathbf{cost}_{\mathbf{exp}}$.)

– For $b \in \mathbb{N}$, in the b-*block* bucketing, the cost of step t is

$$\mathbf{cost}_{b-\mathbf{block}}(C,p) = \sum_{i=p}^{s(p)} C(i),$$

where $s(p)$ is the least multiple of b larger or equal to p. (Note that $\mathbf{cost}_{1-\mathbf{block}} = \mathbf{cost}_{\mathbf{cheap}}$ and $\mathbf{cost}_{k-\mathbf{block}} = \mathbf{cost}_{\mathbf{exp}}$.)

For completeness we remark that the cost function used in previous work [10, 2] is the number of items in buckets $1, \ldots, p$ before the placement:

$$\mathbf{cost}(C,p) = \sum_{i=1}^{p} C(i).$$

Fix a rearrangement rule R and a cost function c. A placement sequence $p_1, \ldots, p_\ell$ and a load sequence $n_1, \ldots, n_\ell$ together determine a sequence of configurations $B = (B_0, B_1, \ldots, B_\ell)$, called a *bucketing* where B_0 is the empty configuration and for $i \in [1, \ell]$, $B_i = R(B_{i-1}, n_i, p_i)$. Each of these ℓ placements is charged a cost according to the cost rule c. We write $c[R](p_1, \ldots, p_\ell | n_1, \ldots, n_\ell)$ for the sum $\sum_{i=1}^{\ell} c(B_{i-1}, p_i)$, which is the sum of the costs of each of the ℓ rearrangements that are done during the bucketing. If R is the prefix rule, we call B a *prefix bucketing* and denote the cost simply by $c(p_1, \ldots, p_\ell | n_1, \ldots, n_\ell)$. In the case of simple bucketing, $n_1 = \ldots = n_\ell = 1$, we write simply $c[R](p_1, \ldots, p_\ell)$ or $c(p_1, \ldots, p_\ell)$ in the case of simple prefix bucketing.

3.1 Tail bucketing and the online labeling

We will also need an alternative rearrangement function, called the *tail rearrangement rule*. The bucketing game with this rule is called *tail bucketing*. The tail rearrangement rule $Tail_\beta$ with parameter β acts on configuration C, bucket p and group size m by first moving all items below bucket p to bucket p so that $w = C(1) + \cdots + C(p) + m$ items are in bucket p (as with the prefix rule), but then for j from p down to 1, β fraction of the items in bucket j are passed to bucket $j-1$, until we reach bucket 1. (Here we allow the number of items in a bucket to be non-integral.) So the number of items in bucket j for $j \in [2, p]$ is $(1-\beta)\beta^{p-j}w$ and the number of items in bucket 1 is $\beta^{p-1}w$.

A bucketing B produced with the tail bucketing rearrangement rule is called a *tail bucketing*.

We will consider tail bucketing with the cheap cost function. We will now relate the expected cost of randomized online labeling algorithm $\mathbf{A}$ on the sequence $y_1, y_2, \ldots, y_n$ which was produced by our adversary $\mathbf{Adversary(A}, n, m\mathbf{)}$ to the cost of a specific tail bucketing instance.

For a lazy online labeling algorithm $\mathbf{A}$ and $t = 1, \ldots, n$, let $\mathbf{f}_{\mathbf{A},t}, S_t(i), q_t, y_t$ be as defined by the $\mathbf{Adversary(A}, n, m\mathbf{)}$ and the algorithm $\mathbf{A}$. Denote $Y = \{y_1, y_2, \ldots, y_n\}$. Set $k = \lfloor 4 \log(m+1) \rfloor$. Let $q_1, \ldots, q_n$ be the sequence of critical levels produced by the algorithm. For integer $i \in [k]$ define $\bar{i}$ to be $\bar{i} = (k+1) - i$. Define the placement sequence $p_1 = \bar{q}_1, \ldots, p_n = \bar{q}_n$, and consider the tail bucketing $B_0, \ldots, B_n$ determined by this placement sequence with parameter $\beta = 1/6$, and all group sizes 1 (so it is a simple bucketing). The following lemma is used to relate the cost of online labeling to the tail bucketing.

Lemma 4. *Let $\{S_t(i) : 1 \le t \le n, 1 \le i \le d_t\}$ be the interval hierarchy computed by* $\mathbf{Adversary(A}, n, m\mathbf{)}$ *and $B_{\mathbf{A}} = (B_0, \ldots, B_n)$ be the corresponding tail-bucketing. Then for any $t \in [0, n]$ and any $j \in [1, d_t]$:*

$$|S_t(j) \setminus S_t(j+1)| \ge B_t(\bar{j}) - 3.$$

Here, for the case $j = d_t$, we take $S_t(j+1)$ to be $\emptyset$.

Proof. We will actually prove:

$$\left\lceil \sum_{i \le \bar{j}} B_t(i) \right\rceil + 2 \ge |S_t(j)| \ge \left\lceil \sum_{i \le \bar{j}} B_t(i) \right\rceil . \qquad (1)$$

Given this we get:

$$|S_t(j) \setminus S_t(j+1)| \geq \left\lceil \sum_{i \leq \bar{j}} B_t(i) \right\rceil - \left(\left\lceil \sum_{i \leq \bar{j}-1} B_t(i) \right\rceil + 2 \right) \geq B_t(\bar{j}) - 3,$$

as required.

