A1-B3-42

Amber Shukla

DAA LAB

PRACTICAL NO. 3

Aim: Perform Fractional Knapsack for the given scenario

A. Load the truck using different methods: Minimum weight, Maximum profit,

Profit/weight ratio. Compute the total profit using each method and infer the best

performing method.

B. Compute the time required in each method.

Given Data:

 Capacity of truck 850 Kgs

 Weight in kg for each box:

[7, 0, 30, 22, 80, 94, 11, 81, 70, 64, 59, 18, 0, 36, 3, 8, 15, 42, 9, 0, 42, 47, 52, 32, 26, 48, 55,6, 29, 84, 2, 4, 18, 56, 7, 29, 93, 44, 71, 3, 86, 66, 31, 65, 0, 79, 20, 65, 52, 13]

 Profit in Rs for each box:

[360, 83, 59, 130, 431, 67, 230, 52, 93, 125, 670, 892, 600, 38, 48, 147, 78, 256, 63, 17, 120,164, 432, 35, 92, 110, 22, 42, 50, 323, 514, 28, 87, 73, 78, 15, 26, 78, 210, 36, 85, 189, 274,43, 33, 10, 19, 389, 276, 312]

Code:

```
#include <bits/stdc++.h>
#include <chrono>
using namespace std;
using namespace std::chrono;
#define pb push_back
#define pop pop_back

double trucksize = 850, n = 50;

// In Vector Box Dimention 1 is Weight || Dimention 2,0 is Profit || Dimention 2,1 is
Profit/Weight

double max_profit(vector<vector<double>> box)
{
    double profit = 0, current_weight = 0;

    sort(box.begin(), box.end(), [](const vector<double> &a, const vector<double> &b)
         { return a[1] > b[1]; });

    int i = 0;

    while (current_weight < trucksize && i < n)
    {
        if (current_weight + box[i][0] >= trucksize)
        {
            double remaining_space = trucksize - current_weight;
            if (box[i][0] > 0)
            { // Avoid division by zero
                profit += (box[i][1] * (remaining_space / box[i][0]));
            }
            current_weight = trucksize;
            break;
        }

        current_weight += box[i][0];
        profit += box[i][1];
        i++;
    }

    return profit;
}

double min_weight(vector<vector<double>> box)
{
    double profit = 0, current_weight = 0;

    sort(box.begin(), box.end(), [](const vector<double> &a, const vector<double> &b)
         { return a[0] < b[0]; });

    int i = 0;

    while (current_weight < trucksize && i < n)
```

```
    {
        if (current_weight + box[i][0] >= trucksize)
        {
            double remaining_space = trucksize - current_weight;
            if (box[i][0] > 0)
            { // Avoid division by zero
                profit += (box[i][1] * (remaining_space / box[i][0]));
            }
            current_weight = trucksize;
            break;
        }

        current_weight += box[i][0];
        profit += box[i][1];
        i++;
    }

    return profit;
}

double pw_ratio(vector<vector<double>> box)
{
    double profit = 0, current_weight = 0;

    sort(box.begin(), box.end(), [](const vector<double> &a, const vector<double> &b)
         { return a[2] > b[2]; });

    int i = 0;

    while (current_weight < trucksize && i < n)
    {
        if (current_weight + box[i][0] >= trucksize)
        {
            double remaining_space = trucksize - current_weight;
            if (box[i][0] > 0)
            { // Avoid division by zero
                profit += (box[i][1] * (remaining_space / box[i][0]));
            }
            current_weight = trucksize;
            break;
        }

        current_weight += box[i][0];
        profit += box[i][1];
        i++;
    }

    return profit;
}

int main()
{

    vector<vector<double>> box(n, vector<double>(3, 0));
```

