KUET_Phoenix

Contents

1 Include

```
/* BISMILLAHIR RAHMANIR RAHIM */
#include<bits/stdc++.h>
#include<cstdio>
#include<sstream>
#include<cstdlib>
#include<cctype>
#include<cmath>
#include<algorithm>
#include<set>
#include<queue>
#include<stack>
#include<list>
#include<iostream>
#include<fstream>
#include<numeric>
#include<string>
#include<vector>
#include<cstring>
#include<map>
#include<iterator>
using namespace std;

#define READ          freopen("in.txt", "r", stdin);
#define WRITE         freopen("out.txt", "w", stdout);
#define ll            int long long
#define ull           unsigned long long
#define ld            long double
#define lld           long long double
#define pi            acos(-1)
#define pb            push_back
#define pff           push_front
#define pbk           pop_back
#define pfk           pop_front
#define mp            make_pair
#define gcd(a,b)      __gcd(a,b)
#define lcm(a, b)     ((a)*((b)/gcd(a,b)))
#define dist(ax,ay,bx,by)
    sqrt((ax-bx)*(ax-bx)+(ay-by)*(ay-by))
#define sf(a)         scanf("%d",&a)
#define sfl(a)        scanf("%lld",&a)
#define sff(a,b)      scanf("%d %d",&a,&b)
```

```
#define sffl(a,b)      scanf("%lld %lld",&a,&b)
#define sfff(a,b,c)    scanf("%d %d %d",&a,&b,&c)
#define sfffl(a,b,c)   scanf("%lld %lld %lld",&a,&b,&c)
#define loop(i,a,b,x)  for(__typeof(b) i=a;i<=b;i+=x)
#define rloop(i,b,a,x) for(__typeof(b) i=b;i>=a;i-=x)
#define lead_zero(x)   __builtin_clzll(x)
#define trail_zero(x)  __builtin_ctz(x)
#define total_1s(x)    __builtin_popcount(x)
#define first_1(x)     __builtin_ffs(x)
#define log2_(x)       __builtin_clz(1) - __builtin_clz(x)
#define Q             int test; scanf("%d", &test);for
    (int z = 1; z<= test; z++)
#define PRINT_CASE     printf("Case %d: ",z)
#define LINE_PRINT_CASE printf("Case %d:\n",z)
#define FAST           ios_base::sync_with_stdio(0);
    cin.tie(0);
#define pf             printf
#define ff             first
#define ss             second
#define all(v)         v.begin(),v.end()
#define SZ(a)          (int)a.size()
/* #### check these paramters before submit ##### */
const int INF = 0x3f3f3f3f;/* useful for memset*/
const ll LL_INF = 0x3f3f3f3f3f3f3f3f;
const int mx = 1e5+5; /*CHECK here for every problem*/
const int mod = 1e9+7
/*-----------------------Graph Moves----------------*/
///const int fx[]={+1,-1,+0,+0};
///const int fy[]={+0,+0,+1,-1};
///const int fx[]={+0,+0,+1,-1,-1,+1,-1,+1}; // Kings Move
///const int fy[]={-1,+1,+0,+0,+1,+1,-1,-1}; // Kings Move
///const int fx[]={-2, -2, -1, -1, 1, 1, 2, 2}; //
    Knights Move
///const int fy[]={-1, 1, -2, 2, -2, 2, -1, 1}; //
    Knights Move
 //int day[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31,
     30, 31};
/*------------------------Bitmask------------------*/
///int Set(int N,int pos){return N=N | (1<<pos);}
///int reset(int N,int pos){return N= N & ~(1<<pos);}
///bool check(int N,int pos){return (bool)(N & (1<<pos));}
```

2 Graph

2.1 DFS

```
void dfs(ll start)
{
    cnt++;
    for(ll i=0; i<adj[start].size(); i++)
    {
        int v=adj[start][i];
        if(!visited[v])
        {
            visited[v]=1;
            dfs(v);
        }
    }

}
```

2.2 BFS

```
vector<int>adj[mx];
int visited[mx];
int cost[mx];

void bfs(int source)
{

    cout<<"bfs"<<endl;
    visited[source]=1;
    cnt++;
    queue<int>q;
    q.push(source);
    while(!q.empty())
    {
        int u=q.front();
        q.pop();
        for(int i=0; i<adj[u].size(); i++)
        {
            int  v=adj[u][i];

            if(visited[v]==0)
            {
                visited[v]=1;
                q.push(v);
                //cnt++;
            }
        }
    }
}
```

2.3 BFS with move (bombs no there mines)

```
int R,C;

void bfs(int sx,int sy){
    int ux,uy,vx,vy;
    visited[sx][sy]=true;
    dis[sx][sy]=0;
    queue<int>q;
    q.push(sx);
    q.push(sy);
    while(!q.empty())
    {
        ux=q.front();
        q.pop();
        uy=q.front();
        q.pop();
        for(int i=0;i<4;i++){
            vx=ux+fx[i];
            vy=uy+fy[i];
            if((vx>=0&&vx<=R)&&(vy>=0&&vy<=C)&&
                adj[vx][vy]==0){
                if(!visited[vx][vy]&&!dis[vx][vy]){
                    visited[vx][vy]=true;
                    dis[vx][vy]=dis[ux][uy]+1;
                    q.push(vx);
                    q.push(vy);
                }
            }
        }
    }
}
```

2.4 Dijkstra

```
//vector<pii>adj[mx];
//int visited[mx];
//int dis[mx];
int parent[mx];
void path(int n)
{
        if(n==1)
        {
            cout<<"1";
        return;
      }
      path(parent[n]);
      cout<<" "<<n;
}
void dijkstra(int s)
{
    CLR(visited);
    for(int i=0;i<100005;i++) dis[i]=inf;
    priority_queue<pii,vector<pii>,greater<pii>>pq;
    dis[s]=0;
    pq.push({dis[s],s});

    while(!pq.empty())
    {
        int u=pq.top().second; pq.pop();
        if(visited[u]) continue;
        visited[u]=1;
        for(int i=0;i<adj[u].size();i++){
            int vx=adj[u][i].first;
            int vy=adj[u][i].second;
            if(dis[vx]>dis[u]+vy)
            {
                dis[vx]=dis[u]+vy;
                pq.push({dis[vx],vx});
                parent[vx]=u; ///path print korte lgbe
            }
        }
    }
}
```

2.5 Dijkstra 2nd Shortest Path Length

```
#define maxn 5009
#define INF 1000000000000
vector<pair<lli,lli> > adj[maxn];
void dijkstra(lli src,lli n,vector<pair<lli,lli> > adj[])
{
    vector<lli>shortest(maxn,INF);
    vector<lli>sec_shortest(maxn,INF);
    priority_queue<pair<lli,lli> >q;
    shortest[src]=0;
    q.push({0,src});
    pair<lli,lli>uu,a;
    lli u,v,c;
    while(!q.empty())
    {
        uu=q.top();
        v=-uu.first;
        u=uu.second;
        q.pop();
        for(lli ii=0; ii<adj[u].size(); ii++)
        {
            a=adj[u][ii];
            c=v+adj[u][ii].first;
            if(shortest[a.second]>c)
            {
                /*second shortest path er jnno ar ekta
                    vector nibo then
                shortest path er path relaxation er
                    condition e ektu change korbo*/
                sec_shortest[a.second]=shortest[a.second];
                shortest[a.second]=c;
                q.push({-shortest[a.second],a.second});
            }
            else if(sec_shortest[a.second]>c &&
                 shortest[a.second]<c)
            {
                sec_shortest[a.second]=c;
                q.push({-sec_shortest[a.second],a.second});
            }}}
    cout<<sec_shortest[n]<<endl;
}
```

