**CSE102**

Part 1

1.Briefly describe the idea of divide-and-conquer technique.

A problem instance is divided into several smaller instances of the same problem, ideally of about same size

The smaller instances are solved, typically recursively

The solutions for the smaller instances are combined to get a solution to the original problem

2. Briefly describe the idea of the dynamic programming technique.

An efficient way to implement some divide and conquer algorithm.

3. what is complexity class P and what is complexity class NP?

4.Given any two decision problems A and B, what is a polynomial time reduction from A to B? Briefly explain how this technique can be used to prove certain problems are NP-hard.

Part 2

1. Given an array A of n numbers, (n>=1)

a) Design a divide-and-conquer algorithm for finding values of both the largest and smallest elements in A.

b) Set up a recurrence relation for the number of key comparison made by your algorithm and justify briefly.

c) For n=2k, solve the recurrence relation set up in b).

d) What is the worst case time complexity of your algorithm (in big-O notation)?

2. Consider the following problem. Given an array A consisting of n distinct integers A[1],…A[n]. It is known that there is a position p between 1 and n, such that A[1]<A[2]<…<A[p-1]<A[p] and A[p]>A[p+1]>…>A[n].

a) Design a divide-and-conquer algorithm to find the position p with running time of O(log n) in the worst case.

b) Set up a recurrence relation for the number of key comparisons made by your algorithm and justify it briefly.

c) Solving the recurrence relation to show that the complexity of your algorithm is O(log n) (for simplicity, you can assume that n=2k).

3.Change-making problem: give change for amount n using the minimum number of coins of values d1 < d2 < … < dm. Assume that there are unlimited quantities of coins for each of them values d1 < d2 < … < dm where d1 = 1.

a) Let F(n) be the minimum number of coins whose values add up to n. For convenience, define F(0)=0. Set up a recurrence relation for F(n) (n>0) that can be used by a dynamic programming algorithm.

b) For the amount n=6 and coin values 1,3, and 4, solving the Changing-making problem using the relation set in a).

c)Write pseudocode of the dynamic programming algorithm for solving this problem and determine its time complexity.

4. There is a row of n coins whose values are some positive integers c1, c2, . . . , cn, not necessarily distinct. Let F(n) be the maximum amount of money that can be picked up from the row subject to the constraint that no two coins adjacent in the initial row can be picked up.

a) Set up a recurrence relation for F(n) that can be used by dynamic programming algorithm. (hint: to derive a recurrence for F(n), you can partition all the allowed coin selections into two groups: those that include the last coin and those without it.)

b) For coin row 5, 1, 2, 10, 6, 2 solve the coin row problem using the relation set in a).

c)Write pseudocode of the dynamic programming algorithm for solving this

problem and determine its time complexity.

5. Apply the branch-and-bound algorithm to solve the travelling salesman problem for the following graph.

