Analysis of Algorithms (*Five Questions*) Points 100

Write down clearly any assumption you make in answering a question that you think is not explicitly stated in the latter.

Q1a. Write a *dynamic programming* algorithm for the following recurrence. **Q1b.** Analyze the complexity for your algorithm. **Q1c.** For C(k, k) =[1, 2, 4, 4, 7], for k = 1 through 5, draw the table for C and show the values C(1,3), C(2,4), and C(3,5) on the table. Computations should be shown.

Recurrence:

C(i, i) are given (like above) for all *i* as input.

C(i, j) = 0, for all i > j

C(i, j) = min{ C(i, k) + C(k, j),

| for all k with i≤k<j},

for all 1 ≤ i < j ≤ n

Q2. Set up a recurrence equation and solve it for the following recursive algorithm, *A* is array of integers and *k* is an integer.

```
Algorithm Unknown(array A, k)
    If k > size(A)  then return False;  // size(A) is the number of elements in A
    if k= = 1 then  return A;  // return the only element in A

    pick a pivot, p = one of the elements of A;
    QuickPartition(A);
        // partitions the array A by exchanging elements such that
        //  elements <=p is on left of p, and elements >p are on its right
    Let, L = left half of A including p;
    Let, R = right half of A;
    if  size(L) ≥ k then Unknown(L, k)
    else Unknown(R, k - size(L) -1);
        // previous call's  k-th  element is  k-|L|-1  in R
End algorithm.
```

Q3a. Write a recursive backtracking algorithm to find a minimum cost-Hamiltonian circuit starting from a given node. (HC: a path from start node to itself covering each node once and only once.) You should use greedy strategy in selecting next node when there are multiple choices of next nodes. No bounding function is necessary.
Q3b. The following is a weighted graph G:
V= {a, b, c, d, e, f, g}
E= {(a, b, 10), (a, d, 5), (b, c, 3), (c, d, 7), (e, f, 3), (f, g, 6), (g, d, 1)}
Draw the graph first.
Run your algorithm on this graph starting from node *a* showing the depth-first traversal to find the minimum-cost HC, if there exists any HC.

Q4. The following is a directed graph G:
V= {a, b, c, d, e, f, g, h, i}
E= {(a, b), (a, d), (b, c), (c, a), (d, a), (d, e), (i, f), (f, g), (g, h), (h, i)}
Draw the graph first.
Find the *strong connected components* (*SCC*) in the graph by using depth first traversals (DFT). You must show the first DFT-spanning tree with post-order numbering, then the reverse graph, and then the second DFT-spanning tree as the SCC's.

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

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

Q5b. What is the *time* complexity of the following algorithm fragment in terms of *k*?

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