We prove (1) by induction on t. For $t = 0$ we have $d_0 = 1$, so we only need to check the case $j = 1$. We have $|S_0(1)| = 2$, and $\bar{j} = k$ and $\sum_{i \leq k} B_0(i) = 0$.

Let $t \geq 1$ and assume the claim is true for $t-1$. Let $j \in [1,k]$. Suppose first $j \leq q_t$. By the definition of the critical level, $q_t \leq d_{t-1}$. Therefore $j \leq d_{t-1}$ and we may apply the induction hypothesis with $t-1$ and j. Since $j \leq q_t$ $|S_t(j)| = |S_{t-1}(j)|+1$. The conclusion then follows by induction if we can show that $\sum_{i \leq \bar{j}} B_t(i) - \sum_{i \leq \bar{j}} B_{t-1}(i) = 1$. This holds because $p_t = \bar{q}_t$ and so $\bar{j} \geq p_t$ and therefore B_t is obtained from B_{t-1} by adding a single item at position p_t and redistributing items among the first p_t buckets, so that the difference in the two sums is indeed 1.

Now assume $j > q_t$. We hold t fixed and prove the equality by induction on j, where we use the already proved case $j = q_t$ as the basis. Suppose that $d_t \geq j > q_t$ and that the desired equality holds for $(t, j-1)$.

Define $w(j) = \sum_{i \leq \bar{j}} B_t(i)$. For $d_t \geq j > q_t$ we have $\bar{j} < p_t$ and the tail-bucketing rule implies $w(j) = w(j-1)/6$. Also, the rebuilding rule for $S_t(j)$ implies $|S_t(j)|$ is between $\lceil S_t(j-1)/6 \rceil$ and $\lceil S_t(j-1)/6 \rceil + 1$ (which is verified by case analysis depending on $|S_t(j-1)| \mod 6$).

Thus:

$$\begin{aligned} |S_t(j)| &\leq \left\lceil \frac{1}{6}|S_t(j-1)| \right\rceil + 1 \\ &\leq \left\lceil \frac{1}{6}(\lceil w(j-1) \rceil + 2) \right\rceil + 1 \\ &\leq \left\lceil \frac{1}{6} w(j-1) \right\rceil + 2 \\ &= \lceil w(j) \rceil + 2, \end{aligned}$$

where the second inequality uses the induction hypothesis and the third is a simple arithmetic fact. This proves the first inequality of (1). Similarly for the second inequality:

$$\begin{aligned} |S_t(j)| &\geq \left\lceil \frac{1}{6}|S_t(j-1)| \right\rceil \\ &\geq \left\lceil \frac{1}{6}(\lceil w(j-1) \rceil) \right\rceil \\ &\geq \left\lceil \frac{1}{6} w(j-1) \right\rceil \\ &= \lceil w(j) \rceil . \end{aligned}$$

□

Corollary 1. *The cost of randomized labeling algorithm* $\mathbf{A}$ *with label space* $[1,m]$ *on* $y_1, \ldots, y_n$ *satisfies:*

$$\chi_{\mathbf{A}}(y_1, y_2, \ldots, y_n) \geq \frac{1}{40}(\min \mathbf{cost_{cheap}}[Tail_{1/6}](p_1, \ldots, p_n) - 10n),$$

where the minimum is over all placement sequences $(p_1, \ldots, p_n)$ *into* $\lfloor 4\log(m+1)\rfloor$ *buckets.*

Proof. Consider the placement sequence p derived from the sequence of critical levels as in Lemma 4. The total cost is $\sum_t B_{t-1}(p_t) = \sum_t B_{t-1}(\bar{q}_t)$, which by Lemma 4 is bounded above by $\sum_t |S_{t-1}(q_t) \setminus S_{t-1}(1+q_t)| + 3n$. Split this latter sum according to $q_t < d_{t-1}$ or $q_t = d_{t-1}$. The terms for which $q_t = d_{t-1}$ are each at most 7 (since $|S_{t-1}(d_{t-1})| \leq 7$) and so:

$$\sum_t B_{t-1}(\bar{q}_t) - 10n \leq \sum_{t:q_t<d_{t-1}} |S_{t-1}(q_t) \setminus S_{t-1}(1+q_t)|.$$

Now apply Lemma 3. □

4 Lower bounds on tail bucketing

Armed with Corollary 1, it now suffices to prove a lower bound on the cheap cost of simple tail bucketing when the number of items is n and the number of buckets is $\lfloor 4\log(m+1)\rfloor$.[5]

The first step is to bound the cost of (simple) tail bucketing by the cost of (simple) prefix bucketing under the cost function $\mathbf{cost}_{\gamma-\mathbf{disc}}$.

