Analysis of Algorithms (*Five Questions*) Points 100

Q1. Write a recursive divide-and-conquer algorithm for computing a sequence of addition of numbers, e.g., 2+3+1+5+7+8. Analyze its space & time-complexity (presuming input size is some power of 2). [20]

Q2. DFS algorithm for a graph is given below:
Input: graph G=(V, E);   *Output*: a DFS spanning tree over G

```
DFS(node v)
(1)   mark v as visited;
(2)   for each not visited node w adjacent to v do
(4)           parent(w) = v;
(5)           DFS(w);      // end for;
(5)   if there exists a not visited node w in V
(6)           DFS(w)       // end if;    & End Algorithm
```

Driver call: DFS(any node of the input graph)
(**2a)** Modify above to print *pre-order numbering* of nodes. [5]
(**2b)** Draw a undirected G=(V={a, b, c, d, e}, E={((a,b), (b,c), (b,e), (b, d), (c,e), (c,d) }. Starting with a call to DFS(a), show your call sequences (i.e., the recursion tree or the traversal of the graph) and your *pre-order numbering* of the nodes. [5]
(**2c)** Write an algorithm that would traverse *each arc* once and only once, by marking arcs. Use the above DFS function. [10]

[page deliberately left blank for answering question 2]

Q3. Suppose a test has 4 questions {q1, q2, q3, q4}, each question number q_i, is associated with (*p_i* points, and *t_i* time-needed-to-answer), which are like the following {(q1, p=2, t=2), (q2, p=3, t=2), (q3, p=5, t=2), (q4, p=1, t=2)}.
No partial grading is allowed, i.e., one gets 0 or full points for answering a question. Maximum time for the test is *T=7.* Find best set of questions to answer by using a *dynamic programming* algorithm. Recurrence for the optimum points P(i, j) is given below, for (q_1 .. q_i) questions in any arbitrary ordering of questions, and an integer *j* being the time limit.
P(i, j) = P(i-1, j) if tj > T,
When tj <=T, P(i, j) = max{ P(i-1, j), pj +P(i-1, j-tj)}
Show the table for P, as you apply the algorithm. You need not write the algorithm.
Both the *optimum set* of questions and the corresponding *optimum aggregate points* must be computed.

[20]

Answer briefly (1 or 2 lines each) [20]

Q4a. Can a *Breadth First Traversal* on a graph produce a spanning tree?

Q4b. Can a *Depth First Traversal* on a graph produce a spanning tree?

Q4c. What are the *time and space* complexities of the *dynamic programming* algorithm above?

Q4d. Which is *higher* for any algorithm: *space complexity* or *time complexity*?

Q4e. In a city highway system we would like to know what is the shortest distance from the post office to all other addresses using only the highway system. Name an algorithm to use for solving this problem.

Answer *true/false* for the following sentences. Explain your answer if you cannot answer as true/false.

Q4f. There are problems which are not NP-complete problems.

Q4g. The set of P-class problems is a subset of the NP-complete problems.

Q4h. There are problems which are not yet proved to be P-class or NP-class.

Q4i. In order to prove a problem *X* to be NP-hard one needs to develop a polynomial transformation from a known NP-hard problem *Y* to *X*. So, if *Y* had a polynomial-time algorithm, then *X* would also have had a polynomial-time algorithm.

Q4j. 4-SAT is a P-class problem.

Q5a. What is the *value* of the variable count in terms of *n* after the following algorithm-fragment is executed? [10]

```
(1) count = 0;
(2) For i = 1 through  n  do
(3)       For p = 1 through  3*i  do
(5)             For k = 1 through 5 do
(4)               count = count +1;
   end for loops;
```

Q5b. What is the *time* complexity of the following algorithm fragment? [10]

```
(0) int count := 0;
(1) For i =1 through 5 do
(2)       For p = 1 through i*i do
(3)                   For k = 1 through i do
(4)                         count++;
end for loops;
(5) print count;
```