On Randomized Online Labeling with Polynomially Many Labels

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

[1] Charles University, Prague
[2] Institute of Mathematics, Academy of Sciences CR, Prague
[3] Department of Mathematics, Rutgers University

Abstract. In this paper 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 *keys* from a totally ordered universe U and must assign each arriving key a label from the label set $\{1, 2, \ldots, m\}$ so that the order of labels (strictly) respects the ordering on U. As new keys 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.

⋆ 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 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].

The parameter m, 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 that 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 problem is trivial if the universe U is a set of size at most m, since then we can simply fix an order preserving bijection from U to $\{1, \ldots, m\}$ in advance. In this paper we assume that $U = \{1, \ldots, 2^n\}$ (as is typical in the literature).

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.

For lower bounds, the present authors recently proved [9] a tight $\Omega(n \log(n)^2)$ lower bound for the case of linearly many labels and tight bound $\Omega(n \log(n)^3)$ for the case $m = n$. Furthermore, together with Babka and Čunát they also proved [2] a tight lower bound $\Omega(n \log(n)/(\log\log(m) - \log\log(n)))$ that is valid for m between $n^{1+\varepsilon}$ and 2^{n^ε}. In particular, this gives the tight bound $\Omega(n \log(n))$ for the case of m being polynomial in n. Both of these papers built on previous partial results of Dieta, Seiferas and Zhang ([16, 11, 10]. All these lower bounds apply only to *deterministic* algorithms, leaving open the possibility of better randomized algorithms.

Indeed, in the area of online algorithms there are known many situations where randomized algorithms perform provably better than deterministic ones. A typical example is the paging problem with k pages for which the best possible deterministic algorithms have competitive ratio k but randomized algorithms have the competitive ratio only $\Theta(\log(k))$ [7]. In contrast, our results show that (at least for the polynomial space regime), randomization does not provide a significant improvement over determinism.

1.1 Our Results

In this paper we prove an $\Omega(n \log(n))$ lower bound on the expected number of relabelings by any *randomized* online labeling algorithm when labeling n items by m labels, for $m = n^C$, $C \geq 1$. This matches the known deterministic upper bounds up to constant factors. Our bound also implies an $\Omega(n \log(n))$ lower bound on the message complexity of randomized protocols for Distributed Controller Problem [12, 1].

2 Our proof

Our overall proof strategy resembles the one from previous papers [10, 2]. The proof consist of two parts in the first part for a given randomized algorithm **A** we design an adversarial sequence of items, and in the second part we prove that this sequence of items is expensive for the algorithm. This later part is done using a new variant of prefix bucketing game [10].

In passing from the deterministic case to the randomized case, both parts of the proof require some significant changes. In the first part we have to design an effective deterministic adversary against a randomized algorithm, and in the second part we have to recourse to a new variant of bucketing game. Not surprisingly, its analysis is more subtle than the analysis of the known variants of the bucketing game.

For the first part of the proof, we assume we are given a randomized algorithm **A**, which we interpret as a probability distribution on deterministic algorithms. From this we construct an adversarial sequence of n items that will force the algorithm to perform $\Omega(n \log(n))$ relabels in expectation. We design the sequence item by item, where we try to chose the next item to make the future expected cost large. Say that an interval of $[a, b]$ of possible label values is *crowded* if there are many items whose label is in that range. In the deterministic case, the adversary constructed in [2] (building on [10]) identifies a small interval $[a, b]$ of label values such that (roughly) every interval containing the selected interval is crowded, and then chooses a new item to be an item for which the algorithm is either forced to give the item a label in the interval $[a, b]$ or to relabel some items. In the randomized setting this viewpoint does not work since we don't know the actual labeling selected by the algorithm, but only a probability distribution over labelings. So we use a dual point of view, focusing on intervals of inserted items rather than intervals of labels. If Y^t denotes the items inserted up to time t, a Y^t interval is a set which is the intersection of Y^t with an interval. The algorithm determines a probability distribution over labelings of Y^t. The adversary looks for a small Y^t-interval with the property that (roughly) for each superinterval of the selected interval, the algorithm maps the superinterval to a relatively crowded range of labels with significant probability.

The interaction between our adversary and the algorithm will generate a sequence of *set hierarchies*. The changes in these hierarchies will be proportional to the cost incurred by the algorithm. In turn we will be able to relate these changes also to the cost of a certain new variant of a prefix bucketing game, *tail bucketing*. Prefix bucketing was originally defined by Dietz at al. for the purpose of lower-bounding the cost of deterministic online labeling [10]. We will prove $\Omega(n \log(n))$ lower bound on the cost of our tail bucketing game which translates into almost the same lower bound for the expected cost of the randomized algorithm.

