CS570 Fall 2019: Analysis of Algorithms Exam I

	Points		Points
Problem 1	20	Problem 5	20
Problem 2	15	Problem 6	10
Problem 3	15	Problem 7	10
Problem 4	10		
	Total	**100**	

ANISH ADNANI

29/09/2021

Instructions:

1. This is a 2-hr exam. Closed book and notes
2. If a description to an algorithm or a proof is required please limit your description or proof to within 150 words, preferably not exceeding the space allotted for that question.
3. No space other than the pages in the exam booklet will be scanned for grading.
4. If you require an additional page for a question, you can use the extra page provided within this booklet. However please indicate clearly that you are continuing the solution on the additional page.
5. Do not detach any sheets from the booklet. Detached sheets will not be scanned.
6. If using a pencil to write the answers, make sure you apply enough pressure so your answers are readable in the scanned copy of your exam.
7. Do not write your answers in cursive scripts.

1) 20 pts

Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE/FALSE]
If path P is the shortest path from u to v and w is a node on the path, then the part of path P from u to w is also the shortest path between u to w.

[TRUE/FALSE]
Consider an alternate version of the interval scheduling problem where there is a positive reward for each job/interval. The following greedy algorithm always gets the maximum possible reward: Sort the jobs by reward and schedule them one by one starting with the highest reward, rejecting any that overlap with jobs already scheduled.

[TRUE/FALSE]
Dijkstra's algorithm may not terminate if the graph contains negative-weight edges.

[TRUE/FALSE]
If a data structure supports an operation 'X' such that a sequence of n 'X' operations takes $\Theta(nlogn)$ time to perform in the worst case, then the amortized time of the 'X' operation is $\Theta(logn)$, while the actual time of a single 'X' operation could be as high as $\Theta(nlogn)$.

[TRUE/FALSE]
In an unweighted graph where the shortest distance between any two vertices is at most T, any BFS tree has depth at most T, but a DFS tree might have a larger depth.

[TRUE/FALSE]
Starting with a node u in a graph G, run DFS and BFS to obtain search trees T and T' respectively. The number of children of u in T cannot be greater than the number of children of u in T'.

[TRUE/FALSE]
In class, we proved that a binary heap can be constructed in linear time by showing that the amortized cost of the insert operation in a binary heap is O(1).

[TRUE/FALSE]
Let T and T' be distinct Minimum Spanning Trees of a graph G. Then T and T' must have at least one common edge

[TRUE/FALSE]
Let G be a connected bipartite graph with n nodes and m edges. Then, m log m = $O(n^2 \log n)$

[TRUE/FALSE]
We know that algorithm A has a worst case running time of $O(nlogn)$ and algorithm B has a worst case running time of $O(n^2)$. It is possible for algorithm B to run faster than algorithm A when $n \to \infty$

2) 15 pts

Which of the following equalities are true and why?

a. $3n^2 + 6n = O(n^2 log n)$ TRUE

$6n^2 \log n$ $6n + 3n^2$

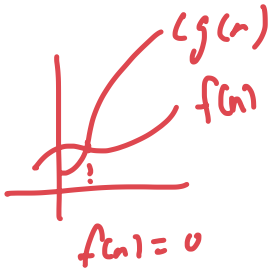
$2n^2 \log n$ $2n + n^2$

$n^2 \log n + n^2 \log n$ $2n + n^2$

b. $3^n = O(2^n)$

Not true

$2^n = 0$



c. $\log n = O((\log \log n)^4)$

FALSE

Logarithmic > Logarithmic(log)

d. $n = O((\log n)^{\log n})$ TRUE

$n = 64$

$\log_2 n = \log_2 64 = \log_2 2^6 = 6$

$64 = (6^6)$

e. $n^{100} = O(2^n)$

TRUE

3) 15 pts

Given a $n \times n$ matrix where each of the rows and columns are sorted in ascending order, find the k-th smallest element in the matrix using a min-heap data structure. You may assume that $k < n$. Your algorithm must run in time O(n + k log n).

we can consider each row as seperate list

To find K^{th} largest element

K:= 4th largest element

if A(i) < B(j)
i=i+1
if i+j = K
return i

if A(i) > B(j)
j=j+1
if i+j = = K
return j

[continued on page 9.]

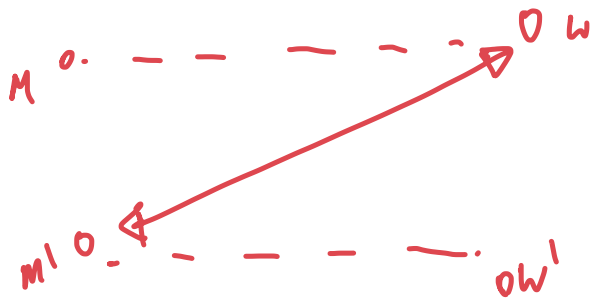
4) 10 pts

Prove or disprove the following statement:

For a given stable matching problem, if m_i and w_j appear as a pair when men propose and they also appear as a pair when women propose, then m_i and w_j must be paired in all possible stable matchings.

