First Come First Served for Online Slot Allocation and Huffman Coding

Monik Khare Claire Mathieu* Neal E. Young†

full version with full proofs‡

Abstract

Can one choose a good Huffman code on the fly, without knowing the underlying distribution? *Online Slot Allocation (OSA)* models this and similar problems: There are n slots, each with a known cost. There are n items. Requests for items are drawn i.i.d. from a fixed but hidden probability distribution p. After each request, if the item, i, was not previously requested, then the algorithm (knowing c and the requests so far, but not p) must place the item in some vacant slot j_i, at cost $p_i\, c(j_i)$. The goal is to minimize the total cost $\sum_{i=1}^n p_i\, c(j_i)$.

The optimal offline algorithm is trivial: put the most probable item in the cheapest slot, the second most probable item in the second cheapest slot, etc. The optimal online algorithm is *First Come First Served* (FCFS): put the first requested item in the cheapest slot, the second (distinct) requested item in the second cheapest slot, etc. The optimal competitive ratios for any online algorithm are $1 + H_{n-1} \sim \ln n$ for general costs and 2 for concave costs. For logarithmic costs, the ratio is, asymptotically, 1: FCFS gives cost $\text{OPT} + O(\log \text{OPT})$.

For Huffman coding, FCFS yields an online algorithm (one that allocates codewords on demand, without knowing the underlying probability distribution) that guarantees asymptotically optimal cost: at most $\text{OPT} + 2\log_2(1 + \text{OPT}) + 2$.

1 Introduction

Modeling an algorithm by its worst-case performance can be overly pessimistic. Modeling by average-case performance on a specific input distribution can be overly optimistic. A natural compromise is modeling by average-case performance on an adversarially chosen distribution: a good algorithm should perform well on inputs drawn from any distribution in some large class. This "worst-distribution" approach is an emerging direction in the study of online algorithms, where standard (worst-case) competitive analysis can lead to overly pessimistic competitive ratios.

Here we introduce a online problem that we call *Slot Allocation (OSA)*. An instance is specified by n *slots*, with (known) respective costs $c(1) \le c(2) \le \cdots \le c(n)$, and a (hidden) probability distribution p on n items. Requests for items are drawn i.i.d. from p. If the requested item has not been requested before, the algorithm must assign the item i to some vacant (not-yet-assigned) slot, j_i, at cost $p_i\, c(j_i)$. Play stops once all items have been requested. The objective is to minimize the assignment cost, that is, $\sum_{i=1}^n p_i\, c(j_i)$. An *online* algorithm must choose each slot j_i as a function of just the slot costs c and the items chosen so far (but not p).

*École Normale Supérieure, Paris, France. Research partially supported by NSF grant CCF-0964037.

†University of California, Riverside, California, U.S.A. Research partially supported by NSF grant CCF-1117954.

‡Extended abstract appears in SODA 2014.

Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 445-454 (2014)
doi: 10.1137/1.9781611973402.33

The optimal offline solution is trivial: for each slot $j \in [n]$, put the jth most probable item in slot j. There is one natural online algorithm, which we call *First Come First Served* (FCFS): put the jth (distinct) item requested in slot j.

The *cost* of a randomized algorithm on (p, c) is its expected assignment cost (over all random choices of the algorithm and draws from p). The *competitive ratio* of an algorithm for a set X of inputs, is the supremum, over instances $(p, c) \in X$, of the algorithm's cost on (p, c) divided by the optimal offline cost for (p, c). We show that FCFS has the minimum competitive ratio of any online algorithm. We also determine the optimal ratio for various classes of costs: it is $1 + H_k$ for general costs (here $k \leq n - 1$ is the number of non-maximal coefficients in the cost vector) and 2 for concave costs. For logarithmic costs, the asymptotic competitive ratio[1] is 1: FCFS gives cost $\text{OPT} + O(\log \text{OPT})$.

We apply FCFS to *Online Huffman Coding (OHC)*. For OHC the offline problem is standard Huffman coding: given probability distribution p, find a prefix-free code χ over $\{0, 1\}$ minimizing the average codeword length, $\sum_{i=1}^{n} p_i |\chi_i|$ [20]. In the online problem, requests are drawn i.i.d. from p and revealed one by one. In response to each request, if the item i has not been requested before, the algorithm (knowing only the requests so far, but not p) must commit to its codeword χ_i for i. (If the requests were ordered adversarially, rather than drawn from p, no online algorithm could have bounded competitive ratio.)

Applying FCFS to OHC yields an algorithm that guarantees cost at most $\text{OPT} + 2\log_2(1 + \text{OPT}) + 2$. The algorithm uses a so-called *universal codeword set [7, 18]* to relax the prefix-free constraint. This makes OHC a special case of OSA; applying FCFS gives the algorithm.

Related work. Analyzing FCFS requires analyzing sampling without replacement from a non-uniform distribution. This poorly understood process (see e.g. [10, 11]) underlies the independent chip model (for valuing chips in poker tournaments, e.g. [15]) and Wallenius' noncentral hypergeometric distribution [40].

Adaptive Huffman coding. The so-called *adaptive Huffman coding* algorithm also requires no a-priori knowledge of the frequency distribution, but the algorithm modifies the code online: the ith item in the sequence is transmitted using the codeword from the optimal Huffman code for just the first $i - 1$ items (except for first occurrences) [8, 14, 26, 37, 38, 39]. Adaptive Huffman coding gives cost at most $\text{OPT} + 1$. Both online and adaptive Huffman coding require only one pass (regular Huffman coding requires two) and nearly minimize the cost, but online Huffman coding does so using a fixed (non-adaptive) code.

List management and paging. These two online problems [3, 36] are similar to OSA with, respectively, linear and 0/1 costs. The differences are that in OSA (a) the requests are drawn i.i.d. from a distribution, and (b) the algorithm is compared to the static OPT, which fixes the list order or cached set at the start and then doesn't change it. For paging, the OSA model is essentially the *independent reference model (IRM)*, which has a long history, e.g. [1, 12] and is still in use. For list management, the OSA model was used by Rivest [34] in a precursor to the worst-case model [36]. For list management, FCFS orders items by first request; for paging, FCFS fixes the first k (or $k - 1$) requested items in the cache. Our Thm. 3.1 implies that these simple static strategies are, respectively, 2-competitive and $(1 + H_k)$-competitive against their static OPTs.)

Other more sophisticated worst-distribution models for paging have been studied [23, 28, 33, 42].

In **online bin packing**, items with sizes arrive online to be packed into n unit-capacity bins. If sizes are i.i.d. from a discrete distribution, an online algorithm achieves cost $\text{OPT} + O(w(n))$, where

[1]Formally, we define the asymptotic competitive ratio to be the limit, as $h \to \infty$, of the competitive ratio against inputs where the entropy of p is at least h. For general costs and for concave costs the optimal asymptotic ratios equal the non-asymptotic ratios. This can be shown by adapting the lower-bound proofs in §4.

$w(n)$ is $\log n$ for distributions with *bounded waste* and $\sqrt{n}$ for those that are *perfectly packable* [4, 19]. In **online knapsack**, items with weights and values arrive online; each must be packed or rejected on arrival. Lueker studied a model where items are drawn randomly from any distribution in a large class [30]. In **online facility location**, points arrive online; the algorithm opens facilities to serve them. The worst-case ratio is $\Theta(\log n)$; for randomly ordered points, the competitive ratio is $O(1)$ [32]. In **online Steiner tree**, given a graph, terminal vertices arrive online and must be connected on arrival by adding a path to the current tree. The worst-case ratio is $\Theta(\log n)$, but if terminals are i.i.d. from a known distribution, the ratio is $O(1)$ [32].

