HW 5 (due Wednesday, at noon, October 10, 2018)

CS 473: Algorithms, Fall 2018 Version: **1.3**

Submission guidelines and policies as in homework 1.

1 (100 PTS.) Storing fast.

Consider an array $A[1 \ldots n]$ which is initially empty. One can check in constant time whether an entry in array is empty, or store a value in an array. Think about this array as a cache. You are given a stream $x_1, \ldots, x_m$ of elements to store in the array. Let rand(n) be a library function that returns a random number chosen uniformly from $\{1, \ldots, n\}$.

In the ith iteration, the algorithm randomly select a location $r_i \leftarrow rand(n)$, and tries to store x_i in the location $A[r_i]$. If this location already contains a value, the algorithm retries by resampling $r_i \leftarrow rand(n)$, and repeat doing it till success.

1.A. (10 PTS.) For $m = n$ prove an exact bound on the expected number of calls to $rand(n)$.

Solution: Let X_i be the number of calls to *rand* for x_i. We have that X_i has a geometric distribution, we probability $(n-i+1)/n$, since this the probability of throwing x_i into an empty bin. As such, we have that $\mathbb{E}[X_i] = n/(n-i+1)$, and the expected number of calls to rand is $\sum_{i=1}^{n} n/(n-i+1) = O(n \log n)$.

1.B. (10 PTS.) For $m = n/2$ prove an exact bound on the expected number of calls to $rand(n)$.

Solution: Let X_i be the number of calls to *rand* for x_i. We have that X_i has a geometric distribution, we probability $(n-i+1)/n$, since this the probability of throwing x_i into an empty bin. As such, we have that $\mathbb{E}[X_i] = n/(n-i+1)$, and the expected number of calls to rand is $\sum_{i=1}^{n/2} n/(n-i+1) \leq n$.

1.C. (20 PTS.) Consider the variant that after trying 3 times for each element to store it, the algorithm simply reject it. For $m = n$, provide an upper bound (as tight as possible [constants matter here]) on the expected number of elements being rejected.

Solution: Assume, pessimistically, that all elements were stored correctly before x_i is handled. As such, when the algorithm tries to store x_i, there are at least $n-(i-1)$ empty slots (if some elements get rejected, then there are event more empty slots). As such, the probability of the ith element to get rejected is at most

$$p_i = \left(1 - \frac{n-(i-1)}{n}\right)^3 = \left(\frac{(i-1)}{n}\right)^3.$$

In particular, let Y_i be an indicator variable that is one if x_i get rejected. We have that $\mathbb{E}[Y_1] \leq (i-1)^3/n^3$. As such, by linearity of expectations, we have that

$$R_1 = \mathbb{E}\left[\sum_{i=1}^{n} Y_i\right] \leq \sum_{i=1}^{n} \mathbb{E}[Y_i] \leq \sum_{i=1}^{n-1} \frac{i^3}{n^3} \leq \frac{(n-1)^2 n^2}{4n^3} \leq \frac{n}{4},$$

by looking up the formula $\sum_{i=1}^{n} i^3 = \frac{n^2(n+1)^2}{4}$.

1.D. (20 PTS.) We take all the rejected elements from the previous part (for simplicity assume the number of these elements is exactly your upper bound from the previous part), and the algorithm tries to store them again, in a new array $B_2[1 \ldots, n]$, which is initially empty, again doing three tries before rejecting an element. Provide an upper bound (as tight as possible) on the expected number of elements being rejected.

Solution:

This is easier – we have $n/4$ elements, and the probability of a ball to get rejected is at most $1/4^3$. As such, the expected number of rejected balls R_2 is at most $n/64$.

1.E. (10 PTS.) Assume we do the above for i layers. Let R_{i-1} be the number of elements that got rejected in the first $i-1$ layers. Assume that $R_{i-1} \leq q_i n$, for some constant $q_i < 1/2$. Let Z_i be the number of elements that get rejected in the ith layer. Prove that $\mathbb{E}[Z_i] \leq q_i^3 n$.

In the following, for the sake of simplicity of exposition, we assume that for all i, $R_i = \mathbb{E}[Z_i]$.

Solution:

We are throwing $q_i n$ elements into an array of size n. The probability that an element get rejected is bounded by $((q_i n)/n)^3 = q_i^3$. The result now readily follows from linearity of expectation. Indeed, let B_{i-1} be the set of balls rejected in the first $i-1$ rounds. For a ball $b \in B_{i-1}$, let X_b be an indicator variable that is one $\iff$ the ball b is rejected in the ith round. We have,

$$\mathbb{E}[Z_i] = \mathbb{E}\left[\sum_{b \in B_{i-1}} X_b\right] = \sum_{b \in B_{i-1}} \mathbb{E}[X_b] \leq q_i^3 \left|B_{i-1}\right| \leq q_i^3 R_i \leq q_i^4 n \leq q_i^3 n.$$

One can prove with some more work that $\mathbb{P}\big[Z_i > 2q_i^4 n\big] < 1/n^{O(1)}$, which is sufficient to prove the rest of the exercise.