Lemma 5. *Let* $k \geq 1$ *be an integer and* $p_1, \ldots, p_\ell$ *be the placement sequence into* k *buckets. Then:*

$$\mathbf{cost}_{\mathbf{cheap}}[Tail_\beta](p_1, \ldots, p_\ell) \geq (1-\beta) \cdot \mathbf{cost}_{\beta-\mathbf{disc}}(p_1, \ldots, p_\ell)..$$

Proof. Refer to the item loaded in step j as item j. We can partition the cost of step s as the sum of the contributions due to each of the items $1, \ldots, s-1$. We now show that for each item j and each step $s > j$, the contribution of item j to the cost at step s using $\mathbf{cost}_{\mathbf{cheap}}$ with the tail rearrangement rule is at least $1-\beta$ times the contribution of item j to the cost at step s using $\mathbf{cost}_{\beta-\mathbf{disc}}$.

Let h be an index in $\{j, j+1, \ldots, s-1\}$ such that p_h is maximum. After step $s-1$, under the prefix rearrangement rule, j is located in bucket p_h. If $p_s \leq p_h$ then the contribution to $\mathbf{cost}_{\beta-\mathbf{disc}}$ by item j is $\beta^{p_h - p_s}$, otherwise the contribution is 0.

Under tail rearrangement j is split among buckets $1, \ldots, p_h$ with $(1-\beta)\beta^{p_h - i}$ of j in bucket i for $2 \leq i \leq p_h$ and $\beta^{p_h - 1}$ located in bucket 1. If $p_s > p_h$ then under $\mathbf{cost}_{\mathbf{cheap}}$ the contribution of item j to step s is 0. If $1 < p_s \leq p_h$ then under $\mathbf{cost}_{\mathbf{cheap}}$ the contribution is $(1-\beta)\beta^{p_h - p_s}$ and for $p_s = 1$ the contribution is $\beta^{p_h - p_s}$. This is at least $1-\beta$ times the contribution to $\mathbf{cost}_{\beta-\mathbf{disc}}$ under prefix bucketing. □

The next step is an easy reduction from $\mathbf{cost}_{\gamma-\mathbf{disc}}$ to $\mathbf{cost}_{b-\mathbf{block}}$.

Lemma 6. *Let* $\gamma \in (0,1]$ *and* $1 \leq b$. *Let* $p_1, \ldots, p_\ell$ *be a placement sequence. Then:*

$$\mathbf{cost}_{\gamma-\mathbf{disc}}(p_1, \ldots, p_\ell) \geq \gamma^b \mathbf{cost}_{b-\mathbf{block}}(p_1, \ldots, p_\ell).$$

[5] *Fun fact:* Before deriving the lower bound we expended several CPU-days (on AMD Phenom II X4 955 3.2GHz with 16GB of RAM) to calculate the optimal cost of tail-bucketing for upto 30 buckets and 500 items. This provided us with confidence that the cost grows in non-linear fashion.

Proof. Since in both games we are using the prefix rearrangement rule, the configuration after each step in the two games is the same. Consider the contribution of the tth step of the bucketing to each side. Items are loaded into bucket p_t. Let s be the least multiple of b with $s \geq p_t$ and let $r = s - p_t$. In b-block bucketing we pay only for items that at step $t-1$ were in buckets of the form $p_t + i$ where $0 \leq i \leq r$. Since $r \leq b$, in γ-discounted bucketing we pay at least γ^b for each of these items. □

Applying this lemma with $b = 1$ gives $\mathbf{cost}_{\gamma-\mathbf{disc}}(p_1, \ldots, p_n) \geq \mathbf{cost}_{\mathbf{cheap}}(p_1, \ldots, p_n)$. This lower bound does not help us directly because it can be shown that for $k = \log(n+1)$ buckets there is an (n, k)-placement sequence with $\mathbf{cost}_{\mathbf{cheap}}(p_1, \ldots, p_n) = 0$. This follows from the following lemma, which we state in greater generality so that we can use it later:

Lemma 7. *For any ℓ, k and for any load sequence $n_1, \ldots, n_\ell$ there is an (ℓ, k)-placement sequence $r_1, \ldots, r_\ell$ into k buckets satisfying:*

$$\mathbf{cost}_{\mathbf{cheap}}(r_1, \ldots, r_\ell | n_1, \ldots, n_\ell) = \sum_{j=1}^{\ell-2^k+1} n_j(m+1-j),$$

where $m = \max(\ell - 2^k + 1, 0)$.

In particular, if $k \geq \log(\ell+1)$ then $\mathbf{cost}_{\mathbf{cheap}}(r_1, \ldots, r_\ell | n_1, \ldots, n_\ell) = 0$.

Proof. The sequence consists of loading all items into bucket 1 for the first m steps. For all steps $m + j$ for $j \leq 2^k - 1$ load new items in step j in bucket $\alpha(j) + 1$ where $\alpha(j)$ is the largest power of 2 dividing j.

It is easy to prove by induction on j that after step $m+j$ the set of occupied buckets are exactly those whose positions correspond to the 1's in the binary expansion of j. Furthermore, for all $j \geq 2$, $\alpha(j) + 1$ is empty at the end of step $j - 1$. It follows that during the last $2^k - 2$ steps there is no cost incurred.

