Part I. Algorithm

1. (20%) For each of the following statements, determine whether it is correct or not. Explain (You get no credit without explanation).

(*a*) If a problem is proved to be an NP–completel problem, then at present it will always take exponential number of steps to solve this problem for all kinds of inputs.
(*b*) If the time complexity of an algorithm is higher than the presently found highest lower bound of the problem, then this algorithm must not be optimal.
(*c*) We can prove the NP–completeness of a problem A by proving that A polynomially reduces to a proven NP–complete problem.
(*d*) An algorithm is optimal if its time complexity is equal to the lower bound of the problem that it tries to solve.

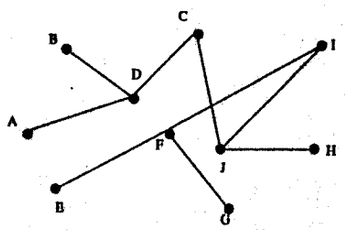
2. (15%) (*a*) Find the *longest common subsequence* (LCS) of $A = aabcdae$ and $B = beadf$. (*b*) How can we use the *dynamic programming* approach to solve the LCS problem.

3. (15%) For each of the following algorithm strategies, give one example to illustrate how it works :
(*a*) *divide – and – conquer*,
(*b*) *prune – and – search*, and
(*c*) *branch – and – bound*.

(背面仍有題目,請繼續作答)

Part II. Data Structure

1.(15%) Assume that a (undirected) graph is represented by a table format in computer such as that shown in the following figure. Write an algorithm which finds all nodes to which a given node, say X, connects. (Your algorithm must be accompanied with clear explanations in order to get score.)



Starting Point	Ending Point
A	D
B	D
D	C
C	J
I	J
H	J
B	F
F	G
I	F

2.(20%) Answer the following questions.

1. The radius of a tree is the maximum distance between any two vertices. Given a connected and undirected graph, write an algorithm for finding a spanning tree of maximum radius.
2. Analyze the complexity of your algorithm.

(Both the algorithm and analysis should be accompanied with explanations to get score.)

3. Answer the following questions about trees.

1. (5%) We all know that the purpose of using a tree is to speed up the search of a data item. However, if we already know what we want to find (e.g., a value 5), then why do we use the tree to find the value? If we do not know what we will find, then a tree does not help. So, what is purpose of having a tree?
2. (10%) We all know that the shorter the height (i.e., the number of levels) of a tree, the less we will spend in time to find a data item in the tree. Then, why do not use a one-level tree (in which the children of the root node are all leaf nodes) to index data items? Do you think this tree will work or will not work? Why?