CS 630: Graduate Algorithms	Released on 9/29

Homework 2

Prof. Charalampos Tsourakakis — *Due to: Friday 5pm, 10/13*

Instructions

- The homework is due on **Friday 10/13 at 5pm ET**.
- No extension will be provided, unless for serious documented reasons.
- **Despite having two weeks, better start early than late!**
- Unless specified differently, the points are distributed equally among the sub-questions.
- Study the material taught in class, and feel free to do so in small groups, but the solutions should be a product of your own work.
- This is not a multiple choice homework; reasoning, and mathematical proofs are required before giving your final answer.
- **Sub-optimal algorithms and lack of proof of correctness will lead to a major deduction in points.**
- If you work with others or utilize any external tools and resources, please make sure to annotate your answers.
- Please submit your work through Gradescope. You can find the access code on Piazza.

Exercise 1 [10 points]

Given an integer array A of length n, and two integers k and ℓ, find the number of pairs of indices i, j (where $1 \leq i, j \leq n$) such that $\texttt{A}[i] \leq \texttt{A}[j]$ and there are exactly k elements in the range $[\texttt{A}[i], \texttt{A}[j]]$ that are divisible by ℓ[1]. Give an efficient algorithm and analyze its time and space complexity.

Exercise 2 [10 points]

The input comprises a real number x and a matrix $A[1..n, 1..m]$ containing nm real numbers. It is given that each row $A[i, 1..m]$ is sorted in ascending order from left (i.e., first column $j = 1$) to right

[1] For example, consider $n = 5, \texttt{A} = [44, 13, 5, 7, 16, 40], k = 1, \ell = 3$. The pair $(6, 1)$ should be counted since $\texttt{A}[6] = 40, \texttt{A}[1] = 44$ and there is a single ($k = 1$) number x such that $40 \leq x \leq 44$ that is divisible by 3 (i.e., $x = 42$).

(i.e., last column $j = m$), and each column $A[1..n, j]$ is sorted in ascending order from top (i.e., first row $i = 1$) to bottom (i.e., last row $j = n$). The objective is to ascertain whether the value x is present within matrix A. Give an efficient algorithm and analyze its time and space complexity.

Exercise 3 [10 points]

Given a sequence of n distinct numbers $\mathtt{A} = \langle a_1, a_2, \dots, a_n \rangle$, we define a pair (a_i, a_j) as an inversion if $i < j$ but $a_i > a_j$. The total number of inversions in the sequence A, denoted as $I(A)$, is the count of pairs (a_i, a_j) that are inversions. Design a divide and conquer algorithm to determine $I(A)$ in $\Theta(n \log n)$ time.

Exercise 4 [10 points]

In a theater, there are n front seats arranged in a straight line. Due to COVID-19 policies, individuals are required to maintain an empty seat's distance from others on both their left and right. As individuals arrive, they randomly select and occupy a seat.

1. Give tight upper and lower bounds on the number of occupied seats. Prove the tightness of your bounds.
2. Write a probabilistic recurrence that expresses the expected number of occupied seats.

Exercise 5 [10 points]

Consider a Monte Carlo algorithm A for a problem Π whose expected running time is at most $T(n)$ on any instance of size n and that produces a correct solution with probability $\gamma(n)$. Suppose further that given a solution to n, we can verify its correctness in time $t(n)$. Show how to obtain a Las Vegas algorithm that always gives a correct answer to Π. What is the expected run time of your procedure?

Exercise 6 [5 points]

Suppose you are given a coin for which the probability of HEADS, say p, is unknown. How can you use this coin to generate unbiased (i.e., $\mathbb{P}[\text{HEADS}] = \mathbb{P}[\text{TAILS}] = 1/2$) coin-flips? Give a scheme for which the expected number of flips of the biased coin for extracting one unbiased coin-flip is no more than $\frac{1}{p(1-p)}$.

Exercise 7 [10 points]

A large dining hall has 137 seats available for a special dinner event, and every seat has been reserved for the night. The guests arrive in a random order to the hall. However, the 17th guest to arrive is confused and can't remember the number of his assigned seat, so he chooses a seat at random and sits down. From then on, whenever a guest arrives and finds their seat already taken, they also choose a random seat to sit. What is the probability that the last guest to arrive finds their assigned seat empty?

Exercise 8 [10 points]

An alternative analysis of the running time of randomized quicksort focuses on the expected running time of each individual recursive call to `RANDOMIZED-QUICKSORT`, rather than on the number of comparisons performed.

1. Argue that, given an array of size n, the probability that any particular element is chosen as the pivot is $\frac{1}{n}$.
2. Show that
$$\mathbb{E}[T(n)] = \frac{2}{n} \sum_{q=2}^{n-1} \mathbb{E}[T(q)] + \Theta(n). \tag{8.1}$$
3. Show that
$$\sum_{k=2}^{n-1} k \lg k \leq \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2. \tag{8.2}$$
4. Using the bound from equation (8.2), show that the recurrence in equation (8.1) has the solution $\mathbb{E}[T(n)] = \Theta(n \lg n)$.

Exercise 9 [10 points]

Prove the lower bound of $\left\lfloor \frac{3n}{2} \right\rfloor - 2$ comparisons in the worst case to find both the maximum and minimum of n numbers.

Exercise 10 [15 points + 10 bonus points]

Theory part [15 points]

For n distinct elements $x_1, x_2 \ldots x_n$ with positive weights $w_1, w_2 \ldots w_n$ such that $\sum_{i=1}^{n} w_i = 1$, the weighted median is the element x_k satisfying $\sum_{x_i < x_k} w_i < \frac{1}{2}$ and $\sum_{x_i > x_k} w_i \leq \frac{1}{2}$. For example, if the elements are $0.1, 0.1, 0.1, 0.3, 0.4$ and each element equals its weight (so $w_i = x_i$ for $1 \leq i \leq 5$), then the median is 0.1, but the weighted median is 0.3.

(a) Argue that the median of $x_1, x_2 \ldots x_n$ is the weighted median of the x_i with weights $w_i = 1/n$ for $i = 1, 2, \ldots n$.

(b) Show how to compute the weighted median of n elements in $O(n \log n)$ worst-case time using sorting.

(c) Show how to compute the weighted median in $\Theta(n)$ worst-case time using a linear-time median algorithm.

Define the distance between two elements a, b as $d(a, b) = |a - b|$. Given points $x_1, x_2 \ldots x_n$ with positive weights $w_1, w_2 \ldots w_n$ as defined previously, we wish to find a point x (not necessarily one of the input points) that minimizes the sum $\sum_{i=1}^{n} w_i \cdot d(x, x_i)$.

(d) Argue that the weighted median is a best solution.

(e) We generalize the problem into 2-dimensions. We are given a set of n points $\{(x_i, y_i)\}_{i=1,\ldots,n}$, and their corresponding positive weights $w_1, w_2 \ldots, w_n$ such that $\sum_{i=1}^{n} w_i = 1$. What is a best solution point (x, y) that minimizes its weighted distance sum to all the n points in the set? Here the weighted distance between points (x, y) and (x_i, y_i) is defined as $w_i \cdot (|x_i - x| + |y_i - y|)$.

Coding part [10 bonus points]

Implement an algorithm that finds the best solution you described in the last question (e) of the theory part. Achieve this by completing the *best_position* function in the Python file available at the Git repo `https://github.com/tsourolampis/bu-cs630-fall23`.