2.6 DSU

```
#define mx 12505
int parent[mx],n,edges;
pair<lli,pair<int,int>>pi[mx];

void set_parent()
{
    for(int i=1; i<=n; i++)
    {
        parent[i]=i;
    }

}

int find_parent(int r)
{
    return (parent[r]==r)?r:find_parent(parent[r]);
}

void unionm(int x,int y)
{
    int u=find_parent(x);
    int v=find_parent(y);
    if(u!=v)
    {
        parent[v]=u;
    }
}
```

2.7 MST(kruskal)

```
#define mx 10005
int parent[mx],n,edges;


struct edge
{
    int u,v,w;
    edge(int a,int b,int c)
    {
       u=a;
       v=b;w=c;
    }

};

bool comp(edge f,edge s)
{
    return f.w>s.w;
}

vector<edge>vec;
int find(int p)
{
    if(parent[p]==p)
        return p;
    else
        return parent[p]=find(parent[p]);
}

int kruskal()
{
    sort(vec.begin(),vec.end(),comp);
    for(int i=1; i<=n; i++)
    {
        parent[i]=i;
    }
    int xx,yy;
    int cnt=0;
    for(int i=0; i<vec.size(); i++)
    {
        xx=find(vec[i].u);
        yy=find(vec[i].v);
        if(xx!=yy)
        {
            parent[yy]=xx;

        }
        else
        {
            cnt+=vec[i].w;
            //cout<<"cnt "<<cnt<<endl;
        }
    }
    vec.clear();
    return cnt;
}
```

2.8 BellmanFord

```
vector<int>adj[2000];
int dis[2000],parent[2000];
int edge,node;
int Bellman_Ford(int source)
{
    for(int i=0; i<2000; i++)
    {
        parent[i] = -1 ;
        dis[i] = 2e9;
    }
    dis[source]=0;

    for(int i=1; i<node; i++)
    {
        for(int j=0; adj[j].size()!=0; j++)
        {
            int u=adj[j][0];
            int v=adj[j][1];
            if(dis[u]+adj[j][2]<dis[v])
            {
                dis[v]= dis[u]+adj[j][2],parent[v]=u;
            }
        }
    }
    int flag=1;
    for(int j=0; adj[j].size()!=0; j++)
    {
        int u=adj[j][0];
        int v=adj[j][1];
        if(dis[u]+adj[j][2]<dis[v])
            flag=0;
    }
    if(flag)
        for(int i=1; i<=node; i++)
        {
            printf("Vertex %d -> cost = %d parent = %d\n"
                                ,i,dis[i],parent[i]);
        }
}

int main()
{
    cin>>node>>edge;

    for(int i = 0; i<edge; i++)
    {
        int from,next,weight;
        cin>>from>>next>>weight;

        adj[i].pb(from);
        adj[i].pb(next);
        adj[i].pb(weight);
    }
    Bellman_Ford(1);

    return 0;
}
```

2.9 Articulation Bridge

```
vector<int>adj[mx];
int visited[mx],node[mx],low[mx],
    parent[mx],disTime[mx],discover_time,root;
set<pii>s;

void articulation_bridge(int u)
{
    visited[u]=1;
    int cnt=0;
    disTime[u]=low[u]=++discover_time;
    for(int i=0; i<adj[u].size(); i++)
    {
        int v=adj[u][i];
        if(parent[u]==v)
            continue;
        if(visited[v])
            low[u]=min(low[u],disTime[v]); //back edge
                thakle
        else
        {
            parent[v]=u;
            articulation_point(v);
            low[u]=min(low[u],low[v]);
            if(disTime[u]<low[v])
            {
                s.insert({min(u,v),max(u,v)});
                //cout<<u<<" "<<v<<endl;
                //node[u]=1;
            }

        }


    }


}

int main()
{
        s.clear();
        int node;
        sf("%d",&node);
        for(int i=1; i<=node; i++)
        {
            int u,n;
            char c1,c2;
            cin>>u;
            getchar();
            cin>>c1>>n>>c2;
            for(int nn=1; nn<=n; nn++)
        {
            int v;
            cin>>v;
            adj[u].pb(v);
        }
    }
    for(int i=0; i<node; i++)
    {
        root=i;
        if(!visited[i])
        {
            articulation_point(i);
        }
    }
}
```

2.10 Articulation Point

```
vector<int>adj[mx];
int visited[mx], node[mx], low[mx], parent[mx],
    disTime[mx], discover_time, root;


void articulation_point(int u)
{
    visited[u]=1;
    int cnt=0;
    disTime[u]=low[u]=++discover_time;
    for(int i=0; i<adj[u].size(); i++)
    {
        int v=adj[u][i];
        if(parent[u]==v)
            continue;
        if(visited[v]) low[u]=min(low[u],disTime[v]);
             //back edge thakle
         else
        {
            parent[v]=u;
            articulation_point(v);
            low[u]=min(low[u],low[v]);
            if(disTime[u]<=low[v] && u!=root)
            {
                node[u]=1;
            }
           cnt++;
        }


    }
    if(cnt>1 && u==root)
        node[u]=1;

}

int main()
{
        memset(visited,0,sizeof visited);
        memset(node,0,sizeof node);
        memset(low,0,sizeof low);
        memset(disTime,0,sizeof disTime);
        discover_time=0;
        for(int i=0; i<mx; i++)
            adj[i].clear();
        int edge,n;
        sf("%d%d",&n,&edge);
        for(int i=1; i<=edge; i++)
        {
            int u,v;
           sf("%d%d",&u,&v);
            adj[u].pb(v);
            adj[v].pb(u);
        }
        root=1;
        articulation_point(1);
}
```

