Discrete Mathematics and Theoretical Computer Science DMTCS vol. **17:1**, 2015, 415–444

How often should you clean your room?

Kimball Martin* Krishnan Shankar†

Department of Mathematics, University of Oklahoma, Norman, Oklahoma, USA

received 10th Jan. 2015, revised 11th May 2015, accepted 15th June 2015.

We introduce and study a combinatorial optimization problem motivated by the question in the title. In the simple case where you use all objects in your room equally often, we investigate asymptotics of the optimal time to clean up in terms of the number of objects in your room. In particular, we prove a logarithmic upper bound, solve an approximate version of this problem, and conjecture a precise logarithmic asymptotic.

Keywords: search with cleanup, combinatorial optimization, coupon collector's problem, sequential occupancy, Stirling numbers of the second kind

Introduction

Suppose you have n objects in your room which are totally ordered. For simplicity, let us say they are books on shelves alphabetized by author and title. If you are looking for a book (assume you remember the author and title, but not its location on the shelves), the most efficient algorithm is a binary search. Namely, look at the book in the middle of the shelf, and because of the ordering, now you can narrow your search by half. Repeat this process of halving your search list, and you can find your book in about $\log_2 n$ steps. (Here is perhaps a better model for humans naturally search: go to where you think the book should be, scan that area, and if need be jump to a different area based on the ordering. However a logarithmic cost still seems like a good model for this process.)

The theory of searching (and sorting) algorithms is of course well studied in computer science—what is not, however, is what happens after that for humans. Namely, after you are done with your book, you can do one of two things: either put it back on the shelf, which we will also say takes about $\log_2 n$ time, or leave it on your desk, which takes no time. The latter is of course more efficient now, but if you keep doing this, eventually all of your books will wind up as an unsorted pile on your desk. Then when you search for a book, you essentially have to go through your pile book by book (a sequential, or linear, search), which takes about $\frac{n}{2}$ time, and thus is not very efficient for n large.

The question we are interested in here is: when is the optimal time to clean up? That is, over the long run, what is the optimal value $m_{\mathrm{opt}}(n)$ of m ($1 \leqslant m \leqslant n$) at which you should put all the books in the pile back, in order, on the shelf, in the sense that the the average search plus cleanup cost (per search) is

*Supported in part by Simons Collaboration Grant 240605.
†Supported in part by NSF Grant DMS #1104352.

minimized. Here we assume the cleanup algorithm is to simply go through the pile, book by book, and find the right location for each book on the shelves via a binary search (see Remark 1.2 for a discussion of other cleanup algorithms).

The paper is organized as follows. (See Section 1.3 for a more detailed overview.) In Section 1, after first formulating this problem precisely, we will discuss four different models and focus on the (generally unreasonable) case of the uniform distribution, i.e., where you use all objects in your room equally often. It might be more realistic to consider a power law distribution, but even the simple case of the uniform distribution is not so easy. The different models correspond to having either complete or no memory of what is in the pile, and having numbered shelves (each object has a designated location on the shelves) or unnumbered shelves (only the relative order of books is important).

In Section 2, we analyze the search and cleanup cost functions in some detail for each of these models. Our first result is that, in each of these models, one should not clean up immediately (see Proposition 1 below). In fact, if n is small enough, one should never cleanup (see Remarks 4.6 and 4.7). In Section 3, we restrict ourselves to complete memory with numbered shelves for simplicity, and prove that one should clean up before about $4\log_2(n)$ objects are in the pile (see Proposition 2). A good lower bound for the $m_{\mathrm{opt}}(n)$ is not so easy, and so we instead consider an approximate problem in Section 4. Based on the analysis from Section 2, we expect the optimal value $\widetilde{m}_{\mathrm{opt}}(n)$ of m for the approximate problem to be a lower bound for $m_{\mathrm{opt}}(n)$. We essentially determine exactly the optimal value of m for the approximate problem (Theorem 3), which is about $3\log_2(n)$, and then based on numerics conjecture that $m_{\mathrm{opt}}(n) \sim 3\log_2(n)$ (Conjecture 4). In fact we expect that for all four models with arbitrary distributions, $m_{\mathrm{opt}}(n)$ grows no faster than $4\log_2(n)$. Therefore, we humbly suggest you clean your room before $4\log_2(n)$ objects are out.

Since we use a fair amount of (often similar looking) notation, we provide a notation guide at the end for convenience (Appendix A).

1 General Setup

1.1 The Statement of the Problem

We now make a general formulation of our problem, which we call a *search with cleanup optimization problem*.

Let $\mathbf{X} = \{X_1, \ldots, X_n\}$ be a finite set of distinct well-ordered objects, which we view as a probability space with probability measure μ. We consider the following discrete-time Markov chain, depending on a parameter $1 \leqslant m \leqslant n$.

1. At time $t = 0$ each X_i is in a sorted list[(i)] $\mathcal{L}$, and there is an unsorted pile $\mathcal{P}$ which is empty.

2. At any time $t \in \{0, 1, 2, \ldots\}$, each $X \in \mathbf{X}$ is in exactly one of $\mathcal{L}$ and $\mathcal{P}$, i.e., $\mathbf{X}$ is a disjoint union $\mathbf{X} = \mathcal{L} \sqcup \mathcal{P}$.

3. At any time $t \geqslant 1$ with $|\mathcal{P}| < m$, exactly one object $X = X_i$ is selected, and X is selected with probability $\mu(X)$. If the selected $X \in \mathcal{P}$, nothing changes. Otherwise, then X is removed from $\mathcal{L}$ and added to $\mathcal{P}$.

[(i)] By list, we mean an indexed list, rather than a linked list.

4. At any time t, if $|\mathcal{P}| = m$, we stop the process.

This process has finite stopping time with probability 1 provided at least m elements of $\mathbf{X}$ have nonzero probabilities, which we will assume.

Note the state of the process at time t is described simply by a subset $\mathcal{P}$ of $\mathbf{X}$, together with a marked element X_{i_t} (the object selected at time $t \geqslant 1$). The set of possible states is then simply all subsets $\mathcal{P}$ of $\mathbf{X}$, together with a marked point X_{i_t}, of cardinality at most m.

Associated to this process are two (nonnegative) cost functions, $S(X;\mathcal{P})$ and $C(\mathcal{P})$, which do not depend upon t. Here $X \in \mathbf{X}$ and $\mathcal{P} \subset \mathbf{X}$. The functions $S(X;\mathcal{P})$ and $C(\mathcal{P})$ are called the search and cleanup costs.

Let $\mathcal{X}_m = \mathcal{X}_{m,n}$ denote the set of finite sequences $\chi = (X_{i_1}, \ldots, X_{i_\ell})$ in $\mathbf{X}$ such that (i) the underlying set $\{X_{i_1}, \ldots, X_{i_\ell}\}$ has cardinality m, and (ii) $X_{i_j} \neq X_{i_\ell}$ for $j < \ell$. We extend the measure μ to a probability measure on $\mathcal{X}_m$ by

$$\mu(\chi) = \prod_{j=1}^{\ell} \mu(X_{i_j}). \tag{1.1}$$

Note the sequences $\chi \in \mathcal{X}_m$ are in 1-1 correspondence with the possible paths of finite length for the Markov process from the initial state up to the stopping state described in Step 4. Namely, for $t = 0, \ldots, \ell$, let $\mathcal{P}_\chi(t)$ denote the set of elements $\{X_{i_1}, \ldots, X_{i_t}\}$ (here $P_\chi(0) = \emptyset$). Thus $\mathcal{P}_\chi(t)$ represents the "unmarked" state of the process from time $t = 0$ until the stopping time $t = \ell$. Furthermore each $\mu(\chi)$ is the probability of that path for the process.

For example, suppose $n \geqslant 3$, $m = 3$ and $\chi = (X_1, X_2, X_1, X_3)$. This corresponds to selecting X_1 at time 1, X_2 at time 2, X_1 at time 3, and X_3 at time 4, after which the process stops, since we have selected $m = 3$ distinct objects. Specifically, at $t = 1$ we have $\mathcal{P} = \mathcal{P}_\chi(1) = \{X_1\}$; at time $t = 2$ we have $\mathcal{P} = \mathcal{P}_\chi(2) = \{X_1, X_2\}$; at time $t = 3$, we have $\mathcal{P} = \mathcal{P}_\chi(3) = \{X_1, X_2\}$; (unchanged); and at the stopping time $t = 4$, we have $\mathcal{P} = \mathcal{P}_\chi(4) = \{X_1, X_2, X_3\}$. If μ is the uniform distribution on $\mathbf{X}$, the probability of this path is $\mu(\chi) = \frac{1}{n^4}$.

Given $\chi \in \mathcal{X}_m$, we let $\ell(\chi)$ be its length, i.e., the corresponding stopping time, and write $\chi = (\chi_1, \chi_2, \ldots, \chi_\ell)$ where $\ell = \ell(\chi)$.

Now we extend $S(X;\mathcal{P})$ and $C(\mathcal{P})$ to $\chi = (X_{i_1}, \ldots, X_{i_\ell}) \in \mathcal{X}_m$ by

$$S(\chi) = \sum_{j=1}^{\ell} S(X_{i_j}; \mathcal{P}_\chi(j-1)) = \sum_{j=1}^{\ell(\chi)} S(\chi_j; \mathcal{P}_\chi(j-1)) \tag{1.2}$$

and

$$C(\chi) = C(\mathcal{P}_\chi(\ell(\chi))). \tag{1.3}$$

These values are called the *total search* and *total cleanup costs* for the path χ.

We want to optimize the *average total cost function*

$$F(m) := F(m;n) = E\left[\frac{S(\chi) + C(\chi)}{\ell(\chi)}\right] = \sum_{\chi \in \mathcal{X}_m} \frac{S(\chi) + C(\chi)}{\ell(\chi)} \mu(\chi). \tag{1.4}$$

Assume $\mu(X) \neq 0$ for each $X \in \mathbf{X}$.

Problem. Given a *model* $\mathcal{M} = (\mathbf{X}, \mu, S, C)$, determine the value $m_{\text{opt}}(n) = m_{\text{opt}}(n; \mathcal{M})$ of $m \in \{1, 2, \ldots, n\}$ that minimizes $F(m; n)$.

In the event that there is more than one such minimizing m—which we do not typically expect—we may take, say, $m_{\text{opt}}(n)$ to be the smallest such m, so that $m_{\text{opt}}(n)$ is a well-defined function.

Here we will study the asymptotic behavior in the simple case of μ being a uniform distribution on $\mathbf{X}$ as $n = |\mathbf{X}| \to \infty$ for certain cost functions S and C specified below. We note the Markov process we consider arises in the coupon collector's (or birthday) problem, and more generally, sequential occupancy problems (see, e.g., [6] or [4]). The cost functions, however, make the analysis of this problem much more delicate than occupancy problems typically studied in the literature. It turns out that the expected value of the reciprocal of the waiting time in a sequential occupancy problem plays a key role in our analysis. Several results for the expected value of the waiting time itself are known (e.g., see [1]), but not, to our knowledge, for its reciprocal.

1.2 Models and Cost Functions

From now on, we assume μ is the uniform distribution on $\mathbf{X}$ unless explicitly stated otherwise. There are four reasonable, simple search models to consider, all based on doing a binary search on $\mathcal{L}$ and a sequential search on $\mathcal{P}$. Here we view $\mathcal{P}$ as an unordered set. The models depend upon whether the positions of $\mathcal{L}$ (the "shelves") are numbered or not and whether the process is memoryless or not. These models correspond to the following search algorithms $\mathbf{A}$ for an element X of $\mathcal{L} \sqcup \mathcal{P}$.

