Analysis of Algorithms (*Five Questions*) Points 100

Q1. How many lines, as a function of *n* (in O(*n*) form), does the following program print? Write a recurrence and solve it. You may assume *n* is a power of 2.
Draw the recurrence tree for *n*=8.

```
function f(n)
if n > 1:
        print_line(''still going'')
        f(n/2)
        f(n/2)
```

[20]

Q2. Consider the following problem:
INPUT: A set S = {(xi; yi) | 1 <= i <= n} of intervals over the real line.
OUTPUT: A maximum cardinality subset S0 of S such that no pair of intervals in S0 overlap.
Write a *backtracking algorithm* to find it. No pruning is necessary. Discuss its time complexity. [20]

Q3. Suppose a test has 4 questions {q1, q2, q3, q4}, each question number q_i, is associated with two integers (*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 full points for answering a question.
Maximum time for the test is *T=7.* Find the 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 arbitrarily fixed 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)}
Write the algorithm, for any general case.
Compute the table for P, as you apply the algorithm to the above specific problem.
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. Define the *Strongly Connected Component* of a directed graph.

Q4c. What is the main difference between the *divide & conquer* strategy and the *dynamic programming* strategy?

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. 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-class of problems.

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

Q4h. No general purpose algorithm may be developed when a problem is proved to be NP-hard.

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. There exists a polynomial transformation from a problem *X* to a known P-class problem *Y*. Is *X* also a P-class problem?

Q5a. Write a pseudo-code to compute a chain of division, e.g., 2/3/7/8. Analyze its time and space complexity.

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

```
(0) int count := 0;
(1) For i =1 through 5 do
(2)         For p = 1 through i+10 do
(3)                 For k = 1 through i do
(4)                         count++;
end for loops;
(5) print count;
```