2.11 LCA for different root

```
#define mx 200005
int L[mx];
int P[mx][50];
int T[mx];
vector<int>g[mx];
void dfs(int from,int u,int dep)
{
    T[u]=from;
    L[u]=dep;
    for(int i=0; i<(int)g[u].size(); i++)
    {
        int v=g[u][i];
        if(v==from)
            continue;
        dfs(u,v,dep+1);
    }
}

int lca_query(int N, int p, int q)
{
    int tmp, log, i;

    if (L[p] < L[q])
        tmp = p, p = q, q = tmp;

    log=1;
    while(1)
    {
        int next=log+1;
        if((1<<next)>L[p])
            break;
        log++;

    }

    for (i = log; i >= 0; i--)
        if (L[p] - (1 << i) >= L[q])
            p = P[p][i];

    if (p == q)
        return p;

    for (i = log; i >= 0; i--)
        if (P[p][i] != -1 && P[p][i] != P[q][i])
            p = P[p][i], q = P[q][i];

    return T[p];
}

void lca_init(int N)
{
    memset (P,-1,sizeof(P));
    int i, j;
    for (i = 0; i < N; i++)
        P[i][0] = T[i];

    for (j = 1; 1 << j < N; j++)
        for (i = 0; i < N; i++)
            if (P[i][j - 1] != -1)
                P[i][j] = P[P[i][j - 1]][j - 1];
}

int main()
{

    for(int i=0; i<=mx; i++)
        g[i].clear();
    int n;
    sf("%d",&n);
    for(int i=1; i<n; i++)
    {
        int u,v;
        sf("%d %d",&u,&v);
        g[u-1].pb(v-1);
        g[v-1].pb(u-1);
    }

    dfs (0, 0,0);
    lca_init(n);
    int q;
    sf("%d",&q);
    while(q--)
    {
        int r,a,b,ans;
        scanf("%d %d %d",&r,&a,&b);

            int f=lca_query(n,r-1,a-1);
            int s=lca_query(n,r-1,b-1);
            int t=lca_query(n,a-1,b-1);
            if(f!=s && f!=t && s==t)
            {
                ans=f;
            }
            else if(s!=f && s!=t && f==t)
                ans=s;
            else if(t!=f && t!=s && f==s)
                ans=t;
            else ans=t;

        printf("%d\n",ans+1);
    }


}
```

2.12 Topsort

```
#define mx 100005
int n,m,f=0;
vector<int>adj[mx],arr;
bool vis[mx],finish[mx];

void topsort(int s){
    vis[s]=true;

    for(int v:adj[s])
    {
        if( !vis[v])
            topsort(v);
        else if(vis[v] && finish[v]==0) f=1;

    }
    finish[s]=1;
    arr.push_back(s+1);


}
```

2.13 single source multiple shortest path

```
#define int long long int
#define pii pair<int,int>
const int inf=9e15;
int node,edge,k;
vector<pair<int,int>>adj[200005];
// bool vis[200005]; // as a node can be visited many time
vector<vector<int>>dis;

void dijkstra(int s)
{
    dis.resize(200005);
    for(int i=0;i<200005;i++)
    {
        dis[i].resize(k);
        for(int j=0;j<k;j++){
            dis[i][j]=inf;
        }
    }
    priority_queue<pii,vector<pii>,greater<pii>>pq;

    pq.push({0,s});

    while(!pq.empty())
    {
        int u=pq.top().second;
        int d= pq.top().first;
         pq.pop();

        if(dis[u][k-1]<d) continue;
        for(auto x:adj[u]){
            int vx=x.first;
            int vy=x.second;
            if(dis[vx][k-1]>d+vy)
            {
                dis[vx][k-1]=d+vy;

                pq.push({dis[vx][k-1],vx});
                sort(dis[vx].begin(),dis[vx].end());

            }
        }
    }
}
```

2.14 SCC

```
vector<int>v[MX],vr[MX];
vector<pii>vp;
bool vis[MX],vis2[MX];
int str[MX],fin[MX],r,p;
//1003-Drunk
void dfs1(int n)
{
    if(vis[n]==1) return;
    vis[n]=1;
    str[n]=++r;
    for(int i=0;i<v[n].size();i++)
    {
        dfs1(v[n][i]);
    }
    fin[n]=++r;
}
void dfs2(int n)
{
    if(vis2[n]==1) return;
    vis2[n]=1;
    for(int i=0;i<vr[n].size();i++)
    {
            dfs2(vr[n][i]);
    }
}
int main()
{
    int t,i,n,m,j,k;
    cin>>t;
    for(int y=1;y<=t;y++)
    {
        k=1,r=0,p=0;
        map<string ,int>ms;
        memset(vis,0,sizeof vis);
        memset(vis2,0,sizeof vis2);
        cin>>m;
        string a,b;
        for(i=0;i<m;i++)
        {
            cin>>a>>b;
            if(ms[a]==0)
            {
              ms[a]=k;
              k++;
            }
            if(ms[b]==0)
            {
              ms[b]=k;
              k++;
            }
            v[ms[a]].pb(ms[b]);
            vr[ms[b]].pb(ms[a]);
        }
        for(i=1;i<k;i++)
        {if(vis[i]==0)
        dfs1(i);}
        for(i=1;i<k;i++)
        {
           vp.pb(mp(fin[i],i));
        }
        sort(vp);
        for(i=vp.size()-1;i>=0;i--)
        {
          if(vis2[vp[i].second]==0)
          {
              dfs2(vp[i].ss);
              p++;
          }
        }
        cout<<"Case "<<y<<": ";
        if(p==(k-1))cout<<"Yes"<<endl;
        else cout<<"No"<<endl;
       for(i=1;i<k;i++)
       {
           v[i].clear();
           vr[i].clear();
       }
       vp.clear();
    }
}
```

2.15 Cycle_found(Undirected)

```
#define ll long long int
#define mx 100005
bool vis[mx];
int n,m,p[mx];

void cycle(int u,int pu=-1){//here pu means parent
      p[u]=pu;
      vis[u] = 1;
      for(int v:adj[u]){
            if(v==pu)
                  continue;
            if(vis[v]){
                  int u2 = u;
                  while(u^v){
                      ans.push_back(u);
                      u = p[u];
                     ///same hoile u^v=0 hoile
                  }
                  ans.push_back(v);
                  ans.push_back(u2);
                  cout<<ans.size()<<endl;
                  for(int a:ans)
                     cout << a+1 <<" ";
                  exit(0);
            }else{
                dfs(v,u);
            }
      }

}
```

2.16 Cycle_found(Directed)

```
#define ll long long int
#define mx 100005
vector<int>adj[mx],ans;
bool vis[mx];
int color[mx];
int n,m,p[mx];

void dfs(int u,int pu=-1){//here pu means parent
      p[u]=pu;
      vis[u] = 1;
      for(int v:adj[u]){
            if(vis[v] == 1 && color[v]!=2){
                  int u2 = u;
                  while(u^v){
                      ans.push_back(u);
                      u = p[u];
                     ///same hoile u^v=0 hoile
                  }
                  ans.push_back(v);
                  ans.push_back(u2);
                  cout<<ans.size()<<endl;
                  reverse(ans.begin(),ans.end());
                  for(int a:ans)
                     cout << a+1 <<" ";
                  exit(0);
            }else{
                dfs(v,u);
            }
      }
      color[u]=2;

}
```

