1. Activity Selection

int activitySelection(vector<int> start, vector<int> end, int n)

{

vector<pair<int, int>> v;

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

v.push\_back({end[i], start[i]});

sort(v.begin(), v.end());

int res = 1, tim = v[0].first;

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

{

if(v[i].second>tim)

{

res++;

tim = v[i].first;

}

}

return res;

}

1. Huffman decoding

void getCodes(struct MinHeapNode\* root, string str, unordered\_map<string, char> &codes) {

if(!root) return;

if(root->data!='$') {

codes[str]=root->data;

return;

}

getCodes(root->left, str+"0", codes);

getCodes(root->right, str+"1", codes);

}

string decodeHuffmanData(struct MinHeapNode\* root, string code)

{

unordered\_map<string, char> codes;

getCodes(root, "", codes);

int i=0, n=code.size();

string decoded;

while(i<n) {

string temp;

temp+=code[i++];

while(i<n&&(!codes.count(temp))) {

temp+=code[i++];

}

decoded+=codes[temp];

}

return decoded;

}

1. N meeting in one room

int maxMeetings(int start[], int end[], int n)

{

vector<pair<int, int>> v;

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

v.push\_back({end[i], start[i]});

sort(v.begin(), v.end());

int res = 1, tim = v[0].first;

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

{

if(v[i].second>tim)

{

res++;

tim = v[i].first;

}

}

return res;

}

1. Fractional knapsack

bool comparism(Item a, Item b)

{

return (1.0\*a.value/a.weight >1.0\*b.value/b.weight);

}

class Solution

{

public:

//Function to get the maximum total value in the knapsack.

double fractionalKnapsack(int W, Item arr[], int n)

{

sort(arr, arr+n, comparism);

double res = 0;

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

{

if(W >= arr[i].weight)

{

W = W - arr[i].weight;

res = res + arr[i].value;

}

else

{

res = res+ 1.0\*W\*arr[i].value/arr[i].weight;

W=0;

}

if(!W) break;

}

return res;

}

};

1. Largest no with given sum

string largestNumber(int n, int sum)

{

if(sum>9\*n) return "-1";

else if(n==1)

{

char ch[] = {char('0'+sum)};

string s(ch);

return s;

}

char arr[n] = {'0'};

int i;

for(i=0; i<sum/9; i++)

arr[i]='9';

if(sum%9!=0)

arr[i++]='0'+char(sum%9);

for(;i<n;i++)

arr[i]='0';

string ans(arr);

return ans.substr(0,n);

}

1. Job sequencing problem

bool comparison(Job a, Job b)

{

return (a.profit > b.profit);

}

class Solution

{

public:

//Function to find the maximum profit and the number of jobs done.

vector<int> JobScheduling(Job arr[], int n)

{

sort(arr, arr+n, comparison);

int count = 0, res = 0;

int result[n];

bool slot[n];

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

slot[i] = false;

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

{

for (int j=min(n, arr[i].dead)-1; j>=0; j--)

{

if (slot[j]==false)

{

result[j] = i;

slot[j] = true;

break;

}

}

}

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

if(slot[i])

{

count++;

res+=arr[result[i]].profit;

}

vector<int> k = {count, res};

return k;

}

};