问题与讨论

2014-3-6

4.1. Consider the problem of summing the salaries of employees earning more than their direct manager, assuming each employee has a single manager. The employees are labeled 1, 2, etc. Write algorithms that solve the problem for each of the following representations of the input data:

(a) The input is given by an integer N and a two-dimensional array A, where N is the number of employees, $A[I, 1]$ is the salary of the Ith employee and $A[I, 2]$ is the label of his or her manager.

(b) The input is given by a binary tree constructed as follows: The root of the tree represents the first employee. For every node V of the tree representing the Ith employee,
- V contains the salary of the Ith employee;
- the first offspring of V is a leaf containing the label of the manager of the Ith employee; and
- if there are more than I employees, the second offspring of V is the node that represents the $I + 1$th employee.

4.2. (a) Write an algorithm which, given a tree T, calculates the sum of the depths of all the nodes of T.

(b) Write an algorithm which, given a tree T and a positive integer K, calculates the number of nodes in T at depth K.

(c) Write an algorithm which, given a tree T, checks whether it has any leaf at an even depth.

4.8. Prove that the maximal distance between any two points on a polygon occurs between two of the vertices.

4.11. Write algorithms that find the two maximal elements in a given vector of N distinct integers (assume $N > 1$).

(a) Using an iterative method.
(b) Using the divide-and-conquer method.

4.12. Write in detail the greedy algorithm described in the text for finding a minimal spanning tree.

The **integer-knapsack** problem asks to find a way to fill a knapsack of some given capacity with some elements of a given set of available items of various types in the most profitable way. The input to the problem consists of:

- C, the total weight capacity of the knapsack;
- a positive integer N, the number of item types;
- a vector Q, where $Q[I]$ is the available number of items of type I;
- a vector W, where $W[I]$ is the weight of each item of type I, satisfying $0 < W[I] \leq C$; and
- a vector P, where $P[I]$ is the profit gained by storing an item of type I in the knapsack.

All input values are non-negative integers. The problem is to fill the knapsack with elements whose total weight does not exceed C, such that the total profit of the knapsack is maximal. The output is a vector F, where $F[I]$ contains the number of items of type I that are put into the knapsack.

The **knapsack** problem is a variation of the integer-knapsack problem, in which instead of discrete items, there are materials. The difference is that instead of working with integer numbers, we may put into the knapsack any *quantity* of material I which does not exceed the available quantity $Q[I]$. The vectors W and P now contain the weight and profit, respectively, of one quantity unit of material I. All input and output values are now non-negative real numbers, not necessarily integers.

4.13. (a) Design a dynamic planning algorithm for the integer-knapsack problem.

(b) What is your algorithm's output for the input

- $N = 5$
- $C = 103$
- $Q = [3,1,4,5,1]$
- $W = [10,20,20,8,7]$
- $P = [17,42,35,16,15]$

and what is the total profit of the knapsack?

4.14. (a) Design a greedy algorithm for the knapsack problem.

(b) What is your algorithm's output for the input given in Exercise 4.13(b), and what is the total profit of the knapsack now?