2.17 Path Print

```
char adj[1005][1005];
int visited[1005][1005];
const int fx[]= {+1,-1,+0,+0};
const int fy[]= {+0,+0,+1,-1};
string path="";
int R,C;
map<pair<int,int>,pair<pair<int,int>,char> >parent;
string bfs(int sx,int sy,int dx,int dy){
```

```
int ux,uy,vx,vy;
visited[sx][sy]=true;

queue<int>q;
q.push(sx);
q.push(sy);
while(!q.empty())
{
    ux=q.front();
    q.pop();
    uy=q.front();
    q.pop();
    //cout<<ux<<" "<<uy<<endl;
    for(int i=0; i<4; i++)
    {
        vx=ux+fx[i];
        vy=uy+fy[i];
        if((vx>=0&&vx<R)&&(vy>=0&&vy<C)&& adj[vx][vy]!='#')
        {
            char dir ;
            if(i==0) dir = 'D';
            if(i==1) dir = 'U';
            if(i==2) dir = 'R';
            if(i==3) dir = 'L';
            if(!visited[vx][vy])
            {
                visited[vx][vy]=true;
                parent[
                     {vx,vy}]=make_pair(make_pair(ux,uy),dir);
                if(vx == dx && vy == dy)
                {
                    auto end = make_pair(vx,vy);
                    while(true)
                    {
                        path += parent[end].second;
                        end=parent[end].first;
                        if(end.first == sx && end.second ==
                             sy)
                            return path;
                    }

                }
                q.push(vx);
                q.push(vy);
            }
        }
    }
}

return path;
}
```

2.18 Edmond Karp

```
#include<bits/stdc++.h>
using namespace std; //lightoj
#define maxn 205    //diagonal sum
int capacity[maxn][maxn];
vector<int>adjac[maxn];
int edmondkarp(int s,int t,int parent[])
{for(int i=0;i<=maxn;i++)
   {parent[i]=-1;} //bfs
    parent[s] = -3;
    queue<pair<int,int> >q;
    q.push({s,INT_MAX});
    while(!q.empty())
    {
        int now = q.front().first;
        int flow = q.front().second;
        q.pop();
        for(int next : adjac[now])
        {
            if(parent[next]== -1 && capacity[now][next])
            {parent[next]= now;
            int nflow = min(flow,capacity[now][next]);
            if(next == t)
                return nflow;
            q.push({next,nflow});}}}}
int main()
{
    int t;
    scanf("%d",&t);
    for(int tt=1; tt<=t; tt++)
    {int source=0,sink=202,a,b,n,m,d,i,j,k;
        memset(capacity,0,sizeof capacity);
        scanf("%d",&n);
        scanf("%d",&m);
        int
             ara1[n+10][m+10],ara2[n+10][m+10],row1[maxn],row2[
        d=n+m-1;
        for(i=1; i<=d; i++)
        {
            scanf("%d",&row1[i]);
            capacity[source][i]=row1[i];
            adjac[i].push_back(source);
            adjac[source].push_back(i);
            if(i>n)
            {
                k=i-n+1; j=n;
            }
            else
            {
                k=1; j=i;
            }
            while(j>=1 && k<=m)
            {
                ara1[j][k] = i; j--; k++;
            }}
        for(i=1; i<=d; i++)
        {
            cin>>row2[i];
            adjac[i+102].push_back(sink);
            adjac[sink].push_back(i+102);
            capacity[i+102][sink] = row2[i];
            if(i>n)
            {
                j=n;  k=m-(i-n+1)+1;}
            else
            {
                j=i;k=m;
            }
            while(j>=1 && k>=1)
            {
                ara2[j][k] = i+102; j--; k--; }}
        for(i=1; i<=n; i++)
        {for(j=1; j<=m; j++)
            {
adjac[ara1[i][j]].push_back(ara2[i][j]);
adjac[ara2[i][j]].push_back(ara1[i][j]);
capacity[ara1[i][j]][ara2[i][j]] = 99;
capacity[source][ara1[i][j]] -= 1;
capacity[ara2[i][j]][sink] -= 1;}}
    int parent[maxn],nflow;
    while(nflow = edmondkarp(source,sink,parent))
    {
        int now = sink;
        while(now!=source)
        {
            int pre = parent[now];
            capacity[pre][now] -= nflow;
            capacity[now][pre] += nflow;
            now = pre;
        }}
        printf("Case %d:\n",tt);
        for(i=1; i<=n; i++)
        {for(j=1; j<=m; j++)
            {printf("%d",capacity[ara2[i][j]][ara1[i][j]]+1);
            if(j!=m)
              cout<<" ";
            }
            cout<<endl;
        }
        for(i=0; i<maxn; i++)
          adjac[i].clear();
    }}
```

2.19 Dinic

```
map<pair<int,int>,int>mp;
struct FlowEdge
{
    int u,v;
    long long int cap,flow=0;
    FlowEdge(int u,int v,long long int cap) :
        u(u),v(v),cap(cap) {}
};
struct Dinic {
    const long long flow_inf = 1e18;
    vector<FlowEdge> edges;
    vector<vector<int>> adj;
    int n, m = 0;
    int s, t;
    vector<int> level, ptr;
    queue<int> q;

    Dinic(int n, int s, int t) : n(n), s(s), t(t) {
        adj.resize(nn);
        level.resize(nn);
        ptr.resize(nn);
    }
    void add_edge(int u, int v, long long cap) {
        edges.emplace_back(u, v, cap);
        edges.emplace_back(v, u, 0);
        adj[u].push_back(m);
        adj[v].push_back(m + 1);
        mp[{u,v}]=m;
        mp[{v,u}]=m+1;
        m += 2;
    }
     bool bfs() {
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            for (int id : adj[u]) {
                if (edges[id].cap - edges[id].flow < 1)
                    continue;
                if (level[edges[id].v] != -1)
                    continue;
                level[edges[id].v] = level[u] + 1;
                q.push(edges[id].v);
            }
        }
        return level[t] != -1;
    }
    long long dfs(int u, long long pushed) {
        if (pushed == 0)
            return 0;
        if (u == t)
            return pushed;
        for (int& cid = ptr[u]; cid < (int)adj[u].size();
         cid++) {
            int id = adj[u][cid];
            int v = edges[id].v;
            if (level[u] + 1 != level[v] || edges[id].cap
             - edges[id].flow < 1)
                continue;
            long long tr = dfs(v, min(pushed,
             edges[id].cap - edges[id].flow));
            if (tr == 0)
                continue;
            edges[id].flow += tr;
            edges[id ^ 1].flow -= tr;
            return tr;
        }
        return 0;
    }
    long long flow() {
        long long f = 0;
        while (true) {
            fill(level.begin(), level.end(), -1);
            level[s] = 0;
            q.push(s);
            if (!bfs())
                break;
            fill(ptr.begin(), ptr.end(), 0);
            while (long long pushed = dfs(s, flow_inf)) {
                f += pushed;
            }
        }
        return f;
    }
    long long int flow_gese(int u,int v)
    {
        int id=mp[{u,v}];
        return edges[id].flow;
    }
};

//Dinic maxflow(n,0,500)
```

