1. (35 points)

A directed graph without cycles has a natural order. There are many useful applications of directed graphs. Consider academic courses. If the vertices represent academic courses, the graph represents the prerequisite (先修課) structure for the courses. For example, if vertex a precedes b which precedes c, course a is a prerequisite to course b, which is a prerequisite to courses c. The frequently-asked question is such as "In what order should a student take all courses so that he/she will satisfy all prerequisistes?" There is a linear order, called a *topological order*, of the vertices in a directed graph without cycles that answers this question.

(a) Given a directed graph,write a pseudocode procedure for finding all possible topological orders.

(b) Consider directed graphs (digraphs) and courses as Abstract Data Types (ADT). Write a C++ class definition of ADT `digraph`. ADT `digraph` supports the following member functions: constructor, destructor, and findAllTopologicalOrders.

(c) Write a C++ class definition of ADT `course`. ADT `course` supports the following member functions: constructor, destructor, and getCourseDescription.

(d) Write a C++ implementation of ADT `digraph` deep copy constructor as well as necessary declaration.

2. (15 points)

Collision resolution technique is important to allow for dynamic growth of the hash table. Explain the design rationale and the method of each of the following two techniques:

(a) separate chaining.

(b) double hashing.

3. (20 points)

Consider binary trees where each node (either an internal node or a leaf) stores a non-negative integer (aside from pointers to the left and the right children). Your task is to design an algorithm that, given such a tree T and a non-negative integer k as input, determines whether T contains a branch (from the root to a leaf) such that the sum of all numbers stored on the nodes of the branch equals k. Please present

接背面

your algorithm in an adequate pseudo code and make assumptions wherever necessary. Give an analysis of its time complexity. The more efficient your algorithm is, the more points you will be credited for this problem. Is there a possibility that your code may overflow? How do you avoid the problem?

4. (20 points)

   Let $G = (V, E)$ be a connected weighted undirected graph and T be a minimum-cost spanning tree (MST) of G. Suppose that the cost of some edge $\{u, v\}$ in G is *increased*; $\{u, v\}$ may or may not belong to T. Design an algorithm that will either find a new MST or determine that T is still an MST. Please present your algorithm in an adequate pseudo code and make assumptions wherever necessary. Explain why your algorithm is correct and analyze its time complexity. The more efficient your algorithm is, the more points you will be credited for this problem.

5. (10 points)

   Describe as precisely as possible two NP-complete problems that are significantly different in nature, for example one about graphs and the other about boolean expressions. Explain how you can prove the NP-completeness of one of the two problems, given that the other has been proven NP-complete.