For a memoryless process, at any time t, we assume we do not know what elements are in $\mathcal{P}$, i.e., we do not remember the state of the system. Thus it is typically worthwhile to search $\mathcal{L}$ first, as searching $\mathcal{L}$ is much more efficient than searching $\mathcal{P}$. Hence for a memoryless process, we will always first search $\mathcal{L}$ for X. If this search is unsuccessful (i.e., $X \in \mathcal{P}$), then we search $\mathcal{P}$.

At the other extreme, one can consider the process where one has complete memory, i.e., at any time t, we know the state $\mathcal{P}$ of the system. Thus if $X \in \mathcal{L}$, we simply search $\mathcal{L}$, and if $X \in \mathcal{P}$, we only search $\mathcal{P}$.

The other option in the model depends on the data structure for $\mathcal{L}$. Imagine $X_1, \ldots, X_n$ are books with a total ordering. The X_i's in $\mathcal{L}$ are the books that are ordered on bookshelves, whereas the X_i's in $\mathcal{P}$ lie in an unorganized pile on a desk. If there is a marking on each shelf for the corresponding book, so each book has a well defined position on the shelf, we say the shelves are numbered. In this case, we think of $\mathcal{L}$ as a list of size n indexed by keys $k_1 < k_2 < \cdots < k_n$, where k_i points to X_i if $X_i \in \mathcal{L}$, and k_i points to null if $X_i \in \mathcal{P}$, and a search on $\mathcal{L}$, amounts to a search on n keys, regardless of how many objects X actually remain in $\mathcal{L}$. Otherwise, the shelves are unnumbered, so only the relative position of the books on the shelves is important (akin to books shelved in a library stack). Here we simply view $\mathcal{L}$ as a sorted binary tree, and a search on $\mathcal{L}$ is really a search on the $|\mathcal{L}|$ objects in $\mathcal{L}$.

While shelf positions are not typically numbered for books, this situation of "numbered shelves" commonly occurs in other situations, such as a collection of files each in their own labelled folder jacket. Namely, you may take out a file to look at, but leave the folder jacket in place so there is a placeholder for where the file goes when you put it back.

With these models in mind, the four search algorithms $\mathbf{A}$ for an object X in $\mathcal{L} \sqcup \mathcal{P}$ can be described as follows.

- $\mathcal{M}_1$ (No memory, unnumbered shelves) $\mathbf{A}$: do a binary search on the $|\mathcal{L}|$ objects in $\mathcal{L}$; if this fails, then do a sequential search on $\mathcal{P}$

- $\mathcal{M}_2$ (No memory, numbered shelves) $\mathbf{A}$: do a binary search on the n keys to find the correct position for X in $\mathcal{L}$; if it is not there, do a sequential search on $\mathcal{P}$
- $\mathcal{M}_3$ (Complete memory, unnumbered shelves) $\mathbf{A}$: if $X \in \mathcal{L}$, do a binary search on the $|\mathcal{L}|$ objects in $\mathcal{L}$; if $X \in \mathcal{P}$, do a sequential search on $\mathcal{P}$
- $\mathcal{M}_4$ (Complete memory, numbered shelves) $\mathbf{A}$: if $X \in \mathcal{L}$, do a binary search on the n keys for $\mathcal{L}$; if $X \in \mathcal{P}$, do a sequential search on $\mathcal{P}$

Each of these algorithms naturally gives rise to a search cost function $S(X;\mathcal{P})$ where $X \in \mathcal{L} \sqcup \mathcal{P}$, namely the number of comparisons needed in this algorithm. However, it is not necessary for us to write down these functions explicitly. Rather, it suffices to explicate the following average search cost functions. (In fact, one could replace the exact search cost $S(X;\mathcal{P})$ by a certain average search cost and be left with the same optimization problem—see Section 5.)

Let $s_{\mathcal{L}}(j)$ denote the average cost of a search for an object in $\mathcal{L}$ when $\mathcal{L}$ contains $n-j$ elements (we average over both the n choose $n-j$ possibilities for $\mathcal{L}$ and the $n-j$ possibilities for the object). Similarly, let $s_{\mathcal{P}}(j)$ denote the average cost of a search for an object in $\mathcal{P}$ given $\mathcal{P}$ contains j objects (again averaging over all possibilities for $\mathcal{P}$ and the object).

We define the following average search cost functions for successful binary, failed binary and sequential searches on j objects:

$$\begin{aligned} b(j) &= \left(1+\frac{1}{j}\right)\log_2(j+1) - 1 \\ b_f(j) &= \log_2(j+1) \\ s(j) &= \frac{j+1}{2}. \end{aligned} \tag{1.5}$$

The formula for $s(j)$ is of course exactly the expected number of steps required for a successful sequential search. It is easily seen that when $j+1$ is a power of 2, $b(j)$ (resp. $b_f(j)$) is the exact expected number of steps required for a successful (resp. failed) binary search on j objects. These functions are not quite the exact average number of steps for binary searches for all j (they are not generally rational), but as we are primarily interested in asymptotic behavior, we will work with the functions given above for simplicity.[(ii)] Note that $b(2^{r+1}-1) - b(2^r-1) < 2$ and is in fact close to 1 for large r. So $b(n)$ in general is a reasonable approximation of the expected cost for a successful binary search.

Then, for the above four algorithms $\mathbf{A}$, the functions $s_{\mathcal{L}}(j)$ and $s_{\mathcal{P}}(j)$ are given as follows.

- $\mathcal{M}_1$ (No memory, unnumbered shelves)

$$\begin{aligned} s_{\mathcal{L}}(j) &= b(j) \\ s_{\mathcal{P}}(j) &= b_f(n-j) + s(j) \end{aligned} \tag{1.6}$$

- $\mathcal{M}_2$ (No memory, numbered shelves)

$$\begin{aligned} s_{\mathcal{L}}(j) &= b(n) \\ s_{\mathcal{P}}(j) &= b_f(n) + s(j) \end{aligned} \tag{1.7}$$

(ii) For a discussion of how this affects the problem for smaller values of n, see Remark 4.1.

- $\mathcal{M}_3$ (Complete memory, unnumbered shelves)

$$\begin{aligned} s_{\mathcal{L}}(j) &= b(j) \\ s_{\mathcal{P}}(j) &= s(j) \end{aligned} \tag{1.8}$$

- $\mathcal{M}_4$ (Complete memory, numbered shelves)

$$\begin{aligned} s_{\mathcal{L}}(j) &= b(n) \\ s_{\mathcal{P}}(j) &= s(j) \end{aligned} \tag{1.9}$$

Remark 1.1 *If μ were a highly skewed distribution, then it might be more efficient in the no memory models to do the pile search before a list search (see Section 5).*

We now define our cleanup cost functions, based on the simple algorithm of doing a binary search for each object in $\mathcal{P}$ to find the appropriate position to insert it into $\mathcal{L}$. (Even if one remembers the general area where the object should go, there is still the time needed to identify/arrange the exact spot and the time to physically place it there, and a logarithmic cost seems like a reasonable model for this.) This leads to two different possible cleanup cost functions, corresponding to the cases of numbered and unnumbered shelves.

If the shelves are numbered, then the cleanup cost should just be the search cost to find the correct position for each object in $\mathcal{P}$, and it makes sense to set

$$C(\mathcal{P}) = \sum_{X \in \mathcal{P}} S(X; \emptyset),$$

where $S(X;\emptyset)$ denotes the search cost to find the position in $\mathcal{L}$ for X. Note that there is no dependence upon what order we replace the objects. However, we can make things a little easier on ourselves if we wish. Since we will just be considering an average of $C(\mathcal{P})$ over χ (weighted by $\frac{1}{\ell(\chi)}$), it will suffice to consider an average cleanup cost

$$C_m = \binom{n}{m}^{-1} \sum_{|\mathcal{P}|=m} C(\mathcal{P}).$$

Hence we have

$$C_m = m \cdot b(n), \qquad \mathcal{M} \in \{\mathcal{M}_2, \mathcal{M}_4\}. \tag{1.10}$$

If the shelves are unnumbered, then the cleanup cost in fact depends upon the order we replace the objects. Let us write $\mathcal{P} = \{X_{i_1}, \ldots, X_{i_m}\}$ and suppose we place them back in order $X_{i_1}, \ldots, X_{i_m}$. Write $S_{\mathcal{L}}(X)$ for the cost of a (failed if $X \notin \mathcal{L}$) binary search on $\mathcal{L}$ for the object X. Then the order-dependent cleanup cost is

$$C_{od}(X_{i_1}, \ldots X_{i_m}) = S_{\mathcal{L}}(X_{i_1}) + S_{\mathcal{L}\cup\{X_{i_1}\}}(X_{i_2}) + \cdots + S_{\mathcal{L}\cup\{X_{i_1},\ldots,X_{i_{m-1}}\}}(X_{i_m}).$$

Since $\mathcal{P}$ is unordered, we consider all cleanup orderings to occur with the same probability. Hence it suffices to consider an average over all possible orderings:

$$C(\mathcal{P}) = \frac{1}{m!} \sum C_{od}(X_{i_1}, \ldots X_{i_m}),$$

where $(X_{i_1}, \ldots, X_{i_m})$ runs through all possible orderings of $\mathcal{P}$.

As before, since we will be taking an average of our cleanup costs over χ (weighted by $\frac{1}{\ell(\chi)}$), we can consider the simpler quantities

$$C_m = \frac{1}{\binom{n}{m}} \sum_{|\mathcal{P}|=m} C(\mathcal{P}),$$

as in the numbered case. By additivity of the expected value, one sees

$$C_m = \sum_{j=1}^{m} b_f(n-j), \qquad \mathcal{M} \in \{\mathcal{M}_1, \mathcal{M}_3\}. \tag{1.11}$$

As with $S(X;\mathcal{P})$, we could replace the exact cleanup cost $C(\mathcal{P})$ with its average over all subsets of size $\mathcal{P}$ (cf. (5.2)).