2.20 Floyd-Warshall

```
#define INF 100000000
/*map<string,string>::iterator it;
   it=ma.find(s3);
   if(it!=ma.end())
   {cout<<it->second<<endl;}
map<vector<string>, int> mp;
mp[{"***","* *","* *","* *","***"}]=0
mp[{"  *","  *","  *","  *","  *"}]=1
mp[{"***","  *","***","*  ","***"}]=2
mp[{"***","  *","***","  *","***"}]=3
mp[{"* *","* *","***","  *","  *"}]=4
mp[{"***","*  ","***","  *","***"}]=5
mp[{"***","*  ","***","* *","***"}]=6
mp[{"***","  *","  *","  *","  *"}]=7
mp[{"***","* *","***","* *","***"}]=8
mp[{"***","* *","***","  *","***"}]=9
vector<string>v(5);
vector<vector<string> >num((v[0].size()+2)/4)
*/
int A[102][102];
vector<pair<int,int>>edges;
for( i=1; i<=n; i++)
{for(j=1; j<=n; j++)
```

```
 {A[i][j]=INF;
}}
for(i=0; i<edges.size(); i++)
{int u,v;
  u=edges[i].first;
  v=edges[i].second;
  A[u][v]=1;
}
for(k=1;k<=n;k++)
{for(i=1;i<=n;i++)
  {for(j=1;j<=n;j++)
   {A[i][j]=min(A[i][j],A[i][k]+A[k][j]);}}}
```

3 Number Theory

3.1 Seieve

```
#define mx 100000005

int isprime[mx];
vector<int>prime;

void seive()
{
    for(int i=4; i<mx; i+=2){
        isprime[i]=1;
    }
    isprime[0]=isprime[1]=1;
    for(int i=3; i*i<mx; i+=2){
        if(isprime[i]==0){
            for(int j=i*i; j<mx; j+=i+i)
                isprime[j]=1;
        }
    }prime.push_back(2);
    for(int i=3;i<mx;i+=2){
        if(!isprime[i]) prime.push_back(i);
    }
}
```

3.2 Big Mod

```
//we can do modular multiplication inverse
//by it,,if m is prime..-->O(log(p))
//done using concept of bit manipulation,
//faster than recurtion

int bigmod ( int b, int p, int m ) {
    int res = 1 % m, x = b % m;
    while ( p ) {
        if ( p & 1 ) //check if 0th bit is 1
           res = ( res * x ) % m;
        x = ( x * x ) % m;
        p >>= 1; //right shift so that we can work only
            with 0th bit
    }
    return res;
}
```

3.3 Prime factor

```
vector<pair<int,int>>prime_fact;
void prime_factor(int a)
{
    for(int i=0; prime[i]*prime[i]<=a; i++){
        while(a%prime[i]==0){
             a=a/prime[i];
             cnt++;
        }
        if(cnt)
           prime_fact.push_back({prime[i],cnt});
    }
    if(a>1){
        prime_fact.push_back({a,1});
    }

}
```

3.4 how many digit in a factorial

```
int stirling(int n)//base 10
{
        return floor(((n+0.5)*log(n)-n+
        0.5*log(2*pi))/log(10))+1;
}


//or


//int digits_in_factorial(n, b=10){
    //return floor( lgammaf(n+1)/log(b) ) + 1;}//number
        too small

int stirling(n, b=10){
    return floor(
        ((n+0.5)*log(n)-n+0.5*log(2*pi))/log(b))+1;}
                                ///number too large
```

3.5 eular phi

```
ll phi[mxN];
void phi_1_to_mxN() {

    phi[0] = 0;
    phi[1] = 1;
    for (int i = 2; i < mxN; i++)
        phi[i] = i - 1;

    for (int i = 2; i < mxN; i++)
        for (int j = 2 * i; j < mxN; j += i)
             phi[j] -= phi[i];
}


//coprime sum of a number phi[n]/2*n
```

3.6 get power of a number in a factorial

```
int get_power(int p,int q)//p!                q
    power
{
    int i=q,cnt=0;
    while(p/i>=1)
    {
        int d=p/i;
        cnt+=d;
        i*=q;
    }
    cout<<"cnt - "<<cnt<<endl;
    return cnt;
}
```

3.7 number of divisor

```
int number_of_divisor(int n) ///have to generete prime
    using seive
```

```
{
        long long int j=0,div=1;
        while(prime[j]*prime[j]<=n)
         {

                long long int d=1;
                while(n%prime[j]==0)
                {
                    n/=prime[j];
                    d++;
                }

                div*=d;

                j++;
            }

            if(n!=1) div*=2;
            return div;
} //disisor count with 1 and n itself
```

3.8 segmented seieve

```
void segmented_seive(lli l,lli r)
{
    bool isPrime[r-l+1];
    mem(isPrime,1);
    if(l==1)
        isPrime[0]=false;

    for(lli i=0;prime[i]*prime[i]<=r;i++ )
    {
        lli curPrime=prime[i];
        lli base=curPrime*curPrime;
        if(base<l)
            base=((l+curPrime-1)/curPrime)*curPrime;
        for(lli j=base;j<=r;j+=curPrime)
        {
                isPrime[j-l]=false;
        }

    }
    vector<lli>v;
        for(lli i=0;i<=r-l;i++)
        {
                if(isPrime[i]==true)
                    {v.pb(l+i);}
        }
}
```

3.9 bit wise seive

```
int prime[M/64+4];
int check(lli n)
{   return prime[n>>6]&(1<<((n>>1)&31));
}
void seter(lli n)
{   prime[n>>6]|=(1<<((n>>1)&31));
}
void bitwise_sieve(lli p)
{   for(lli i=3; i*i<=p; i+=2)
    {   if(check(i)==0)
        {   for(lli j=i*i; j<=p; j+=(2*i))
                seter(j);
        }
    }
    vprime.pb(2);
    for(lli i=3; i<=p; i+=2)
    {   if(check(i)==0)
        {   vprime.pb(i);
        }
    }
    return;
}
```

3.10 modular Inverse

```
pii extended_euclid(lli a,lli b){
        if(b==0){
                return pii(1,0);
        }else{
                pii d = extended_euclid(b,a%b);
                return pii(d.ss,d.ff-d.ss*(a/b));
        }
}

lli modular_inverse(lli a){
    pii ret = extended_euclid(a,mod);
    return ((ret.ff%mod)+mod)%mod;

}
```

3.11 Chinese Remainder Theorem

```
class ChineseRemainderTheorem {
    vector<pll> equations;
   public:
    void clear() { equations.clear(); }
    /* x = r (mod m)*/
    void addEquation(ll r, ll m) { equations.push_back({r,
        m}); }
    pll solve() {
        if (equations.size() == 0) return {-1, -1};
        ll a1 = equations[0].first;
        ll m1 = equations[0].second;
        a1 %= m1;
        for (int i = 1; i < equations.size(); i++) {
            ll a2 = equations[i].first;
            ll m2 = equations[i].second;

            ll g = __gcd(m1, m2);
            if (a1 % g != a2 % g) return {-1, -1};
            ll p, q;
            ext_gcd(m1 / g, m2 / g, &p, &q);

            ll mod = m1 / g * m2;
            ll x = ((__int128)a1 * (m2 / g) % mod * q %
                mod +
                      (__int128)a2 * (m1 / g) % mod * p %
                          mod) %
                     mod;
            a1 = x;
            if (a1 < 0) a1 += mod;
            m1 = mod;
        }
        return {a1, m1};
    }
};
```

3.12 Binary To Ineger

```
int make_int(string s)//s represent binary form of a
    number
{
    int sum=0;
    reverse(all(s));
    for(int i=0; i<s.size(); i++)
    {
        sum+=(s[i]-'0')*pow(2,i);
    }
    return sum;
}
```