2.1 Adversary

We provide here a brief overview of the adversary. The adversary will be completely deterministic although its behavior will be determined by the expected behavior of the randomized algorithm. The adversary will maintain a hierarchy of nested sets of items inserted so far

$$S_1 \supset T_2 \supset S_2 \supset T_3 \supset \cdots \supset T_d \supset S_d$$

and it will always pick the next item to fall in between items in S_d. The set S_1 will consist of all the items inserted so far, and S_d will consist of at most 7 items. To construct the hierarchy one starts from the set S_1. Given S_i we pick T_{i+1} to consist of either the first half of elements of S_i or the second half of the elements according to their order whichever of the two halfs is more likely to be compressed. Namely, we look at the median of S_i and check whether it is more likely that its label is closer to the label of the minimum or to the maximum element of S_i. If the median is more likely to have label close to the minimum we pick the first half as T_{i+1} otherwise the second half. From the set T_{i+1} we remove two thirds of items, the smallest third and the largest third. What is left is S_{i+1}. We repeat this process until S_i is of size at most seven where we stop. Once the adversary determines S_d it choses the next item to lie in between the items of S_d. The adversary checks how the algorithm responds to the new item and updates the hierarchy of sets.

In the deterministic case, the algorithm would respond by relabeling some items and assigning a label to the new item. We can assume that the algorithm is *lazy* in the sense that all the relabeled items form a sub-interval of the inserted items. The adversary would check for which sets T_i their maximum or minimum element got relabeled by the algorithm, it would destroy all such sets and their corresponding sets S_i, and it would rebuild the hierarchy from the smallest remaining set S_q. By the laziness of the algorithm we would destroy all sets from a certain index up. This would be very much akin to the adversary in [2]. A point to note in this case is that the number of items in destroyed sets is within a constant to the number of relabeled items since the items in $T_i \setminus S_i$ form buffers of items that were moved (at least half of them) and their number is proportional to $S_{i-1} \setminus S_i$. Hence the work done by the algorithm and the number of items in the destroyed sets is in linear relation. This allows us to account for the cost of the algorithm.

In the randomized case the problem is that there might be an algorithm having a small but non-zero probability that at each step relabels all the items. Using the same strategy as in the deterministic case would destroy the whole hierarchy. Then we would lose the relationship between the expected cost incurred by the algorithm and the number of items in destroyed sets. The expected cost to the randomized algorithm could be negligible but our adversary would incur the largest possible cost. To mitigate this issue we will destroy a set T_i and all its descendants only when the accumulated expected number of relabels of items in $T_i \setminus S_i$ is proportional to the size of $S_{i-1} \setminus S_i$. This will guarantee at least that the expected cost to the algorithm is proportional to $S_{i-1} \setminus S_i$ although it will *not*

guarantee that the number of items in the destroyed sets is proportional to this expected cost. Still we will be able to relate the expected cost of the algorithm to the behavior of our adversary. The smallest surviving set will be denoted by S_q and rebuilding of the hierarchy will start from it.

One of the crucial parts in designing the adversary is to make sure that the hierarchy never grows too deep. The choice of the set T_i at each level of the hierarchy together with the rule when to destroy it allows us to control the depth so that it never grows to more than $4\log(m+1)$ levels.

We can then relate the hierarchy maintained by the adversary to a particular version of a bucketing game. The cost of the bucketing game and the relationship between the hierarchy and the game is set so that the cost of the game corresponds to the sizes of sets S_q at each step of the game. We describe the bucketing game next.

2.2 Bucketing games

A prefix bucketing game with n items and k buckets starts with k empty buckets indexed $1,\ldots,k$. At each step we insert the next item in a chosen bucket p and we move all the items from buckets $1,\ldots,p-1$ into the bucket p as well. For such a step the original definition of Dietz et al. [10, 2] charges a cost proportional to the number of items in buckets $1,\ldots,p$ before the merge, i.e., proportional to the number of items in bucket p after the merge. 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 the previous papers, there a correspondence was shown that any deterministic labeling algorithm could be associated to a bucketing strategy such that the cost of the labeling algorithm against our adversary was bounded below by a constant times the cost of the bucketing strategy. It is also shown that the minimal cost of any bucketing strategy (for more than $2\log(n)$ buckets) is $\Omega(n\log(n)/(\log(k)-\log\log(n))$ [10, 2]. 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 in relating the cost of an algorithm to the cost of bucketing in the randomized case, 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! So this will not give any lower bound on 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.

Due to space limitations large portion of the proofs is in the appendix. Unless otherwise specified, logarithms in this paper are to base 2.

3 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 $f^0, f^1, \ldots, f^n$ of labelings where f^t is the labeling of $(y^1, \ldots, y^t)$ determined by A. We say that A *relabels* $y \in \{y^1, y^2, \ldots, y^t\}$ *at time* t if $f_A^{t-1}(y) \neq f_A^t(y)$. In particular, y^t is relabeled at time 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 $\chi_A(y^1, \ldots, y^n) = \sum_{t=1}^{n} |Rel_A^t|$.

A *randomized* online labeling algorithm $\mathbf{A}$ is a probability distribution on deterministic online labeling algorithms. Given a 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 $\chi_{\mathbf{A}}(y^1, \ldots, y^n) = \mathbf{E}\left[\chi_A(y^1, \ldots, y^n)\right]$. 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.)

3.1 The adversary

We now specify an adversary **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.

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}, the adversary will select a Y^{t-1}-interval, denoted S^t_*, of size at least 2 and select y^t to be $\min(S^t_*) + 2^{n-t}$. It is easy to see by induction on t that the items chosen through step 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_*. Therefore all of the chosen items are distinct.

