EASY :

MEDIUM :

1. Vertex Cover Problem

```
#include<bits/stdc++.h>
using namespace std;
vector<int>adj[10005];
vector<bool>visited[10005];
void addEdge(int v, int w)
{
     adj[v].push_back(w);
     adj[w].push_back(v);
}

void printVertexCover(int V)
{
     bool visited[V];
     for (int i=0; i<V; i++)
          visited[i] = false;
     for (int u=0; u<V; u++)
    {
          if (visited[u] == false)
          {
               for (auto it:adj[u])
               {
                    if (visited[it] == false)
                   {
                    visited[it] = true;
                         visited[u] = true;
                         break;
                    }
               }
          }
     }
     for(int i=0; i<V; i++)
     if(visited[i])
     cout<<i<< " ";
}
```

```
int main()
{
      addEdge(0, 1);
      addEdge(0, 2);
      addEdge(1, 3);
      addEdge(3, 4);
      addEdge(4, 5);
      addEdge(5, 6);
      printVertexCover(7);

      return 0;
}
```

HARD :

1. Min cost path .

```
class Point
{
    public:
    int x;
    int y;
    int cost;
};
struct comp{
    bool operator()(Point const& p1, Point const& p2)
    {
        return p1.cost>p2.cost;
    }
};

int dx[4]={0,1,0,-1};
int dy[4]={1,0,-1,0};

int Solution::solve(int r, int c, vector<string> &mat) {

    string s="RDLU";
```

```
    vector<vector<int>>dist(r,vector<int>(c,INT_MAX-1));

    priority_queue<Point,vector<Point>,comp>pq;

    pq.push({0,0,0});
    dist[0][0]=0;

    while(!pq.empty())
    {
        Point temp=pq.top();
        pq.pop();
        if(temp.x==r-1 && temp.y==c-1)
        return dist[temp.x][temp.y];

        for(int i=0;i<4;i++)
        {
            int new_x=temp.x+dx[i];
            int new_y=temp.y+dy[i];
            int cost=dist[temp.x][temp.y];
            if(s[i]!=mat[temp.x][temp.y]) cost++;

            if(new_x>=0 && new_y>=0 && new_x<r && new_y<c &&
cost<dist[new_x][new_y])
            {
                pq.push({new_x,new_y,cost});
                dist[new_x][new_y]=cost;
            }
        }
    }
    return dist[r-1][c-1];
}
```