In **the secretary problem**, a random permutation of ranked items is revealed item by item. The algorithm must reject or select each item as it is revealed. Only one item can be selected; the goal is to choose the top-ranked item [6, 16]. Similarly to OSA, the offline problem is trivial but the (random) online setting is interesting. Random online models for many variations of the secretary problem have been studied recently, mainly in the larger context of **online auctions** [2]. (See [9, 13] for older work.) In the **adwords** problem, given a known set of keywords, advertisers arrive online. Each advertiser comes with a subset of keywords, the algorithm must allocate an unused one to the advertiser (if any are still available). The objective is to maximize the number of assigned keywords. If the advertisers arrive in random order, the optimal competitive ratio is 0.696 or more [31, 21], better than the worst-case ratio of $1 - 1/e \approx 0.632$ [24]. Other random online models, including hidden distributions, have been studied too [5].

Open problems. For OSA with logarithmic costs and OHC, what are the optimal non-asymptotic competitive ratios? Are the $O(\log \text{OPT})$ additive gaps in our upper bounds tight? For problems such as paging, list management, and adaptive Huffman Coding (which allow dynamically adjusting the current online solution at some cost), for various "worst-distribution" models, what are the worst-case gaps between the costs of the static OPT and the dynamic OPT?

Preliminary. In the body of the paper, for convenience, we relax the assumption that p is a probability distribution (i.e., that $\sum_{i=1}^{n} p_i = 1$), replacing p by any frequency distribution $f \in \mathbb{R}_+^n$. (To sample from f, use the normalized distribution $f / \sum_{i=1}^{n} f_i$.) Also, $f(i)$ and $c(j)$ are synonymous with f_i and c_j, respectively.

2 FCFS is optimal for Online Slot Allocation

In this section we prove that the First-Come-First-Serve algorithm (FCFS) is optimally competitive for OSA:

Theorem 2.1. FCFS *is optimally competitive for any class X of OSA inputs that is closed under permutation of the frequency distribution (that is, for any $(f, c) \in X$, and any permutation π of $[n]$, the instance (f', c) is also in X, where $f'(i) = f(\pi_i)$).*

Intuitively, the theorem is not surprising, but the proof is technical (due to the unbounded depth of the game tree) and indirect.

For the proof we use the following formal definition of an algorithm for OSA. Define an *algorithm state* $u = (f, c, (i_1, \ldots, i_t), g, i)$ to be a tuple where (f, c) is the OSA instance, $(i_1, \ldots, i_t)$ are the first t requests, g is an assignment of slots to requested items, and i is request $t + 1$, which needs a slot (so $i \notin \{i_1, \ldots, i_t\}$). Formally, a (randomized) algorithm $\mathcal{A}$ is defined by, for each algorithm state u, a distribution $\mathcal{A}(u)$ on vacant slots, from which the slot for item i is chosen at random.

$\mathcal{A}$ is online if, for all $u = (f, c, (i_1, \ldots, i_t), g, i)$, distribution $\mathcal{A}(u)$ is determined by $(c, (i_1, \ldots, i_t), g, i)$ — the state without f.

$\mathcal{A}$ is *unbiased* if, for all $u = (f, c, (i_1, \ldots, i_t), g, i)$ distribution $\mathcal{A}(u)$ is determined by $(f, c, (i_1, \ldots, i_t), g)$, the state without i (the "name" of the requested item).

$\mathcal{A}$ is *stateless* if, for all $u = (f, c, (i_1, \ldots, i_t), g, i)$, distribution $\mathcal{A}(u)$ is determined by (f, c, W, V), where $W = [n] - \{i_1, \ldots, i_t\}$ is the set of items without slots and $V = [n] - \{g(i_1), \ldots, g(i_t)\}$ is the set of vacant slots.

Any stateless algorithm $\mathcal{A}$ is unbiased, but $\mathcal{A}$ can be stateless or unbiased without being online. FCFS is online, unbiased, and stateless.

Here is a summary of the proof:

1. *For any online algorithm there is an equally competitive unbiased algorithm.* This is simply by symmetry — in the event that the requested item i has not yet been requested, the "name" of the item gives no useful information, because the frequency distributions are symmetric (closed under permutation of the item names).

2. *For any input (f, c) and any unbiased algorithm, some stateless algorithm achieves the same cost.* An algorithm is stateless if it ignores repeat requests (requests to items that were previously requested). Intuitively, repeat requests give no useful information, because they do not change the part of the problem that remains to be solved.

3. FCFS *gives minimum cost for (f, c) among stateless algorithms.* Roughly, this is because, in expectation, the frequency of the first item chosen is larger than that of the second (or any subsequent) item. By a simple exchange argument, the optimal strategy may as well give the best slot (slot 1) to the first item chosen. By induction, the optimal strategy may as well be FCFS.

LEMMA 2.1. *For any online algorithm $\mathcal{A}$, for any class of inputs that is closed under permutation of the frequency distribution, there is an unbiased algorithm $\mathcal{A}'$ that is equally competitive.*

Proof. $\mathcal{A}'$ randomly relabels the items then simulates $\mathcal{A}$ on the relabeled instance. Relabeling makes $\mathcal{A}'$ unbiased. $\mathcal{A}'$ is as competitive as $\mathcal{A}$, because relabeling the items doesn't change OPT. Here are the details.

$\mathcal{A}'$, on a given instance (f, c), first renames the n items according to a random permutation π, then simulates $\mathcal{A}$ item by item on the renamed instance (f', c). (Formally, on any algorithm node $u = (f, c, (i_1, \ldots, i_t), g, i)$, the algorithm $\mathcal{A}'$ uses the slot distribution $\mathcal{A}'(u) = \mathcal{A}(u')$, where $u' = (f, c, (\pi(i_1), \ldots, \pi(i_t)), g_\pi, \pi(i))$ and g_π is defined by $g_\pi(\pi(i)) = g(i)$.)

The random renaming of items ensures that $\mathcal{A}'$ is unbiased. This is because, when an item i is presented to $\mathcal{A}'$ for the first time (i.e., $i \in [n] - \{i_1, \ldots, i_t\}$), the item $\pi(i)$ that $\mathcal{A}'$ presents to $\mathcal{A}$ is uniformly distributed over the items not yet seen by $\mathcal{A}$. Therefore, $\mathcal{A}'(u)$ is independent of u's particular item $i \in [n] - \{i_1, \ldots, i_t\}$.

$\mathcal{A}'$ is as competitive as $\mathcal{A}$, because, for any instance (f, c), the cost that $\mathcal{A}'$ incurs is at most the cost that $\mathcal{A}$ incurs on the permuted instance (f', c), while the optimal cost for (f', c) equals the optimal cost for (f, c). $\square$

Recall that unbiased/stateless algorithms "know" (f, c).

LEMMA 2.2. *Fix any instance (f, c) of OSA. For any unbiased algorithm $\mathcal{A}$, there is a stateless algorithm $\mathcal{A}'$ whose cost on (f, c) is at most the cost of $\mathcal{A}$ on (f, c).*

Proof. To reason precisely about running $\mathcal{A}$ on (f, c), we describe the process of choosing an allocation as a one-player game (against chance) in extensive form. To model that $\mathcal{A}$ is unbiased, we change the process slightly: we make $\mathcal{A}$ choose the next slot without knowing which unseen item the slot is for. More specifically, we break the step of choosing the next item i into two steps. First, we choose whether the next item will be taken from the seen items, or from the unseen items. In the former case, we immediately choose the next item from the seen items. In the latter case, before choosing the next item, we consult the algorithm $\mathcal{A}$ to determine the slot that the item will be assigned, and then choose the item after determining its slot.