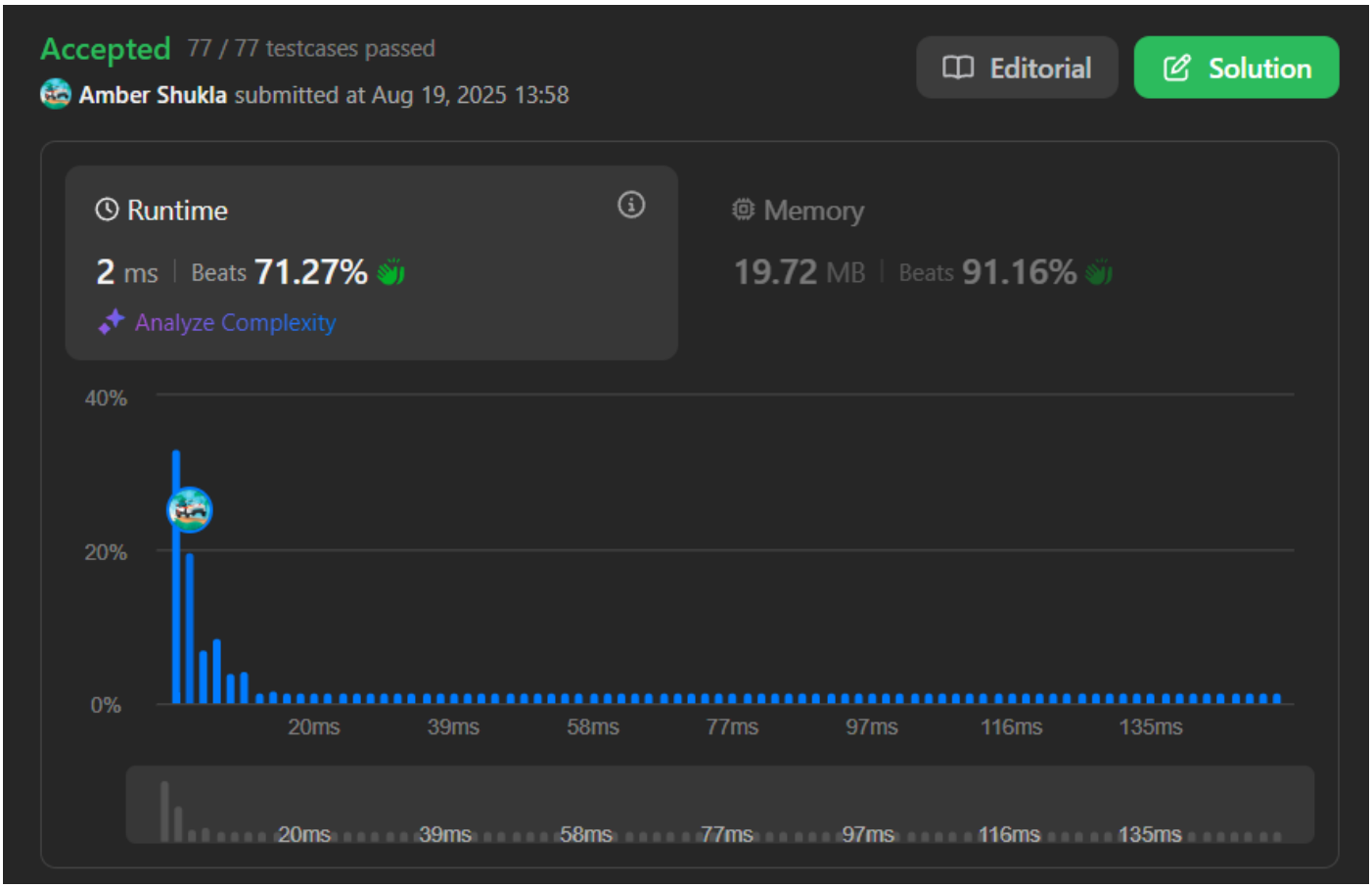
```
    cout << "Enter Weight of all Boxes" << endl;

    for (int i = 0; i < n; i++)
        cin >> box[i][0];

    cout << "Enter Profit Values of all Boxes" << endl;

    for (int i = 0; i < n; i++)
    {
        cin >> box[i][1];
        (box[i][0] != 0) ? (box[i][2] = (box[i][1] / box[i][0])) : (box[i][2] = 9999); //
Profit/Weight Calculation
    }

    auto start_time = high_resolution_clock::now();
    cout << "\n\nProfit By Maximum Profit Method: " << max_profit(box) << endl;
    auto end_time = high_resolution_clock::now();
    auto time_taken = duration_cast<microseconds>(end_time - start_time);
    cout << "Time taken: " << time_taken.count() << " microseconds" << endl;

    start_time = high_resolution_clock::now();
    cout << "\nProfit By Minimum Weight Method: " << min_weight(box) << endl;
    end_time = high_resolution_clock::now();
    time_taken = duration_cast<microseconds>(end_time - start_time);
    cout << "Time taken: " << time_taken.count() << " microseconds" << endl;

    start_time = high_resolution_clock::now();
    cout << "\nProfit By Profit/Weight Ratio Method: " << pw_ratio(box) << endl;
    end_time = high_resolution_clock::now();
    time_taken = duration_cast<microseconds>(end_time - start_time);
    cout << "Time taken: " << time_taken.count() << " microseconds" << endl;

    return 0;
}
```

