Problem Set

sharif

May 24, 2023

1 Wormholes

Link

This problem is based on finding a **negative cycle** in a graph.
ALgorithm used: **Bellman ford**

```
1
2  #include<bits/stdc++.h>
3  using namespace std ;
4
5
6   bool find_negative_cycle(vector<pair<int,int>>adj[],int noVertices ,int source)
7  {
8      vector<int>distance(noVertices,numeric_limits<int>::max());
9      distance[source]=0 ;
10
11     //Relaxation
12     for(int i=0 ;i<noVertices-1;i++)
13     {
14         for(int u=0;u<noVertices;u++)
15         {
16             for(auto edges: adj[u])
17             {
18                 int v=edges.first ;
19                 int weight=edges.second ;
20                 if(distance[u]!=numeric_limits<int>::max() && distance[u]+weight<distance[v])
21                 {
22                     distance[v]=distance[u]+weight ;
23                 }
24             }
25         }
26     }
27
28     //additional iteration to check negative cycle
29     for(int i=0 ;i<noVertices ;i++)
30     {
31         for(auto edges: adj[i])
32             {
33                 int v=edges.first ;
34                 int weight=edges.second ;
35                 if(distance[i]!=numeric_limits<int>::max() && distance[i]+weight<distance[v])
36                 {
37                     return true ;
38                 }
39             }
40     }
41     return false ;
42
43 }
44
```

```
45
46 int main()
47 {
48
49 int t ,n,m;//star system , wormholes
50 scanf("%d",&t);
51 while(t--)
52 {
53  scanf("%d%d",&n,&m);
54  vector<pair<int,int>>adj[n+10] ;
55  for(int i=0 ;i<m ;i++)
56  {
57      int u,v,w ;
58      scanf("%d%d%d",&u,&v,&w);
59      adj[u].push_back({v,w});
60
61  }
62
63  if(find_negative_cycle(adj,n,0))
64  {
65      printf("possible\n");
66  }
67  else
68  {
69      printf("not possible\n");
70  }
71
72 }
73
74 return 0 ;
75 }
```

2 King Escape

Link

ALgorithm used: **dfs**

```cpp

#include<bits/stdc++.h>
using namespace std;
const int N=1e3+10 ;

int visited[N][N] ;
int n,ax,ay,bx,by,cx,cy ;

bool checkValidMove(int nx,int ny) //new x and new y
{
    if(nx==ax || ny==ay) // check horizontal and vertical
    {
        return true ;
    }
    if(abs(nx-ax)==abs(ny-ay)) //check queen and king are same diagonal or not
    {
        return true ;
    }

    return false ;
}

void dfs(int x,int y) {

    int dx[8] = {1, 1, 1, 0, 0, -1, -1, -1};
    int dy[8] = {1, 0, -1, 1, -1, 1, 0, -1};

        if(x<1 || x>n || y<1 || y>n)
        {
            return ;
        }
        if(visited[x][y])
        {
            return  ;
        }

        if(checkValidMove(x,y))
        {
            return ;
        }
    visited[x][y]=1 ;

        for (int i = 0; i < 8; ++i) {
            int nx = x + dx[i];
            int ny = y + dy[i];

            dfs(nx,ny);//check for new cell

        }
}

int main() {
cin >> n;
   // queen location
    cin >> ax >> ay;
    // king location
    cin >> bx >> by;
    // target location
    cin >> cx >> cy;

    dfs(bx,by);
    if (visited[cx][cy])
        cout << "YES" << endl;
    else
        cout << "NO" << endl;

    return 0;
}
```

3 leetCode-Find if Path Exists in Graph

Link

This problem is based on finding a **connected component** in a graph.
ALgorithm used: **dfs**

```
1
2  class Solution {
3
4       void dfs(int source ,vector<int>adj[], vector<bool>&visit )
5      {
6          visit[source]=true ;
7          for(auto child: adj[source]) //traverse adjacent nodes  of source
8          {
9              if(!visit[child])
10             {
11                 visit[child]=true ;
12                 dfs(child,adj,visit);
13             }
14         }
15     }
16 public:
17     bool validPath(int n, vector<vector<int>>& edges, int source, int destination) {
18
19         int x=edges.size() ;
20         vector<int>adj[n];
21         vector<bool>visit(n,false) ;
22
23         for(int i=0 ;i<x ;i++)
24         {
25             adj[edges[i][0]].push_back(edges[i][1]);
26             adj[edges[i][1]].push_back(edges[i][0]);
27
28         }
29
30
31                 dfs(source,adj,visit); // traverse from source if found destination return true
32
33                 if( (visit[source] && !visit[destination]) ||  (!visit[source] &&visit[destination
                       ]))
34                 {
35                     return false ;
36                 }
37
38
39          return true ;
40
41
42     }
43 };
```

4 CodeForces-Bmail Computer Network

Link

This problem is based on finding a **Connected path from source to destination** in a graph.
Algorithm used: **BFS**

```cpp
#include<bits/stdc++.h>
using namespace std ;
const int N=200000 ;
int visited[N];

void bfs(vector<int>graph[],int parent[],int source)
{
    queue<int>q ;
    q.push(source);
    visited[source]=1 ;
    parent[source]=-1 ;
    while(!q.empty())
    {
        int v=q.front();
        for(auto u: graph[v])
        {
            if(!visited[u])
            {
                parent[u]=v ;
                visited[u]=true ;
                q.push(u);

            }
        }
           q.pop() ;
    }

}

void path(int parent[],int src) // using recursion for finding parent ( destination to source)
{

    if(parent[src]==-1){
            printf("%d ",1);
        return ;
    }


   path(parent ,parent[src]);
   printf("%d ",src);
}

int main()
{
 int n ;
 cin>>n ;
 vector<int>graph[n+5];

 int parent[n]={} ;
 //8
 //1 1 2 2 3 2 5

 for(int i=2 ;i<=n ;i++)
 {
     int a;
     cin>>a ;
     graph[i].push_back(a);
     graph[a].push_back(i);
 }

 bfs(graph,parent,1) ;
 path(parent,n);


