CS 344 Exam 1: Asymptotics, Divide and Conquer Algorithms

Due : (July 29, 2023 11:55 pm)

Please sign the honor pledge before working on the exam.

When giving an algorithm for any problem - please follow the format of first explicitly stating what the input and output of the algorithm are. The algorithm description should be *precise* and *self explanatory*. You may write comments to explain what each step does but have this information written separately.

*Make sure to **justify correctness** and analyze **time complexity** of each of your algorithms.*

1. (**Honor Pledge**) - I pledge on my honor that I have neither received nor given any help on this exam.
2. (**Comparing functions**) Indicate whether each of the following statements are true or false. **Justify**.
   (a) $2^{\sqrt{n}} = O((\sqrt{2})^n)$
   (b) $n^{\sqrt{n}} = O((\log n)^n)$
   (c) $n^{n!} = \Theta((n!)^n)$
3. (**Recurrence resolution**) Consider the following recurrence :
   $$T(n) = b^r T(\frac{n}{b}) + n^r \log n$$
   for some constants $b, r > 0$.
   (a) Use the Master theorem and come up with a polynomial upper bound for $T(n)$.
   (b) What value of d did you use in the previous part? Can that be improved to give a tighter upper bound for $T(n)$?
   (c) What is the tightest upper bound that Master theorem allows you to provide for $T(n)$?
   (d) Provide a lower bound for $T(n)$. (*hint : this part should take you lesser time than sneezing)*
   (e) Now solve the above recurrence precisely using a recurrence tree as follows :
   - Draw the recurrence tree. What is the cost at level i? (assume that the top level is labelled level 1)
   - How many levels does this recurrence tree have?
   - What is the cost at level i?
   - Add up all the costs across all the levels to get a Θ bound for $T(n)$.
4. (**Note picking**) Alguss is given an array of n currency notes (some of the note denominations may repeat). If they pick a particular denomination, they get to take home all notes of that denomination. The constraint is Alguss can only pick one denomination. Help Alguss select the denomination that maximizes their take home amount.
   (a) Provide an $O(n^2)$ algorithm to solve this problem.
   (b) Give a more efficient algorithm to solve this problem.

5. (**Function anomalies**) Let $f : \mathbb{Z} \to \mathbb{Z}$ be a strictly increasing function i.e.

$$\forall x, y \in \mathbb{Z} : \quad x > y \Rightarrow f(x) > f(y)$$

Consider an array A of n integers. We call an index pair (i, j) a *function anomaly* for f if

$$i < j \text{ and } f(a_i) > f(a_j)$$

We are interested in computing the number of function anomalies for f on a given array A.

(a) Calculate the number of function anomalies for the array A - [13 71 19 7 3 5].

(b) Based on the example above come up with a simple condition for when an index pair (i, j) is a function anomaly for f for a given array A.

(c) Use the condition above to provide an $O(n^2)$ algorithm that computes the number of function anomalies for f on a given array A.

(d) Use divide and conquer to come up with a more efficient algorithm that solves the above problem. (*Hint : There is an algorithm that runs in time $O(n \log n)$ but you will get credit for any algorithm that runs in time asymptotically better than $\Theta(n^2)$)*

6. (**k-Selects Sort**) Consider an array A of n integers. We will try to sort this array with the help of the SELECT subroutine as follows :

- On input A, use SELECT to pick out elements having ranks $\frac{n}{k}, \frac{2n}{k}, \frac{3n}{k}, \frac{4n}{k}, ..., \frac{kn}{k}$ respectively. (you may assume all the SELECT calls are made on the entire array A)
- Use the elements picked out in the previous step as pivots, partition the array A into k groups as follows :

$$\text{For } 1 \leq i \leq k : \quad B_i = \{a_j : a_j \in A \text{ and } (i-1)\frac{n}{k} \leq \text{rank}(a_j) < (i)\frac{n}{k}\}$$

- Apply Mergesort to sort each of the B_i's.

Answer the following questions :

(a) Prove/Disprove that the above algorithm (k-Selects sort) actually sorts a given input array A.

(b) Analyze the time complexity of the above algorithm. Your answer should be a function of k and n. (carefully analyze the time needed for each step individually)

(c) For what values of k would k-Selects sort be as efficient as mergesort?

7. (**Coconut Breaking**) Alguss lives in a building with n floors. They have a bunch of coconuts and want to find out which floors are safe to throw the coconuts from (and which are not). Assume all the coconuts behave similarly - that is, until a certain floor x satisfying $1 \leq x \leq n$ it is safe to throw the coconuts to the ground without breaking - whereas for any floor starting at $(x + 1)$ dropping a coconut from these "higher" floors results in it breaking. Answer the following questions :

(a) Provide an $O(n)$ time algorithm to find the earliest floor from which dropping a coconut results in it breaking.

(b) Provide a more efficient algorithm that finds the earliest floor from which dropping a coconut results in it breaking. Could Alguss execute this algorithm using a single coconut? Justify.

8. (**Extra Credit**) - Prove/Disprove : If $f(n), g(n), h(n)$ are all strictly increasing functions satisfying $f(n) = O(g(n))$ and $g(n) = O(h(n))$ then $f(g(n)) = O(f(h(n)))$.