CS 2223 D-Term 2020 MIDTERM EXAM Name:

Friday, April 17, 2020

Question 1 *(1 points)*
REQUIRED: Affirmation

“My brain is open. . . .”

I pledge that I am taking this exam on my own, with help from no one else and only the one-page note sheet we are officially permitted.

Question 2 *(10 points)*
Big O TWO PARTS!

PART 1: We showed MERGESORT to be $\Theta(n \lg n)$.

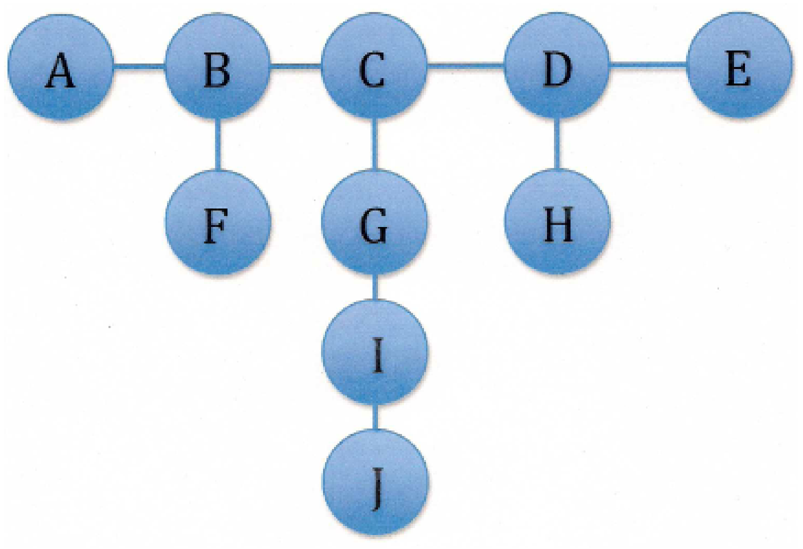
What does that mean? Big theta means the the best and worst case scenario are the same. So in the case of mergesort it will always run in nlogn time.

PART 2: We have also found that QUICKSORT is both $O(n^2)$ and $\Omega(n \log n)$.

Explain why no algorithm can be both $\Omega(n^2)$ and $O(n \log n)$.

No algorithm can be both big Omega n^2 and Big O n log n because that would mean that the best case scenario is worse than the worst case scenario. n^2 is slower than n log n.

Question 3 *(15 points)*
Graph Searching starting at Vertex H.



THREE PARTS!
(Breadth-First Search / Depth-First Search Push Order / Depth-First Search Pop Order)

PART 1:
Perform a traversal of the graph above by executing a Breadth-First Search algorithm, starting at **vertex** H.
Report the vertices in the order they are discovered.

H,D,E,C,B,A,G,F,I,J

PART 2:
Perform a traversal of the graph above by executing a Depth-First Search algorithm, starting at **vertex** H.
Report the vertices in the order they are discovered, i. e. "push" order.

H,D,E,C,G,I,J,B,F,A

PART 3:
For the Depth-First search in Part 2 above also record the vertices in the order they are discarded, i. e. "pop" order.

H,E,D,J,I,G,C,F,B,A

CLEARLY LABEL ALL THREE PARTS IN THE ORDER REQUESTED:

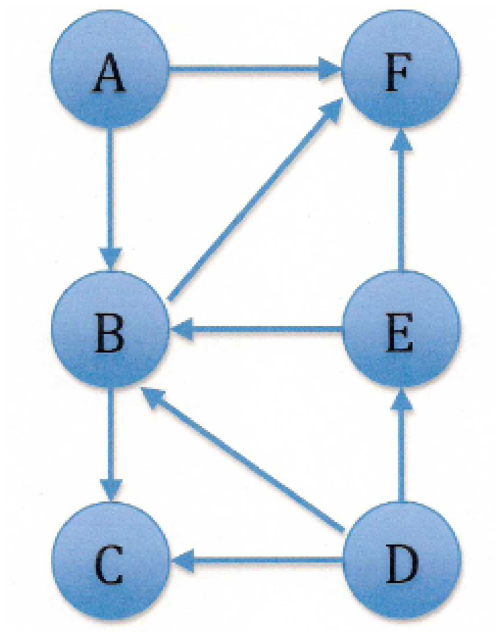
BFS order of traversal:

DFS push order of traversal:

DFS pop order of recording:

Be certain to start your algorithms at **Vertex H**.

Question 4 *(10 points)*
Topological Sort
Perform a Topological Sort of the graph below.



Report the vertices in a linear ordering that preserves the notion of "precedes" suggested by the arrows in the diagram.

A,D,E,B,F,C

Question 5 *(15 points)*
Search algorithm time complexity

We have seen that our Breadth-First and Depth-First Search graph algorithms can be implemented using adjacency lists with $O(|E| + |V|)$ order of growth and with adjacency matrices with $O(|V|^2)$ order of growth.

When is one of them better than the other? When are they both essentially the same? Which should be used with trees (acyclic graphs)? Why?

Question 6 *(15 points)*
QUICKSORT

It is sometimes beneficial to randomize data before sorting it with QUICKSORT.

Explain. . .

(1) under what circumstances this might be advantageous, . . .

(2) what restrictions must/should apply to any randomizing algorithm implemented for this purpose, . . .

and . . .

(3) how it is that possible that invoking an additional procedure–randomization in this case–can actually improve the efficiency of the overall sorting process.

Question 7 *(15 points)*
Heaps

A priority queue can be used to sort a sequence.

If the priority queue is implemented with a heap, we get HEAPSORT which is $\Theta(n \log n)$.

Alternatively, if the priority queue is implemented with a linked list, the algorithm will be $O(n^2)$ but also $\Omega(n)$.

(5 points) Under what conditions on the input sequence is the linked list better? Why?

(5 points) Under what conditions on the input sequence is the linked list worse? Why?

(5 points) Why does HEAPSORT exhibit $\Theta(n \log n)$ order of growth in both those cases?

Be certain to answer all three components of this question.

Question 8 *(10 points)*
Dynamic Programming / Coin Row

Use a dynamic programming algorithm to solve the coin row problem (maximize the sum without taking adjacent coins) for the following arrangement of coins:

4	2	1	6	10	9	3	7	8	5	11

What is the maximum sum achievable from non-adjacent coins? 38

Which coins are taken to create this sum? 4,6,8,9,11

Question 9 *(15 points)*
Recurrence Relations

The minimum number of moves required to solve the n-disc Tower of Hanoi, $T(n)$, is given by the recurrence relation:

$$T(n) = 2T(n-1) + 1 \text{ for } n > 1 \text{ and } T(1) = 1$$

Explain the relationship. That is, how does the recurrence arise from (optimal) solving?

The minimum number of moves required to solve the n-disc Tower of Hanoi, $D(n)$, is also given by this recurrence relation:

$$D(n) = 4D(n-2) + 3 \text{ for } n > 2 \text{ and } D(2) = 3, D(1) = 1$$

Similarly, explain this relationship.