Output:

```
Profit By Maximum Profit Method: 7076.08
Time taken: 16 microseconds

Profit By Minimum Weight Method: 6265.75
Time taken: 7 microseconds

Profit By Profit/Weight Ratio Method: 7566.86
Time taken: 6 microseconds
```

LeetCode:

```
class Solution {
public:
    int maximumUnits(vector<vector<int>>& boxTypes, int truckSize) {

        sort(boxTypes.begin(), boxTypes.end(), [](const vector<int>& a, const vector<int>& b) {
            return a[1] > b[1];
        });

        int totalUnits = 0;
        int Total_Boxes = 0;

        for (auto& box : boxTypes) {
            int Box_Count = box[0];
            int Box_Units = box[1];

            int Box_Min = min(Box_Count, truckSize - Total_Boxes);

            totalUnits += Box_Min * Box_Units;
            Total_Boxes += Box_Min;

            if (Total_Boxes == truckSize) {
                break;
            }
        }
        return totalUnits;
    }
};
```



PART (B)

[SUBMISSION ON HackerRank]

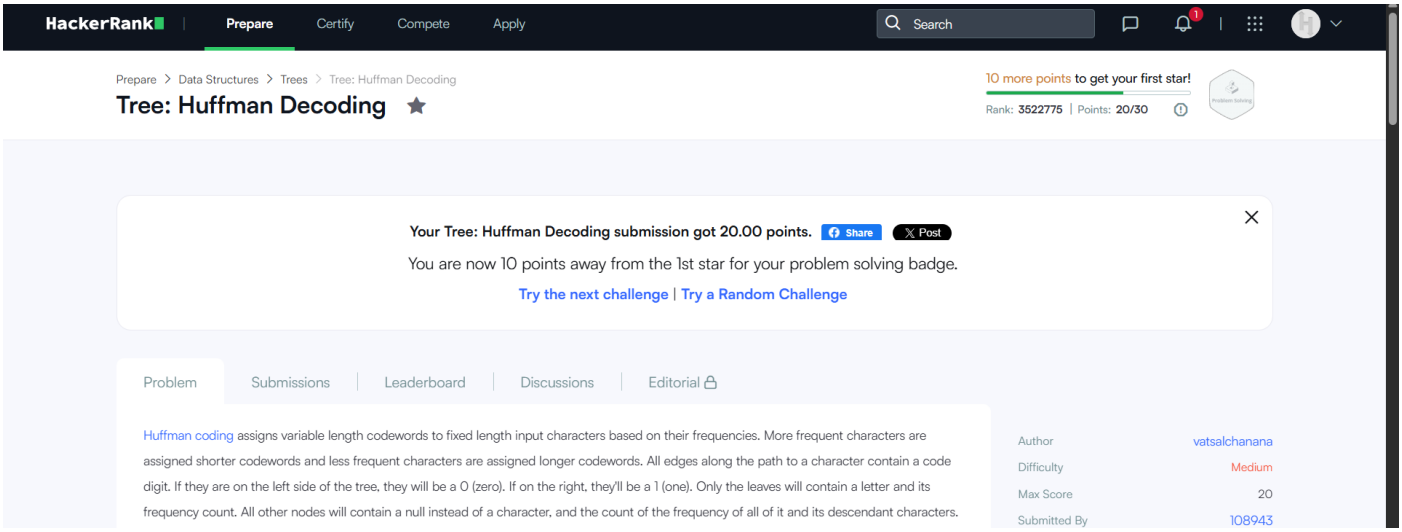
Aim: Huffman coding and decoding

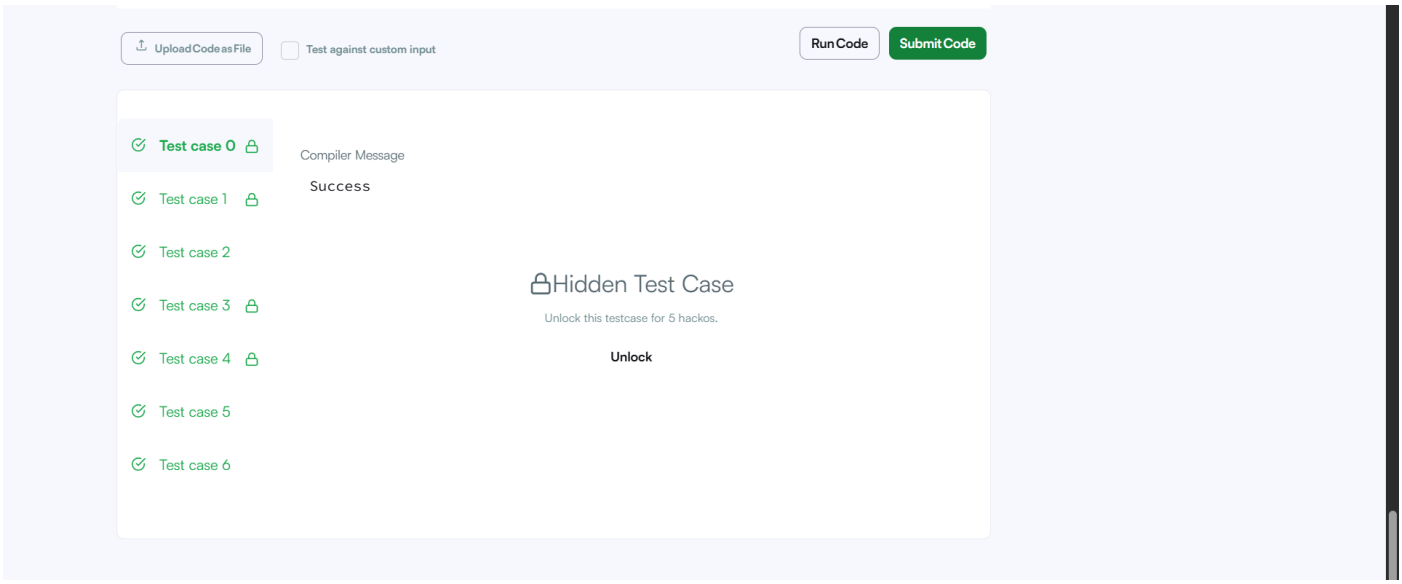
https://www.hackerrank.com/challenges/tree-huffman-decoding/problem

