CSE317 Design & Analysis of Algorithms

Institute of Business Administration Karachi
Leadership and Ideas for Tomorrow

Final Examination

Max Marks: 50 Duration: 3 hours

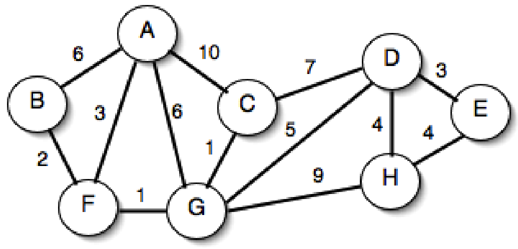
IBA FCS Faculty of Computer Science

Answer all questions. Use seperate answer sheet. Be to the point. Show your work.
Please give clear and rigorous answers.

Name: ____________________ **ERP:** __________

Question 1: Greedy Algorithms .... *8 marks*

(a) [4 marks] Consider the weighted graph below.



i. Run Prim's algorithm starting from vertex A. Write the edges in the order which they are added to the minimum spanning tree.

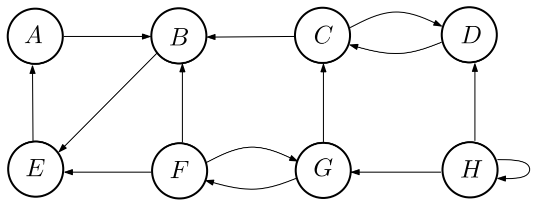
ii. Run Kruskal's algorithm. Write the edges in the order which they are added to the minimum spanning tree.

(b) [4 marks] Let's consider a long, quiet country road with houses scattered very sparsely along it. (We can picture the road as a long line segment, with an eastern endpoint and a western endpoint.) Further, let's suppose that despite the bucolic setting, the residents of all these houses are avid cell phone users. You want to place cell phone base stations at certain points along the road, so that every house is within four kilometers of one of the base stations.

Give an efficient algorithm that achieves this goal, using as few base stations as possible.

Question 2: Graphs .... *8 marks*

(a) [4 marks] Run the strongly connected components algorithm on the following directed graph G. When doing DFS on G^R: whenever there is a choice of vertices to explore, always pick the one that is alphabetically first.



i. In what order the strongly connected components (SCCs) found?

ii. Whcih are sources SCCs and which are sink SCCs?

iii. Draw the "metagraph" (each meta-node is an SCC of G).

(b) [4 marks] Vertex r is a root of digraph (directed graph) $G = (V, E)$, if there exists a path from r to v for each $v \in V$. (Note that a digraph may have several roots, or none at all.) Design an algorithm that finds a root in a digraph G, represented as an adjacency list, in time $O(|V| + |E|)$, or determines that none exist. (Hint: A constant number of DFS passes over G suffices.)

Question 3: Analysis .... *8 marks*

(a) [4 marks] For each pair of expressions $T(n)$ and $f(n)$ below, indicate whether $T(n)$ is O, Ω, or Θ of $f(n)$. Give one-line justification for each answer.

$T(n)$	$f(n)$	$T \overset{?}{=} O(f)$	$T \overset{?}{=} \Omega(f)$	$T \overset{?}{=} \Theta(f)$
$n^{1.01}$	$n \log^2 n$			
$\sqrt{n}$	$\log^3 n$			
$n2^n$	3^n			
$n!$	2^n			

(b) [4 marks] Give formal definitions of Define Big-O, Big-Omega, and Big-Theta notations.

Question 4: Divide & Conquer .... *8 marks*

(a) [4 marks] Suppose you are choosing between the following two algorithms:

i. Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.

ii. Algorithm B solves problems of size n by recursively solving two subproblems of size $n - 1$ and then combining the solutions in constant time.

What are the running times of each of these algorithms (in big-O notation as a function of n), and which would you choose?

(b) [4 marks] Suppose we are given an array $A[1..n]$ with the special property that $A[1] \geq A[2]$ and $A[n-1] \leq A[n]$. We say that an element $A[x]$ is a local minimum if it is less than or equal to both its neighbors, or more formally, if $A[x-1] \geq A[x]$ and $A[x] \leq A[x+1]$. For example, there are six local minima (circled) in the following array:

$$9 \quad \textcircled{7} \quad 7 \quad 2 \quad \textcircled{1} \quad 3 \quad 7 \quad 5 \quad \textcircled{4} \quad 7 \quad \textcircled{3} \quad \textcircled{3} \quad 4 \quad 8 \quad \textcircled{6} \quad 9$$

We can obviously find a local minimum in $O(n)$ time by scanning through the array. Describe and analyze an algorithm that finds a local minimum in $O(\log n)$ time. (Hint: With the given boundary conditions, the array must have at least one local minimum. Why?)

Question 5: Dynamic Programming .... *10 marks*

Suppose you're managing a consulting team of expert computer hackers, and each week you have to choose a job for them to undertake. Now, as you can well imagine, the set of possible jobs is divided into those that are *low-stress* (e.g., setting up a Web site for a class at the local elementary school) and those that are *high-stress* (e.g., protecting the nation's most valuable secrets, or helping a desperate group of IBA students finish FYP project one week before the symposium). The basic question, each week, is whether to take on a low-stress job or a high-stress job.

