CSE317 Design & and Analysis of Algorithms

Final Examination – Spring'15

Max Marks: 70 Time Allowed: 3 hours

Answer the questions in the spaces provided on the question sheets.
If you run out of room for an answer, continue on the provided extra sheet.
Attempt all questions. Be to the point. Show your work.

1. (a) [7 marks] Suppose you are choosing between the following three algorithms:
   - Algorithm A solves problems of size n by dividing them into three subproblems of half the size, recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.
   - Algorithm B solves problems of size n by recursively solving a subproblem of size $n-1$ and then combining the solution in $O(\log n)$ time.
   - Algorithm C solves problems of size n by dividing them into six subproblems of size $n/3$, recursively solving each subproblem, and then combining the solutions in $O(n)$ time.

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

(b) [4 marks] For each pair of expressions (A, B) below, indicate whether A is O, o, Ω, ω, or Θ of B. Your answer should be in the form of table with "yes" or "no" written in each box.

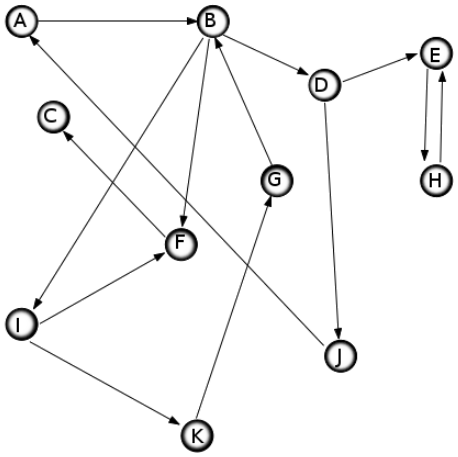
	A	B	O	o	Ω	ω	Θ
(a)	$\log(n!)$	$n \log n$					
(b)	$(n^2 - n)/2$	$6n$					
(c)	$\sqrt{n}$	$n^{\cos(\pi n/4)}$					
(d)	$n\sqrt{n}/2$	$n \log n$					

(c) [4 marks] Consider an array $A[1 \ldots n]$ constructed by the following process: we start with n distinct elements, sort them, and then rotate the array k steps to the right. For example, we might start with the sorted array $[1, 4, 5, 9, 10]$, and rotate it right by $k = 3$ steps to get $[5, 9, 10, 1, 4]$. Give an $O(\log n)$-time algorithm that finds and returns the position of the minimum element in array A. (Your algorithm is given the array $A[1 \ldots n]$ but does not know k.)

2. (a) [2 marks] Draw the graph with the following adjacency matrix:

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

(b) [5 marks] Run the strongly connected components algorithm on the following directed graph G. When doing DFS on G_{rev}: whenever there is a choice of vertices to explore, always pick the one that is alphabetically first. In what order are the strongly connected components (SCCs) found? Draw the metagraph (each meta-node is an SCC of G).



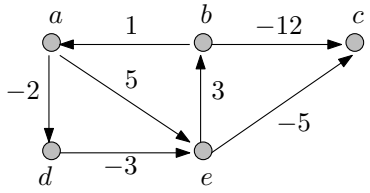
(c) [3 marks] The police department in the city of Computopia has made all streets one-way. The mayor contends that there is still a way to drive legally from any intersection in the city to any other intersection, but the opposition is not convinced. A computer program is needed to determine whether the mayor is right. However, the city elections are coming up soon, and there is just enough time to run a linear-time algorithm. Formulate this problem graph-theoretically, and explain why it can indeed be solved in linear time.

3. (a) [3 marks] What is the probability that given two elements of the input array will be compared during the execution of randomized quicksort?

   (b) [4 marks] In the deterministic linear-time algorithm for selecting the ith largest element, we grouped elements into sets of 5 each (except possibly the last one which may contain fewer). Analyse the algorithm if we grouped them into sets of 3. Give the recurrence for this case and show that the same proof as discussed in the lecture doesn't work.

   (c) [3 marks] Suppose m keys are chosen randomly and the hash function distributes the keys uniformly at random over buckets $\{0, 1, \ldots, n-1\}$. What is the probability that all m keys hash to the first bucket (bucket number 0)?

4. (a) [6 marks] Run the Bellman-Ford shortest path algorithm on the following graph, starting from vertex a. Specifically, fill in the tables below.



	0	1	2	3	4
a	0				
b	∞				
c	∞				
d	∞				
e	∞				

vertex	a	b	c	d	e
prev[vertex]					

(b) [4 marks] Suppose you are given a directed graph that has negative values on some of the edges. The Bellman Ford algorithm will give the correct solution for shortest paths from some starting vertex s, if there are no negative cycles. But suppose you don't know if the given graph has negative cycles. One way to check for negative cycles is to run Bellman Ford for n iterations, instead of $n-1$. Explain.

5. (a) [5 marks] Suppose that a graph has **distinct** edge weights. Does its shortest edge have to belong to the Minimum Spanning Tree (MST)? Can its longest edge belong to the MST? Does a min-weight edge on every cycle have to belong to the MST? Prove your answer to each question or give a counterexample.

(b) [1 mark] Suppose we have an undirected graph with weights that can be either positive or negative. Do Prims and Kruskals algorithim produce a MST for such a graph?

(c) [4 marks] A server has n customers waiting to be served. The service time required by each customer is known in advance: it is t_i minutes for customer i. So if, for example, the customers are served in order of increasing i, then the i^{th} customer has to wait $\sum_{j=1}^{i} t_j$ minutes. We wish to minimize the total waiting time

$$T = \sum_{i=1}^{n} (\text{time spent waiting by customer } i)$$

Describe a greedy algorithm for the problem and show that the obtained solution is always optimal. *Hint:* Use exchange argument.

6. [5 marks] One form of the knapsack problem is as follows: We are given a set of integers $A = \{a_1, a_2, \ldots, a_n\}$ and an integer K. Is there a subset of A whose sum is exactly K?

   Let $F[i, k] = 1$ if there is subset of first i items whose sum is exactly k (otherwise $F[i, k] = 0$). Note that $F[i, 0]$ is always 1. Write a recurrence to compute $F[i, k]$.

7. The *hopping game* is played on row of cells. A player initially stands on the left-most cell and by performing sequence of jumps arrives at the right most cell. Each jump is from left to right and cannot be more cells away than the number written in the current cell. The goal is to *minimize* the number of jumps to reach the right-most cell.

   Following is an example.

1	3	5	8	4	2	6	7	3	1	0

   In the above example, the player initially stands at Cell #1. As the number 1 is written on the current cell, he can jump at most 1 hop, i.e., to Cell #2. At Cell #2 the number 3 is written, so he can jump at most 3 hops (to Cell #3, #4 or #5). Optimally, she can reach the right-most cell by performing 3 jumps (Cell #1 $\rightarrow$ Cell #2 $\rightarrow$ Cell #4 $\rightarrow$ Cell #11).

   (a) [3 marks] Show by giving an example that the strategy of taking the longest jump possible at each cell does not minimize the number of jumps.

   (b) [7 marks] Use dynamic programming to find minimum number of jumps. For each $0 \leq i \leq n$, let $J(i)$ is defined to be minimum numbers of jumps to reach Cell #i starting from Cell #1. Write the recurrence for $J(i)$, What is the running time of the resulting algorithm?