Remarks

1.2 *This is not the only reasonable way to clean up. One could first sort the objects in $\mathcal{P}$, which can be done in $O(m \log m)$ time, though the way humans naturally sort is perhaps better modeled by insertion sort, which takes $O(m^2)$ time. Then one can merge $\mathcal{L}$ and $\mathcal{P}$ in $O(n)$ steps, as in a linear merge sort. This is more efficient than our above algorithm if m is relative large and one efficiently sorts $\mathcal{P}$. Since our optimization problem is one in which m should be at most logarithmic in n (cf. Proposition 2 and Remark 1.6), our cleanup algorithm above is more efficient.*

1.3 *Alternatively, one could do a binary-search-based merge sort after sorting $\mathcal{P}$ as follows. Say the ordering on $\mathbf{X}$ is $X_1 < X_2 < \cdots < X_n$. Let $X_{j_1}, \ldots, X_{j_m}$ be the elements in $\mathcal{P}$ in sorted order, i.e., $j_1 < j_2 < \cdots < j_m$. First do a binary search to insert X_{j_1} in $\mathcal{L}$. Then do a binary search on $\mathcal{L} \cap \{X_{j_1+1}, X_{j_1+2}, \ldots, X_n\}$ to find the position for X_{j_2}. Continue in this manner of binary searches on smaller and smaller subsets of $\mathcal{L}$, to replace all m objects. This may more be efficient than the cleanup algorithm we are using, depending on how we sort $\mathcal{P}$ and the relative size of m and n, and it may be interesting to study our optimization problem with this type of algorithm. However, it is only slightly more efficient when m is relatively small compared to n: suppose $m \approx \log n$ and one does an insertion sort on $\mathcal{P}$; the insertion sort alone takes $O(\log^2 n)$ time, which is the same order as our original cleanup algorithm. In light of the additional complications it brings, we do not consider this type of cleanup here.*

1.4 *One could also consider partial cleanups, where one does not put back all objects at the same time, but only some of the items in $\mathcal{P}$. We do not wish to consider such complications here. Moreover, as it typically takes time and effort for humans to switch between tasks, there seems to be extra efficiency in practice if one clean up all at once (or in a few chunks), than in many small steps.*

1.5 *This model assumes all objects are in relatively close proximity, as in your room. If one wanted to consider a similar problem for objects in large library or warehouse, one should include the cost of transit time for retrieving and putting back the objects in the functions $s_{\mathcal{L}}$ and $C(\mathcal{P})$. The transit time should be $O(\sqrt{n})$ assuming the objects are organized in a 2-dimensional grid, or at least 3-dimensional with bounded height.*

1.3 Overview

Intuitively, there are three reasons why it may be better to wait to cleanup, i.e., why $m_{\text{opt}}(n)$ might be greater than 1. Assume n is large.

(i) If one has complete memory and there are relatively few objects in the pile, the search cost for an object in the pile will be less than the search cost for a random object in the list.

(ii) If the shelves are not numbered and there are relatively few objects in the pile, one will almost surely be searching for objects which are in the sorted list, and this will go slightly faster if there are less than n objects in the list.

(iii) In all four of the above models, the average cleanup cost per search should decrease as m increases.

Thus in the case of complete memory, it is rather evident that we should have $m_{\text{opt}} > 1$. On the other hand, in the case of no memory, if one searches for an object in the pile, one first has to do a binary search on the list, which costs more than just searching for a random element in the list. So in the case of no memory, unnumbered shelves, it is not *a priori* obvious whether this factor or points (ii) and (iii) will win out. This is settled by our first result, which says one should never clean up immediately.

Proposition 1 *Suppose* $\mathcal{M} \in \{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4\}$. *For any* $n \geqslant 2$, *we have* $m_{\text{opt}}(n) > 1$.

This is not hard, and we provide two proofs: one by computing $F(1)$ and $F(2)$ explicitly in each model (see Section 2.3), and another by observing $F(m) < F(1)$ whenever $m < 4b(n-m)$ (see Lemma 2.13).

In Section 3, we restrict ourselves for simplicity to the case of complete memory and numbered shelves (model $\mathcal{M}_4$). An upper bound for m_{opt} is not too difficult, since after the pile is a certain size, each search will have an associated cost that is at least $F(1)$. Specifically, we show

Proposition 2 *Let* $\mathcal{M} = \mathcal{M}_4$. *For* $n \geqslant 1$, $m_{\text{opt}}(n) < 4b(n) \leqslant 4\log_2(n+1)$.

The problem of obtaining a good lower bound seems much more difficult, and we use some bounds shown in Section 4 to construct an approximation $\widetilde{F}(m)$ for $F(m)$ such that the (smallest if not unique) value $\widetilde{m}_{\text{opt}}(n)$ (see Section 2 for the definition) of m which minimizes $\widetilde{F}(m)$ should satisfy $\widetilde{m}_{\text{opt}}(n) \leqslant m_{\text{opt}}(n)$ (Conjecture 4.2). While we can compute $\widetilde{m}_{\text{opt}}(n)$ for fairly large n fairly quickly, the amount of time required to compute $m_{\text{opt}}(n)$ is significant, so we can only compare values of these functions for relatively small n (see Table 4.1), but it appears that $\widetilde{m}_{\text{opt}}(n) \sim m_{\text{opt}}(n)$. Given that this is the case, one would like to determine $\widetilde{m}_{\text{opt}}(n)$.

Theorem 3 (Theorem 4.3) *For any* $n \geqslant 5$, *we have*

$$3b(n) - \tfrac{3}{2} \leqslant \widetilde{m}_{\text{opt}}(n) < 3b(n) + \tfrac{1}{2}$$

i.e., for $n \geqslant 5$, $\widetilde{m}_{\text{opt}}(n)$ *equals* $\lceil 3b(n) - \frac{3}{2} \rceil$ *or* $\lceil 3b(n) - \frac{3}{2} \rceil + 1$.

This leads us to the following conjecture about our original problem.

Conjecture 4 (Conjectures 4.2 and 4.5) *Let* $\mathcal{M} = \mathcal{M}_4$. *For* $n \geqslant 5$, *we have*

$$m_{\text{opt}}(n) \geqslant 3b(n) - \tfrac{3}{2},$$

and, asymptotically,

$$m_{\text{opt}}(n) \sim 3b(n) \sim 3\log_2(n).$$

We briefly touch on the amount of cost savings in this optimization problem in Remark 4.8.

Finally, in Section 5, we make some comments about the problem for non-uniform distributions. In particular, we expect that, as one varies the underlying distribution, $m_{\mathrm{opt}}(n)$ is maximized for the uniform distribution.

Remark 1.6 *Based on the above factors, one would expect that the optimal cleanup point should be greater in the case of complete memory versus no memory, as well as in the case of unnumbered shelves versus numbered shelves. Consequently, we expect that*

$$m_{\mathrm{opt}}(n;\mathcal{M}_1) \leqslant m_{\mathrm{opt}}(n;\mathcal{M}_2) \leqslant m_{\mathrm{opt}}(n;\mathcal{M}_4) \leqslant m_{\mathrm{opt}}(n;\mathcal{M}_3).$$

We verified this numerically for small n, but we do not focus on this here. In particular, we note that preliminary numerics for $\mathcal{M}_3$ suggest $m_{\mathrm{opt}}(n) \sim 4\log_2(n)$ (Remark 4.7). (In this paper, by "numerical calculations" we mean that we used high-precision floating point calculations in PARI/GP, and not to mean that our calculations were provably correct.)

2 Expectation costs

In this paper, m and n denote integers satisfying $1 \leqslant m \leqslant n$. Further, unless otherwise specified, χ will denote a path in $\mathcal{X}_m$. If f is a function on $\mathcal{X}_m$, we sometimes denote $E[f]$ by $E_m[f]$ to specify m, or $E_{m,n}[f]$ if we want to specify both m and n.

In this section, we decompose

$$F(m) = F_{\mathcal{L}}(m) + F_{\mathcal{P}}(m) + F_C(m),$$

where the terms on the right will represent average list search, average pile search and average cleanup costs. We will analyze these terms individually. (In the case of no memory, where one does a list search then a pile search for an object $X \in \mathcal{P}$, we include both of these search costs in the function $F_{\mathcal{P}}$.) It appears that $F_{\mathcal{L}}$ and F_C are increasing in m, whereas $F_{\mathcal{P}}$ is decreasing in m (cf. Remark 2.7 and Lemma 2.8). We also expect that F is unimodal—initially decreasing, then increasing. Thus our optimization problem is about the question of when $F_{\mathcal{P}}$ begins increasing faster than $F_{\mathcal{L}} + F_C$ decreases.

2.1 Expected search cost

In this section, we want to find a way to calculate $E\left[\frac{S}{\ell}\right]$. We can reduce this to studying averages of the form

$$\sum_{\chi\in\mathcal{X}_m^{(\ell)}} S(\chi) = \sum_{\chi\in\mathcal{X}_m^{(\ell)}} \sum_{j=1}^{\ell} S(\chi_j;\mathcal{P}_\chi(j-1)), \tag{2.1}$$

where

$$\mathcal{X}_m^{(\ell)} = \{\chi \in \mathcal{X}_m : \ell(\chi) = \ell\}. \tag{2.2}$$

Namely, note the probability that $\chi \in \mathcal{X}_m^{(\ell)}$ depends only on ℓ, and is

$$\mu(\mathcal{X}_m^{(\ell)}) = \frac{|\mathcal{X}_m^{(\ell)}|}{n^\ell}. \tag{2.3}$$

Hence

$$F_S(m) := E\left[\frac{S}{\ell}\right] = \sum_{\chi\in\mathcal{X}_m} \frac{S(\chi)}{\ell(\chi)}\mu(\chi) = \sum_{\ell=m}^{\infty} \frac{1}{\ell n^\ell} \sum_{\chi\in\mathcal{X}_m^{(\ell)}} S(\chi) \tag{2.4}$$

Proposition 2.1 *We have*

$$|\mathcal{X}_m^{(\ell)}| = m!\binom{n}{m}\left\{\begin{matrix}\ell-1\\ m-1\end{matrix}\right\}.$$

Here $\left\{\begin{matrix}\ell\\ m\end{matrix}\right\}$ denotes the Stirling number of the second kind, i.e., the number of ways to partition a set of ℓ elements into m nonempty subsets.

Proof: Note $\mathcal{X}_m^{(\ell)}$ is in bijection with the set of pairs (α, X) where α is a sequence of length $\ell-1$ in $\mathbf{X}$ consisting of $m-1$ distinct elements, and X is an element of $\mathbf{X}$ not occurring in α. The number of such α is simply the number of surjective maps from $\{1,\dots,\ell-1\}$ to $\{1,\dots,m-1\}$, which is $(m-1)!\left\{\begin{matrix}\ell-1\\ m-1\end{matrix}\right\}$ by the twelvefold way, times the number of possibilities for the set of $m-1$ distinct elements in α, which is $\binom{n}{m-1}$. Observing that for each such α there are $n-(m-1)$ distinct choices for X gives the stated result. □

As an aside, we note this provides a proof of the identity (cf. [4, Thm 2.11])

$$\sum_{\ell=m}^{\infty} \left\{\begin{matrix}\ell-1\\ m-1\end{matrix}\right\}\frac{1}{n^\ell} = \frac{(n-m)!}{n!} \tag{2.5}$$

since $\sum |\mathcal{X}_m^{(\ell)}|/n^\ell = 1$.

Write

$$S(\chi) = S_{\mathcal{L}}(\chi) + S_{\mathcal{P}}(\chi)$$

where

$$S_{\mathcal{L}}(\chi) = \sum_{\chi_j\notin\mathcal{P}_\chi(j-1)} S(\chi_j; \mathcal{P}_\chi(j-1))$$

and

$$S_{\mathcal{P}}(\chi) = \sum_{\chi_j\in\mathcal{P}_\chi(j-1)} S(\chi_j; \mathcal{P}_\chi(j-1)).$$

In other words, $S_{\mathcal{L}}(\chi)$ (resp. $S_{\mathcal{P}}(\chi)$) is the total cost of searches along χ when the sought-after object is in $\mathcal{L}$ (resp. $\mathcal{P}$).

The action of the symmetric group $Sym(\mathbf{X})$ on $\mathbf{X}$ induces an action on $\mathcal{X}_m^{(\ell)}$. Namely, for $\sigma \in Sym(\mathbf{X})$, put

$$\chi^\sigma = (\chi_1^\sigma, \dots, \chi_\ell^\sigma).$$

Also, for $\chi\in\mathcal{X}_m$, we put $\tau_\chi(j)$ to be the number of times one searches along χ for an object in $\mathcal{P}$ when $\mathcal{P}$ has size j. Explicitly, set $t_0 = t_0(\chi) = 0$ and, for $1 \leqslant j \leqslant m$, let $t_j = t_j(\chi)$ be the minimal integer such that $|\mathcal{P}_\chi(t_j)| = j$. Then for $0 \leqslant j < m$, we set $\tau_j(\chi) = t_{j+1}(\chi) - t_j(\chi) - 1$.