1.F. (30 PTS.) Prove an upper bound (as small as possible), using the above, that holds with high probability, on the number of rounds one need to have till all elements are stored somewhere.

Solution:

By the above, $q_1 \leq 1/2$, $q_2 \leq 1/2^{2^1}$, and more generally, setting $q_i \leq 1/2^{2^i}$, we have that $q_{i+1} \leq q_i^3 \leq 1/2^{2^{i+1}}$.

In particular, for $I = \lceil \lg(\lg n) \rceil + 2$, we have that $R_I \leq q_I n \leq n/2^{4 \lg n} \leq 1/n^3$. Namely, with high probability, the number of rounds needed is at most $\lg \lg n + 2$. To see that, observe that

$$\mathbb{E}[Z_I] \leq R_I \leq 1/n^3.$$

As such, we have that

$$\mathbb{P}[Z_I > 0] = \mathbb{P}[Z_I \geq 1] \leq \frac{\mathbb{E}[Z_I]}{1} \leq \frac{1}{n^3},$$

which implies the claim.

Since $\lg \lg n \leq 8$ for all reasonable values (since $2^{2^8} = 2^{256} \geq 10^{25}$, this implies that we need only a constant number of layers!

Note, that the assumption that the number elements getting rejected is exactly the expected bound, is of course false. The same analysis essentially holds, but the details becomes much hairier.

Solution:

For the reader interested in this surprising result, I suggest reading about the power of two choices.

2 (100 PTS.) Cuts and stuff.

2.A. (10 PTS.) Consider a graph $G = (V, E)$ with n vertices, m edges and a min cut of size k. Let $\mathcal{F}$ be the collection of all min-cuts in G (i.e., all the cuts in $\mathcal{F}$ are of size k). What is the probability that MinCut (the simpler variant – see class notes) would output a specific min-cut $S \in \mathcal{F}$?

(Here, we are looking for a lower bound on the probability, since two minimum cuts of the same size might have different probabilities to be output.)

[And yes, this is easy.]

Solution:

Well. The analysis done in class implies that this is $\geq 2/n(n-1)$.

2.B. (10 PTS.) Let S be a set of numbers. Consider a randomized algorithm, that for any $x \in S$, outputs x with probability at least p. Prove that $|S| \leq 1/p$. Here the algorithm just runs, and output one element of S in the output and then stops.

Solution: For $s \in S$, let X_s be an indicator variable that is one $\iff$ s is the output. Observe that $\mathbb{E}[X_s] \geq p$. As such, we have that $1 = \sum_{s \in S} X_s$, which in turn implies that $1 = \mathbb{E}[1] = \sum_{s \in S} \mathbb{E}[X_s] \geq |S| p$, which readily implies that $|S| \leq 1/p$.

2.C. (10 PTS.) Bound the size of $\mathcal{F}$ using (B).

Solution:

Since a cut has probability $\geq 2/n(n-1)$ to be output, it follows that the number of cuts is at most $n(n-1)/2$.

2.D. (10 PTS.) A ***good cut*** is a cut $(S, \overline{S})$ such that the induced subgraphs G_S and $G_{\overline{S}}$ are connected. Consider a specific good cut $C = \left(S, \overline{S}\right)$ with kt edges in it, where $t \geq 1$. What is the probability that MinCut would output this cut? (Again, proof a lower bound on this probability.)

Solution:

The cut we are interested in has kt edges, as such, the probability it is being hit in the ith iteration is $2kt/((n-i+1)k)$. As such, the desired probability, as long as the graph has more than $2t$ vertices, is at least

$$\alpha = \frac{n-2t}{n} \cdot \frac{n-1-2t}{n-1} \cdots \frac{2t+1-2t}{2t+1} \geq \frac{1}{n^{2t}}.$$

For the last few iterations we need to modify the argument. There is always an edge that is not in the cut of interest, and its weight is least one. All the edges in the desired cut have weight at least kt. As such, the probability of picking a good edge in each step of the final $2t-1$ iteration is at least $\delta/(kt+\delta) \geq 1/kt$, where δ is the total weight of the edges that are outside the cut of interest, and the probability of success is

$$\beta \geq \left(\frac{1}{kt}\right)^t.$$

As such, the probability the algorithm outputs the good cut is at least

$$\alpha\beta \geq \frac{1}{n^{2t}}\left(\frac{1}{kt}\right)^t \geq \left(\frac{1}{nkt}\right)^{2t} \geq \frac{1}{n^{6t}}.$$

2.E. (40 PTS.) Consider the set $\mathcal{F}(t)$ of all good cuts in G of size at most kt. Bound the size of $\mathcal{F}(t)$.