The game tree T for the game that representing an instance (f, c) has three types of nodes, each identified with a possible state of the process. Each non-leaf node has edges to its children, representing the possible transitions to the next state.

- A draw node represents the state just before the next item is drawn. The node is a tuple $u = ((i_1, i_2, \ldots, i_t), g)$, where $i_1, \ldots, i_t$ is the sequence of items drawn so far, and $g : \{i_1, \ldots, i_t\} \to [n]$ is the (injective) allocation slots to drawn items. The root node of T is a draw node $u = ((), g)$ where g is the empty allocation.

  If all items have been seen (i.e., $\{i_1, \ldots, i_t\} = [n]$), then the draw node u is a leaf. Otherwise, the edges out of the node are as follows. For each seen item i, there is an edge labeled i to the draw node $w = ((i_1, i_2, \ldots, i_t, i), g)$. This edge is followed with probability proportional to f_i. Following the edge corresponds to drawing item i as the next item in the sequence.

  For the group of unseen items, there is a single edge to the choose node $w = ((i_1, i_2, \ldots, i_t), g, *)$. This edge is labeled $*$, and is followed with probability proportional to the sum of the frequencies of the unseen items. Following this edge corresponds to committing to draw an unseen item as the next item (but not yet drawing the item).

- A choose node represents the state just before the algorithm chooses the slot for the (yet to be chosen) unseen item. The node is a tuple $u = ((i_1, i_2, \ldots, i_t), g, *)$, where $i_1, \ldots, i_t$ and g are as for a draw node. For each unused slot j (i.e., $j \in [n] - \{g(i_1), g(i_2), \ldots, g(i_t)\}$), there is an edge labeled $* \mapsto j$ to the assign node $w = ((i_1, i_2, \ldots, i_t), g, j)$. The probabilities for the edges out of u are unspecified (they will be determined by the strategy chosen to play the game; in other words, by the algorithm $\mathcal{A}$). Following the edge corresponds to choosing the slot j to be assigned to the next (to be chosen) unseen item.

- An assign node represents the state just after the slot has been chosen, and just before the unseen item that will receive that slot has been chosen. The node is a tuple $u = ((i_1, i_2, \ldots, i_t), g, j)$, where $i_1, \ldots, i_t$ and g are as for a draw node, and $j \in [n]$ is a slot not used by g. For each unseen item i (i.e., $i \in [n] - \{i_1, \ldots, i_t\}$), there is an edge labeled $i \mapsto j$ to the draw node $w = ((i_1, \ldots, i_t, i), g')$, where g' is g extended by $g'(i) = j$ (assigning slot j to i). This edge is followed with probability proportional to i's frequency f_i (normalized by the sum of the frequencies of the unseen items). Taking the edge corresponds to assigning slot j to item i.

This defines the game tree T for (f, c). Formally, the game is a one-player game against chance, where the draw and assign nodes are chance nodes, and the choose nodes belong to the player. The payout (at each leaf node) is the cost of the final allocation g.

A (behaviorally randomized) *strategy* A for the player assigns, to every choose node u, a distribution $A(u)$ on the edges out of u. Such strategies correspond bijectively to the unbiased algorithms $\mathcal{A}$ for (f, c). The strategy A corresponding to $\mathcal{A}$ can be determined as follows. For choose node $u = ((i_1, \ldots, i_t), g, *)$, choose any unseen item i, let u' be the algorithm state $(f, c, (i_1, \ldots, i_t), g, i)$, and take $A(u) = \mathcal{A}(u')$, the slot-distribution for u'. Since $\mathcal{A}$ is unbiased, this distribution is independent of the choice of i.

Say that a strategy A is *stateless* if the distribution $A(u)$ at each choose node $u = ((i_1, \ldots, i_t), g, *)$ depends only on the set $W = [n] - \{i_1, \ldots, i_t\}$ of unseen items and the set $V = [n] - \{g(i_1), \ldots, g(i_t)\}$ of vacant slots. Call the pair (W, V) the *configuration* at node u. Crucially, the subtree T_u of T rooted at u (including the leaf costs, all edge labels, and all probabilities except for the probabilities assigned to the choose nodes by $\mathcal{A}$) is determined by the configuration. Intuitively, this means that the distribution $A(u)$ (and A's action throughout T_u) may as well depend only on the configuration. We show that this is true. This will complete the proof, if A is stateless, it's corresponding algorithm $\mathcal{A}$ is also stateless.

Fix any strategy A. Modify A as follows. Consider, one at a time, each equivalence class X of choose nodes, where two choose nodes are equivalent if they have the same configuration. Note that if $u = ((i_1, \ldots, i_t), g, *)$ and u' are both in X, then T_u and $T_{u'}$ are disjoint (because leaving node u increases the set of seen items). Consider the random experiment of playing the game once (using A to determine probabilities out of choose nodes). Letting random variable x be the first node in X (if any) encountered, and abusing notation, we can express the expected payout as

$$\begin{aligned} &\Pr[x \text{ undefined}]\, E\big[\operatorname{cost}(A) \,|\, x \text{ undefined}\big] \\ &\quad + \sum_{u \in X} \Pr[x = u]\big[\operatorname{cost}(g_u) + \operatorname{cost}(A_u)\big] \end{aligned}$$

where $\operatorname{cost}(g_u)$ is the cost of the partial allocation g at node u and $\operatorname{cost}(A_u)$ is the expected cost of the remaining allocation following node u.

Although X is infinite, by an averaging argument, there must be a node $W \in X$ such that

$$\operatorname{cost}(A_W) \;\leq\; \frac{\sum_{u \in X} \Pr[x = u] \operatorname{cost}(A_u)}{\sum_{u \in X} \Pr[x = u]}.$$

In other words, $\sum_{u \in X} \Pr[x = u] \operatorname{cost}(A_W)$ is at most $\sum_{u \in X} \Pr[x = u] \operatorname{cost}(A_u)$. Fix such a W. For each node u other than W in the equivalence class X, modify A by replacing A_u by A_W. (Here A_u represents A restricted to the subtree T_u. Recall that T_u and T_W are isomorphic.) The modified strategy A' has expected payout

$$\begin{aligned} &\Pr[x \text{ undefined}]\, E\big[\operatorname{cost}(A) \,|\, x \text{ undefined}\big] \\ &\quad + \sum_{u \in X} \Pr[x = u]\big[\operatorname{cost}(g_u) + \operatorname{cost}(A_W)\big]. \end{aligned}$$

By the choice of W, this is at most the expected payout for strategy A.

After A is modified in this way for every equivalence class of choose nodes, the resulting algorithm A' is stateless. (Note that the modification for one equivalence class leaves A' stateless within the previously modified equivalence classes.) □

The next and final step is to show that, for any instance (f, c), FCFS is optimal among stateless strategies. For stateless strategies, the underlying one-player game against chance has the following finite form: *Generate a random permutation $i_1, i_2, \ldots, i_n$ of the items by sampling from f without replacement. For each time $t = 1, 2, \ldots, n$, just before the next item i_t is drawn, choose the slot j_t that will hold that item i_t. Finally, pay the cost, $\sum_{t=1}^{n} f(i_t)c(j_t)$.*

Call this the *compact game* for (f, c).

Observation 2.3. *(a) The stateless algorithms for (f, c) correspond to the strategies for the compact game.*

(b) The compact game has an optimal strategy that is deterministic (i.e., each choice j_t is completely determined by $i_1, \ldots, i_{t-1}$).