The selection of S_*^t is done as follows. The adversary constructs a nested sequence of subsets of Y^{t-1}.

$$Y^{t-1} = S_1^t \supset T_2^t \supset S_2^t \supset T_3^t \supset \cdots \supset T_{d^t}^t \supset S_{d^t}^t$$

called the *hierarchy at step* t, and chooses $S_*^t = S_{d^t}^t$.

Note that the superscript for the hierarchy is one larger than the superscript of the containing set Y^{t-1}. This is because the hierarchy is constructed at the beginning of step t in order to determine y^t. Each of the sets S_i^t and T_i^t is a Y^{t-1}-interval of size at least 2. The depth d^t of the hierarchy may vary with t. The sets S_i^t and T_i^t are said to be at *level* i in the hierarchy.

The pseudo-code for the adversary is given in Figure 3.1.

The hierarchy for $t = 1$ has $d^1 = 1$ and $S_1^1 = \{0, 2^n\}$. The hierarchy at step $t > 1$ is constructed based on the hierarchy at the previous step $t - 1$ and the expected behavior of the algorithm on $y^1, \ldots, y^{t-1}$ as reflected by the joint probability distribution over the sequence of functions $f_{\mathbf{A}}^1, \ldots, f_{\mathbf{A}}^{t-1}$.

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-1}) 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 S_i^t for $i \leq q^t$ are *preserved*, which means that S_i^t is obtained simply by adding y^{t-1} to S_i^{t-1}, and the rest are *rebuilt*. The rule for rebuilding the hierarchy for $i > q^t$ is defined by induction on i as follows: If $|S_{i-1}^t| < 7$ then the hierarchy is terminated with $d^t = i - 1$. Otherwise, consider the labeling of S_{i-1}^t by f^{t-1} (which is randomly distributed depending on $\mathbf{A}$). If the probability that S_{i-1}^t is left-leaning with respect to f^{t-1} is at least 1/2, then set $T_i^t = \textbf{left-half}(S_{i-1}^t)$ otherwise $T_i^t = \textbf{right-half}(S_{i-1}^t)$. Set $S_i^t = \textbf{middle-third}(T_i^t)$. Observe that since $|S_{i-1}^t| \geq 7$, we have $|T_i^t| \geq 4$ and $|S_i^t| \geq 2$.

It remains to explain how the critical level q^t is selected. When constructing each set S_i^t of the hierarchy for $i \geq 2$, the adversary defines a parameter $\textbf{birth}_i^t$ which is set equal to t if S_i^t is rebuilt, and is otherwise set to $\textbf{birth}_i^{t-1}$. It is easy to see (by induction on t), that $\textbf{birth}_i^t$ is equal to the largest time $u \leq t$ such that S_i^u was rebuilt. It follows that for each $u \in [\textbf{birth}_i^t, t]$, $\min(T_i^u) = \min(T_i^t)$ and $\max(T_i^u) = \max(T_i^t)$.

Say that item y has *stable label* during interval $[a, b]$ if for each step u in $[a, b]$, $f^u(y)$ has the same value, and has unstable label on $[a, b]$ otherwise. We define the event $\textbf{Stable}_i^t$ to be the event (depending on $\mathbf{A}$) that $\min(T_i^t)$ and $\max(T_i^t)$ have stable labels during interval $[\textbf{birth}_i^t - 1, t]$.

We are finally ready to define q^t. If there is at least one level $i \geq 2$ for which $\mathbf{Pr}[\textbf{Stable}_i^{t-1}] \leq 3/4$, let $i_{\min}$ be the least such level, and choose $q^t = i_{\min} - 1$. Otherwise set $q^t = d^{t-1}$.

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

Adversary$(\mathbf{A}, n, m)$

$t = 1$: $S_1^1 \longleftarrow \{0, 2^n\}$ and $\mathbf{birth}_1^1 = 1$, $y^1 = 2^{n-1}$.

For $t = 2, \ldots, n$ do

- $S_1^t \longleftarrow S_1^t \cup \{y^{t-1}\}$;
- (Choose critical level) Consider the sequence of (dependent) random functions $f^1, \ldots, f^{t-1}$ produced by $\mathbf{A}$ in response to $y^1, \ldots, y^{t-1}$. If there is an index $i \geq 2$ for which $\mathbf{Pr}[\mathbf{Stable}_i^{t-1}] \leq 3/4$, let $i_{\min}$ be the least such index and let $q^t = i_{\min} - 1$. Otherwise set $q^t = d^{t-1}$.
- $i \longleftarrow 1$.
- (Preserve intervals up to the critical level) While $i < q^t$ do:
  - $i \longleftarrow i + 1$.
  - $T_i^t \longleftarrow T_i^{t-1} \cup \{y^{t-1}\}$
  - $S_i^t \longleftarrow S_i^{t-1} \cup \{y^{t-1}\}$
  - $\mathbf{birth}_i^t \longleftarrow \mathbf{birth}_i^{t-1}$
- (Build intervals after the critical level) While $|S_i^t| \geq 7$ do:
  - $i \longleftarrow i + 1$
  - If S_{i-1}^t is left-leaning with respect to f^{t-1} with probability at least $1/2$ then $T_i^t \longleftarrow$ **left-half**(S_{i-1}^t) otherwise $T_i^t \longleftarrow$ **right-half**(S_{i-1}^t)
  - $S_i^t \longleftarrow$ **middle-third**(T_i^t).
  - $\mathbf{birth}_i^t \longleftarrow t$. [Record that S_i^t and T_i^t were rebuilt]
- $d^t \longleftarrow i$.
- $y^t \longleftarrow \min(S_{d^t}^t) + 2^{n-t}$.