Lemma 2.2 *For any $\chi \in \mathcal{X}_m^{(\ell)}$, we have the following average cost formulas:*

$$\frac{1}{n!}\sum_{\sigma} S_{\mathcal{L}}(\chi^{\sigma}) = \sum_{j=0}^{m-1} s_{\mathcal{L}}(n-j)$$

and

$$\frac{1}{n!}\sum_{\sigma} S_{\mathcal{P}}(\chi^{\sigma}) = \sum_{j=0}^{m-1} \tau_j(\chi) s_{\mathcal{P}}(j).$$

Proof: To see the first equality, observe that for any χ, there must be exactly one search for an object X_{i_j} which is in $\mathcal{L}$ when $\mathcal{L}$ has $n-j$ objects for each $j = 0, 1, \ldots, m-1$. Fixing one such j and averaging the contribution of this search cost over the permutations σ yields $s_{\mathcal{L}}(n-j)$.

The second equality is similar. □

This yields the following expected cost formulas:

$$E[S_{\mathcal{L}}(\chi) \,|\, \chi \in \mathcal{X}_m^{(\ell)}] = |\mathcal{X}_m^{(\ell)}|^{-1} \sum_{\chi \in \mathcal{X}_m^{(\ell)}} S_{\mathcal{L}}(\chi) = \sum_{j=0}^{m-1} s_{\mathcal{L}}(n-j)$$

and

$$E[S_{\mathcal{P}}(\chi) \,|\, \chi \in \mathcal{X}_m^{(\ell)}] = |\mathcal{X}_m^{(\ell)}|^{-1} \sum_{\chi \in \mathcal{X}_m^{(\ell)}} S_{\mathcal{P}}(\chi) = \sum_{j=0}^{m-1} E[\tau_j \,|\, \mathcal{X}_m^{(\ell)}] s_{\mathcal{P}}(j).$$

Consequently, one has

Lemma 2.3

$$F_S(m) = E\left[\frac{S}{\ell}\right] = F_{\mathcal{L}}(m) + F_{\mathcal{P}}(m)$$

where

$$F_{\mathcal{L}}(m) = \sum_{\ell=m}^{\infty} \frac{|\mathcal{X}_m^{(\ell)}|}{\ell n^{\ell}} \sum_{j=0}^{m-1} s_{\mathcal{L}}(n-j) = E_m\left[\frac{1}{\ell}\right] \sum_{j=0}^{m-1} s_{\mathcal{L}}(n-j)$$

and

$$F_{\mathcal{P}}(m) = \sum_{\ell=m}^{\infty} \frac{|\mathcal{X}_m^{(\ell)}|}{\ell n^{\ell}} \sum_{j=1}^{m-1} E_m[\tau_j \,|\, \mathcal{X}_m^{(\ell)}] s_{\mathcal{P}}(j).$$

2.2 Expected cleanup cost

The expected cleanup cost per item is simply

$$F_C(m) := E_m\left[\frac{C}{\ell}\right] = \sum_{\chi \in \mathcal{X}_m} \frac{C(\chi)}{\ell(\chi)}\mu(\chi) = C_m \sum_{\chi \in \mathcal{X}_m} \frac{\mu(\chi)}{\ell(\chi)} = C_m E_m\left[\frac{1}{\ell}\right]$$

where C_m is as in (1.10) or (1.11) according to whether the shelves are numbered or not.

With this notation, the expected search-and-cleanup cost is

$$F(m) = F_{\mathcal{L}}(m) + F_{\mathcal{P}}(m) + F_C(m).$$

Note that in the case of numbered shelves, we have

$$F_C(m) = F_{\mathcal{L}}(m).$$

In the case of unnumbered shelves,

$$F_C(m) = E_m\left[\frac{1}{\ell}\right]\sum_{j=0}^{m-1} b_f(n-j-1)$$

and

$$F_{\mathcal{L}}(m) = E_m\left[\frac{1}{\ell}\right]\sum_{j=0}^{m-1} b(n-j).$$

Consequently, we have

$$F_C(m) = \left(1 - \frac{m - \sum_{j=0}^{m-1}\log_2(n-j+1)/(n-j)}{\sum_{j=0}^{m-1}\log_2(n-j+1)}\right) F_{\mathcal{L}}(m).$$

We remark that this implies $F_C(m) \leqslant F_{\mathcal{L}}(m)$.

In any case, we have reduced our problem to studying the expected list search cost $F_{\mathcal{L}}(m)$ and $F_{\mathcal{P}}(m)$.

2.3 Some simple calculations

Here, we calculate $F(1;n)$ and $F(2;n)$ for each of the four models discussed above, which we hope will be instructive. In all cases, these calculations, together with the observation (2.6) that $E_2\left[\frac{1}{\ell}\right] < \frac{1}{2}$, imply that $F(2;n) < F(1;n)$ for all $n \geqslant 2$, giving one proof of Proposition 1. We remark the proof of this inequality does not depend on the specific definitions of the functions $b(j)$ and $b_f(j)$, just that these functions are increasing. In fact, it does not depend upon the definition of $s(j)$ either, as long as $s(1) \geqslant 0$.

Calculations for $m = 1$

First consider $m = 1$. Then $\mathcal{X}_1 = \mathcal{X}_1^{(1)} = \{(1), (2), (3), \ldots, (n)\}$. Consequently $E[\frac{1}{\ell}] = 1$ and $F_{\mathcal{P}}(1) = 0$.

2.3.1 Unnumbered shelves

Suppose we have unnumbered shelves, i.e., $\mathcal{M}_1$ or $\mathcal{M}_3$. Then

$$F(1;n) = F_{\mathcal{L}}(1;n) + F_C(1;n) = b(n) + b_f(n-1).$$

2.3.2 Numbered shelves

Suppose we are in the case of numbered shelves, i.e., $\mathcal{M}_2$ or $\mathcal{M}_4$. Then

$$F(1;n) = 2F_{\mathcal{L}}(1;n) = 2s_{\mathcal{L}}(n) = 2b(n).$$

Calculations for $m = 2$

Now take $m = 2$. Then, for $\ell \geqslant 2$, $\mathcal{X}_2^{(\ell)}$ consists of the $2\binom{n}{2}$ sequences of the form $(X_1, X_1, \ldots, X_1, X_2)$, where there are $\ell - 1$ occurrences of X_1. Then

$$E\left[\frac{1}{\ell}\right] = n(n-1)\sum_{\ell=2}^{\infty}\frac{1}{\ell n^{\ell}} < n(n-1)\frac{1}{2}\sum_{\ell=2}^{\infty}\frac{1}{n^{\ell}} = \frac{1}{2}. \tag{2.6}$$

As an aside, we note that the equality in (2.6) yields the closed form expression

$$E_2\left[\frac{1}{\ell}\right] = n(n-1)\left(\log\frac{1}{1-1/n} - \frac{1}{n}\right). \tag{2.7}$$

Since

$$E[\tau_{\chi}(1)\,|\,\chi \in \mathcal{X}_2^{(\ell)}] = \ell - 2,$$

we have

$$F_{\mathcal{P}}(2;n) = s_{\mathcal{P}}(1)\sum_{\ell=m}^{\infty}\frac{|\mathcal{X}_2^{(\ell)}|}{\ell n^{\ell}}(\ell-2) = s_{\mathcal{P}}(1)\left(1 - 2E\left[\frac{1}{\ell}\right]\right).$$

2.3.3 No memory, unnumbered shelves

Suppose $\mathcal{M} = \mathcal{M}_1$, so

$$F(2;n) = F_{\mathcal{L}}(2;n) + F_{\mathcal{P}}(2;n) + F_C(2;n).$$

Here $F_{\mathcal{L}}(2;n) = E\left[\frac{1}{\ell}\right](b(n) + b(n-1))$, $F_C(2;n) = E\left[\frac{1}{\ell}\right](b_f(n-1) + b_f(n-2))$, and $s_{\mathcal{P}}(1) = b_f(n-1) + s(1)$, so

$$F(2;n) = b_f(n-1) + s(1) + E\left[\frac{1}{\ell}\right](b(n) + b(n-1) - b_f(n-1) + b_f(n-2) - 2s(1)).$$

2.3.4 No memory, numbered shelves

Suppose $\mathcal{M} = \mathcal{M}_2$, so

$$F(2;n) = 2F_{\mathcal{L}}(2;n) + F_{\mathcal{P}}(2;n).$$

Here $F_{\mathcal{L}}(2;n) = 2E\left[\frac{1}{\ell}\right]b(n)$ and $s_{\mathcal{P}}(1) = b_f(n) + s(1)$, so

$$F(2;n) = b_f(n) + s(1) + E\left[\frac{1}{\ell}\right](4b(n) - 2b_f(n) - 2s(1)).$$

2.3.5 Complete memory, unnumbered shelves

Suppose $\mathcal{M} = \mathcal{M}_3$, so

$$F(2;n) = F_{\mathcal{L}}(2;n) + F_{\mathcal{P}}(2;n) + F_C(2;n).$$

Here $F_{\mathcal{L}}(2;n) = E\left[\frac{1}{\ell}\right](b(n)+b(n-1))$, $F_C(2;n) = E\left[\frac{1}{\ell}\right](b_f(n-1)+b_f(n-2))$, and $s_{\mathcal{P}}(1) = s(1)$, so

$$F(2;n) = s(1) + E\left[\frac{1}{\ell}\right](b(n) + b(n-1) + b_f(n-1) + b_f(n-2) - 2s(1)).$$

2.3.6 Complete memory, numbered shelves

Suppose $\mathcal{M} = \mathcal{M}_4$, so

$$F(2;n) = 2F_{\mathcal{L}}(2;n) + F_{\mathcal{P}}(2;n).$$

Here $F_{\mathcal{L}}(2;n) = 2E\left[\frac{1}{\ell}\right] b(n)$ and $s_{\mathcal{P}}(1) = s(1)$, so

$$F(2;n) = s(1) + E\left[\frac{1}{\ell}\right](4b(n) - 2s(1)).$$

2.4 Expected list search cost

We now return to studying search costs, in particular we consider the expected list search and cleanup costs, $F_{\mathcal{L}}(m)$ and $F_C(m)$. Since

$$F_{\mathcal{L}}(m) = E_m\left[\frac{1}{\ell}\right]\sum_{j=0}^{m-1} s_{\mathcal{L}}(n-j) \quad \text{and} \quad F_C(m) = C_m E_m\left[\frac{1}{\ell}\right],$$

studying these quantities reduces to studying

$$E_m\left[\frac{1}{\ell}\right] = E_{m,n}\left[\frac{1}{\ell}\right] = \sum_{\ell=m}^{\infty} \frac{|\mathcal{X}_m^{(\ell)}|}{\ell n^\ell} = m!\binom{n}{m}\sum_{\ell=m}^{\infty}\left\{ {\ell-1 \atop m-1} \right\}\frac{1}{\ell n^\ell}. \tag{2.8}$$

Note this is the expected value of the reciprocal of a waiting time for a sequential occupancy problem.

First we obtain the following finite formula, which allows us to compute $E_m\left[\frac{1}{\ell}\right]$ quickly.