It remains to bound the total cost during the first $m + 1$ steps. Each item loaded at step $j \leq m$ is charged $m + 1 - j$ steps (at each step in $j + 1, \ldots, m + 1$). Thus the total charge is $\sum_{j=1}^{m+1} n_j(m + 1 - j)$. □

As mentioned, this gives an upper bound of 0 if the number of buckets is at least $\log(\ell + 1)$. We now show that a small reduction in the number of buckets is enough to give a good lower bound on $\mathbf{cost}_{\mathbf{cheap}}$.

Lemma 8. *For any (ℓ, k)-placement sequence $p_1, \ldots, p_\ell$,*

$$\mathbf{cost}_{\mathbf{cheap}}(p_1, \ldots, p_\ell) \geq (\ell + 1)(\log(\ell + 1) - 2k).$$

Proof. We lower bound $\mathbf{cost}_{\mathbf{cheap}}(p_1, \ldots, p_\ell)$ by induction on ℓ, where the base case $\ell = 0$ is trivial. Let $m_1 < m_2 < .... < m_r$ be the indices such that $p_{m_i} = k$. Also define $m_0 = 0$ and $m_{r+1} = \ell + 1$. For $i \in [1, r + 1]$, the interval $[m_{i-1} + 1, m_i - 1]$ is called *phase i*. Each phase consists only of placements to buckets $k - 1$ or lower and (except possibly the last phase) is followed immediately by a placement to bucket k. We define $\ell_i = m_i - m_{i-1} - 1$ to be the length of the phase. Let $\gamma_i = (\ell_i + 1)/(\ell + 1)$ so that $\sum_{i=1}^{r+1} \gamma_i = 1$.

Let us now analyze the cost of the sequence phase by phase. At the beginning of phase i there are no items in any bucket below k. The phase itself is an $(\ell_i, k - 1)$ bucketing so by induction has cost at least $(\ell_i + 1)(\log(\ell_i + 1) - 2(k-1)) = (\ell+1)\gamma_i(2 + \log(\gamma_i) + \log(\ell+1) - 2k)$. Except for $i = r+1$, the placement p_{m_i} immediately following the phase costs $m_{i-1} = (\ell+1)(\sum_{j=1}^{i-1}\gamma_j)$ since that is the number of items in bucket k prior to that placement. Summing over phases and rearranging gives:

$$\begin{aligned}\mathbf{cost_{cheap}}(p_1,\ldots,p_\ell) \geq (\ell+1)\left(\sum_{j=1}^{r}(r-j)\gamma_j + 2 + \sum_{i=1}^{r+1}\gamma_i\log(\gamma_i)\right)\\ +(\ell+1)(\log(\ell+1)-2k)\end{aligned}$$

Note that the final term is the lower bound we are aiming for so it suffices to show:

$$\sum_{j=1}^{r}(r-j)\gamma_j + 2 \geq \sum_{i=1}^{r+1}\gamma_i\log(1/\gamma_i).$$

Since $\sum_{i=1}^{r+1}\gamma_i = 1$ the lefthand side is at least $\sum_{j=1}^{r+1}(r-j+2)\gamma_j$. Observing that $\sum_{i=1}^{r+1}2^{-(r-j+2)} \leq 1$, the desired inequality follows from:

Proposition 1. *Let $\alpha_1,\ldots,\alpha_s$ be nonnegative reals summing to 1. Then for all choices of $x_1,\ldots,x_s$ of nonnegative reals with sum at most 1, the function $\sum_i \alpha_i\log(1/x_i)$ is minimized when $(x_1,\ldots,x_s) = (\alpha_1,\ldots,\alpha_s)$.*

This is essentially equivalent to the well known fact that the KL-divergence of two distributions is always nonnegative and is easily proved by first noting that we may assume $\sum_i x_i = 1$, and then using Lagrange multipliers, or induction on s. □

4.1 From $\mathbf{cost}_{b-\mathbf{block}}$ to $\mathbf{cost_{cheap}}$

So far we have shown that the cost of online labeling can be bounded below by the cheap cost of tail-bucketing, which can be bounded below by the $\mathbf{cost}_{b-\mathbf{block}}$ for simple bucketing.

Below we will prove Lemma 11 which shows that $\mathbf{cost}_{b-\mathbf{block}}$ can be bounded below by $\mathbf{cost_{cheap}}$ with fewer buckets. In preparation, we begin by bounding $\mathbf{cost_{exp}}$ from below by $\mathbf{cost_{cheap}}$ with fewer buckets.