Observation 2.3 (b) holds by a standard leaves-to-root induction (because the game is a finite one-player game against chance).

The final step requires an additional observation. In a permutation $i_1, \ldots, i_n$, generated by drawing items without replacement from f, items with larger frequency tend to be drawn earlier, so, not surprisingly, given two adjacent indices j and $j+1$, the expected frequency $E[f(i_j)]$ of the jth item drawn is as large as the expected frequency $E[f(i_{j+1})]$ of the $(j+1)$st item. Indeed, this holds even if we condition on the outcomes of all the other $n-2$ draws:

Observation 2.4. *Fix a frequency distribution $f(1) \geq \cdots \geq f(n)$. Let $i_1, i_2, \ldots, i_n$ be a random permutation of $[n]$ generated by sampling without replacement from f. Then, for any index $j \in [n-1]$, given any outcome J of the $n-2$ random draws $i_1, \ldots, i_{j-1}$ and $i_{j+2}, \ldots, i_n$ (all the draws except i_j and i_{j+1}) the expectation of $f(i_j)$ is at least that of $f(i_{j+1})$:*

$$E[f(i_j) \,|\, J] \;\geq\; E[f(i_{j+1}) \,|\, J].$$

Proof. Fix any J. We show $E[f(i_j) - f(i_{j+1}) \,|\, J] \geq 0$.

Let a and b be the two slots not occurring in $i_1, \ldots, i_{j-1}$ or $i_{j+2}, \ldots, i_n$, so there are two possible permutations consistent with J: one in which $(i_j, i_{j+1}) = (a, b)$, the other in which $(i_j, i_{j+1}) = (b, a)$. Call these two permutations A and B, respectively, so that $E[f(i_j) - f(i_{j+1}) \,|\, J]$, the expectation in question, is

$$\begin{aligned} &\Pr[A \,|\, J]\,\big(f(a) - f(b)\big) \;+\; \Pr[B \,|\, J]\,\big(f(b) - f(a)\big) \\ &= \frac{\Pr[A] - \Pr[B]}{\Pr[J]}\,\big(f(a) - f(b)\big). \end{aligned}$$

Assume without loss of generality that $a < b$, so $f(a) \geq f(b)$. To show the expectation is non-negative, we verify $\Pr[A] \geq \Pr[B]$.

By calculation, the (unconditioned) probability that a given permutation $I = (i_1, \ldots, i_n)$ occurs is $\Pr[I] = \prod_{t=1}^{n} f(i_t)/S_I(t)$, where $S_I(t)$ is the tail sum $\sum_{s=t}^{n} f(i_s)$.

Applying this identity to A and to B, then canceling common terms, $\Pr[A]/\Pr[B]$ is $S_B(j+1)/S_A(j+1)$, which equals $[f(a) + S_B(j+2)]/[f(b) + S_A(j+2)]$. Since $f(a) \geq f(b)$ and $S_A(j+2) = S_B(j+2) \geq 0$, the ratio $\Pr[A]/\Pr[B]$ is at least 1. □

Next we complete the final step:

Lemma 2.5. *Fix any instance (f, c) of OSA. Among stateless algorithms,* FCFS *gives minimum cost for (f, c).*

Proof. We show that FCFS gives minimum cost among deterministic strategies for the compact game (this proves the lemma by Observation 2.3). If $n \leq 1$ the lemma is trivial. Assume $n \geq 2$ and fix any optimal deterministic strategy A for the compact game. Let j_1 be the first slot that A chooses for the instance (f, c). Recall that $c_1 \leq c_2 \leq \cdots \leq c_n$.

After A chooses its first slot j_1 (deterministically) and assigns it to the item i_1 drawn subsequently from f, the rest of the game corresponds to a compact game for an instance (f', c') of cardinality $n-1$, where f' is obtained from f by deleting item i_1, and c' is obtained from c by deleting slot j_1. By induction, the FCFS strategy is optimal for that smaller instance. So (by modifying A, if necessary, but without increasing its cost) assume that A chooses all remaining slots in order of increasing cost (breaking ties among slots of equal cost by preferring slots with minimum index j).

```
input: frequency distribution $f$ over $[n]$, partition of $[n]$ into two sets $U$ and $\overline{U} = [n] - \overline{U}$.
output: random permutation $i_1, \ldots, i_n$ of $[n]$ that is distributed as if sampled without replacement from $f$
1. Draw random permutation $\pi = \pi_1, \ldots, \pi_{|U|}$ of $U$ by sampling without replacement from $f$ restricted to $U$.
2. Draw random permutation $\overline{\pi} = \overline{\pi}_1, \ldots, \overline{\pi}_{|\overline{U}|}$ of $\overline{U}$ by sampling without replacement from $f$ restricted to $\overline{U}$.
3. Define $f(\pi) = \sum_{i \in \pi} f_i$ to be the sum of the frequencies of items in sequence $\pi$.
4. For $j = 1, 2, \ldots, n$:                                                          (merge)
5.     With probability $f(\pi)/\big(f(\pi) + f(\overline{\pi})\big)$, let $i_j \leftarrow$ pop($\pi$);
6.     otherwise let $i_j \leftarrow$ pop($\overline{\pi}$).        (pop($\pi$) deletes & returns the first item remaining in $\pi$)
```

Figure 1: A procedure for sampling without replacement, per Lemma 3.1 (merging).

If slot j_1 is the minimum-cost slot 1, then A is FCFS, and we are done. Otherwise, since A plays FCFS after the first item (and ties are broken consistently) A chooses the minimum-cost slot among $[n] - \{j_1\}$ for the second item. This is slot 1. That is, A plays first slot j_1, and then slot 1. Consider the strategy A' differs from A only in that for the first two items A' plays slot 1 and then slot j_1. The cost of A' minus the cost of A is

$$E\big[\, c(1)\, f(i_1) \;+\; c(j_1)\, f(i_2)\,\big] \\ -\, E\big[\, c(j_1)\, f(i_1)\big] \;-\; c(1) f(i_2)\,\big],$$

which equals $\big(c(1) - c(j_1)\big)\, E[f(i_1) - f(i_2)]$.

This is non-positive because $c(1) \le c(j_1)$ and, by Observation 2.4, $E[f(i_1)] \ge E[f(i_2)]$. Therefore, this algorithm A' also gives minimum expected cost.

A' plays slot 1 first. By induction on n, replacing A's subsequent $n-1$ choices by the choices that would be made by FCFS does not increase the expected cost. The resulting algorithm is FCFS. Thus, FCFS has minimum expected cost for (f, c). □

Lemmas 2.1, 2.2, and 2.5 imply Thm. 2.1.

3 Competitive ratio of FCFS for OSA

We have established that FCFS is optimally competitive. In this section we bound its competitive ratio from above. (Section 4 has matching lower bounds.)

Recall that $c_1 \le c_2 \le \cdots \le c_n$. The cost vector is *concave* if $c_{i+2} - c_{i+1} \le c_{i+1} - c_i$ for all $i \in [n-2]$. Recall $H_j = 1 + \frac{1}{2} + \cdots + \frac{1}{j} \sim \ln j$.

THEOREM 3.1. *The competitive ratio of the First-Come-First-Served algorithm for Online Slot Allocation is bounded as follows:*

(i) For general costs, the ratio is at most $1 + H_K$, where there are K non-maximum coefficients in the cost vector: $K = |\{i \in [n] \mid c_i < \max_j c_j\}| \le n - 1$.

(ii) For concave costs, the ratio is at most 2.

