1.KNAPSACK PROBLEM

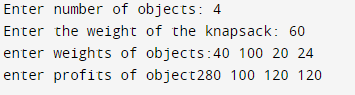
OBJECTIVE

Knapsack problem using greedy method.

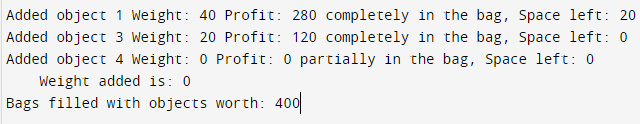
METHODOLOGY

The basic idea of the **greedy approach** is to calculate the ratio value/weight **for** each item and sort the item on basis of this ratio(decreasing order). Then take the item with the highest ratio and add them until we can't add the next item as a whole and at the end add the next item as much as we can partially.

INPUT



OUTPUT



CODE

#include<stdio.h>

#include<iostream>

using namespace std;

int main()

{

int array[2][100], n, w, i, curw, used[100], maxi = -1, totalprofit = 0;

//input number of objects

cout << "Enter number of objects: ";

cin >> n;

//input max weight of knapsack

cout << "Enter the weight of the knapsack: ";

cin >> w;

/\* Array's first row is to store weights

second row is to store profits \*/

cout<<"enter weights of objects:";

for (i = 0; i < n; i++)

{

cin >> array[0][i];

}

cout<<"enter profits of object";

for (i = 0; i < n; i++)

{

cin >> array[1][i];

}

for (i = 0; i < n; i++)

{

used[i] = 0;

}

curw = w;

//loop until knapsack is full

while (curw >= 0)

{

maxi = -1;

//loop to find max profit object

for (i = 0; i < n; i++)

{

if ((used[i] == 0) && ((maxi == -1) || (((float) array[1][i]

/ (float) array[0][i]) > ((float) array[1][maxi]

/ (float) array[0][maxi]))))

{

maxi = i;

}

}

used[maxi] = 1;

//decrease current wight

curw -= array[0][maxi];

//increase total profit

totalprofit += array[1][maxi];

if (curw >= 0)

{

cout << "\nAdded object " << maxi + 1 << " Weight: "

<< array[0][maxi] << " Profit: " << array[1][maxi]

<< " completely in the bag, Space left: " << curw;

}

else

{

cout << "\nAdded object " << maxi + 1 << " Weight: "

<< (array[0][maxi] + curw) << " Profit: "

<< (array[1][maxi] / array[0][maxi]) \* (array[0][maxi]

+ curw) << " partially in the bag, Space left: 0"

<< " Weight added is: " << curw + array[0][maxi];

totalprofit -= array[1][maxi];

totalprofit += ((array[1][maxi] / array[0][maxi]) \*(array[0][maxi]+curw));

}

}

//print total worth of objects filled in knapsack

cout << "\nBags filled with objects worth: " << totalprofit;

return 0;

}