4 Dynamic Problem basic

4.1 0-1 Knapsack

```
ll int weight[2005];
ll int cost[2005];

ll int dp(int w,int n)
{
    ll int k[n+1][w+1];
    for(int i=0;i<=n;i++)
    {
        for(int j=0;j<=w;j++)
        {
            if(i==0 || j==0)
                k[i][j]=0;
            else if(j-weight[i]>=0)
             k[i][j]=max(cost[i]+k[i-1][j-weight[i]],k[i-1][j]);
            else
                k[i][j]=k[i-1][j];
        }
    }
    return k[n][w];
}
```

4.2 Coin Change fixed

```
int coin_change_fiexdLength(int amount,int i)
{
    if(i>=n)
    {
        if(amount==0)
            return 1;
        else
            return 0;
    }
    if(dp[amount][i]!=-1)
        return dp[amount][i];
    ll rep1=0,rep2=0;
    for(int ii=1; ii<=n_coin[i]; ii++)
    {
        if((amount - coin[i]*ii>=0))
            rep1+=coin_change_fiexdLength(amount-coin[i]*ii,i+1);
    }
    rep2+=coin_change_fiexdLength(amount,i+1);
    return dp[amount][i]=(rep1%mod+rep2%mod)%mod;
}
```

4.3 Coin Change infinite

```
int coin[111],n;

int call(int amount)
{

    int dp[n][amount+1];
    for(int j=1; j<=amount; j++)
        {
                if(j%coin[0]==0) dp[0][j]=1;
                else dp[0][j]=0;
        }
    for(int i=1; i<n; i++)
        {
            for(int j=0; j<=amount; j++)
            {

                if(j==0) dp[i][j]=1;
                else if(coin[i]>j)
                {
                    dp[i][j]=(dp[i-1][j])%mod;
                }
                else
                {
                    dp[i][j]=(dp[i-1][j]+dp[i][j-coin[i]])%mod;
                }
                //cout<<dp[i][j];
            }
        }
}
```

4.4 Long Increasing sub-sequence

```
////from Shakil Ahmed's Blog
int input[ 100005] , n ;
void LIS_with_set()
{
    set < int > lis ;
    set < int > :: iterator it ;
    scanf("%d",&n);
    for( int i = 0 ; i < n ; i++ )
    {
        scanf("%d",&input[i]);
        lis.insert( input[i]);
        it = lis.find( input[i]);
        it++;
        /*if its not the end of lis then removing it
            position
        value for better small value on that position */
        if( it != lis.end()) lis.erase(it);
    }
    cout << lis.size() << endl ;
}
//using multiset then apply upper_bound we can find LIS
    for duplicate value .
void LIS_with_multiset()
{
    multiset < int > lis ;
    multiset < int > :: iterator it ;
    scanf("%d",&n);
    for( int i = 0 ; i < n ; i++ )
    {
        scanf("%d",&input[i]);
        lis.insert( input[i]);
        it = lis.upper_bound( input[i]);
        if( it != lis.end()) lis.erase(it);
    }
    cout << lis.size() << endl ;
}
```

4.5 Combination

```
ll dp[mx][mx];
ll nCr(ll n,ll r)
{
        if(r==1)
        {
                return n;
        }
        if(n==r)
        {
                return 1;
        }
        if(dp[n][r]!=-1)
        {
                return dp[n][m];
        }
        else
        {
                dp[n][r]=nCr(n-1,r)+nCr(n-1,r-1);
                return dp[n][r];
        }

}
```

4.6 Total number of sub-array

```
// Function to find number of subarrays
// with sum exactly equal to k.
ll findSubarraySum(vector<ll>arr, int n, ll sum)
{
        unordered_map<ll, ll> prevSum;

        ll res = 0;
        ll currsum = 0;

        for (int i = 0; i < n; i++) {
                currsum += arr[i];
                if (currsum == sum)
                        res++;
                if (prevSum.find(currsum - sum)
                    !=prevSum.end())
                        res += (prevSum[currsum - sum]);
                prevSum[currsum]++;
        }
        return res;
}
```

4.7 fibonacci using Marix exponential

```
#define see(args...) \
{ \
    string _s = #args; replace(_s.begin(), _s.end(), ',',
         ' ');\
    stringstream _ss(_s); istream_iterator<string>
         _it(_ss); err(_it, args);\
}
void err(istream_iterator<string> it) {}
template<typename T, typename... Args>
void err(istream_iterator<string> it, T a, Args... args)
{
    cout<< *it << " = " << a
         <<",\n"[++it==istream_iterator<string>()];
    err(it, args...);
}
const int N = 1e6+9, mod = 1e9+7;
void multiply(ll F[2][2], ll M[2][2]) {
   ll a = (F[0][0] * M[0][0])%mod + (F[0][1] *
        M[1][0])%mod;
   ll b = (F[0][0] * M[0][1])%mod + (F[0][1] *
        M[1][1])%mod;
   ll c = (F[1][0] * M[0][0])%mod + (F[1][1] *
        M[1][0])%mod;
   ll d = (F[1][0] * M[0][1])%mod + (F[1][1] *
        M[1][1])%mod;
   F[0][0] = a%mod;
   F[0][1] = b%mod;
   F[1][0] = c%mod;
   F[1][1] = d%mod;
}
void power(ll F[2][2], ll n) {
   if (n == 0 || n == 1)
      return;
   ll M[2][2] = {{1,1},{1,0}};
   power(F, n / 2);
   multiply(F, F);
   if (n % 2 != 0)
      multiply(F, M);
}
ll fibonacci_matrix(ll n) {
    if(n == 0)return 1;
   ll F[2][2] = {{1,1},{1,0}};
   if (n == 0)
      return 0;
   power(F, n - 1);
   return F[0][0] % mod;
}
int main()
{
    cin >> n;
    cout<<fibonacci_matrix(n);

    return 0;
}
```

5 pbds

```
//https://cses.fi/problemset/task/2169
//Nested Ranges Count

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
template<class T> using o_set = tree<T,
     null_type,less<T>,
     rb_tree_tag,tree_order_statistics_node_update>;

void solve(){
    int n;
    cin>>n;
    o_set<pair<int, int>>b,c;
    vector<array<int, 3>>a(n);
    vector<int>aa(n),bb(n);

    for(int i=0; i<n; i++){
        cin>>a[i][0]>>a[i][1];
        a[i][2] = i;
    }
    // sorting using lambda//
    sort(a.begin(), a.end(),[&](const array<int, 3> x,
         const array<int, 3> y){
        if(x[0]==y[0]) return x[1]>y[1];
        else return x[0]<y[0];
    });
    for(int i=0,j=n-1; i<n; i++,j--){
        aa[a[j][2]] = (c.order_of_key({a[j][1]+1, -1})) ;
        c.insert({a[j][1],i});
    }
    for(int i=0; i<n; i++){
        bb[a[i][2]] = (i-b.order_of_key({a[i][1], -1})) ;
        b.insert({a[i][1],i});
    }
    for(int i=0; i<n; i++) cout<<aa[i]<<" ";cout<<endl;
    for(int i=0; i<n; i++) cout<<bb[i]<<" ";cout<<endl;

}
```

