Exercise 11 Greedy Algorithm

11. Write a C++ program for implementation of following algorithms a) Knapsack algorithm b) Job sequence with dead line

Objective:

The objective of this exercise is to enable you to perform knapsack algorithm and job sequence with dead line algorithm.

Procedure and Description

a) Knapsack algorithm

The knapsack problem is a problem in combinatorial optimization (finding an optimal object from a finite set of objects). Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items.

Algorithm:

n: No. of objects
m&u: capacity
i&j: used as index of objects
P[i]: profit index i
W[j]: Weight index j
X: benefit

The steps for knapsack problem are given below

Step 1: Greedy knapsack (m, n)
Step 2: p [1: n} and w [1: n] contain the profits and weights respectively
Of the n objects ordered such that p[i]/w[i]≥p [i+1]/w [i+1]
Step 3: m is the knapsack size and x [1: n] is the solution vector
Step 4: for i: =1 to n do x[i]:=0.0;//Initialize x
Step 5: u: =m;
Step 6: for i: =1 to n do
Step 7: if (w[i]>u) then break;
Step 8: x[i]:=1.0; U: =U-w[i];
If (i ≤ n) then x[i]:=u/w[i];
Step 9: end

Expected output:

After executing program enter set of items, each with ratio, weight and a profit, then by performing algorithm steps it determines the benefit.

Input: capacity: 7

ratio:2.5 Wgt: 2 profit: 5.0
ratio: 2.0 wgt: 2 profit: 4.0
ratio: 1.67 wgt: 3 profit: 5.0
ratio: 1.5 wgt: 4 profit: 6.0

Output: Cumulative benefit: 14.0

b) Job sequence with dead line

In this job sequence with dead line, given a set of n jobs, each having a "deadline" (an integer) d[i] and a profit p[i] associated with it. For a job i the profit is earned if and only if the job is completed within its deadline. Each job takes one unit of time on a machine (processor) and only one machine is available. We want to maximize the profit. The jobs are arranged in decreasing order of profit in an array.

Algorithm:

n: set of jobs
i&j: used as index of jobs
d[i]:an integer deadline
P[i]: job i profit
J[i]: ith job with optimal solution
r & k: variables used to store result values.

The steps for Job sequence with dead line are given below

Step 1: Algorithm JS (d, j, n)

Step 2: d[i]≥1, 1≤i≤n are the deadlines, n≥1. The jobs are

ordered such that p [1] ≥p[2] ≥....≥ P[n].

Step 3: J[i] is the Ith job in the optimal solution, 1≤i≤k.

Step 4: Also, at termination d [J[i]] ≤ d[J[i+1]], 1≤i≤k.

Step 5: d [0]:=J[0]:=0; //initialize

Step 6: J [1]:=1; include job 1

K: =1;

Step 7: for i: =2 to n do

Step 8: Consider jobs in non-increasing order of p[i]
Find position for i and check feasibility for insertion

Step 9: r: =k;

Step 10: While((d[J[r]]>d[i]) and (d[J[r]]≠r)) dp r:= r – 1;

Step 11: if ((d [J[r]] ≤d[i]) and (d[i]>r)) then

Step 12: Insert i into J [ ].

Step 13: for q: =k to (r+1) step – 1 do J [q+1]:=J[q];
J[r+1]:=i; k: =k+1;

Step 14: end

Step 15: return k;

Expected output:

Input:

After executing program enter the Time slot, number of jobs, profit deadline and profit/Time.

Example:

Time slots 1, 2, and 3. (Slot 0 is sentinel)

Number of jobs=5

Job (i)	Profit	Deadline	Profit/Time
A	100	2	100
B	19	1	19
C	27	2	27
D	25	1	25
E	15	33	15

Output:

Jobs done: Job c at time 1
Job a at time 2
Job e at time 3

Number of jobs done: 3, total profit: 142