If you select a low-stress job for your team in week i, then you get a revenue of $l_i > 0$ dollars; if you select a high-stress job, you get a revenue of $h_i > 0$ dollars. The catch, however, is that in order for the team to take on a high-stress job in week i, it's required that they do no job (of either type) in week $i - 1$; they need a full week of prep time to get ready for the crushing stress level. On the other hand, it's okay for them to take a low-stress job in week i even if they have done a job (of either type) in week $i - 1$.

So, given a sequence of n weeks, a *plan* is specified by a choice of "low-stress," "high-stress," or "none" for each of the n weeks, with the property that if "high-stress" is chosen for week $i > 1$, then "none" has to be chosen for week $i - 1$. (It's okay to choose a high-stress job in week 1.) The *value* of the plan is determined in the natural way: for each i, you add l_i to the value if you choose "low-stress" in week i, and you add h_i to the value if you choose "high-stress" in week i. (You add 0 if you choose "none" in week i.)

The problem. Given sets of values $l_1, l_2, \ldots, l_n$ and $h_1, h_2, \ldots, h_n$, find a plan of maximum value. (Such a plan will be called *optimal.*)

Example. Suppose $n = 4$, and the values of l_i and h_i are given by the following table. Then the plan of maximum value would be to choose "none" in week 1, a high-stress job in week 2, and low-stress jobs in weeks 3 and 4. The value of this plan would be $0 + 50 + 10 + 10 = 70$.

	Week 1	**Week 2**	**Week 3**	**Week 4**
l	10	1	10	10
h	5	50	5	1

(a) [5 marks] Show that the following algorithm does not correctly solve this problem, by giving an instance on which it does not return the correct answer.

```
$i = 1$
while $i \leq n$ do
    if $h_{i+1} > l_i + l_{i+1}$ then
        Output "Choose no job in week $i$"
        Output "Choose a high-stress job in week $i + 1$"
        $i = i + 2$
    else
        Output "Choose a low-stress job in week $i$"
        $i = i + 1$
```

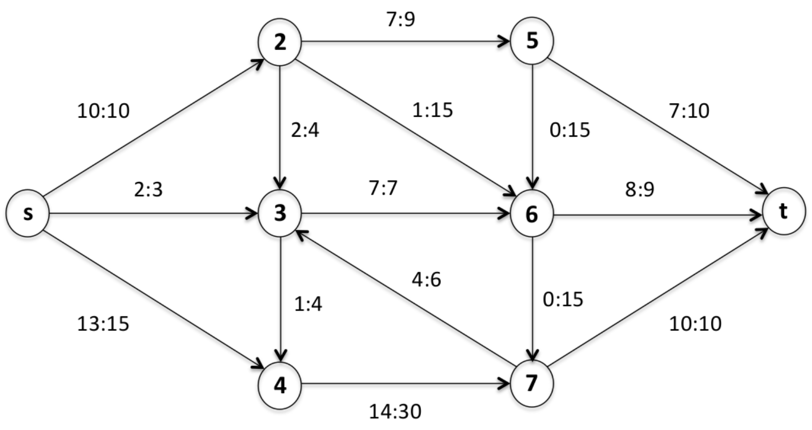
To avoid problems with overflowing array bounds, we define $h_i = l_i = 0$ when $i > n$.

In your example, say what the correct answer is and also what the above algorithm finds.

(b) [5 marks] Give an efficient algorithm that takes values for $l_1, l_2, \dots, l_n$ and $h_1, h_2, \dots, h_n$ and returns the value of an optimal plan.

Question 6: Network Flows .... *8 marks*

Consider the following flow network G and initial flow f_{init}. We will continue with the Ford-Fulkerson algorithm to find a maximum flow in G.



(a) [4 marks] Compute max flow f in G. List all the augmenting s-t paths you encountered.

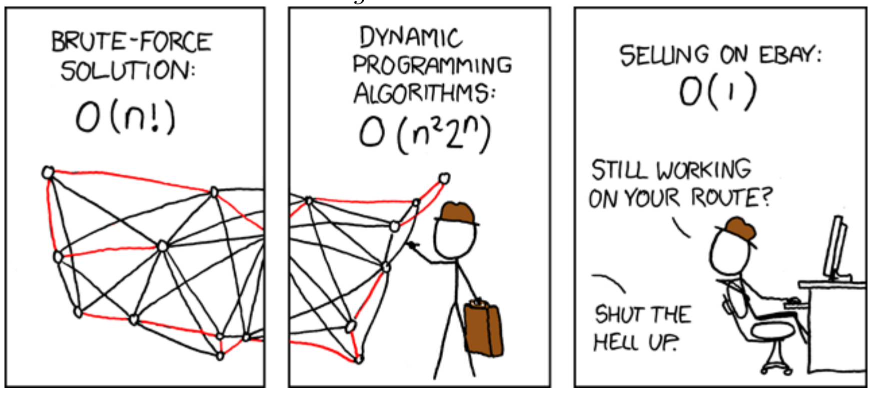
(b) [1 mark] What is the size of max flow f?.

(c) [1 mark] Draw the final residual graph G_f of G for max flow f.

(d) [2 marks] Find a cut in G with size equals the size of flow f found in part (a).

******* end of exam *******

Traveling Salesman Problem



BRUTE-FORCE SOLUTION: O(n!)
DYNAMIC PROGRAMMING ALGORITHMS: O(n²2ⁿ)
SELLING ON EBAY: O(1)
STILL WORKING ON YOUR ROUTE?
SHUT THE HELL UP.