

1. Write a *dynamic programming* algorithm for computing $C(1,n)$ from the following formula. Before setting up the iteration loops carefully observe that all the needed values should be available. Analyze the complexity for your algorithm. Drawing a table for C may help.

$$C(i, j) = 0, \text{ for all } i \geq j$$

$$C(i, j) = \max\{ C(i, k) + C(k, j) + 2 \\ \mid \text{for all } k \text{ with } i \leq k \leq j\}, \\ \text{for all } 1 \leq i < j \leq n$$

2a. What is the asymptotic time-complexity and space complexity of the following code fragment in terms of n ?

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

2b. What is the asymptotic time-complexity and space complexity of the following code fragment in terms of n ?

```
(1) For  $i = 1$  through  $n$  do
(2)   For  $p = 1$  through  $i*i$  do
(3)     create a new node and add to a link-list;
        // presume  $O(n)$  time-complexity for the step (3)
    end for loops;
(4) print count;
```

3a. What is a strongly connected component (SCC) in a directed graph?

(*Algorithm SCC Step 1.* Develop a depth first search (DFS) algorithm that would label a directed graph G in post-order traversal.

Step 2. Subsequently traverse the corresponding reverse graph G' , where arcs of G are reversed, again with the depth first algorithm, but according to the high-to-low numberings of labels on the nodes from the previous traversal.

Output: The spanning trees generated from *step 2* above are the SCC's of G .)

3b. Write a DFS algorithm directed graph-traversal first (must restart repeatedly until all nodes are traversed).

3c. Complete the SCC algorithm.

It may be worthwhile for you to dry run the above steps on the following example, and illustrate that in your answer below,

after further connecting a to d , b to c , c to a , b to f , g to f , g to h , d to e , c to e , h to j , and i to h (total 15 arcs)

$a \text{ ----} \rightarrow b \qquad \qquad g$

$d \leftarrow \text{----} c \leftarrow \text{-----} f \leftarrow \text{-----} h$

$e \qquad \qquad j \text{ -----} \rightarrow i$

4. Write a recursive divide-and-conquer algorithm for computing a sequence of divisions of numbers, e.g., $2/3/1/5/7/8$. Analyze its space & time-complexity.

5. Answer *true/false* for the following sentences. You may explain your answer in a line.

a. Name an algorithm for finding shortest path on a weighted graph from a given starting node. (not a true/false question)

b. Name an algorithm for finding all pairs shortest path in a distance graph..

(not a true/false question)

c. There exists an NP-class problem that is not known to belong to the NP-complete class or to the P-class.

d. To prove a problem X to be NP-hard one can develop a polynomial transformation from X to a known NP-hard problem Y . So, if X were to have a polynomial-time algorithm, then Y would also have a polynomial-time algorithm.

e. 4-SAT is an NP-class problem.

f. There is a polynomial-time algorithm for finding the shortest paths between all pairs of nodes in a weighted graph.

g. *QuickSort* algorithm takes $O(n^2)$ time for running on an already sorted array of size n , when the pivot is chosen always as the last element of the array.

h. What is the number of arcs on a tree with m nodes? (not a true/false question)

i. Why is the space complexity of an algorithm always less than or the same as that of its time-complexity? (not a true/false question)

j. 2-SAT problem is an NP-hard problem.