Lemma 9. *Let $k \geq 1$ and $b = 2^k - 1$. Let $n_1,\ldots,n_\ell$ be an arbitrary load sequence. Then for any placement sequence $p_1,\ldots,p_\ell$ into b buckets there is a placement sequence $r_1,\ldots,r_\ell$ into k buckets such that*

$$\mathbf{cost_{cheap}}(r_1,\ldots,r_\ell|n_1,\ldots,n_\ell) \leq \mathbf{cost_{exp}}(p_1,\ldots,p_\ell|n_1,\ldots,n_\ell).$$

Proof. If $\ell \leq b$ then $k \geq \log(\ell+1)$ so by Lemma 7 there is a placement sequence $r_1,\ldots,r_\ell$ with zero cheap cost and the lemma follows. Hence, assume $\ell > b$. We begin with a lower bound on $\mathbf{cost_{exp}}(p_1,\ldots,p_\ell|n_1,\ldots,n_\ell)$. At step j, any item inserted before j that is in bucket p_j or higher incurs a charge of 1. Any previously loaded item that is in a bucket less than p_j incurs no charge, but is moved to bucket p_j. Thus, once an item is loaded, in every step it incurs a charge of 1 or increases its bucket number. An item loaded at step j incurs no cost at step j and incurs a cost of 1 in every step that it does not move, which means that it incurs a cost of one in at least $(\ell - j) - (b-1)$ steps. Summing over the first $\ell - b$ items we get.

$$\mathbf{cost_{exp}}(p_1, \dots, p_\ell | n_1, \dots, n_\ell) \geq \sum_{j=1}^{\ell-b} n_j(\ell - j - b + 1).$$

Now, setting $b = 2^k - 1$, Lemma 7 completes the proof of the lemma. □

For a step i let $c^i(p_1, \dots, p_\ell | n_1, \dots, n_\ell)$ be the cost of the placement into p_i at step i. For $I \subseteq [1, \ell]$, let

$$c^I(p_1, \dots, p_\ell | n_1, \dots, n_\ell) = \sum_{i \in I} c^i(p_1, \dots, p_\ell | n_1, \dots, n_\ell). \tag{2}$$

Lemma 10. *Let $p_1, \dots, p_\ell$ be a placement sequence with b buckets. Let $\theta \in [1, b]$ and let $I = \{i_1 < \dots < i_h\}$ be the indices in $[1, \ell]$ such that $p_{i_j} > \theta$. Let $s_1, \dots, s_h$ be the placement sequence into $b - \theta$ buckets given by $s_j = p_{i_j} - \theta$ and let $n_1, \dots, n_h$ be given by $n_1 = i_1$ and for $j > 1$, $n_j = i_j - i_{j-1}$. Then for cost function $c \in \{\mathbf{cost_{cheap}}, \mathbf{cost_{exp}}\}$,*

$$c^I(p_1, \dots, p_\ell) = c(s_1, \dots, s_h | n_1, \dots, n_h).$$

Proof. It suffices to show that for each $j \in [1, h]$, $c^{i_j}(p_1, \dots, p_\ell) = c^j(s_1, \dots, s_h | n_1, \dots, n_h)$. Let $B_1, \dots, B_\ell$ be the bucketing sequence associated to $(p_1, \dots, p_\ell)$, and let $\tilde{B}_1, \dots, \tilde{B}_h$ be the bucketing sequence associated to $(s_1, \dots, s_h | n_1, \dots, n_h)$.

We claim that for each $j \in [1, h]$ the configuration B_{i_j} restricted to $[\theta + 1, b]$ is identical to the configuration $\tilde{B}_j$ restricted to $[1, b - \theta]$. This is easily shown by induction on j. The base case $j = 0$ is trivial. Assume $j > 0$. The result holds for $j - 1$ so $B_{i_{j-1}}$ restricted to $[\theta + 1, b]$ is identical to B_{j-1} restricted to $[1, b - \theta]$.

For the sequence $s_1, \dots, s_h$, at step j, all buckets above s_j are unchanged, all buckets below s_j are emptied, and s_j increases by the number of items that were in buckets below s_j, together with the load of n_j.

Now consider the change in the configuration B from $B_{i_{j-1}}$ to B_{i_j}. For each $s \in i_{j-1} + 1$ to $i_j - 1$, $p_s \leq \theta$, which implies that B restricted to $[\theta + 1, b]$ is unchanged. Next consider the placement p_{j_i} at step j_i. All buckets above $p_{j_i} = s_j + \theta$ are unchanged and all buckets below p_j are emptied, and bucket p_j gets all of the items that were in buckets $[\theta + 1, p_{i_j} - 1]$ after step i_{j-1} together with all of the n_j new items that arrived since i_{j-1} of the buckets in B. This exactly matches the change in bucket s_j at step j in the other bucketing, as required to establish the claim.

By the claim, the cost of step i_j for $p_1, \dots, p_\ell$ is the same as the cost of step j for $s_1, \dots, s_h | n_1, \dots, n_h$ as required to prove the lemma. □

Next we come to a crucial reduction which lower bounds $\mathbf{cost}_{b-\mathbf{block}}$ in terms of $\mathbf{cost_{cheap}}$ with a fewer number of buckets.

Lemma 11. *Let $k \geq 1$, $m \geq 1$ and $b = 2^k - 1$. Let $p_1, \dots, p_\ell$ be a placement sequence into bm buckets. There exists a placement sequence $s_1, \dots, s_\ell$ for km buckets such that*

$$\mathbf{cost_{cheap}}(s_1, \dots, s_\ell) \leq \mathbf{cost}_{b-\mathbf{block}}(p_1, \dots, p_\ell).$$

Proof. Fix $p_1, \dots, p_\ell$. We first describe the construction of the sequence $s_1, \dots, s_\ell$ and then prove the properties.

