Kick Start 2018 - Round A

Analysis: Lucky Dip

Lucky Dip: Analysis

Given the array of **N** integers, define E[k] as the optimal expected value of the item drawn given k redips.

Small dataset

There are two cases to consider for the small dataset: **K** = 0 and **K** = 1.

When **K** = 0, we must accept the first item which we draw. Since the chance of obtaining each item is the same, the expected value is the average value of all the items in the bag.

When **K** = 1, we note that if we redip, the expected value that we draw is going to be the same as the case when **K** = 0. Thus, the best strategy is to keep the item if the value of the item is greater than the average value of the items in the bag, or return the item otherwise.

Thus, we have the following equations to compute the values for E[0] and E[1]:

$$E[0] = \Sigma\, \mathbf{V_i} \;/\; \mathbf{N}$$
$$E[1] = \Sigma\, \max(\mathbf{V_i},\, E[0]) \;/\; \mathbf{N}$$

The values for E[0] and E[1] can be computed in O(**N**) time.

Large dataset

When **K** > 1, we may proceed with a similar recursion as in the case when **K** = 1. Given k redips, we should keep the item if the value of the item is greater than the expected value for k-1 redips, and we should return the item otherwise. This gives rise to the following recursion:

$$E[k] = \Sigma\, \max(\mathbf{V_i},\, E[k-1]) \;/\; \mathbf{N}$$

Computing E[**K**] using the above approach requires us to compute E[0], E[1], ..., E[**K** - 1], each of which requires O(**N**) time. Thus, the total runtime amounts to O(**NK**), which is fast enough to pass the large dataset. However, it is possible to derive an even faster solution.

Suppose we have already computed the value for E[k - 1]. If our array of $\mathbf{V_i}$s is sorted, we can use a binary search to quickly determine the quantity of items which have value less than or equal to E[k - 1]. Let us denote this quantity with x_k. Note that the probability of drawing one of these x_k items is x_k / **N**. This corresponds to the case when we return the item and redip.

On the other hand, there is an equal likelihood for drawing each of the remaining **N** - x_k items. This gives us the following equation for E[k]:

$$E[k] = x_k * E[k - 1] \;/\; \mathbf{N} + \Sigma_{i>x_k}\, \mathbf{V_i} \;/\; \mathbf{N}$$

For fast computation, we precompute by sorting the array of $\mathbf{V_i}$s in increasing order, and computing the suffix sums for the array. We will then compute E[0], E[1], ..., E[**K**] accordingly, in which each step takes O(log **N**) time. The total runtime is then O(**N** + **N** log **N** + **K** log **N**) = O((**N** + **K**) log **N**).