CSE317 Design & and Analysis of Algorithms

Final Examination – Fall'15

Max Marks: 99 Time Allowed: 3 hours

Attempt all questions. Be to the point. Show your work.

1. [20 marks] For each of the following statements, decide whether it is true or false. Give brief justification of your answer. (1 mark for each correct answer and 1 mark for each explanation)

   (a) $2^{n+1} = O(2^n)$

   (b) Consider $f(n) = \log\log n$ and $g(n) = 10^{10^{10^{10^{10}}}}$. Then $f(n)$ is $O(g(n))$.

   (c) Kruskal's algorithm works if there are negative edge weights in the input graph.

   (d) For any graph and a source vertex, there is a unique BFS tree for it.

   (e) Every computational problem on input size n can be solved by an algorithm running on super-computer with running time polynomial in n.

   (f) Let G be a graph with a negative cycle. Then there is no pair of vertices that has a finite cost shortest path.

   (g) Given n integers $a_1, \ldots, a_n$, the third smallest number among $a_1, \ldots, a_n$ can be computed in $O(n)$ time.

   (h) If $T_1(n) = O(f(n))$ and $T_2(n) = O(f(n))$, then $T_1(n) = O(T_2(n))$.

   (i) The number of comparisons performed by binary search algorithm in worst case is $\Omega(n \log n)$.

   (j) If a graph has a unique *source* and a unique *sink* then it must be directed acyclic.

2. (a) [3 marks] Algorithms A, B, C, D, E, F, G each solves problem P. For an input of size n,

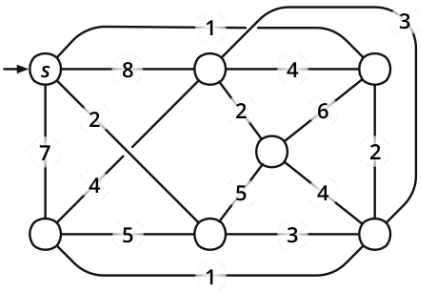
   Algorithm A executes n^2 operations,
   Algorithm B executes 2^n operations,
   Algorithm C executes $\log\log(n)$ operations,
   Algorithm D executes n operations,
   Algorithm E executes $\log(n)$ operations,
   Algorithm F executes $n!$ operations, and
   Algorithm G executes $n\log(n)$ operations.

   Rank the algorithms from the fastest to the slowest.

   (b) [4 marks] Show that the recurrence $T(n) = 2 \cdot T(n/2) + c \cdot n$, where c is a constant, solves to $T(n) = \Theta(n \log n)$ without using Master theorem.

3. *Graph Algorithms*

   (a) [6 marks] Clearly indicate the following trees in the weighted graph pictured below. Some of these subproblems have more than one correct answer. (You do not need to show the intermediate steps, just give the required trees)



   i. A BFS tree rooted at s

   ii. A shortest-path tree rooted at s

   iii. A minimum spanning tree

   (b) [6 marks] You are given a strongly connected directed graph $G = (V, E)$ with positive edge weights along with a particular node $v_0 \in V$. Give an efficient algorithm for finding shortest paths between all pairs of nodes, with the one restriction that these paths must all pass through v_0.

4. Consider the set of intervals $\{(1,3), (2,4), (3,6), (6,7), (5,12), (8,12), (11,13)\}$ in which each interval has a start time and a finish time. These intervals for instance represent jobs for a machine and we are interested in finding maximum cardinality subset of jobs such that no two jobs in the subset are intersecting.

   (a) [6 marks] Find a subset of mucompatible intervals using the following greedy strategies.

   i. Shortest job first

   ii. Minimum conflict jobs first

   iii. Latest start time first

   (b) [4 marks] We have seen in the lecture that greedy algorithm of picking one by one the interval with *earliest finish time* (that is compatible with previously selected intervals) is optimal. Show that the greedy strategy of selecting the interval that starts as late as possible is also optimal.

5. In this question we will compute an alignment of minimum possible cost between the strings $X =$ *your-firstname* and $Y =$ *lastname*.

   (a) [3 marks] How many subproblems we get? Write the recurrence for computing the optimal cost of a problem given the optimal solution of relevant subproblems.

   (b) [7 marks] Fill in the appropriate table using the recurrence from previous part. (If your first-name or last-name have more than 5 letters then consider the first 5 letters only).

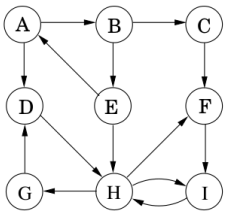
6. There is a row of n coins whose values are some positive integers $c_1, c_2, \ldots, c_n$, not necessarily distinct. Our task is to pick up maximum amount of money subject to the constraint that no two coins adjacent can be picked up.

   (a) [3 marks] Propose a greedy approach and show that it fails, i.e., give an example where the greedy solution is not an optimal solution.

   (b) [5 marks] Identify subproblems and write a dynamic-programming recurrence to compute maximum amount of money that can be picked using optimal sub-solutions.

   (c) [2 marks] Write pseudocode of dynamic programming algorithm based on the above recurrence that runs in linear time.

7. [10 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.



   (a) In what order are the strongly connected components (SCCs) found?

   (b) Which are source SCCs and which are sink SCCs?

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

   (d) What is the minimum number of edges you must add to this graph to make it strongly connected?

8. *Multithreaded Algorithms*

   (a) [6 marks] Define *work*, *span*, and *parallelism* of a parallel algorithm.

   (b) [4 marks] Consider the following multithreaded pseudocode for transposing an $n \times n$ matrix A in place:

```
P-TRANSPOSE(A)
1  n = A.rows
2  parallel for j = 2 to n
3      parallel for i = 1 to j − 1
4          exchange a_ij with a_ji
```

   Analyze the work, span, and parallelism of this algorithm.

9. [10 marks] You're consulting for a small computation-intensive investment company, and they have the following type of problem. They look at n consecutive days of a given stock, at some point in the past. Lets number the days $i = 1, 2, \ldots, n$; for each day i, they have a price $p(i)$ per share for the stock on that day. (Well assume for simplicity that the price was fixed during each day.) Suppose during this time period, they wanted to buy $1,000$ shares on some day and sell all these shares on some (later) day. They want to know: When should they have bought and when should they have sold in order to have made as much money as possible?

   For example, suppose $n = 3$, $p(1) = 9$, $p(2) = 1$, $p(3) = 5$. Then you should return "buy on 2, sell on 3" (buying on day 2 and selling on day 3 means they would have made Rs 4 per share, the maximum possible for that period). Clearly, there's a simple algorithm that takes time $O(n^2)$: try all possible pairs of buy/sell days and see which makes them the most money. Your investment friends were hoping for something a little better.

   Give an $O(n \log n)$ algorithm to find the correct numbers i and j using divide and conquer approach. (*Hint:* Divide into two subproblems and combine the solutions in linear time)