TRUE

let there be some stable matching



let m' be paired to w

IF m' prefers w' over w he want propose w unless he is rejected by w'

IF m has already proposed w
- They are matched and (w, m') cannot form

contradiction

IF m has not already proposed w
- m' propose w gets accepted
- But when m proposes w he gets accept

Hence again $m'w$ cant be paired

contradiction

5) 20 pts

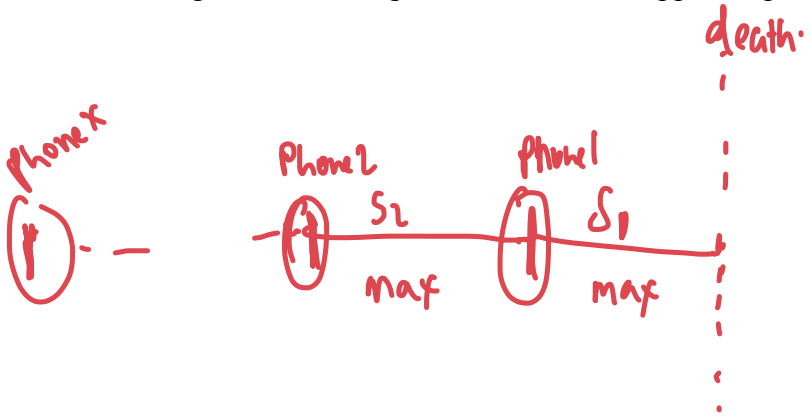
Tom is looking to buy a new smartphone, and is looking at the upcoming phone releases. Each phone i releases to the public at some time t_i and is given software support for some number of years s_i. Tom wants to buy as few phones as possible over the rest of his (unfortunately finite) lifetime. Assuming that we know the date of Tom's demise and all the phone release data until that time,

a) design an efficient algorithm to minimize the number of phones Tom needs to buy for the rest of his life while ensuring that he never goes without an unsupported phone. (10 pts)

ti → release year

Si → support in years

ti ⟶ ti+si



b) Prove the correctness of your solution. (10 pts)

Algo choose phone with greatest S_i sort T O(nlogn)

T = current-time phone = 0

while T < Tom's death:

O(n) { For x in range of all available phones at T:

max = maximum(Q) of all phones

T = T + Q

phone t = 1

O(nlogn)

return phone

6) 10 pts

Consider the divide and conquer solution described in class to find the closest pair of points in a 2D plane. Assume that we did not have a driver routine to sort the points. So our recursive function did not receive the points in sorted orders of their X and Y coordinates and the sorting had to be done for each subproblem (at every level). What would be the worst-case complexity of this algorithm assuming that the rest of the algorithm remains the same?

$$f(n) = C(n) + D(n)$$

$$= \Theta(n) + \Theta(n \log n)$$

$$= n \log n$$

$$n^{\log_b a} = n^{\log_2 2} = (n)^1$$

Case #2

$$O(n(\log n)^2)$$

7) 10 pts

a) Sam is trying to compute the complexity of merge-sort. In writing the recurrence relation, he erroneously considers the complexity of the merging step to be $O(n^2)$ instead of $O(n)$. Assuming no other mistakes, what is the complexity of merge-sort he ends up with? (5 points)

$f(n) = 2f(n) + O(n)$

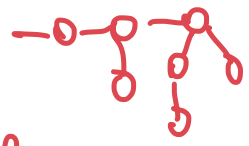
$= O(n^2) + 1$

$= O(n^2)$

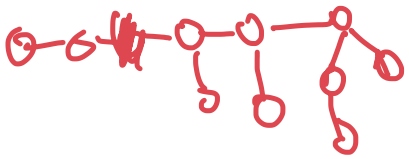
$O(n^2)$

b) Show that the number of nodes in the highest-order binomial tree in a binomial heap with n elements is $\Theta(n)$. (5 points)

7 elem



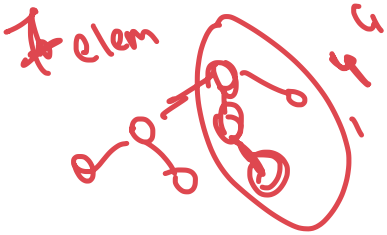
10 elements



$h^{\log_b a} = n^{\log_2 2} = n$

1 2 3 4 5 6 7 8 9

elem



$1 + 2^0 + 2^1 + 2^2 \ldots < O(n^2)$

$\frac{n(n+1)}{2}$ S_n

$\frac{k(2^k-1)}{2-1} = n$

$(2^0 + 2^1 + 2^2 + \ldots + 2^k) = n$ # Case 3

$O(n^2)$

$2^k = n+1$

$k = \log_2(n+1)$

$2^0 + 2^1 + \ldots + 2^k = n$

$\frac{1(2^{k+1}-1)}{(2-1)} = n$

$2^{k+1} = n+1$

$2^k = \left(\frac{n+1}{2}\right)$

$\therefore$ last order tree has atleast $(n/2)$ nodes

$O(n)$

Additional Space

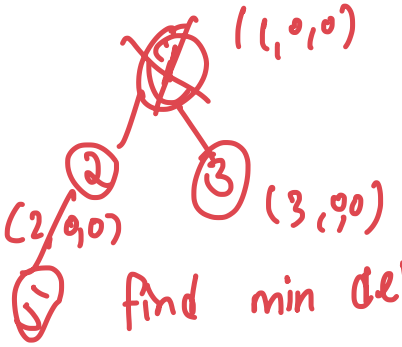
---

(3)

So rather than 2 arrays

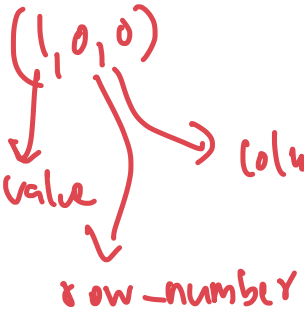
we have 'n' arrays

insert 1st element from all rows in min heap



find min delete min

new element



∴ Complexity

O(n) → min heap initialize

k * log(n)

looping k times

insertion (heapify)

do this 'k' times

O(n + k log(n))

Additional Space