6 String

6.1 kmp

```
int lps[1000007];

 void computePrefixFunction(string P)
 {
     int n=SZ(P);
     int k=-1;
     lps[0]=-1;
     for(int i=1;i<n;i++)
     {
         while(k>-1 && P[i]!=P[k+1])
            k=lps[k];
         if(P[i]==P[k+1])
            k++;
         lps[i]=k;
     }
 }

 void KMP(string& T, string& P)
 {
     int m=SZ(P);
     int n=SZ(T);
     computePrefixFunction(P);
     int k=-1;
     for(int i=0;i<n;i++)
     {
         while(k>-1 && T[i]!=P[k+1])
            k=lps[k];
         if(T[i]==P[k+1])
            k++;
         if(k==m-1)
         {
             cout<<"Pattern found in position "<<i-k<<endl;
             k=lps[k];
         }
     }
 }

int main()
{
    KMP(text,pattarn);

    return 0;
}
```

6.2 kmp2

```
int b[MX];
vector<int>v;

void bff(string a)
{
    int i=0,j=-1,m=a.size();
    b[0]=-1;
    while(i<m)
    {
        while(j>=0 && a[i]!=a[j])j=b[j];
        i++;j++;
        b[i]=j;
    }
}

void kmp(string t,string p)
{
    int n=t.size(),m=p.size(),i=0,j=0,ans=0;
    while(true)
    {
        if(j==n) break;
        else if(t[j]==p[i])
        {
            i++;
            j++;
            if(i==m)
            {
               v.pb(j-m+1);
               i=b[i];
            }
        }
        else{
            if(i==0)j++;
            else i=b[i];
        }
    }
    return ;
}

int main()
{
    int t;
    sf("%d",&t);
    while(t--)
   {
       string s,ss;
       cin>>s>>ss;
       bff(s);
       kmp(s,ss);
       if(v.size()>0)
       {
          cout<<v.size()<<endl;
          for(int i=0;i<v.size();i++)
          {
              cout<<v[i]<<" ";
          }cout<<endl;
       }
       else cout<<"Not Found"<<endl;
       v.clear();
       if(t)cout<<endl;
   }

    return 0;
}
```

6.3 trie

```
int ans;

struct Trie{
    int next[2][MX];
    int endmark[MX];
    bool vis[MX];
    int sz;

  void nn()
  {
    mem(vis,0);
    mem(next,0);
    sz=0;
  }
    int insertTrie(int k)
    {
        int v=0;
        for(int i=31;i>=0;i--)
        {
            bool c=k&(1<<i);
```

```
           if(!vis[next[c][v]])
           {
               next[c][v]=++sz;
               vis[sz]=true;
           }
           v=next[c][v];
       }
       endmark[v]=k;
   }

   int maxsearchTrie(int p)
   {
      int v=0;
       for(int i=31;i>=0;i--)
       {
           bool c=p&(1<<i);
           if(vis[next[1-c][v]])v=next[1-c][v];
           else v=next[c][v];

       }

       return endmark[v]^p;
   }

}t;

int main()
{
  // WRITE
   int tt,n,i,j;
   cin>>tt;
   t_c(tt)
   {
       t.nn();
       cin>>n;
       int a,xx=0,maxx=MINI,minn=M;
       t.insertTrie(0);
       for(i=0;i<n;i++)
       {
           cin>>a;
          xx=xx^a;
          t.insertTrie(xx);
          maxx=max(maxx,t.maxsearchTrie(xx));

       }
       cout<<maxx<<endl;
   }
   return 0;
}
```

7 Geometry

```
*******************************************************
bool is_Power_Of_Two(unsigned int x)
{
  return x && !(x & (x   1));
}
****************************************************

/*------------pair sort according second
    element---------------*/
bool sortbysec(pair<int,int> &a,
            pair<int,int> &b)
{
   return (a.second < b.second);
}

//*---------------------structure sort-----------*/
point a[10]; //point type array
bool compair(point a, point b) //here point is a struct
    type data type //
{
       return a.x>b.x;
}
//just call
sort(a,a+10,compair);

/*-------------------------------------*/

///find the centre of a circle which passthrew the corner
    point of a triangle///
point centre(point a,point b,point c)
{
       point ab,ac,ans;
       ab.x=(a.x+b.x)/2; ab.y=(a.y+b.y)/2;
       ac.x=(a.x+c.x)/2; ac.y=(a.y+c.y)/2;
       double a1=(a.x-b.x); double b1=a.y-b.y; double
           c1=ab.x*a1+ab.y*b1;
       double a2=a.x-c.x; double b2=a.y-c.y; double
           c2=ac.x*a2+ac.y*b2;
       double det=a1*b2-a2*b1;
       double d=b2*c1-b1*c2;
       double dd=c2*a1-a2*c1;
       ans.x=d/det;
       ans.y=dd/det;
       return ans;

}

//a segment intersect another segment or not

int orintation(point p, point q, point r)
{
   int val=(q.y-p.y)*(r.x-q.x)-(q.x-p.x)*(r.y-q.y);
   if(val==0)
       return 0;
   return (val>0)?1:2;
}
bool onSeg(point p, point q, point r)
{
   if (q.x <= max(p.x,r.x)&&q.x>=min(p.x,r.x)&&
       q.y<=max(p.y,r.y)&&q.y>=min(p.y,r.y))
       return true;
   return false;
}

bool is_intersect(point p1,point q1,point p2,point q2)
{
   // cout<<"Ok"<<endl;
   int o1=orintation(p1,q1,p2);
   int o2=orintation(p1,q1,q2);
   int o3=orintation(p2,q2,p1);
   int o4=orintation(p2,q2,q1);

   if(o1!=o2 && o3!=o4)
       return true;
   else if(o1==0 && onSeg(p1,p2,q1))
       return true ;
   else if(o2==0 && onSeg(p1,q2,q1))
       return true;
   else if(o3==0 && onSeg(p2,p1,q2))
       return true;
   else if(o4==0 && onSeg(p2,q1,q2))
       return true;
   else
       return false;
}
///a line intersect another line or not or lie on
    it-----------

int area(int x1,int y1,int x2,int y2,int x3,int y3){
    return x1*y2-y1*x2+x2*y3-y2*x3+x3*y1-y3*x1;
}

void line_line_intersect(int x1,int y1,int x2,int y2,int
    x3,int y3,int x4,int y4) {
    int a1 = y2-y1 ;
    int b1 = x1 -x2 ;
    int c1 = y1*x2 - x1*y2 ;
    int a2 = y4 -y3 ;
    int b2 = x3 -x4 ;
    int c2 = y3*x4 - x3*y4 ;
    int m = (a1*b2 - b1*a2) ;
    if( area(x1,y1,x2,y2,x3,y3)==0 &&
        area(x1,y1,x2,y2,x4,y4)==0)
                                    ///given points are
                                        on the line
    {
        pf("LINE\n") ;
    }

    if(m==0) ////never intersect each other
    {
        pf("NONE\n") ;
    }

    double x = ((b1*c2 - b2*c1)*1.0) / ((a1*b2 -
        a2*b1)*1.0) ;
    double y = ((c1*a2 - c2*a1)*1.0) / ((a1*b2 -
        a2*b1)*1.0) ;
    pf("POINT %.2lf %.2lf\n",x,y) ; ////intersect point
}
////if a point in a triangle or in a rectangle or in a
    circle-------

struct point{
    double x,y;
};
struct rectangle{
    point r1,r2;
};

struct circle{
    point c1;
    double radii;
};

struct triangle{
    point a,b,c;
};

bool in_rectangle(point p1,point p2,point p){
///p1=upper left point, p2=lowert right point,p=given
    point
    if(p.x>p1.x && p.x<p2.x && p.y<p1.y && p.y>p2.y)
        return true;
    else
        return false;
}

bool in_circle(point c1,double r,point p)
{///c1=centre of triangle,r=radious,,p=given point
        if(dist(p.x,p.y,c1.x,c1.y)<r) return true;
        return false;
}
```

```
double area(point a,point b,point c){///area of a triangle
       return
           abs(0.5*(a.x*b.y+b.x*c.y+c.x*a.y-b.x*a.y-c.x*b.y-a
}

bool in_triangle(point a,point b,point c,point p){
      double area1=area(a,b,p);
      double area2=area(b,c,p);
      double area3=area(a,c,p);
      double main_area=area(a,b,c);
      double total_area=area1+area2+area3;
      if(area1!=0 && area2!=0 && area3!=0)
      {
             if(total_area-main_area<=1e-1) return
                 true;///alada alda vabe area main area
                             //thke samanno boro hoy..
                           ///you know that baby:)'
             else return false;
      }
      return false;
}
```