(iii) For costs c with $c_j = \log_2 j + O(\log\log j)$, FCFS returns an allocation with cost at most $\mathcal{H}(f) + O(\log \mathcal{H}(f))$, where $\mathcal{H}(f)$, the entropy of $f/\sum_i f_i$, is a lower bound on OPT $+ O(\log \text{OPT})$.

Proof. Assume throughout that, for any instance (f, c) of OSA, the frequency distribution is non-increasing: $f_1 \ge f_2 \ge \cdots f_n > 0$ (this is without loss of generality, as FCFS is unbiased). Hence, $\text{OPT}(f, c) = \sum_{j=1}^{n} c_j f_j$.

Preliminaries. Throughout we model FCFS on (f, c) via the following equivalent random process (the compact game from Section 2): *Generate a random permutation $i_1, i_2, \ldots, i_n$ of $[n]$ by sampling from f without replacement. For $j = 1, 2, \ldots, n$, put item i_j in slot j.* The cost of FCFS on (f, c) is then $\sum_{j=1}^{n} c_j f(i_j)$.

To bound the probabilities of various events related to this process, we observe that it can be viewed as a recurse-then-merge process, similar to mergesort, as shown in Fig. 1.

LEMMA 3.1. (MERGING) *Let f be a frequency distribution on $[n]$, and let $(U, \overline{U})$ be a partition of $[n]$ into two sets. Given f and $(U, \overline{U})$, the random permutation $i_1, \ldots, i_n$ of $[n]$ generated by the procedure in Fig. 1 is distributed as if it were obtained by sampling without replacement from f.*

Proof. The case that U or $\overline{U}$ is empty is trivial, so assume otherwise.

The probability that a given item $i \in U$ is taken first is $[f(\pi)/(f(\pi) + f(\overline{\pi}))] \times [f_i/f(\pi)] = f_i/[f(\pi) + f(\overline{\pi})]$.

The probability that a given item $\overline{i} \in \overline{U}$ is taken first is $[f(\overline{\pi})/(f(\pi) + f(\overline{\pi}))] \times [f_{\overline{i}}/f(\overline{\pi})] = f_{\overline{i}}/[f(\pi) + f(\overline{\pi})]$.

Thus, the first item is distributed correctly. By induction on n, the remaining items are distributed as a random permutation drawn without replacement, from f with the first item deleted. This gives the correct distribution on the entire permutation. $\square$

It suffices to consider cost vectors in a spanning set:

LEMMA 3.2. *If the competitive ratio of* FCFS *is at most λ for all inputs having cost vectors in some set X, where each vector in X is non-decreasing, then the competitive ratio of* FCFS *is also at most λ for all inputs having cost vectors in the positive linear span of X.*

Proof. Fix any instance (f, c') where c' is in the positive linear span of X, that is, $c' = \sum_{c \in X} \alpha_c \, c$ where each $\alpha_c \in \mathbb{R}_+$. By assumption each vector $c \in X$ is non-decreasing, so c' is also.

Let $i_1, i_2, \ldots, i_n$ be a random permutation obtained by sampling without replacement from f. The expected cost of FCFS on (f, c') is (by linearity of expectation)

$$E\Big[\sum_{j=1}^{n} c'_j \, f_{i_j}\Big] \;=\; \sum_{c \in X} \alpha_c \; E\Big[\sum_{j=1}^{n} c_j \, f_{i_j}\Big].$$

Each term $E\big[\sum_{j=1}^{n} c_j \, f_{i_j}\big]$ is the expected cost of FCFS on instance (f, c), which by assumption is at most $\lambda \, \text{OPT}(f, c) = \lambda \sum_{j=1}^{n} c_j f_j$. Thus, the right-hand side above is at most

$$\sum_{c \in X} \alpha_c \, \Big[\lambda \sum_{j=1}^{n} c_j \, f_j\Big] \;=\; \lambda \sum_{j=1}^{n} c'_j \, f_j \;=\; \lambda \, \text{OPT}(f, c') \, . \qquad \square$$

Now we prove each part in turn.

Part (i) – general costs. Fix any instance (f, c). Recall that f is non-decreasing and c is non-increasing. Assume further that, for some $k \le K$, the cost vector c satisfies $c_1 = c_2 = \cdots = c_k = 0$ and $c_{k+1} = \cdots = c_n = 1$. (This is without loss of generality by Lemma 3.2, as such cost vectors span all non-decreasing cost vectors in $\mathbb{R}^n_+$ with at most K non-maximal coefficients.) Call items $1, \ldots, k$ *large* and the remaining items $k + 1, \ldots, n$ *small*. Let ϵ denote OPT's cost, $\epsilon = \sum_{i=k+1}^{n} f_i$, the sum of the small items' frequencies.

Let $\mathcal{I} = (i_1, \ldots, i_n)$ of be a random permutation of the items obtained by sampling from f without replacement. For the analysis, generate the random permutation $\mathcal{I}$ via the process described in Lemma 3.1: choose a random permutation $\mathcal{L} = (\ell_1, \ldots, \ell_k)$ of the large items (sampling from f restricted to large items); choose a random permutation $\mathcal{S}$ of the small items (sampling from f restricted to small items); then, in the *merge phase*, for each $j = 1, 2, \ldots, n$, obtain i_j by taking either the first remaining large item (with probability proportional to the frequency of the remaining large items) or the first remaining small item (with probability proportional to the frequency of the remaining small items).

The small items can't contribute more than ϵ to the cost of FCFS. We focus just on the large items, and show that they contribute in expectation at most ϵH_k.

FCFS pays for any item that isn't chosen within the first k iterations of the merge phase. Focus on just these iterations: fix any $j \le k$ and consider the start of iteration j. Let h be the number of large items chosen so far; these items will definitely be free. Further, FCFS will definitely pay for p large items, where $p = j - 1 - h$ is the number of not-yet-chosen large items (i.e., $k - h$) minus the number of iterations left within the first k (i.e., $k - j + 1$). The h free items are first in $\mathcal{L}$; the p to-be-paid items are last:

$$\mathcal{L} = (\underbrace{\ell_1, \ell_2, \ldots, \ell_h}_{h \text{ free}}, \underbrace{?, ?, \ldots, ?}_{\text{not determined}}, \underbrace{\ell_{k-p+1}, \ldots, \ell_k}_{p \text{ paid}}). \tag{3.1}$$

During iteration j, the status (paid or free) of exactly one large item will be determined. Define X_j to be the corresponding contribution to the cost: If iteration j chooses a large item, ℓ_{h+1}, then that item becomes *free*, and X_j is zero. Otherwise, the last not-determined item, ℓ_{k-p}, becomes *paid*, and X_j is that item's frequency, $f(\ell_{k-p})$.

FCFS pays in total $\sum_{j=1}^{k} X_j$ for the large items.

Let $\Phi_j(F, P)$ denote the event that, at the start of iteration j of the merge phase, the sequence of free items is F and the sequence of paid items is P. Let N denote the unordered set of not-determined large items (abusing notation, $N = [k] - F - P$). Define $f(N) = \sum_{i \in N} f_i$.