Lemma 2.4 *We have*

$$E_m\left[\frac{1}{\ell}\right] = \binom{n-1}{m-1}(-1)^{m+1} + m\binom{n}{m}\sum_{j=1}^{m-1}(-1)^{m-j}\binom{m-1}{j}\frac{\log(1-j/n)}{j}. \tag{2.9}$$

Note the first term can be interpreted as the $j=0$ term for the sum on the right.

Proof: The generating function for Stirling numbers of the second kind is given by

$$\sum_{\ell=m}^{\infty}\left\{ {\ell-1 \atop m-1} \right\} x^{\ell-1} = \frac{x^{m-1}}{(1-x)(1-2x)\cdots(1-(m-1)x)}. \tag{2.10}$$

Hence

$$E_m\left[\frac{1}{\ell}\right] = \frac{n!}{(n-m)!}\int_0^{1/n} \frac{x^{m-1}}{(1-x)(1-2x)\cdots(1-(m-1)x)}\,dx. \tag{2.11}$$

We compute the integral using partial fractions.

$$\frac{x^{m-1}}{(1-x)(1-2x)\cdots(1-(m-1)x)} = \frac{A_1 x}{1-x} + \frac{A_2 x}{1-2x} + \cdots + \frac{A_{m-1}x}{1-(m-1)x},$$

where $A_j = \dfrac{(-1)^{m-1-j}}{(j-1)!\,(m-1-j)!}$. Now $\displaystyle\int \frac{A_j x}{1-jx}\,dx = -\frac{A_j x}{j} - \frac{A_j\,\log(1-jx)}{j^2}$, and so we have

$$E_m\left[\frac{1}{\ell}\right] = m!\binom{n}{m}\sum_{j=1}^{m-1} -\frac{A_j}{j}\cdot\frac{1}{n} - \frac{A_j}{j^2}\cdot\log(1-j/n) \tag{2.12}$$

We can simplify this sum a little by observing that $m!\,\dfrac{A_j}{j} = m\cdot(-1)^{m-1-j}\dbinom{m-1}{j}$. Then the first part of the above summation simplifies to

$$\frac{m}{n}\binom{n}{m}(-1)^m\sum_{j=1}^{m-1}(-1)^j\binom{m-1}{j} = \frac{m}{n}\binom{n}{m}(-1)^m\cdot(-1)$$

Putting all this together yields the lemma. □

Since the above formula is an alternating sum, it not so useful in studying the behavior of $E_m\left[\frac{1}{\ell}\right]$ as m varies, which is our goal, though it is useful for numerics.

Now we observe some elementary bounds.

Lemma 2.5 *For* $1 \leqslant m \leqslant n$*,*

$$\frac{1}{n(\log(n)-\log(n-m))} \leqslant \frac{1}{E_m[\ell]} \leqslant E_m\left[\frac{1}{\ell}\right] \leqslant \frac{1}{m}.$$

(We interpret the leftmost term as 0 when $m = n$.)

Proof: As is well known, ℓ is a sum of m independent geometric distributions with means $\frac{n}{n}$, $\frac{n}{n-1}$, …, $\frac{n}{n-m+1}$. Thus

$$E_m[\ell] = \sum_{j=0}^{m-1}\frac{n}{n-j} = n(H_n - H_{n-m}), \tag{2.13}$$

where H_j is the j-th harmonic number. This implies the first inequality. The second is Jensen's inequality. The third follows as $\ell(\chi) \geqslant m$ for any $\chi \in \mathcal{X}_m$. □

If $n \to \infty$ and m grows slower than $\sqrt{n}$, $\ell \to E_m[\ell]$ in probability [1]. Thus we might expect $\frac{1}{E_m[\ell]}$ to be a good approximation for $E_m[\frac{1}{\ell}]$, as the following bound shows.

Lemma 2.6 *For* $1 < m < \sqrt{2n}$,

$$E_m\left[\frac{1}{\ell}\right] < \frac{1}{E_{m-1}[\ell]}.$$

For $1 < m \leqslant n$,

$$E_m\left[\frac{1}{\ell}\right] \leqslant \frac{1}{E_{m-1}[\ell]} + \frac{(m-1)(n-m)}{2n^2}.$$

Proof: Let $x > 0$. Chebyshev's inequality tells us

$$Pr(|\ell - E_m[\ell]| \geqslant x) \leqslant \frac{\mathit{Var}(\ell)}{x^2}.$$

Consequently,

$$E_m\left[\frac{1}{\ell}\right] \leqslant \frac{1}{E_m[\ell]-x}(1-y) + \frac{y}{m} = \frac{1}{E_m[\ell]-x} + y\left(\frac{1}{m} - \frac{1}{E_m[\ell]-x}\right) \tag{2.14}$$

for any $y \geqslant \frac{\mathit{Var}(\ell)}{x^2}$. Note

$$\mathit{Var}(\ell) = \sum_{j=0}^{m-1} \frac{jn}{(n-j)^2} \leqslant \frac{n}{(n-m+1)^2}\frac{(m-1)m}{2}.$$

Set $x = \frac{n}{n-m+1}$ so $E_m[\ell] - x = E_{m-1}[\ell]$. Now we apply (2.14) with

$$y = \frac{(n-m+1)^2}{n^2}\frac{n}{(n-m+1)^2}\frac{(m-1)m}{2} = \frac{(m-1)m}{2n}.$$

Observe

$$\frac{1}{m} - \frac{1}{E_{m-1}[\ell]} = \frac{1}{mE_{m-1}[\ell]}\left(\sum_{j=1}^{m-2}\frac{j}{n-j} - 1\right) \leqslant \frac{1}{mE_{m-1}[\ell]}\left(\frac{(m-2)(m-1)}{2(n-m+2)} - 1\right),$$

which is negative if $m < \sqrt{2n}$, and one gets the first part. The second part follows from the crude bound

$$\frac{1}{mE_{m-1}[\ell]}\left(\sum_{j=1}^{m-2}\frac{j}{n-j} - 1\right) \leqslant \frac{1}{m}\frac{n-m}{n}.$$

Remark 2.7 *(i) The preceding two lemmas imply that* $E_m\left[\frac{1}{\ell}\right]$ *is decreasing in* m *for* $1 \leqslant m < \sqrt{n}$. ◻ *However, note that it follows easily that* $E_m\left[\frac{1}{\ell}\right]$ *is decreasing in* m *for all* $m \geqslant 1$ *because of the inequality of probabilities,* $\mu_m(\ell \leqslant X) \leqslant \mu_{m-1}(\ell \leqslant X)$ *for all* X.

(ii) In fact we expect the first part of Lemma 2.6 to hold for all $1 < m \leqslant n$*, as well as the stronger bound* $E_m\left[\frac{1}{\ell}\right] \leqslant \frac{m-1}{m}\frac{1}{E_{m-1}[\ell]}$*, which appears true numerically. This stronger bound together with Lemma 2.5 would imply*

$$mE_m\left[\frac{1}{\ell}\right] > (m+1)E_{m+1}\left[\frac{1}{\ell}\right], \tag{2.15}$$

which means that for any of our four models $\mathcal{M}$, the expected list search and cleanup costs, $F_{\mathcal{L}}(m)$ and $F_C(m)$, are strictly decreasing in m. Furthermore, numerics suggest that the sequence $\frac{1}{E_m[\ell]}$ decreases in m faster than the sequence $E_m[\frac{1}{\ell}]$. For instance, it appears

$$mE_m\left[\frac{1}{\ell}\right]-(m+1)E_{m+1}\left[\frac{1}{\ell}\right]\leqslant\frac{m}{E_m[\ell]}-\frac{m+1}{E_{m+1}[\ell]};\quad \textit{and}\quad \frac{E_{m+1}[\ell]}{E_m[\ell]}\geqslant\frac{E_m[1/\ell]}{E_{m+1}[1/\ell]}.$$

Lemma 2.8 *Fix $m \geqslant 1$, and let $\epsilon > 0$. Then for sufficiently large n,*

$$F_{\mathcal{L}}(m;n)>F_{\mathcal{L}}(m+1;n)-\epsilon\quad \textit{and}\quad F_C(m;n)>F_C(m+1;n)-\epsilon.$$

In the case of unnumbered shelves, we may take $\epsilon = 0$, i.e., for m_0 sufficiently small relative to n, $F_{\mathcal{L}}(m;n)$ and $F_C(m;n)$ are decreasing in m for $1 \leqslant m < m_0$.

Proof: It is easy to see that (e.g., by Lemma 2.5 or [1]), for fixed m, $\frac{1}{E_{m,n}[\ell]}$ and $E_{m,n}\left[\frac{1}{\ell}\right]$ are increasing sequences with limit $\frac{1}{m}$. This implies that for n sufficiently large

$$E_{m,n}\left[\frac{1}{\ell}\right]>\frac{m+1}{m}E_{m+1,n}\left[\frac{1}{\ell}\right]-\epsilon.$$

This implies the lemma as

$$\frac{m+1}{m}\sum_{j=0}^{m-1}s_{\mathcal{L}}(n-j)-\sum_{j=0}^{m}s_{\mathcal{L}}(n)\geqslant 0\quad\text{and}\quad\frac{m+1}{m}C_m-C_{m+1}\geqslant 0,$$

and these differences can be bounded away from 0 in the unnumbered case. □

2.5 Expected pile search cost

Now we consider the expected pile search cost

$$F_{\mathcal{P}}(m)=\sum_{j=1}^{m-1}E_m\left[\frac{\tau_j}{\ell}\right]s_{\mathcal{P}}(j)=\sum_{\ell=m}^{\infty}\frac{|\mathcal{X}_m^{(\ell)}|}{\ell n^\ell}\sum_{j=1}^{m-1}E_m[\tau_j(\chi)\,|\,\chi\in\mathcal{X}_m^{(\ell)}]s_{\mathcal{P}}(j).$$

First we remark the following explicit formula for the expected values in the inner sum.