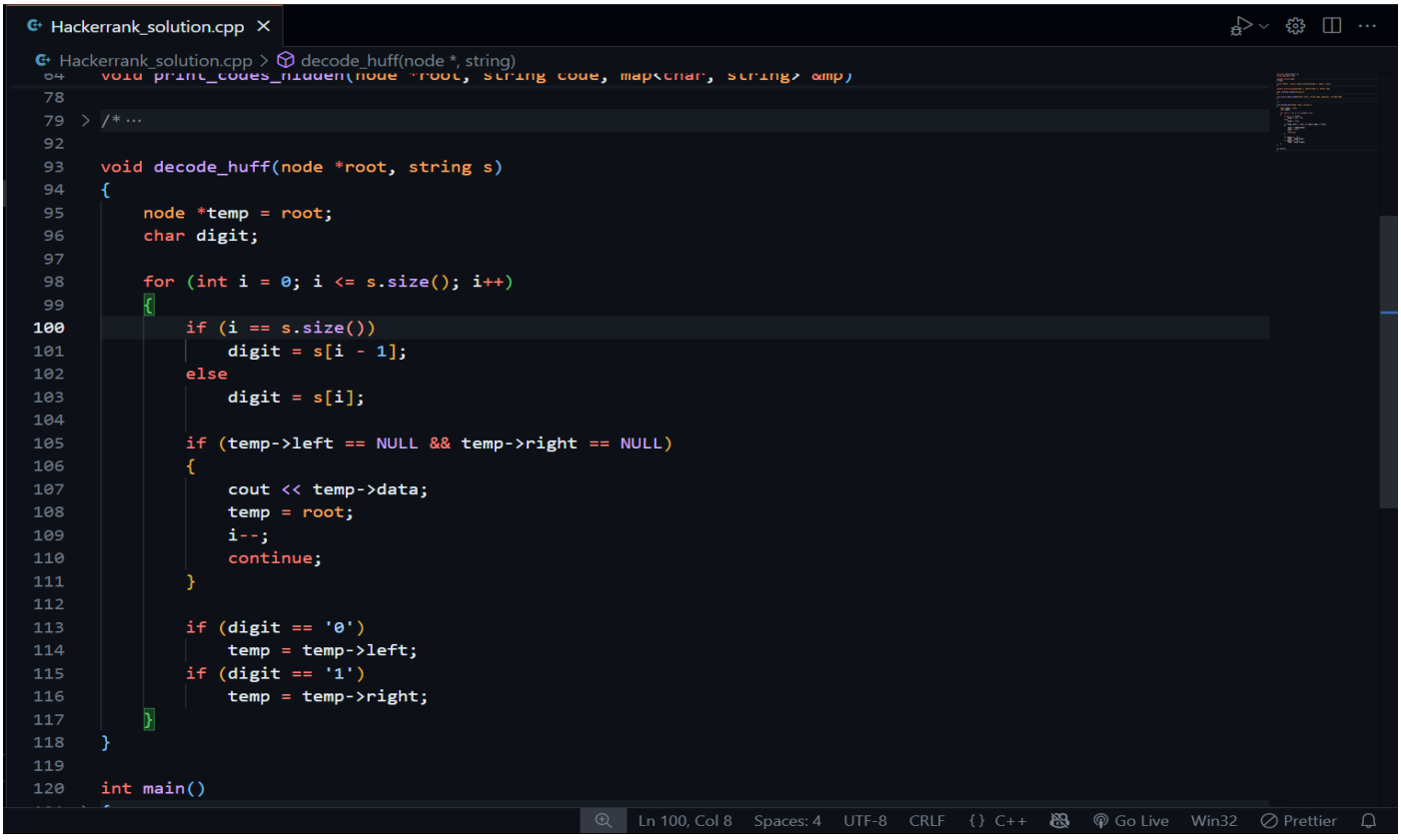
Code:

```
void decode_huff(node *root, string s)
{
    node *temp = root;
    char digit;

    for (int i = 0; i <= s.size(); i++)
    {
        if (i == s.size())
            digit = s[i - 1];
        else
            digit = s[i];

        if (temp->left == NULL && temp->right == NULL)
        {
            cout << temp->data;
            temp = root;
            i--;
            continue;
        }

        if (digit == '0')
            temp = temp->left;
        if (digit == '1')
            temp = temp->right;
    }
}
```

Outputs:


HackerRank
Prepare
Certify
Compete
Apply
Search
Prepare > Data Structures > Trees > Tree: Huffman Decoding
Tree: Huffman Decoding
10 more points to get your first star!
Rank: 3522775 | Points: 20/30
Your Tree: Huffman Decoding submission got 20.00 points.
Share
Post
You are now 10 points away from the 1st star for your problem solving badge.
Try the next challenge | Try a Random Challenge
Problem
Submissions
Leaderboard
Discussions
Editorial
Huffman coding assigns variable length codewords to fixed length input characters based on their frequencies. More frequent characters are assigned shorter codewords and less frequent characters are assigned longer codewords. All edges along the path to a character contain a code digit. If they are on the left side of the tree, they will be a 0 (zero). If on the right, they'll be a 1 (one). Only the leaves will contain a letter and its frequency count. All other nodes will contain a null instead of a character, and the count of the frequency of all of it and its descendant characters.
Author
vatsalchanana
Difficulty
Medium
Max Score
20
Submitted By
108943


Upload Code as File
Test against custom input
Run Code
Submit Code
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Hidden Test Case
Unlock this testcase for 5 hackos.
Unlock


Hackerrank_solution.cpp
Hackerrank_solution.cpp > decode_huff(node *, string)
/*···
void decode_huff(node *root, string s)
{
node *temp = root;
char digit;
for (int i = 0; i <= s.size(); i++)
{
if (i == s.size())
digit = s[i - 1];
else
digit = s[i];
if (temp->left == NULL && temp->right == NULL)
{
cout << temp->data;
temp = root;
i--;
continue;
}
if (digit == '0')
temp = temp->left;
if (digit == '1')
temp = temp->right;
}
}
int main()
Ln 100, Col 8
Spaces: 4
UTF-8
CRLF
C++
Go Live
Win32
Prettier