LEMMA 3.3. *For any (F, P), the probability that item i_j is small, conditioned on $\Phi_j(F, P)$, is at most $\epsilon / f(N)$.*

Proof. Condition further on $\mathcal{S} = S$ and $\mathcal{L} = L$, where S is any permutation of the small items and L is any permutation of the large items that starts with F and ends with P. With $\Phi_j(F, P)$, this completely determines the entire state at the start of iteration j of the merge phase, including the outcomes of iterations $1, \ldots, j-1$. Starting in any such state, the probability that iteration j chooses a small item is the following ratio: the total frequency of the not-yet-chosen small items (the last $n - k - |P|$ items in S), divided by the total frequency of all not-yet-chosen items (the last $n - k - |P|$ in S and the last $k - |F|$ in L). The numerator is at most ϵ; the denominator is at least $f(N) + f(P) \ge f(N)$. So, for any (S, L),

$$\Pr[i_j \text{ small} \mid \mathcal{S} = S;\, \mathcal{L} = L;\, \Phi_j(F, P)] \;\le\; \epsilon / f(N).$$

So $\Pr[i_j \text{ small} \mid \Phi_j(F, P)] \le \epsilon / f(N)$. ☐

LEMMA 3.4. *For any (F, P), the conditional expectation of X_j, given $\Phi_j(F, P)$ and that item i_j is small, is at most $f(N)/|N|$.*

Proof. Let Ψ_j denote the event that $\Phi_j(F, P)$ occurs and i_j is small. This event determines that the large-item permutation $\mathcal{L}$ has F as a prefix and P as a suffix.

We claim that this is the only way that it conditions $\mathcal{L}$: that is, the conditional distribution of $\mathcal{L}$ is the same as that of a random permutation of the large items that is obtained just by sampling with repetition from f, conditioned on having F as a prefix and P as a suffix. Here's why: Fix any ordering S of the small items, and any two orderings L and L' of the large items that are consistent with Ψ_j (L and L' order N differently). All choices made in iterations $1, \ldots, j$ of the merge phase are independent of N's order, so

$$\Pr[\Psi_j \mid \mathcal{S} = S;\, \mathcal{L} = L] \;=\; \Pr[\Psi_j \mid \mathcal{S} = S;\, \mathcal{L} = L'].$$

This holds for any S, so

$$\Pr[\Psi_j \mid \mathcal{L} = L] \;=\; \Pr[\Psi_j \mid \mathcal{L} = L'].$$

That is, Φ_j reveals no information about the ordering of N within $\mathcal{L}$. This proves the claim.[2]

Now recall that $\ell_{|F|+1}, \ldots, \ell_{k-|P|}$ denotes the order of N within $\mathcal{L}$, so that $X_j = f(\ell_{k-|P|})$. By Observation 2.4, even conditioning $\mathcal{L}$ on F and P, we have $E[f(\ell_{|F|+1})] \geq E[f(\ell_{|F|+2})] \geq \cdots \geq E[f(\ell_{k-|P|})]$. The sum of these $|N|$ expectations is $f(N)$, so the last (and minimum) one is at most their average $f(N)/|N|$. $\square$

The two lemmas imply that, for any (F, P), the conditional expectation $E[X_j \mid \Phi_j(F, P)]$ is at most the product $\big(\epsilon/f(N)\big)\big(f(N)/|N|\big) = \epsilon/|N|$. Since $|N|$ is necessarily $k - j + 1$, this implies $E[X_j] \leq \epsilon/(k-j+1)$. Thus, the large items cost in expectation at most $\sum_{j=1}^{k} E[X_j] \leq \sum_{j=1}^{k} \epsilon/(k-h+1) = \epsilon H_k$.

This proves Part (i) of Thm. 3.1.

Part (ii) – concave costs. Fix any instance (f, c) where c is concave and f is non-decreasing. Assume $c_1 = 0$ (without loss of generality, otherwise, subtracting c_1 from all costs decreases the cost of any allocation by $n\, c_1$, only improving the competitive ratio). Assume that, for some integer $k \geq 2$, each $c_j = \min(j, k) - 1$ (without loss of generality, by Lemma 3.2, as such vectors span all concave cost vectors with $c_1 = 0$).

Generate random permutation $i_1, \ldots, i_n$ by drawing without replacement from f. Let j_i denote the position of item i in the permutation (and the slot FCFS puts i in, at cost $c_{j_i} = \min(j_i, k) - 1$). The cost of FCFS is upper-bounded by $\sum_{i=1}^{k-1} (j_i - 1) f_i + \sum_{i=k}^{n} (k-1) f_i$.

For any two items $h, i \in [n]$, applying Lemma 3.1 (merging) to the two singleton lists (h) and (i), the probability that h occurs before i in the random permutation is $f_h/(f_i + f_h)$. Hence, for $i \in [k-1]$, the term $(j_i - 1)$ in the upper bound, which equals the number of items drawn before item i, has expectation

$$\sum_{h \neq i} \Pr[j_h < j_i] \;=\; \sum_{h \neq i} \frac{f_h}{f_i + f_h} \;\leq\; i - 1 + \sum_{h=i+1}^{n} \frac{f_h}{f_i}.$$

Hence, the expectation of the upper bound is at most

$$\sum_{i=1}^{k-1} \Big(i - 1 \;\; + \sum_{h=i+1}^{n} \frac{f_h}{f_i} \Big) f_i \;\; + \sum_{i=k}^{n} (k-1) f_i.$$

[2]Here's a sketch of a more detailed proof of this conclusion. Let $\mathcal{U}$ be the set of large-item permutations that start with F and end with P. For $L \notin \mathcal{U}$, $\Pr[\mathcal{L} = L \mid \Psi_j] = \Pr[\mathcal{L} = L \mid \mathcal{L} \in \mathcal{U}] = 0$. For $L, L' \in \mathcal{U}$ the equation before the footnote implies (by calculation)

$$\frac{\Pr[\mathcal{L} = L \mid \Psi_j]}{\Pr[\mathcal{L} = L' \mid \Psi_j]} = \frac{\Pr[\mathcal{L} = L]}{\Pr[\mathcal{L} = L']} = \frac{\Pr[\mathcal{L} = L \mid \mathcal{L} \in \mathcal{U}]}{\Pr[\mathcal{L} = L' \mid \mathcal{L} \in \mathcal{U}]}.$$

Hence, for every L, $\Pr[\mathcal{L} = L \mid \Psi_j] = \Pr[\mathcal{L} = L \mid \mathcal{L} \in \mathcal{U}]$.

Canceling f_i's and simplifying, this is

$$\begin{aligned}
&\sum_{i=1}^{k-1}(i-1)f_i \;+\sum_{h=2}^{n} f_h \sum_{i=1}^{\min(h,k-1)} 1 \;+\sum_{i=k}^{n}(k-1)f_i \\
&= \;2\sum_{h=1}^{k-1}(h-1)f_i \;+2\sum_{h=k}^{n}(k-1)f_h \;=\; 2\,\text{OPT}(f,c).
\end{aligned}$$

Part (iii) – logarithmic cost. Fix any instance (f, c) with $c_j = \log_2 j + O(\log\log j)$. Assume to ease notation that $\sum_i f_i = 1$. Let $i_1, \ldots, i_n$ be a random permutation obtained by drawing from f without replacement. Let j_i be the location of i in the permutation.

LEMMA 3.5. *For any item i and concave function F, $E[F(j_i)] \le F(1/f_i)$.*

Proof. For any other item $h \ne i$, applying Lemma 3.1 (merging) to the two singleton lists (h) and (i), the probability that h occurs before i in the random permutation is $f_h/(f_i + f_h)$. Since $j_i - 1$ is the number of items drawn before item i, by linearity of expectation,

$$E[j_i] \;=\; 1 + \sum_{j\ne i} \frac{f_j}{f_i + f_j} \;\le\; 1 + \frac{1-f_i}{f_i} \;=\; \frac{1}{f_i}.$$

F's concavity gives $E[F(j_i)] \le F(E[j_i]) \le F(1/f_i)$. □

Recall $c_j = \log_2 j + O(\log\log j)$. By Lemma 3.5 and the concavity of the logarithm, the expected cost of FCFS is

$$\begin{aligned}
\sum_{i=1}^{n} f_i\, E[c(j_i)] \;&=\; \sum_{i=1}^{n} f_i\, E[\log_2 j_i + O(\log\log j_i)] \\
&\le\; \sum_{i=1}^{n} f_i \left[\log_2 \frac{1}{f_i} + O\Big(\log\log \frac{1}{f_i}\Big)\right] \\
&=\; \mathcal{H}(f) + O(\log \mathcal{H}(f)).
\end{aligned}$$

(Above $\mathcal{H}(f)$ is the entropy of f.)