Lemma 2.9 *For $1 \leqslant j \leqslant m-1$, we have*

$$E_m[\tau_j\,|\,\mathcal{X}_m^{(\ell)}]=\left\{{\ell-1 \atop m-1}\right\}^{-1}\times\sum_{k=1}^{\ell-m}j^k\left\{{\ell-k-1 \atop m-1}\right\}.$$

Proof: If $\ell = m$, then $\tau_j(\chi) = 0$ for all $\chi \in \mathcal{X}_m^{(\ell)}$ and the formula trivially holds, so suppose $\ell > m$ and let $\chi \in \mathcal{X}_m^{(\ell)}$. If $r = \tau_j(\chi) \geqslant 1$, we can remove the element at position $t_{j+1} - 1$ to get an element $\chi' \in \mathcal{X}_m^{(\ell-1)}$ such that

$$\tau_j(\chi')=r-1,\ \tau_{j'}(\chi')=\tau_{j'}(\chi)\quad j'\neq j.$$

This map from χ to χ' is a j-to-1 surjective map, i.e., for $j, r \geqslant 1$ we have

$$\#\{\chi \in \mathcal{X}_m^{(\ell)} : \tau_j(\chi) = r\} = j \cdot \#\{\chi \in \mathcal{X}_m^{(\ell-1)} : \tau_j(\chi) = r-1\}.$$

Summing over all $r \geqslant 1$, we see

$$\#\{\chi \in \mathcal{X}_m^{(\ell)} : \tau_j(\chi) \geqslant 1\} = j|\mathcal{X}_m^{(\ell-1)}|.$$

Similarly if $r \geqslant k$ we can remove the last k elements before position t_{j+1} to get a j^k-to-1 map into $\mathcal{X}_m^{(\ell-k)}$, and we have

$$\#\{\chi \in \mathcal{X}_m^{(\ell)} : \tau_j(\chi) \geqslant k\} = j^k|\mathcal{X}_m^{(\ell-k)}|. \tag{2.16}$$

Now observe that

$$E_m[\tau_j \mid \mathcal{X}_m^{(\ell)}] = \sum_{k=1}^{\ell-m} k\mu\{\chi \in \mathcal{X}_m^{(\ell)} : \tau_j(\chi) = k\} = \sum_{k=1}^{\ell-m} \mu\{\chi \in \mathcal{X}_m^{(\ell)} : \tau_j(\chi) \geqslant k\}$$

and apply Proposition 2.1. □

Consequently, we have

$$\begin{aligned} E_m\left[\frac{\tau_j}{\ell}\right] &= m!\binom{n}{m}\sum_{\ell=m}^{\infty}\frac{1}{\ell n^\ell}\sum_{k=1}^{\infty} j^k \left\{ \begin{matrix} \ell-k-1 \\ m-1 \end{matrix} \right\} \\ &= m!\binom{n}{m}\sum_{k=1}^{\infty} j^k \sum_{\ell=m+k}^{\infty}\frac{1}{\ell n^\ell}\left\{ \begin{matrix} \ell-k-1 \\ m-1 \end{matrix} \right\} \\ &= m!\binom{n}{m}\sum_{k=1}^{\infty}\frac{j^k}{n^k}\sum_{\ell=m}^{\infty}\frac{1}{(\ell+k)n^\ell}\left\{ \begin{matrix} \ell-1 \\ m-1 \end{matrix} \right\} \end{aligned} \tag{2.17}$$

This expression allows us to get the following upper bound.

Proposition 2.10 *For* $1 \leqslant j \leqslant m-1$, *the covariance* $\mathrm{Cov}(\tau_j, \frac{1}{\ell}) < 0$, *i.e.,*

$$E_m\left[\frac{\tau_j}{\ell}\right] < \frac{j}{n-j}E_m\left[\frac{1}{\ell}\right].$$

Proof: From (2.17) we have

$$E_m\left[\frac{\tau_j}{\ell}\right] < \sum_{k=1}^{\infty}\frac{j^k}{n^k}E_m\left[\frac{1}{\ell}\right] = \frac{j}{n-j}E_m\left[\frac{1}{\ell}\right].$$

That this is equivalent to the condition of negative covariance asserted above follows as

$$\mu\{\chi \in \mathcal{X}_m : \tau_j(\chi) = k\} = \left(\frac{j}{n}\right)^k \frac{n-j}{n}$$

implies

$$E_m[\tau_j] = \frac{n-j}{n}\sum_{k=1}^{\infty} k\left(\frac{j}{n}\right)^k = \frac{j}{n}\frac{n-j}{n}\left(1-\frac{j}{n}\right)^{-2} = \frac{j}{n-j}.$$

□

Remark 2.11 *Suppose $n \geqslant 4$ and $1 \leqslant j < m < n$. Then numerically it appears that*

$$E_m\left[\frac{\tau_j}{\ell}\right] \leqslant E_{m+1}\left[\frac{\tau_{j+1}}{\ell}\right].$$

This would imply that $F_{\mathcal{P}}(m)$ is increasing in m, and, in the case of complete memory,

$$F_{\mathcal{P}}(m+1) > \frac{m+1}{m}F_{\mathcal{P}}(m).$$

Lastly we note

Lemma 2.12 *We have*

$$mE_m\left[\frac{1}{\ell}\right] + \sum_{j-1}^{m-1} E_m\left[\frac{\tau_j}{\ell}\right] = 1.$$

Proof: This follows from the observation that $\sum_{j-1}^{m-1}\tau_j(\chi) = \ell(\chi) - m$. □

2.6 Reinterpreting $F(m)$

From above, we can rewrite

$$F(m) = mE_m\left[\frac{1}{\ell}\right]s^*(m) + \sum_{j=1}^{m-1} E_m\left[\frac{\tau_j}{\ell}\right]s_{\mathcal{P}}(j), \tag{2.18}$$

where

$$s^*(m) = \frac{\sum_{j=0}^{m-1} s_{\mathcal{L}}(n-j) + C_m}{m}$$

denotes the average total (search plus cleanup) cost of taking an object out of $\mathcal{L}$. This expression yields the following interpretation (cf. Lemma 2.12): we can think of $mE_m\left[\frac{1}{\ell}\right]$ as the probability that a given search will cost $s^*(m)$, and $E_m\left[\frac{\tau_j}{\ell}\right]$ as the probability that a given search will cost $s_{\mathcal{P}}(j)$.

It is easy to see that $mE_m\left[\frac{1}{\ell}\right] = 1$ if and only if $m = 1$, so we have $F(1) > F(m)$ whenever $m > 1$ satisfies $s_{\mathcal{P}}(m-1) < s^*(m)$. This yields

Lemma 2.13 *Let $1 < m \leqslant n$.*

1. *In the case of unnumbered shelves, if $m < 4b(n-m)$, then $F(m) < F(1)$.*
2. *In the case of numbered shelves, if $m < 4b(n)$, then $F(m) < F(1)$.*

3 An upper bound

For simplicity now, we will assume we are in model $\mathcal{M}_4$ (complete memory, numbered shelves), though a similar argument can be used for $\mathcal{M}_2$ as well. In this case we have

$$F(m) = 2F_{\mathcal{L}}(m) + F_{\mathcal{P}}(m).$$

Recall that $F(1) = 2b(n)$. Thus, once the pile $\mathcal{P}$ has more than $4b(n)$ elements, a single average pile search must cost more than $F(m_{\mathrm{opt}}(n))$. This idea gives the following upper bound.

Proposition 3.1 *Suppose* $n \geqslant 1$. *Then* $m_{\mathrm{opt}}(n) < 4b(n)$.

We will first prove a lemma. We say a function $f : \mathcal{X}_m \to \mathbb{R}$ is *additive* if, for any $\chi = (\chi_1, \ldots, \chi_\ell)$ and any $1 \leqslant k < \ell$, we can write f as a sum of terms

$$f(\chi) = \sum_{j=1}^{\ell} f(\chi_j; \mathcal{P}_\chi(j-1))$$

where the $f(\chi_j; \mathcal{P}_\chi(j-1))$ depends only upon χ_j and what is in the pile before time j. We can naturally restrict such functions f to functions of $\mathcal{X}_k$ for $k < m$. Note that all the cost functions we considered above are additive, and any linear combination of additive functions is additive.

Let $m \geqslant k \geqslant 1$. Define a *restriction map* $R_k^m : \mathcal{X}_m \to \mathcal{X}_k$, given by

$$R_k^m \chi = (\chi_1, \ldots, \chi_{t_k(\chi)}).$$

We let $T_k^m : \mathcal{X}_m \to \bigcup_{k=1}^{\infty} \mathbf{X}^k$ be the truncated tail from the restriction map, i.e.,

$$T_k^m \chi = (\chi_{t_k(\chi)+1}, \ldots, \chi_{\ell(\chi)}).$$

Put $\mathcal{P}_{\chi,k} = \mathcal{P}_\chi(t_k(\chi))$ to be the pile after time $t_k(\chi)$.

Lemma 3.2 *Suppose* $f : \mathcal{X}_m \to \mathbb{R}$ *is additive and* $m > k \geqslant 1$. *If*

$$f(X_i; \mathcal{P}_{\chi,k}) \geqslant \frac{f(R_k^m \chi)}{\ell(R_k^m \chi)} \tag{3.1}$$

for all $\chi \in \mathcal{X}_m$ *and* $X_i \in \mathbf{X}$, *then*

$$E_m\left[\frac{f}{\ell}\right] \geqslant E_k\left[\frac{f}{\ell}\right]. \tag{3.2}$$

If, further, the inequality in (3.1) *is strict for some* χ *and* X_i, *then the inequality in* (3.2) *is also strict.*

Proof: Since f and ℓ are additive, we can write

$$E_m\left[\frac{f}{\ell}\right] = \sum_{\chi \in \mathcal{X}_m} \mu(\chi) \frac{f(R_k^m \chi) + f(T_k^m \chi; \mathcal{P}_{\chi,k})}{\ell(R_k^m \chi) + \ell(T_k^m \chi)}.$$

Then the above condition guarantees, for any χ,

$$\frac{f(R_k^m\chi)+f(T_k^m\chi;\mathcal{P}_{\chi,m-1})}{\ell(R_k^m\chi)+\ell(T_k^m\chi)} \geqslant \frac{f(R_k^m\chi)}{\ell(R_k^m\chi)}$$

as $\frac{a+b}{c+d} \geqslant \frac{a}{c}$ if and only if $\frac{b}{d} \geqslant \frac{a}{c}$. □

Proof Proof of Proposition 3.1: Set $k = \lfloor 4b(n) \rfloor$ and let $k < m \leqslant n$. Let $\bar{S}_{\mathcal{L}}(\chi)$ (resp. $\bar{S}_{\mathcal{P}}(\chi)$) be the average of $S_{\mathcal{L}}(\chi^\sigma)$ (resp. $S_{\mathcal{P}}(\chi^\sigma)$), where σ ranges over $Sym(\mathbf{X})$. Let $f = 2\bar{S}_{\mathcal{L}} + \bar{S}_{\mathcal{P}}$, so $F(m) = E_m[\frac{f}{\ell}]$.

Then note that

$$\frac{f(R_k^m\chi)}{\ell(R_k^m\chi)} = \frac{2kb(n)+\bar{S}_{\mathcal{P}}(R_k^m\chi)}{\ell(R_k^m\chi)} \leqslant 2b(n),$$

since

$$\bar{S}_{\mathcal{P}}(R_k^m\chi) \leqslant (\ell(R_k^m\chi)-k)s_{\mathcal{P}}(k-1) \leqslant 2b(n)(\ell(R_k^m\chi)-k)$$

Furthermore, this inequality must be strict for some χ (in fact, one only gets equality when $n+1$ is a power of 2 and $t_j(\chi) = j$ for $j < k$).

On the other hand, for any $X_i \in \mathbf{X}$, we have $f(X_i;\mathcal{P}_{\chi,k}) \geqslant 2b(n)$ (with equality if $X_i \notin \mathcal{P}_{\chi,k}$). Applying the above lemma, we see $F(m) > F(k)$. □

4 An approximate problem

Here we make a conjectural lower bound and asymptotic for $m_{\mathrm{opt}}(n)$ by comparing our problem with a simpler optimization problem. We continue, for simplicity, in the case of $\mathcal{M}_4$, though similar approximate problems could be considered for $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ also.

