

Randomized Frequency Count

1 Introduction

As discussed in the previous lectures, we have a list of n items: $1, 2, \dots, n$. We receive a sequence of lookups. When we have a lookup to item i , we must pay a cost of $\text{depth}(i)$ (the depth of the first item is defined to be 1, the last is n). After the lookup, we are allowed to rearrange the list as follows. We can move the requested item forward anywhere in the list, between its current position and the front, at no cost. We can then perform arbitrary swaps of adjacent items at a cost of 1 per swap.

In the previous lecture, we analyzed Frequency Count and showed that it will perform well if examples are drawn *independently* from a probability distribution. In this lecture, we show how to get similar guarantees against an arbitrary sequence of lookups, using the following *Randomized Frequency Count* (RFC) algorithm:

- Initialize counters $c_1 = c_2 = \dots = c_n = 0$.
- Keep items sorted in reverse order of number of lookups.
- On lookup to item i , with probability $1/2$, $c_i \leftarrow c_i + 1$, and with probability $1/2$, do nothing.

Surprisingly, this little randomization enables one to show the following:

Theorem 1 *For any sequence of lookups s_1, s_2, \dots, s_T , the expected cost of RFC is at most,*

$$E[\text{total cost of RFC}] \leq \text{total cost of best fixed list} + O(n^2\sqrt{T}).$$

Here, the total cost of the best fixed list refers to the best single list, where best is chosen in hindsight. (This list is the last list used by the FC algorithm.) **Proof.** We follow the proof of the previous lecture, considering two cases.

Case 1. $n = 2$. In a length-2 list, the cost of each lookup is 1 or 2. Let us call the subsequence in which we did increment the counters the “awake” subsequence. In expectation, the number of elements in the awake subsequence is $T/2$. Let us refer to awake-cost as the total cost on the awake periods only. The first thing to observe is,

$$E[\text{awake-cost}] = \frac{1}{2} \text{cost}$$

This is because, on any period, the probability of being awake is exactly $1/2$, and the decision to be awake or not is independent of the true cost. Now, in the proof from last lecture, we showed that,

$$\text{cost of FC} \leq \text{min-static-cost} + \# \text{ changes to FC.} \tag{1}$$

Next, notice that the state of RFC during the awake lookups and the awake-cost is identical to the FC algorithm run on the awake subsequence. Therefore,

$$\text{awake-cost of RFC} \leq \text{min-awake-cost} + \# \text{ changes to RFC.} \quad (2)$$

Here, min-awake-cost refers to the cost of the best single list, chosen in hindsight, that one would use if one had to use a single list on only the awake subsequence. The number of changes to RFC is simply the number of times that RFC changed its list order (from 1,2 to 2,1 or vice versa).

Next, we will argue that the expected number of changes to RFC is at most $O(\sqrt{T})$ for a 2-item list. A change can only happen after an awake requests when the counts were equal $c_1 = c_2$. Let's think about what happens after the a th request to element 1 (not the a th awake request). We can bound the probability that there the list changes. In order for it to change, they must have been tied, and we must decide to increment. Say the count of 1 is $c_1 = m$. (Of course $E[c_1] = a/2$.) Say element 2 has been requested b times. Then the probability that it's count is equal to m is

$$\frac{C(b, m)}{2^b} = \frac{b!}{m!(b-m)!2^b}.$$

First, we can see that the above expression is maximized for $b = 2m$. To see this, call the above expression $f(b)$ and observe that

$$\frac{f(b)}{f(b-1)} = \frac{b}{2(b-m)} = 1 + \frac{2m-b}{2(b-m)}.$$

Therefore $f(b) \geq f(b-1)$ exactly when $2m - b \geq 0$, i.e. $b \leq 2m$, and we see that the maximum is at $b = 2m$.

As in Homework 1, problem 3, it is not difficult to show that this is at most c/\sqrt{m} for some constant c . Thus the expected sum over all requests is at most $\sum_{m=1}^T c/\sqrt{m} = O(\sqrt{T})$ because we increment each counter at most T times.

Case 2. $n > 2$. By the same argument as in Case 2 of the proof from last lecture, the regret for more than 2 items can be decomposed into the sum of the regrets for each pair of items, and thus the total regret is at most $O(n^2\sqrt{T})$. \square