Output: $y^1, y^2, \ldots, y^n$.

Fig. 1. Pseudocode for the adversary

Lemma 1. *Let $n \leq m$ be integers. 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:*

$$\chi_{\mathbf{A}}(y^1, y^2, \ldots, y^n) \geq \frac{5}{96}\left(\frac{1}{6}\right)^{2^8 c \log(2c)} (n+1)\log(n+1) - \frac{n}{4},$$

where $c = \log(m+1)/\log(n+1)$.

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**$(\mathbf{A}, n, m)$. In what follows, we fix $\mathbf{A}, n, m$. **Adversary**$(\mathbf{A}, n, m)$ determines $y^1, \ldots, y^n$ and the critical levels $q^1, \ldots, q^n$. Note that the definition of q^j only depends on the expected behavior of the algorithm through the construction of f^{j-1}. For purposes of analysis, we also define the critical level q^{n+1} based on the expected behavior of $f^1, \ldots, f^n$ in exactly the same way.

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:*

$$\chi_{\mathbf{A}}(y^1, y^2, \ldots, y^n) \geq \frac{1}{40}\sum_t |S^t_{q^{t+1}} \setminus S^t_{1+q^{t+1}}|,$$

where the sum ranges over time steps $t \in [1, n]$ for which $q^{t+1} < d^t$.

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 $\mathbf{A} = A$.

For any subset S of Y^t, $\mathbf{length}^t_{\mathbf{A}}(S) = f^t_{\mathbf{A}}(\max(S)) - f^t_{\mathbf{A}}(\min(S))$.
For a (i, t) such that $i < d^t$, $\mathbf{shrink}^t_{i,\mathbf{A}}$ is the 0-1 indicator of the event that $\mathbf{length}^{t-1}_{\mathbf{A}}(S^t_{i+1}) \leq \mathbf{length}^{t-1}_{\mathbf{A}}(S^t_i)/2$.
Define $\mathbf{shrink}^t_{\mathbf{A}} = \sum_{i=1}^{d^t-1} \mathbf{shrink}^t_{i,\mathbf{A}}$.

Proof of Lemma 2. For $t = 1$ the claim is trivial so assume $t > 1$. For any algorithm A, $\mathbf{length}^{t-1}_A(S^t_1) = m+1$ and $\mathbf{length}^{t-1}_A(S^t_{d^t}) \geq 2$, and $\mathbf{length}^{t-1}_A(S^t_i) > \mathbf{length}^{t-1}_A(S^t_{i+1})$ for $i \in [1, d^t - 1]$. Therefore $\mathbf{shrink}^t_{i,A}$ can be 1 for at most $\log(m+1) - 1$ values of i. Thus $\mathbf{shrink}^t_A \leq \log(m+1) - 1$.

Next we claim that for $i \in [1, d^t - 1]$, $\mathbf{Pr}[\mathbf{shrink}_i^t = 1] \geq 1/4$. This claim implies $\mathbf{E}\left[\mathbf{shrink}_{\mathbf{A}}^t\right] \geq (d^t - 1)/4$ 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_{i+1}^t and S_{i+1}^t are rebuilt at time t. By definition of the adversary T_{i+1}^t is either $\mathbf{left\text{-}half}(S_i^t)$ or $\mathbf{right\text{-}half}(S_i^t)$. Furthermore this choice is made so that $\mathbf{length}^{t-1}(T_{i+1}^t) \leq \mathbf{length}^{t-1}(S_i^t)/2$ with probability at least $1/2$ and since $S_{i+1}^t \subseteq T_{i+1}^t$, $\mathbf{Pr}[\mathbf{shrink}_i^t = 1] \geq 1/2$.

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

To see the final claim, assume that the event $\mathbf{Stable}_{i+1}^{t-1}$ occurred. For each endpoint of T_{i+1}^{t-1}, its label remained the same under each of the functions $f^{s-1}, \ldots, f^{t-1}$, and by the laziness of the algorithm, it also happened that for each endpoint of S_i^{t-1}, its label remained the same under each of the functions $f^{s-1}, \ldots, f^{t-1}$. Thus if, in addition, $\mathbf{shrink}_i^s$ happens then so does $\mathbf{shrink}_i^t$. □

Proof of Lemma 3.

An *item-step pair* (y, u) is a pair where $y \in Y^u$. For each step t such that $q^{t+1} < d^t$ 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 $\sum_{t:q^{t+1}<d^t} \mathbf{E}\left[\mathbf{relabs}^t\right]$. We will show that $\mathbf{E}\left[\mathbf{relabs}^t\right] \geq \frac{1}{40}|S_{q^{t+1}}^t \backslash S_{1+q^{t+1}}^t|$, which will suffice to prove the lemma.