Based on the bounds for $E_m[\frac{1}{\ell}]$ and $E_m[\frac{\tau_j}{\ell}]$ above, we consider the approximate expectation cost

$$\widetilde{F}(m) = 2\widetilde{F}_{\mathcal{L}}(m) + \widetilde{F}_{\mathcal{P}}(m),$$

where

$$\widetilde{F}_{\mathcal{L}}(m) = mb(n)\frac{1}{E_m[\ell]}$$

and

$$\widetilde{F}_{\mathcal{P}}(m) = \sum_{j=1}^{m-1} \frac{j+1}{2}\frac{j}{n-j}\frac{1}{E_m[\ell]}.$$

Specifically, Lemma 2.5 and Proposition 2.10 imply $F_{\mathcal{L}}(m) \geqslant \widetilde{F}_{\mathcal{L}}(m)$ and

$$F_{\mathcal{P}}(m) \leqslant \sum_{j=1}^{m-1} \frac{j+1}{2}\frac{j}{n-j}E_m\left[\frac{1}{\ell}\right].$$

We suspect that the approximation $\widetilde{F}_{\mathcal{P}}(m)$ is much closer to this upper bound for $F_{\mathcal{P}}(m)$ than $F_{\mathcal{P}}(m)$ itself is, and so we should have $F_{\mathcal{P}}(m) \leqslant \widetilde{F}_{\mathcal{P}}(m)$. This is supported by numerical evidence. (See

Table 4.1 for some numerical calculations.) Moreover, since conjecturally $F_{\mathcal{L}}(m)$ is decreasing in m (and, empirically, faster than $\widetilde{F}_{\mathcal{L}}(m)$ is), while $F_{\mathcal{P}}(m)$ is increasing in m (and, empirically, slower than $\widetilde{F}_{\mathcal{P}}(m)$), we make the following conjecture.

Let $\widetilde{m}_{\mathrm{opt}}(n)$ be the value of $m \in \{1, 2, \ldots, n\}$ which minimizes $\widetilde{F}(m)$. We call the problem of determining $\widetilde{m}_{\mathrm{opt}}(n)$ an *approximate search with cleanup problem*.

Remark 4.1 *Recall that for $m_{\mathrm{opt}}(n)$, we used the function $b(n)$ from* (1.5)*, is an exact average search cost if n is a power of 2 and approximate otherwise (since we are working with a complete memory model here, $b_f(n)$ does not arise here). It is then natural to wonder whether using an exact formula for $b(n)$ for all n makes much of a difference. This is relevant for small n since, but should not matter for asymptotics for large n. However, choosing $b(n)$ as in* (1.5) *does not make much of a difference for small n either; see the emboldened entries the third row in Table 4.1 for the differences. The optimal value in the case of exact average search cost functions is labeled $m_{\mathrm{opt}}^{exact}(n)$.*

Conjecture 4.2 *Let $\mathcal{M} = \mathcal{M}_4$. Then $m_{\mathrm{opt}}(n) \geqslant \widetilde{m}_{\mathrm{opt}}(n)$.*

Tab. 4.1: Small values of $\widetilde{m}_{\mathrm{opt}}$, m_{opt} and $m_{\mathrm{opt}}^{\mathrm{exact}}$

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\widetilde{m}_{\mathrm{opt}}$	1	2	3	4	5	6	7	8	8	8	9	9	9	9	9	10	10	10	10	11
m_{opt}	1	2	3	4	5	6	7	8	8	8	9	9	9	10	10	10	10	10	11	11
$m_{\mathrm{opt}}^{\mathrm{exact}}$	1	2	3	4	5	6	7	8	8	**9**	9	9	9	10	10	10	10	**11**	11	11

n	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
$\widetilde{m}_{\mathrm{opt}}$	11	11	11	11	11	12	12	12	12	12	12	12	12	13	13
m_{opt}	11	11	11	12	12	12	12	12	12	12	13	13	13	13	13
$m_{\mathrm{opt}}^{\mathrm{exact}}$	11	**12**	**12**	12	12	12	12	12	12	**13**	13	13	13	13	13

Theorem 4.3 *For any $n \geqslant 5$, we have*

$$3b(n) - \tfrac{3}{2} \leqslant \widetilde{m}_{\mathrm{opt}}(n) < 3b(n) + \tfrac{1}{2}.$$

In other words, for $n \geqslant 5$, $\widetilde{m}_{\mathrm{opt}}(n)$ is either $\lceil 3b(n) - \frac{3}{2} \rceil$ or $\lceil 3b(n) - \frac{3}{2} \rceil + 1$.

We note that in fact both possibilities of this proposition occur: sometimes $\widetilde{m}_{\mathrm{opt}}(n)$ is $\lceil 3b(n) - \frac{3}{2} \rceil$ and sometimes it is $\lceil 3b(n) - \frac{3}{2} \rceil + 1$, though numerically there seems to be a tendency for $\widetilde{m}_{\mathrm{opt}}(n)$ to be in the right half of this interval, i.e., most of the time $\widetilde{m}_{\mathrm{opt}}(n) > 3b(n) - \frac{1}{2}$.

Proof: Let $1 \leqslant m < n$. We want to investigate when the difference

$$\begin{aligned} \widetilde{F}(m) - \widetilde{F}(m+1) = &\left(2mb(n) + \frac{1}{2} \sum_{j=1}^{m-1} \frac{j(j+1)}{n-j} \right) \left(\frac{1}{E_m[\ell]} - \frac{1}{E_{m+1}[\ell]} \right) \\ &- \left(2b(n) + \frac{1}{2} \frac{m(m+1)}{n-m} \right) \frac{1}{E_{m+1}[\ell]} \end{aligned} \tag{4.1}$$

is positive, i.e., when is $\widetilde{F}(m)$ is decreasing in m? The above expression is positive if and only if

$$2\widetilde{F}_{\mathcal{L}}(m) - 2\widetilde{F}_{\mathcal{L}}(m+1) > \widetilde{F}_{\mathcal{P}}(m+1) - \widetilde{F}_{\mathcal{P}}(m).$$

Since $E_{m+1}[\ell] - E_m[\ell] = \frac{n}{n-m}$, this is equivalent to

$$4b(n)(n-m)\left(\frac{nm}{n-m} - E_m[\ell]\right) > m(m+1)E_m[\ell] - n\sum_{j=1}^{m-1}\frac{j(j+1)}{n-j}. \tag{4.2}$$

The left hand side of (4.2) is

$$4nb(n)\sum_{j=0}^{m-1}\frac{m-j}{n-j},$$

whereas the right hand side of (4.2) is

$$n\sum_{j=0}^{m-1}\frac{m(m+1)-j(j+1)}{n-j} = n\sum_{j=0}^{m-1}\frac{(m-j)(m+j+1)}{n-j}.$$

Hence (4.2) is positive if and only if

$$\sum_{j=0}^{m-1}\frac{m-j}{n-j}\left(4b(n) - (m+j+1)\right) > 0. \tag{4.3}$$

Lemma 4.4 *Let $a, b, c \in \mathbb{Z}$ with $a > 1$ and $c \geqslant \max\{a, b\}$. The sum*

$$\sum_{j=0}^{a}\frac{(a-j)(b-j)}{c-j}$$

is negative if $b \leqslant \frac{a-2}{3}$. This sum is positive if if $b > \frac{a}{3}$ and $c \geqslant 5a^2$.

Proof: All nonzero terms of the sum are positive (resp. negative) if $b \geqslant a$ (resp. $b \leqslant 0$), so assume $0 < b < a$. Now note the above sum is negative if and only if

$$\sum_{j=0}^{b}\frac{(a-j)(b-j)}{c-j} < \sum_{j=b}^{a}\frac{(a-j)(j-b)}{c-j}. \tag{4.4}$$

This is certainly the case if

$$\frac{1}{6}b(b+1)(3a-b+1) = \sum_{j=0}^{b}(a-j)(b-j) \leqslant \sum_{j=b}^{a}(a-j)(j-b) = \frac{1}{6}(a-b)(a-b+1)(a-b-1).$$

Writing $d = a - b$, we see this is true if

$$b(b+1)(3d+2b+1) \leqslant d^3 - d,$$

which holds if $b \leqslant \frac{d}{2} - 1$, i.e., if $b \leqslant \frac{a-2}{3}$.

Similarly, the sum in the lemma is positive if

$$\frac{1}{c}\sum_{j=0}^{b}(a-j)(b-j) \geqslant \frac{1}{c-a}\sum_{j=b}^{a}(a-j)(j-b),$$

i.e., if

$$b(b+1)(3d+2b+1) \geqslant \frac{c}{c-a}\left(d^3-d\right).$$

Suppose $b \geqslant \frac{a}{3}$, i.e., $b \geqslant \frac{d}{2}$. Then the above inequality is satisfied if

$$\frac{d}{2}(\frac{d}{2}+1)(4d+1) \geqslant \left(1+\frac{a}{c-a}\right)\left(d^3-d\right),$$

which holds if

$$\frac{d^2}{4}(4d+1) \geqslant \left(1+\frac{a}{c-a}\right)d^3,$$

which holds if

$$\frac{c-a}{4a} \geqslant a \geqslant a-b = d.$$

This holds if $c \geqslant 5a^2 \geqslant 4a^2 + a$. □

Now applying the first part of this lemma with $a = m$, $c = n$ and $b = \lceil 4b(n) - m - 1\rceil \leqslant 4b(n) - m$, we see

$$m \geqslant 3b(n) + \tfrac{1}{2} \implies \widetilde{F}(m) < \widetilde{F}(m+1),$$

hence $\widetilde{m}_{\mathrm{opt}}(n) < 3b(n) + \frac{1}{2}$.

Similarly, applying the second part of the lemma with $a = m$, $c = n$ and $b = \lfloor 4b(n) - m - 1\rfloor \geqslant 4b(n) - 2$ we see

$$m < 3b(n) - \tfrac{3}{2} \text{ and } n \geqslant 5m^2 \implies \widetilde{F}(m) > \widetilde{F}(m+1).$$

Note $m < 3b(n) - \frac{3}{2}$ implies $5m^2 \leqslant n$ when $45(b(n)-1.5)^2 \leqslant n$. If n is large so that $45(b(n)-1.5)^2 \leqslant n$, then we have $\widetilde{m}_{\mathrm{opt}}(n) \geqslant 3b(n) - \frac{3}{2}$. This is satisfied if $n > 4050$. When $n \leqslant 4050$, one can compute directly that (4.3) holds for all $m < 3b(n) - \frac{3}{2}$. Lastly, note that $n > 3b(n) - \frac{3}{2}$ for $n \geqslant 5$. □

Hence, we should have $3b(n) - \frac{3}{2} \leqslant m_{\mathrm{opt}}(n) < 4b(n)$. Here only the lower bound is conjectural. At least for n small, the table above suggests $\widetilde{m}_{\mathrm{opt}}(n)$ is to be a very good approximation for $m_{\mathrm{opt}}(n)$. This suggests that $m_{\mathrm{opt}}(n)$ grows like $\widetilde{m}_{\mathrm{opt}}(n)$ plus some term of smaller order, and we are led to

Conjecture 4.5 *As $n \to \infty$, we have the asymptotic $m_{\mathrm{opt}}(n) \sim 3b(n)$.*

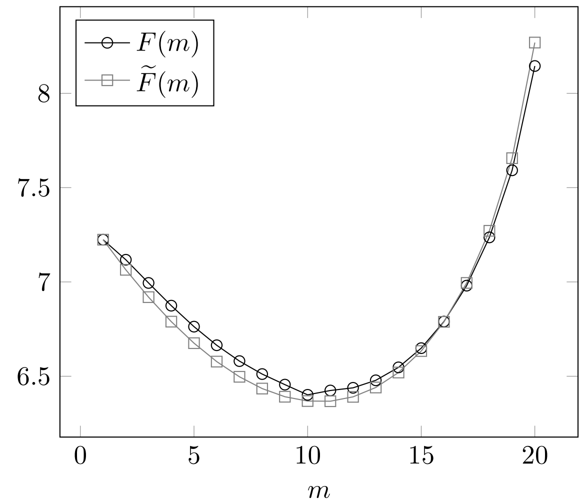

Fig. 1: Comparing $\widetilde{F}(m)$ with $F(m)$ for $n = 20$

Remarks

4.6 *From Table 4.1, we note that for $n \leqslant 8$, one should not clean up until all the objects are in the pile. One might ask for $n \leqslant 8$ if one should ever clean up, i.e., is $F(n)$ at least less than the cost of an average pile search, $\frac{n+1}{2}$? Calculations show this is only true for $n = 7$ and $n = 8$, i.e., one should* never *clean up if $n \leqslant 6$. (This entire remark does not depend on whether one uses the approximate average search cost function for $b(n)$ in* (1.5) *or replaces $b(n)$ with the exact average search cost as discussed in Remark 4.1.)*

