LAB - 9

1. Implement fractional Knapsack problem using Greedy technique.

#include <stdio.h>

void knapsack(int n, int p[], int w[], int W) {

int used[n];

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

used[i] = 0;

int cur\_w = W;

float tot\_v = 0.0;

int i, maxi;

while (cur\_w > 0) {

maxi = -1;

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

if ((used[i] == 0) &&

((maxi == -1) || ((float)w[i] / p[i] > (float)w[maxi] / p[maxi]))) {

maxi = i;

}

}

used[maxi] = 1;

if (w[maxi] <= cur\_w) {

cur\_w -= w[maxi];

tot\_v += p[maxi];

printf("Added object %d (%d, %d) completely in the bag. Space left: %d.\n",

maxi + 1, w[maxi], p[maxi], cur\_w);

} else {

int taken = cur\_w;

cur\_w = 0;

tot\_v += (float)taken / w[maxi] \* p[maxi];

printf("Added %d%% (%d, %d) of object %d in the bag.\n",

(int)((float)taken / w[maxi] \* 100), w[maxi], p[maxi], maxi + 1);

}

}

printf("Filled the bag with objects worth %.2f.\n", tot\_v);

}

int main() {

int n, W;

printf("Enter the number of objects: ");

scanf("%d", &n);

int p[n], w[n];

printf("Enter the profits of the objects: ");

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

scanf("%d", &p[i]);

}

printf("Enter the weights of the objects: ");

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

scanf("%d", &w[i]);

}

printf("Enter the maximum weight of the bag: ");

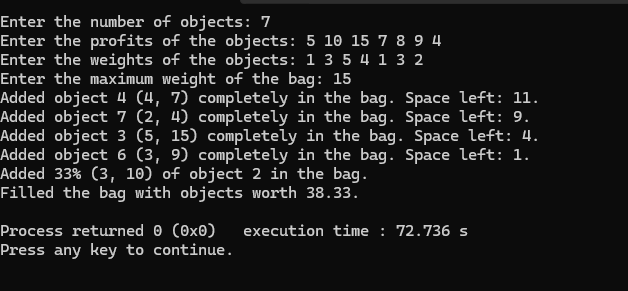
scanf("%d", &W);

knapsack(n, p, w, W);

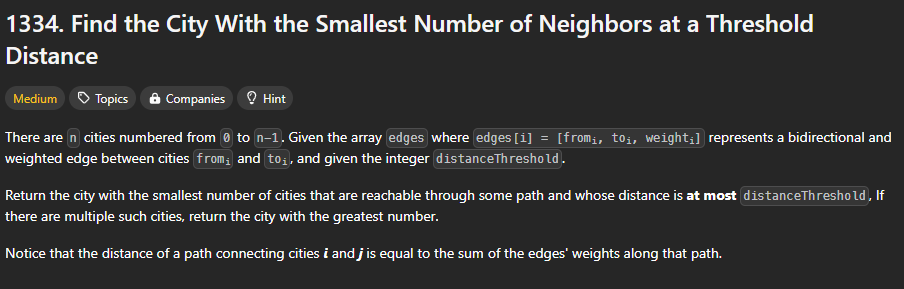
return 0;

}

**OUTPUT-**

****

**LEETCODE QUESTION**

****

int findTheCity(int n, int\*\* edges, int edgesSize, int\* edgesColSize, int distanceThreshold) {

short i,j,k,\*\*dist=(short\*\*)malloc(sizeof(short\*)\*n);

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

dist[i]=(short\*)malloc(2\*n);

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

dist[i][j]=i==j?0:SHRT\_MAX;

}

for(i=0;i<edgesSize;i++){

dist[edges[i][0]][edges[i][1]]=edges[i][2];

dist[edges[i][1]][edges[i][0]]=edges[i][2];

}

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

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

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

dist[j][k]=dist[j][k]<dist[j][i]+dist[i][k]?dist[j][k]:dist[j][i]+dist[i][k];

short \*reached=(short\*)malloc(2\*n);

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

reached[i]=0;

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

dist[i][j]<=distanceThreshold?reached[i]++:1;

}

for(i=0;i<n;free(dist[i++]));

free(dist);

short res=n-1,min=reached[n-1];

for(i=n-1;i>=0;i--)

if(reached[i]<min){

min=reached[i];

res=i;

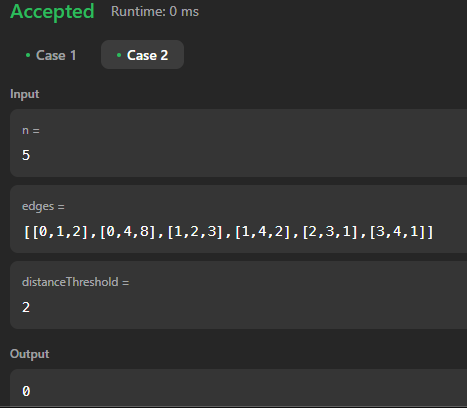
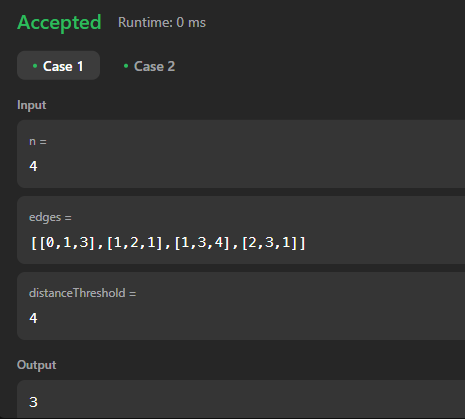
}

free(reached);

return res;

}

**OUTPUT –**

****