We now define W^t for each t such that $q^{t+1} < d^t$. Let $b^t = \mathbf{birth}_{1+q^{t+1}}^t$. For all steps $u \in (b^t, t]$ the sets $T_{1+q^{t+1}}^u$ are preserved and also the sets $S_{1+q^{t+1}}^u$ are preserved and so from step $u - 1$ to u they each change only by the addition of y^{u-1}. Defining for all steps s and levels i, $\Delta_i^s = T_i^s \backslash S_i^s$, we have that the sets $\Delta_{1+q^{t+1}}^u$ are all the same for each $u \in [b^t, t]$. We define W^t to be the set of pairs (y, u) with $y \in \Delta_{1+q^{t+1}}^u$ and $u \in [b^t, t]$, i.e., $W^t = \Delta_{1+q^{t+1}}^t \times [b^t, t]$.

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 $b^{t'} \leq t$. This means that level $1 + q^{t'+1}$ is not rebuilt at step $t + 1$ but level $1 + q^{t+1}$ is rebuilt at step $t + 1$, so $q^{t+1} > q^{t'+1}$. But then this contradicts $\Delta_{1+q^{t+1}}^u \cap \Delta_{1+q^{t'+1}}^u \neq \emptyset$ since $\Delta_{1+q^{t+1}}^u \subset T_{1+q^{t+1}}^u \subset S_{1+q^{t'+1}}^u$ while $\Delta_{1+q^{t'+1}}^u \cap S_{1+q^{t'+1}}^u = \emptyset$.

Finally, let us bound $\mathbf{E}\left[\mathbf{relabs}^t\right]$ from below. By the definition of the adversary $\Delta_{1+q^{t+1}}^t$ is the union of the two equal-sized sets $\mathbf{left\text{-}third}(T_{1+q^{t+1}}^t) \cup \mathbf{right\text{-}third}(T_{1+q^{t+1}}^t)$. By the definition of q^{t+1}, the probability that both $\min(T_{1+q^{t+1}}^t)$

and $\max(T^t_{1+q^{t+1}})$ have stable label during $[b^t - 1, t]$ is at most 3/4. By the laziness of the algorithm, on any run in which the left (resp. right) endpoint of $T^t_{1+q^{t+1}}$ has unstable label during $[b^t - 1, t]$ all items in **left-third**$(T^t_{1+q^{t+1}})$ (resp. **right-third**$(T^t_{1+q^{t+1}})$) have unstable label during $[b^t - 1, t]$ and so at least half the items of $\Delta^t_{1+q^{t+1}}$ have unstable label during $[b^t - 1, t]$. Since this occurs with probability at least 1/4, thus the expected number of relabel events is at least $|\Delta^t_{1+q^{t+1}}|/8$.

To complete the proof of the lemma, we show that $|\Delta^t_{1+q^{t+1}}| \geq \frac{1}{5}|S^t_{q^{t+1}} \backslash S^t_{1+q^{t+1}}|$. The sets $S^u_{q^{t+1}} \backslash S^u_{1+q^{t+1}}$ are the same for all $u \in [b^t, t]$ and the same is true for the sets $\Delta^u_{1+q^{t+1}}$. We compare these two sets for $u = b^t$. Letting $c = |S^{b^t}_{q^{t+1}}|$ we have $c \geq 7$ since q^{t+1} is not the last level at step b^t. Since $T^{b^t}_{1+q^{t+1}}$ and $S^{b^t}_{1+q^{t+1}}$ are rebuilt, $|T^{b^t}_{1+q^{t+1}}| \geq \lceil c/2 \rceil$ and $|\Delta^{b^t}_{1+q^{t+1}}| \geq 2\lfloor(\lceil c/2 \rceil)/3\rfloor \geq c/5$ (where the final inequality uses $c \geq 7$, and is tight for $c = 10$). □

4 Prefix bucketing and Tail bucketing

We will need several different variants of prefix bucketing game introduced by Dietz, Seiferas and Zhang [10]. We have k buckets numbered $1, \ldots, k$ in which items are placed. A *bucket configuration* is an arrangement of items in the buckets; formally it is a mapping $C : \{1, \ldots, k\}$ 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 $n^1, \ldots, n^\ell$ and must sequentially place each group of items into a bucket. The case that the sequence $n^1 = \cdots = n^\ell = 1$ is called *simple bucketing*. The placement is done in a sequence of 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 4.1 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:
$$\mathbf{cost_{cheap}}(C, p) = C(p).$$
- In *expensive* bucketing, the cost is the number of items in buckets p or higher before the placement:
$$\mathbf{cost_{exp}}(C, p) = \sum_{i=p}^{k} C(i).$$
- 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 $\mathbf{cost}_{1-\mathbf{disc}} = \mathbf{cost_{exp}}$.)
- 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 $\mathbf{cost}_{1-\mathbf{block}} = \mathbf{cost_{cheap}}$ and $\mathbf{cost}_{k-\mathbf{block}} = \mathbf{cost_{exp}}$.)

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.

4.1 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 $w(1-\beta)(\beta)^{p-j}$ and the number of items in bucket 1 is $w\beta^{p-1}$.

