EXERISE 9: KNAPSACK USING BRANCH AND BOUND

PRATEEK MOHANTY

20BCE1482

KNAPSACK USING BRANCH AND BOUND

* ALGORITHM

1. Firstly take the value , weight in a structure
2. Sort the array of Items in ascending order
3. Now using greedy method , find the max profit which will be equal to the upper bound
4. Extract a item from queue
5. Compute the profit of the item next level , if profit is greater than max Profit update the max Profit
6. Compute Bound of the next node, if Bound if more than the max Profit then add next level node to Q
7. Otherwise discard it
8. When next level node is not considered as part of solution, then add next level node to Q.

CODE

#include <iostream>

#include <vector>

#include <algorithm>

#include <queue>

using namespace std;

// knapsack using branch and bound

struct Item

{

    float weight;

    int value;

};

// calculating upper bound using greedy method

bool cmp(Item a, Item b)

{

    double a1 = (double)a.value / a.weight;

    double b1 = (double)b.value / b.weight;

    return a1 > b1;

}

struct Node

{

    int level, profit, bound;

    float weight;

};

int upperBound(Node u, int n, int w, Item arr[])

{

    if (u.weight >= w)

        return 0;

    int profit\_bound = u.profit;

    int j = u.level + 1;

    int totweight = u.weight;

    while ((j < n) && (totweight + arr[j].weight <= w))

    {

        totweight += arr[j].weight;

        profit\_bound += arr[j].value;

        j++;

    }

    if (j < n)

        profit\_bound += (w - totweight) \* arr[j].value / arr[j].weight;

    return profit\_bound;

}

int knapsack(int W, Item arr[], int n)

{

    // sort Item on basis of value per unit

    sort(arr, arr + n, cmp);

    queue<Node> Q;

    Node u, v;

    u.level = -1;

    u.profit = u.weight = 0;

    Q.push(u);

    int maxProfit = 0;

    while (!Q.empty())

    {

        u = Q.front();

        Q.pop();

        if (u.level == -1)

            v.level = 0;

        if (u.level == n - 1)

            continue;

        v.level = u.level + 1;

        v.weight = u.weight + arr[v.level].weight;

        v.profit = u.profit + arr[v.level].value;

        if (v.weight <= W && v.profit > maxProfit)

            maxProfit = v.profit;

        v.bound = upperBound(v, n, W, arr);

        if (v.bound > maxProfit)

            Q.push(v);

        v.weight = u.weight;

        v.profit = u.profit;

        v.bound = upperBound(v, n, W, arr);

        if (v.bound > maxProfit)

            Q.push(v);

        v.weight = u.weight;

        v.profit = u.profit;

        v.bound = upperBound(v, n, W, arr);

        if (v.bound > maxProfit)

            Q.push(v);

    }

    return maxProfit;

}

int main()

{

    vector<int> weight, profit;

    int W = 10;

    Item arr[] = {{2, 40}, {3.14, 150}, {1.98, 10}, {5, 90}, {3, 25},{6,50}};

    int n = sizeof(arr) / sizeof(arr[0]);

    cout << "Max profit is: " << knapsack(W, arr, n);

    return 0;

}

OUTPUT