4.7 *For $\mathcal{M}_3$, calculations also say that $m_{\text{opt}}(n) = n$ for $n \leqslant 8$, but here it is only better to never clean up if $n \leqslant 2$. Furthermore, calculations suggest that $m_{\text{opt}}(n) \sim 4b(n)$ for $\mathcal{M}_3$.*

4.8 *To see how close $\widetilde{F}(m)$ is to $F(m)$, we plotted both for $n = 20$ in Figure 1. Note that there is significant cost savings to be had by waiting until m_{opt} to clean up. However, for large n, the graph of $F(m)$ will be more lop-sided, i.e., the right endpoint $F(n)$ will be much larger than the left endpoint $F(1)$. It is not feasible to compute all values of $F(m)$ for some large n, but we graph $\widetilde{F}(m)$ for $n = 100$ in Figure 2. We expect that the graph of $F(m)$ will have a similar shape, and that for large n, the cost savings of waiting until m_{opt} to clean up is proportionally smaller. However, the cost savings should be more pronounced for non-uniform distributions.*

5 Non-uniform distributions

Finally we comment on the problem for general probability distributions on $\mathbf{X}$. Now if one defines the cost functions $S(X;\mathcal{P})$ and $C(\mathcal{P})$ using algorithm $\mathbf{A}$ as in Section 1.2, these cost functions do not just depend upon the multiset of probabilities $\{\mu(X_i)\}$, but upon the specific distribution.

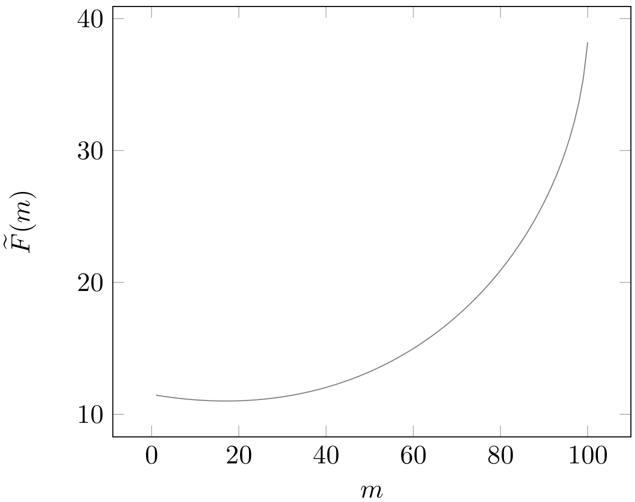

Fig. 2: $\widetilde{F}(m)$ for $n = 100$

Example 5.1 *Fix $1 \leqslant r \leqslant n$ and $0 \leqslant \epsilon \leqslant 1$. Now take the distribution given by $\mu(X_r) = 1 - \epsilon$ and $\mu(X_i) = \epsilon/(n-1)$. Assuming ϵ is small, then most of the time one will be searching for X_r. Depending on what r is, the search (as well as cleanup) cost associated to X_r might be as low as 1 or as high as $b_f(n) \approx \log_2(n)$. Hence, at least for certain values of ϵ and n, one might expect the answer to the associated search with cleanup optimization problem depends upon the choice of r.*

Therefore, we define our cost functions not using the exact search costs given by algorithm $\mathbf{A}$, but rather on the associated average search costs. Specifically, in the complete memory case, we set

$$S(X;\mathcal{P}) = \begin{cases} s_{\mathcal{L}}(n - |\mathcal{P}|) & X \notin \mathcal{P} \\ s_{\mathcal{P}}(|\mathcal{P}|) & X \in \mathcal{P} \end{cases} \tag{5.1}$$

and

$$C(\mathcal{P}) = \sum_{j=1}^{|\mathcal{P}|} s_{\mathcal{L}}(n-j). \tag{5.2}$$

In the case of the uniform distribution on $\mathbf{X}$, this gives us the same optimization problem we studied above.

Note that in the case of no memory, it may be better to always search the pile first, depending on how skewed the distribution is. For instance, in Example 5.1, if ϵ is sufficiently small, then with high probability at any $t \geqslant 1$, we will be looking for X_r and it will be in the pile. Thus we should always search the pile first. Furthermore, by this reasoning (in either the complete or no memory case), for ϵ small enough, we should clean up whenever another object gets in the pile, i.e., $m_{\mathrm{opt}}(n) = 2$.

Consequently, we can decompose the average total cost as in the uniform case

$$F(m) = mE_m\left[\frac{1}{\ell}\right]\cdot 2b(n) + \sum_{j=1}^{m-1} E_m\left[\frac{\tau_j}{\ell}\right] s_{\mathcal{P}}(j), \tag{5.3}$$

though now the quantities $E_m\left[\frac{1}{\ell}\right]$ and $E_m\left[\frac{\tau_j}{\ell}\right]$ will be more complicated. In this case, the probability functions for the underlying Markov process will follow more general sequential occupancy distributions (see, e.g., [6] or [4]).

Note that for a nonuniform distribution, typically objects with higher probabilities will be in the pile at any given time, so the pile search costs will be higher than in the uniform case. Put another way, the expected waiting time $E_m[\ell]$ until cleanup is minimized for the uniform distribution (see, e.g., [7], [5], [3] and [2] for results on $E_m[\ell]$). Therefore, the more skewed the distribution is, the faster the probabilities $E_m\left[\frac{\tau_j}{\ell}\right]$ should be increasing in m, i.e., the smaller $m_{\mathrm{opt}}(n)$ should be, as indicated in our example above. In particular, we expect $m_{\mathrm{opt}}(n)$ is maximized for the uniform distribution.

Acknowledgements

It is a pleasure to thank our colleague Alex Grigo for helpful comments and suggestions. We thank the referee whose careful reading has helped improve the exposition. We also thank our parents for never making us clean up our rooms.

References

[1] Leonard E. Baum and Patrick Billingsley. Asymptotic distributions for the coupon collector's problem. *Ann. Math. Statist.*, 36:1835–1839, 1965.

[2] Petra Berenbrink and Thomas Sauerwald. The weighted coupon collector's problem and applications. In *Computing and combinatorics*, volume 5609 of *Lecture Notes in Comput. Sci.*, pages 449–458. Springer, Berlin, 2009.

[3] Shahar Boneh and Vassilis G. Papanicolaou. General asymptotic estimates for the coupon collector problem. *J. Comput. Appl. Math.*, 67(2):277–289, 1996.

[4] Charalambos A. Charalambides. *Combinatorial methods in discrete distributions*. Wiley Series in Probability and Statistics. Wiley-Interscience [John Wiley & Sons], Hoboken, NJ, 2005.

[5] Philippe Flajolet, Danièle Gardy, and Loÿs Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Appl. Math.*, 39(3):207–229, 1992.

[6] Norman L. Johnson and Samuel Kotz. *Urn models and their application*. John Wiley & Sons, New York-London-Sydney, 1977. An approach to modern discrete probability theory, Wiley Series in Probability and Mathematical Statistics.

[7] Harmindar B. Nath. On the collector's sequential sample size. *Trabajos Estadíst. Investigación Oper.*, 25(3):85–88, 1974.

A Notation guide

Section 1.1

$\mathbf{X}$	a set of n objects (the books) $X_1, \ldots, X_n$
μ	a probability measure on $\mathbf{X}$ (and later $\mathcal{X}_m$)
$\mathcal{L}$	a sorted list (the shelves)
$\mathcal{P}$	an unsorted list (the pile)
$\mathcal{X}_m = \mathcal{X}_{m,n}$	the finite sequences (paths) of objects in $\mathbf{X}$ consisting of m distinct objects, where the last object is distinct from the previous ones
χ	a path in $\mathcal{X}_m$
χ_t	the t-th object in χ
$\ell(\chi)$	the length of χ
$\mathcal{P}_\chi(t)$	the set of objects in $\mathcal{P}$ at time t along path χ
$S(X;\mathcal{P})$	the search cost for object $X \in \mathcal{L} \sqcup \mathcal{P}$ given a certain pile $\mathcal{P}$
$C(\mathcal{P})$	the cleanup cost for a certain pile $\mathcal{P}$
$S(\chi)$	the total search cost along path χ
$C(\chi)$	the cleanup cost for path χ
$F(m) = F(m;n)$	the average total per-search cost for cleaning up when $\lvert\mathcal{P}\rvert = m$
$m_{\mathrm{opt}}(n) = m_{\mathrm{opt}}(n;\mathcal{M})$	the argument which minimizes $F(m)$

Section 1.2

$\mathcal{M}_1$	the no memory, unnumbered shelves model
$\mathcal{M}_2$	the no memory, numbered shelves model
$\mathcal{M}_3$	the complete memory, unnumbered shelves model
$\mathcal{M}_4$	the complete memory, numbered shelves model
$\mathbf{A}$	a search algorithm for the model
$b(j)$	the average case successful binary search cost on a sorted list of length j
$b_f(j)$	the average case failed binary search cost on a sorted list of length j
$s(j)$	the average case sequential search cost on a list of length j
$s_{\mathcal{L}}(j)$	the average cost to search for an element of $\mathcal{L}$ when the list size is j
$s_{\mathcal{P}}(j)$	the average cost to search for an element of $\mathcal{P}$ when the pile size is j
C_m	the average cleanup cost for a pile of size m

Section 2.1

$E[f] = E_m[f] = E_{m,n}[f]$	the expected value of a function on $\mathcal{X}_{m,n}$
$\mathcal{X}_m^{(\ell)}$	the paths in $\mathcal{X}_m$ of length ℓ
$F_S(m)$	the expected search cost per search
$\left\{ {a \atop b} \right\}$	the Stirling number of the second kind
$S_{\mathcal{L}}(\chi)$	the contribution to $S(\chi)$ from searches for objects in $\mathcal{L}$
$S_{\mathcal{P}}(\chi)$	the contribution to $S(\chi)$ from searches for objects in $\mathcal{P}$
$Sym(\mathbf{X})$	the symmetric group on $\mathbf{X}$
$\tau_j(\chi)$	the number of times one does j-element pile search along χ

Section 2.2

$F_{\mathcal{L}}(m)$	the expected value of $S_{\mathcal{L}}$ per search
$F_{\mathcal{P}}(m)$	the expected value of $S_{\mathcal{P}}$ per search
$F_C(m)$	the expected cleanup cost per search

Section 4

$\widetilde{F}(m)$, $\widetilde{F}_{\mathcal{L}}(m)$, $\widetilde{F}_{\mathcal{P}}(m)$	certain approximations for $F(m)$, $F_{\mathcal{L}}(m)$, $F_{\mathcal{P}}(m)$
$\widetilde{m}_{\mathrm{opt}}(n)$	the argument minimizing $\widetilde{F}(m)$