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 $\mathbf{f}_{\mathbf{A}}^t, S_i^t, q^t, y^t$ 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. As mentioned prior to stating Lemma 2, we define q^{n+1} for analysis purposes. For integer $i \in [k]$ define $\bar{i}$ to be $\bar{i} = (k+1) - i$. Define the placement sequence $p^1 = \bar{q}^2, \ldots, p^n = \bar{q}^{n+1}$, and consider the tail bucketing $B^0, \ldots, B^{n+1}$ 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 $\{S_i^t : 1 \le t \le n, 1 \le i \le d^t\}$ 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 $t \in [1,n]$ and any $j \in [1, d^t]$:*

$$|S_j^t \backslash S_{j+1}^t| \ge B_{\bar{j}}^{t-1} - 3.$$

Here, for the case $j = d^t$, we take S_{j+1}^t to be $\emptyset$.

Proof. We will actually prove:

$$\lceil \sum_{i \le \bar{j}} B_i^{t-1} \rceil + 2 \ge |S_j^t| \ge \lceil \sum_{i \le \bar{j}} B_i^{t-1} \rceil. \tag{1}$$

Given this we get:

$$|S_j^t \backslash S_{j+1}^t| \ge \lceil \sum_{i \le \bar{j}} B_i^{t-1} \rceil - (\lceil \sum_{i \le \bar{j}-1} B_i^{t-1} \rceil + 2) \ge B_{\bar{j}}^{t-1} - 3,$$

as required.

We prove (1) by induction on t. For $t = 1$ we have $d^1 = 1$, so we only need to check the case $j = 1$. We have $|S_1^1| = 2$, and $\bar{j} = k$ and $\sum_{i \le k} B_i^0 = 0$.

Let $t \ge 2$ and assume the claim is true for $t-1$. Let $j \in [1,k]$. Suppose first $j \le q^t$. By the definition of the critical level, $q^t \le d^{t-1}$. Therefore $j \le d^{t-1}$ and we may apply the induction hypothesis with $t-1$ and j. Since $j \le q^t$ $|S_j^t| = |S_j^{t-1}| + 1$. The conclusion then follows by induction if we can show that $\sum_{i \le \bar{j}} B_i^{t-1} - \sum_{i \le \bar{j}} B_i^{t-2} = 1$. This holds because $p^{t-1} = \bar{q}^t$ and so $\bar{j} \ge p^{t-1}$ and

therefore B^{t-1} is obtained from B^{t-2} by adding a single item at position p^{t-1} and redistributing items among the first p^{t-1} 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 $\beta(j) = \sum_{i \leq \bar{j}} B_i^{t-1}$. For $d^t \geq j > q^t$ we have $\bar{j} < p^{t-1}$ and the tail-bucketing rule implies $\beta(j) = \beta(j-1)/6$. Also, the rebuilding rule for S_j^t implies $|S_j^t|$ is between $\lceil S_{j-1}^t/6 \rceil$ and $\lceil S_{j-1}^t/6 \rceil + 1$ (which is verified by case analysis depending on $|S_{j-1}^t| \mod 6$).

Thus:

$$\begin{aligned}
|S_j^t| &\leq \lceil \frac{1}{6} |S_{j-1}^t| \rceil + 1 \\
&\leq \lceil \frac{1}{6} (\lceil \beta(j-1) \rceil + 2) \rceil + 1 \\
&\leq \lceil \frac{1}{6} \beta(j-1) \rceil + 2 \\
&= \lceil \beta(j) \rceil + 2,
\end{aligned}$$

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:

$$\begin{aligned}
|S_j^t| &\geq \lceil \frac{1}{6} |S_{j-1}^t| \rceil \\
&\geq \lceil \frac{1}{6} (\lceil \beta(j-1) \rceil) \rceil \\
&\geq \lceil \frac{1}{6} \beta(j-1) \rceil \\
&= \lceil \beta(j) \rceil.
\end{aligned}$$

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

$$\chi_{\mathbf{A}}(y^1, y^2, \ldots, y^n) \geq \frac{1}{40} (\min \mathbf{cost}_{\mathbf{cheap}}[Tail_{1/6}](p^1, \ldots, p^n) - 10n),$$

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 $\sum_t B_{p^t}^{t-1} = \sum_t B_{\bar{q}^{t+1}}^{t-1}$, which by Lemma 4 is bounded above by $\sum_t |S_{q^{t+1}}^t \setminus S_{1+q^{t+1}}^t| + 3n$. Split this latter sum according to $q^{t+1} < d^t$ or $q^{t+1} = d^t$. The terms for which $q^{t+1} = d^t$ are each at most 7 (since $|S_{d^t}^t| \leq 7$) and so:

$$\sum_t B^{t-1}_{\bar{q}^{t+1}} - 10n \leq \sum_{t:q^{t+1}<d^t} |S^t_{q^{t+1}} \backslash S^t_{1+q^{t+1}}|.$$

Now apply Lemma 3.

5 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 $\lfloor 4\log(m+1)\rfloor$.

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 β^{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:*

$$\mathbf{cost}_{\gamma-\mathbf{disc}}(p^1, \ldots, p^\ell) \geq \gamma^b \mathbf{cost}_{b-\mathbf{block}}(p^1, \ldots, p^\ell).$$

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 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 with $\log(n-1)$ buckets there is an (n,k)- placement sequence with $k = \log(n+1)$ 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:*

$$\mathbf{cost}_{\mathbf{cheap}}(r^1, \ldots, r^\ell | n^1, \ldots, n^\ell) = \sum_{j=1}^{\ell - 2^k + 1} n^j (\ell - j - 2^k + 2).$$

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

$$\mathbf{cost_{cheap}}(p^1,\ldots,p^\ell) \geq (\ell+1)(\sum_{j=1}^{r}(r-j)\gamma_j + 2 + \sum_{i=1}^{r+1}\gamma_i \log(\gamma_i)) + (\ell+1)(\log(\ell+1) - 2k)$$

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.