```

```
 //** you can use this path(parent,n) method  or this block of code to print path **

 /*
 int src=n ;
 set<int>path ;
 while(src!=-1)
 {
     path.insert(src);
     src=parent[src] ;
 }
 for(auto u: path)
    cout<<u<<" " ;*/


}
```

5 LeetCode - Convert Sorted Array to Binary Search Tree

Link

This problem is based on **convert a array to binary search tree**
Algorithm used: **Binary search Tree**

```
class Solution {
public:
    TreeNode* sortedArrayToBST(vector<int>& nums) {
        int size = nums.size();
        int half = size/2;

        if(size == 0)
            return NULL; // if no child


        TreeNode* root = new TreeNode(nums[half]);
        //create a root node
        if(size == 1)
            return root;

          //left child ( start to half of nums vector )
        vector<int>left(nums.begin(), nums.begin() + half);
        //right child( half to end  of nums vector )
        vector<int>right(nums.begin() + half + 1, nums.end());
        root->left = sortedArrayToBST(left);
        root->right = sortedArrayToBST(right);

        return root;
    }
};
```

6 Uva - Dividing coins

Link

This problem is based on **optimization technique**
Algorithm used: **0/1 knapSack**

```cpp

#include<bits/stdc++.h>
using namespace std ;
const int N = 110;
const int MAXN = 102*500;
int weight[N];
int dp[N][MAXN] ;

int sum ;

//used 2D array

int knapSack(int m)
{
    int max_weight=sum/2 ; // here max capacity will be half of  total weight

    for(int i=0; i <=max_weight; i++)
     dp[0][i] = 0;


    for(int i=1 ;i<=m;i++)
    {
        for(int w=0 ;w<=max_weight ;w++)
        {
            if(weight[i]<=w)
            {
               dp[i][w]=max(dp[i-1][w],(dp[i-1][w-weight[i]]+weight[i]));
            }
            else
            {
                dp[i][w]=dp[i-1][w] ;
            }

        }
    }


    return dp[m][max_weight];
}

//using 1D Array (memory optimization)
//int  knapSack(int m)
//{
//    int w=sum/2 ;
//  for(int i=1 ;i<=m ;i++)
//  {
//      for(int j=w ;j>0 ;j--)
//      {
//          if(weight[i]<=j)
//          {
//              dp[j]=max(dp[j],weight[i]+dp[j-weight[i]]);
//          }
//      }
//  }
//    return dp[w];
//}
//

int main()
{

int t ;
scanf("%d",&t);
while(t--)
{
int m ;
```

```
scanf("%d",&m);

sum=0 ;
for(int i=1 ;i<=m ;i++)
{
    scanf("%d",&weight[i]) ;
    sum+=weight[i] ;
}

  printf("%d\n", sum - 2*knapSack(m) );
}
return 0 ;
}
```

7 Codeforces-KnapSack

Link

This problem is based on **greedy technique**
Algorithm used:

```
#include<bits/stdc++.h>
using namespace std ;


long long  total_weight ;
long long  oneItem ;
long long  weight ;
vector<int>items ;

int main()
{

    int t ;
    cin>>t ;
    while(t--)
    {
        long long n,w ;
        cin>>n>>w ;


        total_weight=0 ;
        oneItem=0 ;

        for(int i=1 ; i<=n ; i++)
        {
            cin>>weight ;
            if(weight <= w )
            {
                if(weight>=(w+1)/2)
                {
                    oneItem=i ;
                }
                else if ( total_weight < (w+1)/2)
                {
                    items.push_back(i);
                    total_weight+=weight ;
                }
            }

        }

        if(oneItem>0)
        {
```

```
            cout<<1<<'\n'<<oneItem ;

        }
        else
        {
            if(total_weight>=(w+1)/2)
            {
                cout<<items.size()<<endl ;
                for(auto u: items)
                    cout<<u<<" " ;

            }
            else
            {
                cout<<-1 ;
            }
        }

        cout<<endl;
        items.clear();

    }

}

\begin{figure}[H]
\end{figure}
```

8 LeetCode-Is Subsequence

Link

This problem is based on **Longest common subsequence**
Algorithm used: **LCS**

```
class Solution {
public:
    bool isSubsequence(string s, string t) {

        int sub=s.size();
        int  original=t.size();

        int dp[sub+1][original+1];

        for(int i=0 ;i<=sub ;i++)
        dp[i][0]=0 ; //fill first row and first column with zero
        for(int i=0 ; i<=original;i++)
        dp[0][i]=0 ;

        for(int i=1 ;i<=sub;i++)
        {
            for(int j=1 ;j<=original ;j++)
            {
                if(s[i-1]==t[j-1])
                {
                    dp[i][j]=1+dp[i-1][j-1];//if match subsequence with original string
                }
                else
                {
                    dp[i][j]=max(dp[i][j-1],dp[i-1][j]);// if not match
                }
            }
        }

```

```
30          if(dp[sub][original]==sub)
31          return true ;
32          else
33          return false ;
34
35      }
36  };
```