COL351: Analysis and Design of Algorithms **Fall 2024**

Tutorial Sheet 3

Announced on: Aug 08 (Thurs)

Problems marked with ($\star$) will not be asked in the tutorial quiz.

1. In this problem, we will design an algorithm to fairly divide a (mathematical) cake among a bunch of kids. Let the interval $[0, 1]$ denote a *cake*. There are n kids, and kid i's preferences over the cake are given by a nonnegative and integrable *value density* function $v_i : [0, 1] \to \mathbb{R}_{\geqslant 0}$, which should remind you of the probability density function. For any subinterval (or *piece*) $[a, b] \subseteq [0, 1]$ of the cake, kid i's value for it is given by $V_i([a, b]) := \int_a^b v_i(x)dx$. We will assume that every kid values the entire cake at 1, i.e., for every i, $V_i([0, 1]) := \int_0^1 v_i(x)dx = 1$.

   A cake *division* $X = (x_1, x_2, \ldots, x_n)$ refers to a partition of the cake into n subintervals $[x_0, x_1]$, $[x_1, x_2]$, ..., $[x_{n-1}, x_n]$ where $x_0 = 0$ and $x_n = 1$.[1] A division is said to be *proportional* if each agent's value for its piece is at least $1/n$ of its value for the entire cake. For example, if $n = 3$ and kid 1 gets the piece $[0, x_1]$, kid 2 gets $[x_1, x_2]$, and kid 3 gets $[x_2, 1]$, then proportionality would require that $V_1[0, x_1] \geqslant 1/3$, $V_2[x_1, x_2] \geqslant 1/3$, and $V_3[x_2, 1] \geqslant 1/3$.

   The valuations can be accessed via two types of *queries*:

   - a `cut`(i, x, α) query, which, given an index i, a starting point $x \in [0, 1]$ on the cake and a value α, returns a point $y \in [0, 1]$ on the cake such that kid i's value for the piece $[x, y]$ equals α (if such a point exists) and return `null` otherwise, and
   - an `eval`(i, x, y) query, which, given an index i and two points $x, y \in [0, 1]$ on the cake such that $x \leqslant y$, returns kid i's value for the piece $[x, y]$.

   a) Suppose there are $n = 2$ kids. Design an algorithm that always computes a proportional cake division using at most two queries.

   b) Let us now talk about a general number n of agents. Consider the following algorithm: Each agent i makes a mark at point x_i such that its value for the piece $[0, x_i]$ is exactly $1/n$. The agent with the leftmost mark is given the enclosed piece, and the algorithm resumes with the remaining agents and the remaining cake. Is this algorithm proportional? How many queries does it need in the worst case?

   c) For a general n, design a divide-and-conquer algorithm that, given any preferences of the kids, always returns a proportional cake division using at most $\mathcal{O}(n \log n)$ queries. Is this algorithm faster or slower than the one in part (b) (in terms of number of queries)? You may assume n to be a power of 2.

   d) For part (c), can you design an algorithm without using `eval`$(\cdot)$ queries?

[1] The endpoints of the subintervals have zero value and can be arbitrarily assigned to either of the adjacent pieces.

2. Consider the "Twenty Questions" game. In this game, the first player thinks of a number in the range 1 to n. The second player has to figure out this number by asking the fewest number of true/false questions. Assume that all responses are honest.

   a) What is an optimal strategy if n is known?

   b) What is a good strategy if n is not known?

3. In class, we saw an $\mathcal{O}(n \log n)$ time algorithm to count the number of inversions in an array $A[1:n]$. Recall, an inversion is a pair (i, j) with $1 \leqslant i < j \leqslant n$ and $A[i] > A[j]$. Let us call an inversion (i, j) *bounded* if $A[j] < A[i] \leqslant A[j] + 10$. That is, $A[i]$ is bigger than $A[j]$ but not by much. Design an $\mathcal{O}(n \log n)$ time algorithm to count the number of bounded inversions.

4. We are given n wooden sticks, each of integer length, where the i^{th} piece has length $L[i]$. We seek to cut them so that we end up with k pieces of exactly the same length, in addition to other fragments. Furthermore, we want these pieces to be as large as possible.

   a) Given four wooden sticks of lengths $L = [10, 6, 5, 3]$, what are the largest sized pieces you can get for $k = 4$? (Hint: The answer is not 3.)

   b) Design a polynomial-time algorithm that, for a given L and k, returns the maximum possible length of the k equal pieces cut from the original n sticks.

5. Suppose an undirected graph $G = (V, E)$ is represented both as an adjacency matrix and an adjacency list. What is the time complexity of the following operations under each of these representations?

   a) Checking whether two vertices u and v are adjacent in G.

   b) Computing the total number of vertices adjacent to the vertex u.

   c) Computing the number of common neighbors of two vertices u and v.

   d) Adding or removing a vertex or an edge.

   Which of your answers would change, and how, if you additionally knew whether the graph was sparse or dense?

6. A *bipartite* graph is a graph $G = (V, E)$ whose vertices can be partitioned into two sets (i.e., $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$) such that there are no edges between vertices in the same set. Give a linear-time algorithm to determine whether a given graph is bipartite.

7. ($\star$) Let X and Y be two independent random variables whose domain is $\{0, 1, 2, ..., n\}$. You are given their explicit distributions $(p_0, p_1, ..., p_n)$ and $(q_0, q_1, ..., q_n)$ where $p_i = \Pr[X = i]$

and $q_j = \Pr[Y = j]$. Let $Z = X + Y$ be the sum of these two random variables. Design an algorithm that outputs the distribution of Z; that is, for all k you should be able to read out $\Pr[Z = k]$. The running time of your algorithm should be much faster than $\Theta(n^2)$.