5.1 From $\mathbf{cost_{b-block}}$ 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 $\mathbf{cost_{b-block}}$ for simple bucketing.

Below we will prove Lemma 11 which shows that $\mathbf{cost_{b-block}}$ 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. 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.

$$\mathbf{cost_{exp}}(p^1, \ldots, p^\ell | n^1, \ldots, n^\ell) \geq \sum_{j=1}^{\ell-b} n^j(\ell - j - b + 1).$$

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

For a step i let $c^i(p^1, \ldots, p^\ell | n^1, \ldots, n^\ell)$ be the cost of the placement into p^i at step i. For $I \subseteq [1, \ell]$, let

$$c^I(p^1, \ldots, p^\ell | n^1, \ldots, n^\ell) = \sum_{i \in I} c^i(p^1, \ldots, p^\ell | n^1, \ldots, n^\ell). \tag{2}$$

Lemma 10. *Let $p^1, \ldots, p^\ell$ be a placement sequence with b buckets. Let $\theta \in [1, b]$ and let $I = \{i_1 < \cdots < i_h\}$ be the indices in $[1, \ell]$ such that $p^{i_j} > \theta$. Let $s^1, \ldots, s^h$ be the placement sequence into $b - \theta$ buckets given by $s^j = p^{i_j} - \theta$ and let $n^1, \ldots, n^h$ be given by $n^1 = i_1$ and for $j > 1$, $n^j = i_j - i_{j-1}$. Then for cost function $c \in \{\mathbf{cost_{cheap}}, \mathbf{cost_{exp}}\}$,*

$$c^I(p^1, \ldots, p^\ell) = c(s^1, \ldots, s^h | n^1, \ldots, n^h).$$

Proof. It suffices to show that for each $j \in [1, h]$, $c^{i_j}(p^1, \ldots, p^\ell) = c^j(s^1, \ldots, s^h | n^1, \ldots, n^h)$. Let $B^1, \ldots, B^\ell$ be the bucketing sequence associated to $(p^1, \ldots, p^\ell)$, and let $\tilde{B}^1, \ldots, \tilde{B}^h$ be the bucketing sequence associated to $(s^1, \ldots, s^h | n^1, \ldots, n^h)$.

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 $s^1, \ldots, s^h$, 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 $p^1, \ldots, p^\ell$ is the same as the cost of step j for $s^1, \ldots, s^h | n^1, \ldots, n^h$ 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 $p^1, \ldots, p^\ell$ be a placement sequence into bm buckets. There exists a placement sequence $s^1, \ldots, s^\ell$ for km buckets such that*

$$\mathbf{cost_{cheap}}(s^1, \ldots, s^\ell) \leq \mathbf{cost}_{b-\mathbf{block}}(p^1, \ldots, p^\ell).$$

Proof. Fix $p^1, \ldots, p^\ell$. We first describe the construction of the sequence $s^1, \ldots, s^\ell$ 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 $\hat{\mathbf{h}}$ in the partition separately specify s^i for $i \in \hat{\mathbf{h}}$.

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 $[\mathbf{h}_0, \mathbf{h}_{v+1}] - \{\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{v+1}\}$, $\lambda^i < j$.

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 $\hat{\mathbf{h}}$ for the interior of $\mathbf{h}$. Thus the order of $\mathbf{h}$ is equal to $|\hat{\mathbf{h}}|$. 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_j and g_{j+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 of length $v(\mathbf{h})$ to be the sequence $\Delta^1_{\mathbf{h}}, \ldots, \Delta^{v(\mathbf{h})}_{\mathbf{h}}$ given by $\Delta^j_{\mathbf{h}} = \mathbf{h}_j - \mathbf{h}_{j-1}$. (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 $i \in \hat{\mathbf{h}}$. Now view the remainder sequence $r^{\mathbf{h}_1}, \ldots, r^{\mathbf{h}_v}$ of $\mathbf{h}$ as a placement for the load sequence $\Delta^{\mathbf{h}}_1, \ldots, \Delta^{\mathbf{h}}_{v(\mathbf{h})}$. 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:

$$\mathbf{cost_{cheap}}(u^1, \ldots, u^{v(\mathbf{h})} | \Delta^1_{\mathbf{h}}, \ldots, \Delta^{v(\mathbf{h})}_{\mathbf{h}}) \leq \mathbf{cost_{exp}}(r^1, \ldots, r^{v(\mathbf{h})} | \Delta^1_{\mathbf{h}}, \ldots, \Delta^{v(\mathbf{h})}_{\mathbf{h}}).$$

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 $i \in \hat{\mathbf{h}}$, and by doing this for all $\mathbf{h} \in \mathcal{H}$ we get the sequence $s^1, \ldots, s^n$.

Since $j \in [0, m-1]$ 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 $\mathbf{h}^- = (\mathbf{h}_1 - \mathbf{h}_0, \mathbf{h}_2 - \mathbf{h}_0, \ldots, \mathbf{h}_{v(\mathbf{h})} - \mathbf{h}_0)$. Thus $\mathbf{h}^-$ is the set of indices of $p_{\mathbf{h}}$ corresponding to $\hat{\mathbf{h}}$.