The minimum cost is $\text{OPT} = \sum_{i=1}^{n} c_i f_i$. By Kraft's inequality [29] and a calculation, there is a prefix-free code of cost $\sum_{i=1}^{n} f_i(c_i + O(\log c_i)) = \text{OPT} + O(\log \text{OPT})$. Since that code is prefix-free, its cost is at least the entropy of f, so $\mathcal{H}(f) \le \text{OPT} + O(\log \text{OPT})$. □

4 Lower bounds on optimal competitive ratio

In this section we prove the following tight lower bounds on the optimal competitive ratios of any online algorithm for OSA:

THEOREM 4.1. *For Online Slot Allocation:*

(i) For 0/1 costs with at most K non-zero coefficients ($\{|i \in [n] \mid c_i \ne \max_j c_j\} \le K$), the optimal competitive ratio is at least $1 + H_K$ (where $H_K = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{K} \sim \ln K$).

(ii) For concave 0/1 costs, the optimal competitive ratio is at least 2.

Proof. We prove bounds (i) and (ii) for FCFS. Since FCFS is optimally competitive, the theorem follows.

Part (i) – general costs. Fix arbitrarily large n and let $\epsilon \to 0$. Consider the instance (f, c) whose cost vector $c \in \{0,1\}^n$ has K zeros, where the first K frequencies equal 1 (call them *large*) and the last $n' = n - K$ frequencies equal ϵ/n' (call them *small*). The expected cost that FCFS pays for the large items is at least the probability that some large item ends up in position $K+1$ or higher, which equals the probability that some small item comes before some large item. Applying Lemma 3.1 (merging) to merge the small item into the large ones, the probability that all small items come *after* all K large items is

$$\prod_{\ell=0}^{K-1} \frac{K-\ell}{K-\ell+\epsilon} = \prod_{\ell=0}^{K-1} 1 - \frac{\epsilon}{K-\ell+\epsilon}$$
$$= 1 - \sum_{\ell=0}^{K-1} \frac{\epsilon}{K-\ell+\epsilon} + O(\epsilon^2) = 1 - \epsilon H_K + O(\epsilon^2).$$

Hence, the probability that some small item comes before some large item (a lower bound on FCFS's expected cost for the large items) is $\epsilon H_K - O(\epsilon^2)$.

FCFS's cost for the small items is at least $(n' - K)\epsilon/n' = \epsilon - O(\epsilon K/n)$. Thus, FCFS total expected cost is at least $\epsilon[1 + H_K - O(K/n + \epsilon)]$.

The minimum allocation cost is ϵ. The ratio tends to $1 + H_K$ (as $\epsilon \to 0$ and $n/K \to \infty$), proving Part (i).

Part (ii) – concave costs. Fix any n. Consider the instance (f, c) with cost vector $c = (0, 1, 1, \ldots, 1)$ and frequency vector $f = (1, \epsilon, \epsilon, \ldots, \epsilon)$, where $\epsilon \to 0$. Let $i_1, \ldots, i_n$ be a random permutation obtained by drawing without replacement from f. FCFS's expected cost is *at least*

$$\Pr[i_1 \neq 1] \cdot 1 + \Pr[i_1 = 1] \cdot (n-1)\epsilon$$
$$= \frac{(n-1)\epsilon}{1 + (n-1)\epsilon} + \frac{(n-1)\epsilon}{1 + (n-1)\epsilon} = \frac{2(n-1)\epsilon}{1 + (n-1)\epsilon}.$$

The minimum allocation cost is $(n-1)\epsilon$. The ratio is $2/(1 + (n-1)\epsilon)$, which tends to 2 as $\epsilon \to 0$. □

5 Online Huffman coding (OHC)

In this section we prove the following performance guarantee for a FCFS algorithm for OHC:

THEOREM 5.1. *There exists an algorithm for Online Huffman Coding such that, for any instance f, the algorithm returns a prefix-free code of expected cost at most $\mathcal{H}(f) + 2\log_2(1 + \mathcal{H}(f)) + 2 \le 5\,\mathcal{H}(f)$, where $\mathcal{H}(f)$, the entropy of $f/\sum_{i=1}^n f_i$, is a lower bound on* OPT.

Online Huffman Coding is not a special case of Online Slot Allocation because of the prefix-free constraint: assigning a codeword j to a given item precludes the use of other codewords (those with j as a prefix). To work around this, the algorithm uses a so-called *universal (codeword) set* — an infinite prefix-free subset $\mathcal{U}$ of $\{0,1\}^*$ — to effectively relax the prefix-free constraint.

We first fix a particular universal codeword set with "small" codewords. Let $c_{\mathcal{U}}(j)$ be the length of the jth smallest string in $\mathcal{U}$. Call $c_{\mathcal{U}}$ the *cost function* for $\mathcal{U}$.

LEMMA 5.1. *There is a universal set $\mathcal{U}$ with cost $c_{\mathcal{U}}(j) = \lfloor 2 + \log_2 j + 2\log_2(1 + \log_2 j) \rfloor$.*

Proof. By calculation, the cost function satisfies $\sum_{j=1}^{\infty} 1/2^{c_{\mathcal{U}}(j)} \le 1$. The existence of the prefix-free set $\mathcal{U}$ follows by Kraft's inequality [29]. □

(For concreteness, here is a more explicit description of $\mathcal{U}$, following e.g., [17, 18]. For each string $x \in \{0,1\}^+$, add the string wx to $\mathcal{U}$, where w is as computed as follows. Let ℓ be the binary encoding of the length of x. Delete the first "1", replace each "0" or "1" by "00" or "11", respectively, then append a "01". This gives w. $\mathcal{U}$ is prefix-free because w encodes the length ℓ of x in a prefix-free fashion, and ℓ determines when x ends.)

For the rest of the section, fix $\mathcal{U}$ and $c_{\mathcal{U}}$ from Lemma 5.1. Define the Online Huffman Coding algorithm $\textsc{fcfs}_{\mathcal{U}}$ as follows: *Given a new item i, allocate the next smallest unused codeword in $\mathcal{U}$ to item i.* $\textsc{fcfs}_{\mathcal{U}}$ returns a prefix-free code because $\mathcal{U}$ is prefix-free.

Note that $\textsc{fcfs}_{\mathcal{U}}$ is $\textsc{fcfs}$ applied to the OSA instance $(f, c_{\mathcal{U}})$, which is equivalent to the OHC instance f with the additional constraint that codewords must be chosen from $\mathcal{U}$. The jth smallest string in $\{0,1\}^+$ has length $c_{\mathcal{U}}(j) - O(\log\log j)$, so it is easy to show that the added constraint increases $\textsc{opt}$ by at most an additive $O(\log \textsc{opt})$. This observation and Thm. 3.1, Part (iii), imply the following looser performance guarantee:

LEMMA 5.2. *For any instance f of Online Huffman Coding, the prefix-free code returned by $\textsc{fcfs}_{\mathcal{U}}$ has expected cost at most $\textsc{opt}(f) + O(\log \textsc{opt}(f))$.*

But a tighter, more direct analysis proves Thm. 5.1:

Proof. (Thm. 5.1.) Assume without loss of generality that $\sum_i f_i = 1$. Let $i_1, \ldots, i_n$ be a random permutation generated by drawing without replacement from f. Let r.. j_i be the position of i in the permutation.