Solution:

The probability of the algorithm to output a good cut is $\geq 1/n^{6t}$ by the above. As such, arguing as above, it follows that the number of good cuts is at most n^{6t}.

2.F. (20 PTS.) Consider the set $\mathcal{F}'(t)$ of all cuts in G (not necessarily all good) of size at most kt. Bound the size of $\mathcal{F}'(t)$.

Solution:

The number of such cuts is bounded from above by

$$\sum_{i=0}^{kt} \binom{n(n-1)/2}{i} \leq 2\left(\frac{en^2}{kt}\right)^{kt}$$

(this is just the number of subsets of this size). This estimate is not way off. Indeed, consider a path of length $n-1$ with n vertices, and observe that any subset of at most kt edges is a valid cut of size $\leq kt$, and there are at least

$$\sum_{i=0}^{kt} \binom{n-1}{i} \geq \left(\frac{n-1}{kt}\right)^{kt}$$

3 (100 PTS.) Cut, cut, cut.

3.A. (40 PTS.) Given a connected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with n vertices, and $m \geq n$ edges, consider the algorithm that assign edges random weights (say in $[0, 1]$), and computes the MST $\mathcal{T}$ of the resulting graph. Let e be the heaviest edge in T and consider the cut induced by the two connected components of $T - e$. Prove that with probability $\geq 2/n(n-1)$, this is a mincut of G.

Solution:

One can interpret the running of the algorithm of Kruskal algorithm on the randomly weighted edges of this graph, as run of the min-cut algorithm. The result then immediately follows from the proof show in class/notes.

3.B. (Not for submission - for fun.) Provide an algorithm that in (expected or deterministic) $O(m)$ time returns the heaviest edge in the MST of G. [Without computing the MST, of course.]

Solution: Not for you, and not for me.

3.C. (Harder – not for submission.) Consider the task of computing any spanning tree $\mathcal{T}_1$ of G, then computing any spanning (forest) $\mathcal{T}_2$ of $\mathsf{G} - \mathsf{E}(\mathcal{T})$, and continuing in this fashion, where $\mathcal{T}_i$ is a spanning forest of the graph remaining from G after removing the edges of $\mathcal{T}_1, \ldots, \mathcal{T}_{i-1}$. Prove, that one can compute a sequence of such spanning forests $\mathcal{T}_1, \mathcal{T}_2, \ldots$ in $O(m \operatorname{polylog} n)$ time.

Solution: Not for you, and not for me.

3.D. (60 PTS.) Assume you are given that the mincut in G is of size k. Present an algorithm, with running time $O(m \operatorname{polylog} n)$, that computes a graph $\mathsf{H} \subseteq \mathsf{G}$, such that:

(i) The mincut of H is a mincut of G.

(ii) $|\mathsf{E}(\mathsf{H})| = O(nk)$.

Solution:

Run the algorithm of part (C), and let $\mathcal{T}_1, \mathcal{T}_2, \ldots$ be the resulting forests. The claim is that $\mathsf{H} = \mathcal{T}_1 \cup \ldots \cup \mathcal{T}_{k+1}$ is the desired graph. The bound on the bound on the number of edges in the resulting graph is immediate, as is the running time bound. As for correctness, consider any cut $X = (S, \overline{S})_{\mathsf{G}}$. Assume that $\mathcal{T}_i$, for some i, does not contain any edge of X. But then, this implies that all the edges of X are contained in $\mathcal{T}_1 \cup \ldots \cup \mathcal{T}_{i-1}$ – indeed, otherwise, any spanning tree of the graph $\mathsf{G} \setminus (\mathcal{T}_1 \cup \cdots \cup \mathcal{T}_{i-1})$ must cross the cut X. Namely $\left|(S, \overline{S})_{\mathsf{G}}\right| = \left|(S, \overline{S})_{\mathsf{H}}\right|$ in this case.

Otherwise, $Y = (S, \overline{S})_{\mathsf{H}}$ must contain at least one edge of $\mathcal{T}_i$, for all i. Namely, we have that $|Y| \geq k + 1$, and it is not the minimum cut in H.

Observe that H being a subgraph of G, has a mincut that can only be smaller than the mincut in G, which establishes the claim.