Q1 18

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(n log n) 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 $n \to \infty$

Q2 13

2) 15 pts
Which of the following equalities are true and why?

a. $3n^2 + 6n = O(n^2 logn)$ assume the base is 2 for logarithmic here

True. $3n^2+6n \le cn^2\log_2 n$, we can find constants $c=4$, and $n_0=6$ when $n>n_0$, $cn^2\log_2 n = 4n^2\log_2 n > 4n^2 \ge 3n^2+6n$ is always true. Therefore, $3n^2+6n = O(n^2\log n)$ is true.

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

False. Since we can never find positive constants c and n_0 such that $3^n \le c2^n$ is true for all $n>n_0$. $3^n = O(2^n)$ is not true.

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

False. Since we cannot find positive constants c and n_0 such that $\log n \le c(\log\log n)^4$ is true for all $n>n_0$, $\log n = O((\log\log n)^4)$ is not true

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

True. We can find very large positive constants c and n_0, when $n>n_0$, $n \le c(\log n)^{\log n}$ is always true.
Therefore, $n = O((\log n)^{\log n})$ is true

-1 for incomplete reasoning 2

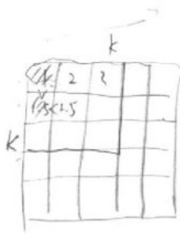
e. $n^{100} = O(2^n)$

True. We can find very large positive constants $C = C_0$ and n_0, when $n>n_0$, $n^{100} \le C_0 2^n$ is always true.
Therefore $n^{100} = O(2^n)$ is true

-1 for incomplete reasoning 2

Q3 0

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 $O(n + k \log n)$.



First create a min-heap of size k using the first row of the matrix This takes O(n).

Q4 10

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.

Here I need to assume when men or women propose, it uses G-S algorithm to get the pair of (m_i, w_j), then the statement is true. However, if it is not, we can only know (m_i, w_j) are just valid partners to each other.

From the lecture, we already know in G-S Alg., when men proposing, men will always end up with their best valid partner and women with their worst valid partner. While when women proposing, women will always end up with their best valid partner and men with their worst valid partner

Here, for m_i, w_j is both his best and worst valid partner, and for w_j, so does m_i. Then m_i, w_j only have one valid partner. Therefore, m_i and w_j must be paired in all possible stable matchings.

Q5 18

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)

Buy a phone and take it as long as possible until Tom has to buy a new one before his current phone reaching its unsupported time.

Repeat this until Tom's demise.

"As long as possible" is vague, use the numbers in the text **8**

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

See more on Page 9!

Assume there is an optimal solution. Firstly we can use mathematical induction to prove that for every new phone, the time it bought in our solution is no earlier than the optimal solution.

Right **10**

Base case: for the first phone, every solution need to choose this one to start. It is true that our solution is no earlier than the optimal one for the first phone

$t_k \geq t_k'$

Induction step: Assume for the k^{th} phone. Our solution is no earlier than the optimal one. Then for the $k+1^{st}$ phone, if the optimal solution choose it later than our solution, since the k^{th} phone is no later than our solution, there must be a gap between $t_k' + s_k'$ and t_{k+1}' because t_{k+1} is the last one in $t_k + S_k$ and $t_k' \leq t_k$, $t_{k+1}' < t_{k+1}$

Therefore. Our solution is always no earlier than the optimal solution

Now let's look at the last phone Tom will buy. Since this phone is also no earlier than optimal solution, and it's impossible that the optimal solution will have closes-pair-rec(X_r)

Q6 5

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?

Now for each step in Divide, we have to perform the sorting of points on their X coordinates to divide the current problem to its subproblem

$\delta = \min(\delta_1, \delta_2)$

So $D(n) = \Theta(n \log n)$.

For combine step. we need to sort the points on their Y coordinates and then compare each one to its next 11 points. The sorting takes $\Theta(n \log n)$ and the rest of compare still takes linear time

Therefore the $f(n) = \Theta(n \log n)$.

This time the recurrence equation is $T(n) = 2T(n/2) + f(n)$

$f(n) = \Omega(n^{\log_b a + \epsilon})$ for some $\epsilon > 0$

this is case 3 of Master Theorem. we can get

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

$f(n)$ $f(n/2)$ $f(n/2)$

$O(n)$ cn $n\log n$

Q7 5

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)

$f(n/2)$ $f(n/2)$

The recurrence equation that Sam gets is $T(n) = 2T(n/2) + O(n^2)$

rather than $T(n) = 2T(n/2) + O(n)$.

According to the Master Method, $a=2$, $b=2$, $f(n) = O(n^2)$,

$f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, this is case 3.

$T(n) = \Theta(n^2)$

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)

Firstly we need to know the order number of the highest-order binomial tree.

Additional Space

one more phone because if that is true, our solution will also has that phone.

Therefore, our solution has the same number of phone as the optimal solution Then our solution is optimal

Additional Space