By Lemma 3.5 and the concavity of the logarithm, $E[\log_2 j_i] \le \log_2(1/f_i)$. Summing over i gives

$$\sum_{i=1}^{n} f_i\, E[\log_2 j_i] \;\le\; \sum_{i=1}^{n} f_i \log_2 \frac{1}{f_i} \;=\; \mathcal{H}(f).$$

Using this inequality (twice), the choice of $c_{\mathcal{U}}$ in Lemma 5.1, and the concavity of log (twice), the expected cost of the allocation, $\sum_{i=1}^n f_i\, c_{\mathcal{U}}(j_i)$, is

$$2 + \Big(\sum_{i=1}^{n} f_i\, E[\log_2 j_i]\Big) + \; 2\log_2\big(1 + \textstyle\sum_{i=1}^n f_i\, E[\log_2 j_i]\big)$$

$\le\; 2 + \mathcal{H}(f) + 2\log_2(1 + \mathcal{H}(f)).$ □

References

[1] A. V. Aho, P. J. Denning, and J. D. Ullman. Principles of optimal page replacement. *Journal of the ACM*, 18(1):80–93, 1971.

[2] M. Babaioff, N. Immorlica, D. Kempe, and R. Kleinberg. Online auctions and generalized secretary problems. *SIGecom Exch.*, 7(2):7:1–7:11, June 2008.

[3] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*, volume 53. Cambridge University Press Cambridge, 1998.

[4] J. Csirik, D. S. Johnson, C. Kenyon, J. B. Orlin, P. W. Shor, and R. R. Weber. On the Sum-of-Squares algorithm for bin packing. *Journal of the ACM*, 53(1):1–65, 2006.

[5] N. R. Devanur. Online algorithms with stochastic input. *SIGecom Exch.*, 10(2):40–49, June 2011.

[6] E. E. B. Dynkin. The optimum choice of the instant for stopping a markov process. *Sov. Math. Dokl.*, 4, 1963.

[7] P. Elias. Universal codeword sets and representations of the integers. *Information Theory, IEEE Transactions on*, 21(2):194–203, 1975.

[8] N. Faller. An adaptive system for data compression. In *Record of the 7th Asilomar Conference on Circuits, Systems, and Computers*, pages 593–597, 1973.

[9] T. Ferguson. Who solved the secretary problem? *Statistical Science*, 4:282–289, Aug. 1989.

[10] A. Fog. Calculation methods for Wallenius' noncentral hypergeometric distribution. *Communications in Statistics, Simulation and Computation*, 37(2):258–273, 2008.

[11] A. Fog. Sampling methods for Wallenius' and Fisher's noncentral hypergeometric distributions. *Communications in Statistics, Simulation and Computation*, 37(2):241–257, 2008.

[12] P. A. Franaszek and T. J. Wagner. Some distribution-free aspects of paging algorithm performance. *Journal of the ACM*, 21(1):31–39, 1974.

[13] P. R. Freeman. The secretary problem and its extensions: A review. *International Statistical Review*, pages 189–206, 1983.

[14] R. Gallager. Variations on a theme by Huffman. *Information Theory, IEEE Transactions on*, 24(6):668–674, 1978.

[15] G. T. Gilbert. The independent chip model and risk aversion. *CoRR*, arxiv.org preprint 0911.3100, 2009. http://arxiv.org/abs/0911.3100.

[16] J. P. Gilbert and F. Mosteller. Recognizing the maximum of a sequence. *Journal of the American Statistical Association*, 61(313):35–73, 1966.

[17] M. J. Golin, C. Kenyon, and N. E. Young. Huffman coding with unequal letter costs. In *Proceedings of the thiry-fourth annual ACM symposium on theory of computing*, pages 785–791. ACM, 2002.

[18] M. J. Golin, C. Mathieu, and N. E. Young. Huffman coding with letter costs: A linear-time approximation scheme. *SIAM Journal on Computing*, 41(3):684–713, 2012.

[19] V. Gupta and A. Radovanovic. Online stochastic bin packing. *CoRR*, arxiv.org preprint 1211.2687, 2012. http://arxiv.org/abs/1211.2687.

[20] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[21] C. Karande, A. Mehta, and P. Tripathi. Online bipartite matching with unknown distributions. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011*, pages 587–596. ACM, 2011.

[22] A. R. Karlin, S. J. Phillips, and P. Raghavan. Markov paging. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science, FOCS '92*, pages 208–217. IEEE, 1992.

[23] A. R. Karlin, S. J. Phillips, and P. Raghavan. Markov paging. *SIAM Journal on Computing*, 30(3):906–922, 2000.

[24] R. M. Karp, U. V. Vazirani, and V. V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the twenty-second annual ACM symposium on Theory of computing, STOC '90*, pages 352–358. ACM, 1990.

[25] M. Khare, C. Mathieu, and N. E. Young. First-come-first-served for online slot allocation and huffman coding. *CoRR*, arxiv.org preprint 1307.5296, 2013. http://arxiv.org/abs/1307.5296.

[26] D. E. Knuth. Dynamic Huffman coding. *Journal of Algorithms*, 6(2):163–180, 1985.

[27] E. Koutsoupias and C. Papadimitriou. Beyond competitive analysis. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science, FOCS '94*, pages 394–400. IEEE, 1994.

[28] E. Koutsoupias and C. H. Papadimitriou. Beyond competitive analysis. *SIAM Journal on Computing*, 30(1):300–317, 2000.

[29] L. G. Kraft. *A device for quantizing, grouping, and coding amplitude-modulated pulses.* PhD thesis, Massachusetts Institute of Technology, 1949.

[30] G. S. Lueker. Average-case analysis of off-line and on-line knapsack problems. In *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms, SODA '95*, pages 179–188. SIAM, 1995.

[31] M. Mahdian and Q. Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing lps. In *Proceedings of the 43rd annual ACM symposium on Theory of computing, STOC 2011*, pages 597–606. ACM, 2011.

[32] A. Meyerson. Online facility location. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001*, pages 426–431. IEEE, 2001.

[33] P. Raghavan. A statistical adversary for on-line algorithms. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 7:79–83, 1992.

[34] R. L. Rivest. On self-organizing sequential search heuristics. *Communications of the ACM*, 19(2):63–67, 1976.

[35] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update rules. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing, STOC '84*, pages 488–492. ACM, 1984.

[36] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, Feb. 1985.

[37] J. S. Vitter. Design and analysis of dynamic Huffman coding. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science, FOCS '85*, pages 293–302. IEEE, 1985.

[38] J. S. Vitter. Design and analysis of dynamic Huffman codes. *Journal of the ACM*, 34(4):825–845, 1987.

[39] J. S. Vitter. Algorithm 673: dynamic Huffman coding. *ACM Transactions on Mathematical Software (TOMS)*, 15(2):158–167, 1989.

[40] K. T. Wallenius. *Biased Sampling: The Non-Central Hypergeometric Probability Distribution.* PhD thesis, Department of Statistics, Stanford University, 1963. Also published with the same title as Technical report No. 70.

[41] N. E. Young. Bounding the diffuse adversary. In *Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms, SODA '98*, pages 420–425. Society for Industrial and Applied Mathematics, 1998.

[42] N. E. Young. On-line paging against adversarially biased random inputs. *Journal of Algorithms*, 37(1):218–235, 2000.