To specify the sequence $s_1, \ldots, s_\ell$ we will define a partition of $[1, \ell]$ into (generally non-consecutive) subsequences, and for each set $\widehat{\mathbf{h}}$ in the partition separately specify s_i for $i \in \widehat{\mathbf{h}}$.

The definition of the partition takes a few steps. Define the *level of a bucket w for block size b* to be the largest λ such that $\lambda b < w$, and the *remainder of w* to be $w - \lambda b$. For $i \in [1, n]$, define λ_i to be the level of p_i and r_i to be the remainder of p_i. By the hypotheses of the lemma each $\lambda_i \in [0, m-1]$ and each remainder is in $[1, \ldots, b]$. We also define $\lambda_0 = \lambda_{\ell+1} = \infty$.

A chain of level j and order v is a sequence $\mathbf{h}$ of indices $\mathbf{h}_0 < \mathbf{h}_1 < \cdots < \mathbf{h}_v < \mathbf{h}_{v+1}$ (with possibly $\mathbf{h}_0 = 0$ or $\mathbf{h}_{v+1} = \ell + 1$) satisfying the following properties:

– $\lambda_{\mathbf{h}_0} > j$ and $\lambda_{\mathbf{h}_{v+1}} > j$,
– $\lambda_{\mathbf{h}_1} = \cdots = \lambda_{\mathbf{h}_v} = j$,
– For any index i belonging to $[\mathbf{h}_0, \mathbf{h}_{v+1}] \setminus \{\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{v+1}\}$, $\lambda_i < j$.

The indices $\mathbf{h}_0, \mathbf{h}_{v+1}$ are the *endpoints* of $\mathbf{h}$, and the other indices are the *interior indices*. We write $\widehat{\mathbf{h}} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_v\}$ for the interior of $\mathbf{h}$. Thus the order of $\mathbf{h}$ equals to $|\widehat{\mathbf{h}}|$. A chain of order 0 is *trivial*, others are non-trivial. Every chain of level j can be obtained in the following way: consider the sequence $0 = g_0 < g_1 < \cdots < g_{w-1} < g_w = \ell + 1$ consisting of all indices at level higher than j. Then between each consecutive pair g_i and g_{i+1} from the sequence there is a unique chain of level j. The interiors of these chains partition the set of indices at level j. The collection of all nontrivial chains is denoted $\mathcal{H}$, and the set of interiors of these chains partitions $[1, \ell]$.

We write $\lambda(\mathbf{h})$ for the level of $\mathbf{h}$ and $v(\mathbf{h})$ for the order of $\mathbf{h}$.

For a chain $\mathbf{h}$ as above, we define its *difference sequence* to be the sequence $\Delta_1^{\mathbf{h}}, \ldots, \Delta_{v(\mathbf{h})}^{\mathbf{h}}$ given by $\Delta_i^{\mathbf{h}} = \mathbf{h}_i - \mathbf{h}_{i-1}$. (We could also define $\Delta_{v(\mathbf{h})+1}^{\mathbf{h}}$ but we won't need it.) The sum of the difference sequence is just $\mathbf{h}_{v(\mathbf{h})} - \mathbf{h}_0$. Finally we define the remainder sequence of $\mathbf{h}$ to be the subsequence $r_{\mathbf{h}_1}, \ldots, r_{\mathbf{h}_{v(\mathbf{h})}}$ of the remainder sequence $r_1, \ldots, r_{v(\mathbf{h})}$ corresponding to the interior indices of $\mathbf{h}$.

At last we are ready to define s_i. Fix $\mathbf{h} \in \mathcal{H}$; we define s_i for $i \in \widehat{\mathbf{h}}$. Now view the remainder sequence $r_{\mathbf{h}_1}, \ldots, r_{\mathbf{h}_v}$ of $\mathbf{h}$ as a placement for the load sequence $\Delta_1^{\mathbf{h}}, \ldots, \Delta_{v(\mathbf{h})}^{\mathbf{h}}$. All of these placements are in the range $[1, 2^k - 1]$ so by Lemma 9, there is a placement $u_1, \ldots, u_{v(\mathbf{h})}$ into buckets in the range $[1, k]$ such that:

$$\mathbf{cost}_{\mathbf{cheap}}(u_1, \ldots, u_{v(\mathbf{h})} | \Delta_1^{\mathbf{h}}, \ldots, \Delta_{v(\mathbf{h})}^{\mathbf{h}}) \leq \mathbf{cost}_{\mathbf{exp}}(r_1, \ldots, r_{v(\mathbf{h})} | \Delta_1^{\mathbf{h}}, \ldots, \Delta_{v(\mathbf{h})}^{\mathbf{h}}).$$

Now for each $i \in [1, v(\mathbf{h})]$ let $s_{\mathbf{h}_i} = \lambda(\mathbf{h})k + u_i$. This defines the values s_i for $i \in \widehat{\mathbf{h}}$, and by doing this for all $\mathbf{h} \in \mathcal{H}$ we get the sequence $s_1, \ldots, s_n$.