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

$$
\begin{aligned}
\mathbf{cost}_{b-\mathbf{block}}(p^1, \ldots, p^\ell) &\overset{(A1)}{=} \sum_{\mathbf{h}\in\mathcal{H}} \mathbf{cost}_{b-\mathbf{block}}^{\{\hat{\mathbf{h}}_1,\ldots,\hat{\mathbf{h}}_{v(\mathbf{h})}\}}(p^1, \ldots, p^\ell) \\
&\overset{(A2)}{=} \sum_{\mathbf{h}\in\mathcal{H}} \mathbf{cost}_{\mathbf{exp}}^{\{\mathbf{h}_1^-,\ldots,\mathbf{h}_{v(\mathbf{h})}^-\}}(p^1_{\mathbf{h}}, \ldots, p_{\mathbf{h}}^{\mathbf{h}_{v(\mathbf{h})}-\mathbf{h}_0}) \\
&\overset{(A3)}{=} \sum_{\mathbf{h}\in\mathcal{H}} \mathbf{cost}_{\mathbf{exp}}(r^{\mathbf{h}_1}, \ldots, r^{\mathbf{h}_{v(\mathbf{h})}} | \Delta^1_{\mathbf{h}}, \ldots, \Delta^{v(\mathbf{h})}_{\mathbf{h}}) \\
&\overset{(A4)}{\geq} \sum_{\mathbf{h}\in\mathcal{H}} \mathbf{cost}_{\mathbf{cheap}}(u^{\mathbf{h}_1}, \ldots, u^{\mathbf{h}_{v(\mathbf{h})}} | \Delta^1_{\mathbf{h}}, \ldots, \Delta^{v(\mathbf{h})}_{\mathbf{h}}) \\
&\overset{(A5)}{=} \sum_{\mathbf{h}\in\mathcal{H}} \mathbf{cost}_{\mathbf{cheap}}^{\{\mathbf{h}_1^-,\ldots,\mathbf{h}_{v(\mathbf{h})}^-\}}(s^1_{\mathbf{h}}, \ldots, s_{\mathbf{h}}^{\mathbf{h}_{v(\mathbf{h})}-\mathbf{h}_0}) \\
&\overset{(A6)}{=} \sum_{\mathbf{h}\in\mathcal{H}} \mathbf{cost}_{\mathbf{cheap}}^{\{\hat{\mathbf{h}}_1,\ldots,\hat{\mathbf{h}}_{v(\mathbf{h})}\}}(s^1, \ldots, s^\ell) \\
&\overset{(A7)}{=} \mathbf{cost}_{\mathbf{cheap}}(s^1, \ldots, s^\ell).
\end{aligned}
$$

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 at the start of epoch E, all items currently stored are in buckets higher than level j and so contribute nothing to the $\mathbf{cost}_{b-\mathbf{block}}$ during epoch E so in accounting for the cost of steps of F we can omit all placements prior to E. During E 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 $p^1, \ldots, p^n$ be an arbitrary placement sequence into $\lfloor 4\log(m+1)\rfloor$ buckets. Then*

$$\mathbf{cost_{cheap}}[Tail_{1/6}](p^1,\ldots,p^n) \geq \frac{5}{12}\left(\frac{1}{6}\right)^{2^8 c\log(2c)}(n+1)\log(n+1),$$

where $c = \log(m+1)/\log(n+1)$.

Now Lemma 1 follows from the above Lemma and Corollary 1.

Proof of Lemma 12. Using Lemmas 5, 6 and 11 we obtain the following chain of inequalities

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

where $(p^1,\ldots p^n)$ is $(n,\lfloor 4\log(m+1)\rfloor)$-placement sequence and $(s^1,\ldots s^n)$ is (n,km)-placement sequence, where km is defined as in Lemma 11. To obtain the lower bound we use Lemma 8. However we have to define the size of block b first, because it determines the actual number of buckets km. We choose b so that we obtain $\Omega(n\log n)$ lower bound, therefore we choose $b = 2^8 c\log(2c)$ where $c = \log(m+1)/\log(n+1)$. This implies

$$\begin{aligned}km &= \frac{\lfloor 4\log(m+1)\rfloor}{b}\cdot k\\ &\leq \frac{4c\log(n+1)\cdot\log(2^8 c\log(2c)+1)}{2^8 c\log(2c)}\\ &\leq \frac{\log(n+1)\log(2^{10}c^2)}{2^6\log(2c)}\\ &\leq \frac{\log(n+1)\log(2^{10}c^2)}{4\log(2^{16}c^{16})} \leq \frac{\log(n+1)}{4},\end{aligned}$$

where the first equality follows from the fact that $bm = \lfloor 4\log(m+1)\rfloor$ and the second one uses that $b = 2^k - 1$. The third inequality uses $x > \log x$ for $x > 0$ and $c \geq 1$. Therefore we obtain from Lemma 8 that $\mathbf{cost_{cheap}}(s^1,\ldots,s^n) \geq \frac{1}{2}(n+1)\log(n+1)$ and thus the proof is finished. □

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).