/ 83

Name(last, first): ________________

20 20 20 20 3

UCLA Computer Science Department

CS 180 **Algorithms & Complexity** **ID (4 digit): ____**

Midterm **Total Time: 1.5 hours** **October 28, 2013**

(each problem has 20 points)

20

1 **a.** Describe Topological sort algorithm in a directed acyclic graph (DAG) (English bullet by bullet). **(10 pts)**
b. Analyze its time complexity. **(5 pts)**
c. Prove its correctness. **(5 pts)**
i) Show that given a DAG, your algorithm will output a Topological sort.
ii) Show that if your graph is not a DAG, then your algorithm will not find a Topological sort.

A.
- Choose any node with no incoming edges.
- Append this node to the topological ordering.
- Remove this node from the DAG, along with its outgoing edges.
- Repeat the first three bullets in order until there are no more nodes in the DAG.

B. Maintain a set S of all active nodes with no incoming edges from other active nodes. O(n), since each node will be added to S exactly once.
Maintain the number of incoming edges for each node. O(e+e), since each edge will be added exactly once, then removed exactly once.
Total time: O(n+m)

C. i) Assume the algorithm outputs a topological sort where v_j is output before v_i, and there is an outgoing edge from v_i to v_j. Then v_j is not a source, and it would not have been output by the algorithm before v_i, so the algorithm must output a topological sort, given a DAG

ii) If the graph is not a DAG, it must have a cycle. If there is a cycle, there is no source in the cycle, since each vertex in the cycle has an incoming edge. The algorithm will not find a topological sort for this cycle part of the graph, so if the graph is not a DAG, the algorithm will not find a topological sort.

Name(last, first): ______,__________

2 **a.** We are given a set of activities $I_1 \ldots I_n$: each activity I_i is represented by its left-point L_i and its right-point R_i. Design a very efficient algorithm that finds the maximum number of mutually overlapping subset of activities. (In the example below the answer is 2 as indicated by the dotted vertical line). Write your solution in English, bullet by bullet. **(15pts)**

20

[redacted]

b. Analyze the time complexity of your algorithm. **(5pts)**

A. • Sort all the left- and right-points so that $L_1 \le L_2 \ldots \le L_n$ and $R_1 \le R_2 \ldots \le R_n$, where L_i corresponds to R_i and the sorted list is of the form : $L_1 \ldots \le R_1 \ldots \le L_n \le \ldots \le R_n$

- Keep a count of the number of mutually overlapping activities in OVERLAP, and a count of the maximum number in MAX.
- Start at L_1 and set OVERLAP = 1 and MAX = 1.
- Look at the next entry. If it is any L_i, add one to OVERLAP. If it is any R_i, subtract one from OVERLAP.
- See if OVERLAP > MAX. If it is, set MAX = OVERLAP.
- Repeat the above two bullets until R_n has been looked at.

B. Sorting $O(2n\log(2n))$

Going through the list $O(2n)$

Time: $O(n\log n)$

Name(last, first): ____________

20

3. Suppose that you are given **n** red and **n** blue water jars, all of different shape and sizes. Every red jar potentially holds a different amount of water (has a different capacity) than all other red jars. Similarly, every blue jar potentially holds a different amount of water (has a different capacity) than all other blue jars. For each red jar, there is a blue jar that has the same capacity.

Group the jars into pairs of red and blue that can hold the same amount of water. You can do the following **basic operation**: pick a red jar, fill it with water, and pour into a blue jar. This will tell you if the red jar has more, less, or same amount of water as the blue jar.

a. Describe an **$O(n^2)$** algorithm for solving the problem. **(15pts)**

b. Explain why your algorithm is $O(n^2)$. **(5pts)**

A. While there is a red jar that has not been paired with a blue jar
Choose any unpaired red jar, R
While R has not been paired
Fill R with water
Pour R into any blue jar, B, that has not been paired and has never had water poured into by R
If B holds the same amount of water as R,
Pair R with B.
Pour the water out of B.

B. This algorithm is $O(n^2)$ because for each red jar, you test it against a possible max of n blue jars, and there are n red jars to test, so $n \cdot n = n^2$.

20

Name(last, first): ________________

4. Consider a DAG (directed acyclic graph) with the longest path having k edges in it.

a. Design an algorithm that partitions the vertices into exactly k+1 groups such that there are no edges between any two vertices in the same group. **(10pts)**

b. Prove the correctness of your algorithm. **(10pts)**

A. Let $I = 0$.
While there are nodes in the DAG,
Take all the sources and put them in partition I.
✓ Remove these sources and their outgoing edges from the DAG.
Increment I by 1.

B. Prove there are no edges between vertizes in a partition.
By the algorithm, each partition contains only sources. These sources have no incoming edges, only outgoing edges.
if two vertices A and B were in the same partition and had an edge between them, then one of the vertices, say A, must have an incoming edge from B. However, this means A is not a source and should not be in the same partition as B. This is a contradiction, ✓ so there are no edges between vertices in a partition.

Prove there are k+1 groups: ✓
In the path with k edges, there are k+1 nodes starting from v_1 to v_{k+1}.
v_1 must be a source, otherwise there is some other v_n which comes before v_1 in the path, where v_n is the start of the longest path in the DAG.
Each iteration of the while loop removes all the sources from the DAG, so v_2 will then be a source. If v_2 is not a source, then there must be at least two v_n and v_m come before it in a path, which contradicts that v_1 is the start of the path, so v_2 is a source. The same goes for all the other $v_3, \dots, v_{k+1}$ in the longest path, so each v_{i+1} becomes a source when v_i is removed from the DAG. By the algorithm, each $v_1, \dots, v_{k+1}$ will be in partitions $1, \dots, k+1$, and there cannot be fewer or more partitions because

that would contradict $v_1, \ldots, v_{k+1}$ being the longest path, so the algorithm is correct.

Name(last, first): ______,______

3

5. Consider a sorted sequence $a_1, \ldots, a_n$ of distinct integers. Design an efficient algorithm that decide whether there is an integer a_i such that $a_i = i$ (for example, if the sequence is -1, 3, 4, 5, 7, 9 then the answer is NO. if the sequence is –1,2,4,5,7,8 the answer is yes for i=2) – note that an O(n) time algorithm would be trivial.

a. Describe your algorithm in English bullet-by-bullet. **(15pts)**

b. Analyze the time complexity of your algorithm. **(5pts)**

A. • Look through them from a_1 to a_n and see if there is an a_i such that $a_i = i$

linear : -13

B. O(n) why? -4