Since $\lambda(\mathbf{h}) \in [0, m-1]$ and $u_i \in [1, k]$ we have that all s values are in $[1, km]$. When we refer to the level of an s_i we mean its level with respect to block size k. Observe that the sequence λ_i of levels (with respect to block size b) corresponding to the placement sequence p is the same as the sequence of levels (with respect to block size k) corresponding to placements in s.

To prove the inequality of the lemma, we need a bit more notation. Write $p^{\mathbf{h}}$ for the consecutive subsequence of p of length $\mathbf{h}_v - \mathbf{h}_0$ starting with $p_{\mathbf{h}_0+1}$. Thus $p_i^{\mathbf{h}} = p_{\mathbf{h}_0+i}$. Define $s^{\mathbf{h}}$ analogously. Also let $\widehat{\mathbf{h}}_{\text{Rel}} = \{\mathbf{h}_1 - \mathbf{h}_0, \mathbf{h}_2 - \mathbf{h}_0, \ldots, \mathbf{h}_{v(\mathbf{h})} - \mathbf{h}_0\}$. Thus $\widehat{\mathbf{h}}_{\text{Rel}}$ is the set of indices of $p^{\mathbf{h}}$ corresponding to $\widehat{\mathbf{h}}$.

The inequality of the lemma is obtained from the following chain (where we use the notation from (2)) of relations:

$$
\begin{aligned}
\mathbf{cost}_{b-\mathbf{block}}(p_1,\ldots,p_\ell) &\overset{(A1)}{=} \sum_{\mathbf{h}\in\mathcal{H}} \mathbf{cost}^{\widehat{\mathbf{h}}}_{b-\mathbf{block}}(p_1,\ldots,p_\ell)\\
&\overset{(A2)}{=} \sum_{\mathbf{h}\in\mathcal{H}} \mathbf{cost}^{\widehat{\mathbf{h}}_{\mathrm{Rel}}}_{\mathbf{exp}}(p^{\mathbf{h}}_1,\ldots,p^{\mathbf{h}}_{\mathbf{h}_{v(\mathbf{h})}-\mathbf{h}_0})\\
&\overset{(A3)}{=} \sum_{\mathbf{h}\in\mathcal{H}} \mathbf{cost}_{\mathbf{exp}}(r_{\mathbf{h}_1},\ldots,r_{\mathbf{h}_{v(\mathbf{h})}}|\Delta^{\mathbf{h}}_1,\ldots,\Delta^{\mathbf{h}}_{v(\mathbf{h})})\\
&\overset{(A4)}{\geq} \sum_{\mathbf{h}\in\mathcal{H}} \mathbf{cost}_{\mathbf{cheap}}(u_{\mathbf{h}_1},\ldots,u_{\mathbf{h}_{v(\mathbf{h})}}|\Delta^{\mathbf{h}}_1,\ldots,\Delta^{\mathbf{h}}_{v(\mathbf{h})})\\
&\overset{(A5)}{=} \sum_{\mathbf{h}\in\mathcal{H}} \mathbf{cost}^{\widehat{\mathbf{h}}_{\mathrm{Rel}}}_{\mathbf{cheap}}(s^{\mathbf{h}}_1,\ldots,s^{\mathbf{h}}_{\mathbf{h}_{v(\mathbf{h})}-\mathbf{h}_0})\\
&\overset{(A6)}{=} \sum_{\mathbf{h}\in\mathcal{H}} \mathbf{cost}^{\widehat{\mathbf{h}}}_{\mathbf{cheap}}(s_1,\ldots,s_\ell)\\
&\overset{(A7)}{=} \mathbf{cost}_{\mathbf{cheap}}(s_1,\ldots,s_\ell).
\end{aligned}
$$

We now justify each of these steps. We work from both ends to the middle. Equalities (A1) and (A7) follow from the fact that $\mathcal{H}$ is a partition of $[1,\ell]$. For all of the other relations, we fix an $\mathbf{h}\in\mathcal{H}$ and show it holds term by term. For (A2) observe first that after step $\mathbf{h}_0$, items stored during steps $[1,\mathbf{h}_0]$ are in buckets higher than level j and so contribute nothing to the $\mathbf{cost}_{b-\mathbf{block}}$ during steps $[\mathbf{h}_0+1,\mathbf{h}_{v(\mathbf{h})}]$ so in accounting for the cost of steps of $\widehat{\mathbf{h}}$ we can omit all placements prior to $\mathbf{h}_0$. During $[\mathbf{h}_0+1,\mathbf{h}_{v(\mathbf{h})}]$ there are no placements above block j so $\mathbf{cost}_{\mathbf{exp}}$ coincides with $\mathbf{cost}_{b-\mathbf{block}}$. This proves (A2) and a similar argument gives (A6). For (A3), we apply Lemma 10 with $\theta = jb$, and for (A5) we apply the same lemma with $\theta = jk$. Finally Lemma 9 implies (A4). □

