Algorithms, 2023 Fall, Homework 12
(Due: Dec 21)

December 12, 2023

Problem 1 : KT book pp 782 problem 2

Problem 2 : KT book pp 787 problem 7

Problem 3 : Consider the balls-and-bins game with m balls and n bins. Let ϵ be some small positive constant (say 0.1). Suppose $m = cn \ln n$ for some large constant c (c could depends on ϵ). Show that with high probability the load of each bin is approximately the same, i.e.,

$$(1-\epsilon)m/n \leq load_i \leq (1+\epsilon)m/n \quad \text{for all} \quad i$$

Problem 4 : Prove the statement I made in the class: Suppose $G \subseteq U$. We know the cardinality $|U|$ and we can take samples uniformly from U efficiently. Moreover, there is an membership oracle for G: for any sample x, the oracle tells you whether $x \in G$ or not in constant time. Our goal is to estimate $|G|$, the cardinality of G. Let $\alpha = |G|/|U|$. Prove the following theorem: Suppose we take N samples $x_1, \ldots, x_N$ uniformly from U, and estimate $|G|$ as follows:

$$g = \frac{\sum_{i=1}^{N} 1(x_i \in G)}{N} \times |U|.$$

For $0 < \epsilon < 1/2$ and $N \geq \frac{1}{\epsilon^2 \alpha} \log 1/\delta$, the following holds:

$$\Pr[g \in (1 \pm \epsilon)|G|] \geq 1 - \delta.$$

Problem 5 : Suppose we want to evaluate the value of $g = \int_0^1 f(x)dx$. However, $f(x)$ is not a nice function that we can compute the integral using formulas we learnt in calculus class. Nevertheless, for each x, you can compute the value of $f(x)$ easily, say in $O(1)$ time. Now, we estimate the integral by

$$(\hat{g}) = \frac{1}{n} \sum_{i=1}^{n} f(x_i),$$

where x_i is an uniformly random chosen points from $[0, 1]$. Argue why this is a good estimation. For fixed value ϵ and δ, show how many samples we need so that

$$\Pr[|g - \hat{g}| \leq \epsilon] \geq 1 - \delta.$$

Problem 6 : There are n straight lines $\ell_1, \ldots, \ell_n$ in $\mathbb{R}^2$ (assuming general positions, meaning that no 3 points are co-linear).

1. Design an $O(n \log n)$ time algorithm for the following decision problem: Given two vertical slab W bounded by two vertical line $x = a$ and $x = b$, compute how many intersection points of $\{\ell_i\}_{i \in [n]}$ are there in W.

2. Given a vertical slab, show how to uniformly sample q intersection points in W in $O(n \log n + q)$ time.

3. (bonus) Use the above result, show how to find the intersection point with the kth largest x-coordinate in $O(n \log^2 n)$ time (Hint: you may design a randomized algorithm that succeeds with high probability. One way to do this is to sample q intersection points, for some appropriate value q. If q is small, you can afford to sort those sampled elements. You can show with high probability, the point we wanted should be between two particular sampled points. Then, we recurse on the vertical slab bounded by the two points.).

(in fact, you can even get an $O(n \log n)$ time (randomized) algorithm. It is even possible to get a deterministic $O(n \log n)$ time algorithm, but it is very complicated).

Problem 7 : Randomized algorithms have been extensively used for approximate counting. Suppose we want to estimate some number Z (say it is the number of independent sets of a certain graph). We say that we have an FPRAS if there is an algorithm that for any constant $\epsilon > 0$, can output an estimate Z' such that

$$\Pr[(1-\epsilon)Z \leq Z' \leq (1+\epsilon)Z] \geq 0.99.$$

in $poly(1/\epsilon, n)$ time. Here n denotes the size of the input.

There is a close connection between counting and sampling (sampling is inherently randomized). Suppose we have an algorithm A that can take a uniform sample from the set of all independent sets. Then, we can use A (as an oracle) to obtain a FPRAS for counting the number of independent set.

(1) Consider a graph G and G' which is obtained by deleting an edge from G. We use $I(G)$ to denote the number of independent set in G. Show

$$1 \geq I(G)/I(G') \geq c$$

for some constant c.

(2) Show how to A to obtain a FPRAS for counting the number of independent set.

sidenote: one can also achieve the reverse direction (if one has a FPRAS, one can get a sampling algorithm).

Bonus problems: Bonus problems are problems that may be (may be not) more challenging than other problems. You do not lose points if you do not answer bonus problems. But you will get extra points if you get correct answers for such problems. **To make your life easier, we give you two weeks for the bonus problems**).

Problem 1 : We are given a set of points in the plane. We would like to compute the convex hull of those points. Now we design a randomized algorithm for computing convex hull in expected $O(n \log n)$ time. We first permute the points randomly. Suppose that they are $P_1, P_2, \ldots, P_n$. The initial convex hull is the triangle. $H_2 = P_0P_1P_2$. Suppose the origin o is inside H_2. Then we process the rest of points one by one, and update the convex hull accordingly. Suppose we are now processing P_i. If P_i is inside the current convex hull, we do nothing (don't worry about how to do this for now). If P_i is outside the current convex hull, we need to update the convex hull H_{i-1} to be the convex hull of $H_i = \text{ConvexHull}\{P_1, \ldots, P_i\}$.

In order to implement the above steps efficiently, for each point P_j outside the current convex hull H_{i-1}, we maintain a pointer pointing to one edge e of H_{i-1}, which is the one that intersects the segment oP_j (recall that o is the origin). For each edge e, we maintain a list of points (outside the current convex hull) which point to e.

When we process P_i (which is outside H_{i-1}), we need to add two new edges, delete a few old edges, and a few points (which was outside H_{i-1} but inside H_i), and update some pointers (to the newly added edges).

1. Show how to do the above using the data structure I mentioned.

2. Prove that the expected running time is $O(n \log n)$. (hint: the analysis for counting the number of pointer updates is very similar to the analysis of the close pair problem. In particular, you want to show that the expected number of pointer updates for H_i at most $2n/(i+1)$).

I have provided enough hints. So try to do this by yourself (without finding answers from the internet).