Lemma 12. *Let $c \geq 1$ be an arbitrary constant. For any large enough integers n, m satisfying $m < (n+1)^c$ and any placement sequence $p_1,\ldots,p_n$ into $\lfloor 4\log(m+1)\rfloor$ buckets the following is true:*

$$\mathbf{cost}_{\mathbf{cheap}}[Tail_{1/6}](p_1,\ldots,p_n) \geq \frac{5}{12}\left(\frac{1}{6}\right)^{512c^2}(n+1)\log(n+1).$$

Now Lemma 1 follows from this lemma and Corollary 1.
Proof of Lemma 12. Choose $b\in[256c^2, 512c^2]$ such that $b = 2^k-1$ for some integer k. By Lemmas 5 and 6 we have:

$$
\begin{aligned}
\mathbf{cost}_{\mathbf{cheap}}[Tail_{1/6}](p_1,\ldots,p_n) &\geq (5/6)\cdot\mathbf{cost}_{1/6-\mathbf{disc}}(p_1,\ldots,p_n)\\
&\geq (5/6)(1/6)^b\cdot\mathbf{cost}_{b-\mathbf{block}}(p_1,\ldots,p_n).
\end{aligned}
$$

Set $k_1 = \lfloor 4\log(m+1)\rfloor$ and $k_2 = k\cdot\lceil k_1/b\rceil$. By Lemma 11, for any (n,k_1)-placement sequence $p_1,\ldots,p_n$ there is a (n,k_2)-placement sequence $s_1,\ldots s_n$ such that:

$$(5/6)(1/6)^b\cdot\mathbf{cost}_{b-\mathbf{block}}(p_1,\ldots,p_n) \geq (5/6)(1/6)^b\cdot\mathbf{cost}_{\mathbf{cheap}}(s_1,\ldots,s_n).$$

We want to apply Lemma 8 to this final expression. Notice,

$$\begin{aligned} k_2 &= \log(b+1) \cdot \left\lceil \frac{\lfloor 4\log(m+1) \rfloor}{b} \right\rceil \\ &\leq \log(b+1) + \frac{\log(b+1)}{b} \cdot 4c \cdot \log(n+1) \\ &\leq \frac{\log(n+1)}{4} \end{aligned}$$

provided that n is large enough and $\log(b+1)/b < 1/16c$. Indeed, since $\log(x+1)/x$ is a decreasing function, $\log(b+1)/b \leq \log(256c^2+1)/256c^2 \leq (11/16)(1/16c)$ as can be easily verified. Lemma 8 applied on $s_1, \ldots s_n$ implies the lower bound. □

References

1. Afek, Y., Awerbuch, B., Plotkin, S., Saks, M.: Local management of a global resource in a communication network. *J. ACM*, 43(1), 1–19 (1996)
2. Babka, M., Bulánek, J., Čunát, V., Koucký, M., Saks, M.: On Online Labeling with Polynomially Many Labels. In *ESA*, 121–132 (2012)
3. Bender, M., Cole, R., Demaine, E., Farach-Colton, M., Zito, J.: Two simplified algorithms for maintaining order in a list. In *ESA*, 152–164 (2002)
4. Bender, M., Demaine, E., Farach-Colton, M.: Cache-oblivious B-trees. *SIAM J. Comput.*, 35(2), 341–358 (2005)
5. Bender, M., Duan, Z., Iacono, J., Wu, J.: A locality-preserving cache-oblivious dynamic dictionary. *J. Algorithms*, 53(2), 115–136 (2004)
6. Bird, R., Sadnicki, S.: Minimal on-line labelling. *Inf. Process. Lett.*, 101(1), 41–45 (2007)
7. Borodin, A., El-Yaniv, R.: *Online computation and competitive analysis.* Cambridge University Press, (1998)
8. Brodal, G., Fagerberg, R., Jacob, R.: Cache oblivious search trees via binary trees of small height. In *SODA*, 39–48, (2002)
9. Bulánek, J., Koucký, M., Saks, M.: Tight lower bounds for online labeling problem. In *STOC*, 1185–1198 (2012)
10. Dietz, P., Seiferas, J., Zhang, J.: A tight lower bound for online monotonic list labeling. *SIAM J. Discrete Math.*, 18(3), 626–637 (2004)
11. Dietz, P., Zhang, J.: Lower bounds for monotonic list labeling. In *SWAT*, 173–180 (1990)
12. Emek, Y., Korman, A.: New bounds for the controller problem. *Distributed Computing*, 24(3-4), 177–186 (2011)
13. Itai, A., Konheim, A., Rodeh, M.: A sparse table implementation of priority queues. In *ICALP*, 417–431 (1981)
14. Korman, A., Kutten, S.: Controller and estimator for dynamic networks. In *PODC*, 175–184 (2007)
15. Willard, D.: A density control algorithm for doing insertions and deletions in a sequentially ordered file in good worst-case time. *Inf. Comput.*, 97(2), 150–204 (1992)
16. Zhang, J.: Density Control and On-Line Labeling Problems. *PhD thesis*, University of Rochester (1993).