8 Data structure

8.1 segment tree (Normal)

```
// ------------------segment
    tree(normal)hakerearth------------
 long long int tree[mx*4];
 long long int tree_min[mx*4];
 long long int arr[mx];
 using namespace std;
void init( int node, int b, int e)
{
    if(b==e)
    {
        tree[node]=arr[b];
        tree_min[node]=arr[b];
        return;
    }
     int mid=(b+e)/2;
    init(node*2,b,mid);
    init(node*2+1,mid+1,e);
    tree[node]=tree[node*2]+tree[node*2+1];
    tree_min[node]=min(tree_min[node*2],tree_min[node*2+1]);
}

ll int query_sum( int node, int b, int e, int i, int j)
{
    if(i>e || j<b)
        return 0;
    if(b>=i && e<=j)
        return tree[node];
     int mid=(b+e)/2;
    ll int p1=query_sum(node*2,b,mid,i,j);
    ll int p2=query_sum(node*2+1,mid+1,e,i,j);
    return p1+p2;
}

void update_sum( int node, int b, int e, int i, int
    newvalue)
{
    if(i>e || i<b)
        return;
    if(b>=i && e<=i)
    {
        tree[node]=newvalue;
        tree_min[node]=newvalue;
        return;
    }
    int mid=(b+e)/2;
    update_sum(node*2,b,mid,i,newvalue);
    update_sum(node*2+1,mid+1,e,i,newvalue);
    tree[node]=tree[node*2]+tree[node*2+1];
    tree_min[node]=min(tree_min[node*2],tree_min[node*2+1]);
}
```

8.2 segment tree (Lazy)

```
struct info
{
    ll prop,sum;
} tree[mx*4];
int a[mx];
void init(int node,int b, int e)
{
    if(b==e)
    {
        tree[node].sum=a[b];
        tree[node].prop=0;
        return;
    }
    int mid=(b+e)/2;
    init(node*2,b,mid);
    init(node*2+1,mid+1,e);
    tree[node].sum=tree[node*2].sum+tree[node*2+1].sum;
    tree[node].prop=tree[node*2].prop+tree[node*2+1].prop;
}

void update(int node,int b,int e,int i,int j,int x)
{
    if(i>e || j<b)
        return;
    if(b>=i && e<=j)
    {
         tree[node].sum=((e-b+1)*x)-tree[node].sum;
        tree[node].prop+=x;
        return;
    }
    int mid=(b+e)/2;
    update(node*2,b,mid,i,j,1);
    update(node*2+1,mid+1,e,i,j,1);

    tree[node].sum=tree[node*2].sum+tree[node*2+1].sum+
                                        (e-b+1)*tree[node].prop;

}
int query(int node,int b,int e,int i,int j,int carry=0)
{

    if(i>e || j<b)
        return 0;
    if(b>=i && e<=j)
        {
                //cout<<"node"<<node<<endl;
                return tree[node].sum+carry*(e-b+1);
        }
    int left=node<<1;
    int right=(node<<1)+1;
    int mid=(b+e)>>1;
     int p1=query(left,b,mid,i,j,tree[node].prop+carry);
     int p2=query(right,mid+1,e,i,j,tree[node].prop+carry);
     return p1+p2;
```

8.3 BIT and eular tree traversal

```
vector<int> adj[mx];
lli in[mx], out[mx], temp[2 * mx], BITree[2 * mx],
     tax[mx], t = 0;
lli get(int idx)
{
    lli sum = 0;idx = idx + 1;
    while (idx > 0){
        sum += BITree[idx];
        idx -= idx & (-idx);
    }
    return sum;
}
void update(int n, int idx, lli val){
    idx = idx + 1;
    while (idx <= n)
    {
        BITree[idx] += val;
        idx += idx & (-idx);
    }
}
void build_BITree(int n){
    for (int i = 1; i <= n; i++)
        BITree[i] = 0;
    for (int i = 0; i < n; i++){
        update(n, i, temp[i]);
    }
}
void dfs(int u, int parent){            ///euler tree
     traversal using dfs
    in[u] = t++; ///first visiting time
    for (auto child : adj[u]){
        if (child != parent)
            dfs(child, u);
    }
    out[u] = t++; ///last visiting time
}
int main(){
    ll node, query;
    cin >> n >> q;
    for (int i = 1; i < n; i++){
        int x, y;
        cin >> x >> y;
        adj[x].pb(y);
        adj[y].pb(x);
    }
    for (int i = 1; i <= n; i++)
        cin >> tax[i];
    dfs(1, 0);
    tax[1] = 0;
    for (int i = 1; i <= n; i++){ ///tree ke akta array te
         convert
        /// korlm euler tree traversal algo diye
        temp[in[i]] = tax[i];
        temp[out[i]] = -(tax[i]);
        //cout<<temp[in[i]]<<" "<<temp[out[i]]<<endl;
    }
    build_BITree(t);
}
```

8.4 Bitmask

```
#define EMPTY_VALUE -1
#define MAX_N 10
#define INF 1061109567

int w[MAX_N][MAX_N];
int mem[MAX_N][1<<MAX_N];

int turnOn(int x, int pos) {
    return N | (1<<pos);
}

bool isOn(int x ,int pos) {
    return (bool)(x & (1<<pos));
}

int n;
int f(int i, int mask) {
    if (mask == (1<<n) - 1) {
        return w[i][0];
    }

    if (mem[i][mask] != -1) {
        return mem[i][mask];
    }

    int ans = INF;
    for (int j = 0;j < n;j++) {
        if (w[i][j] == INF) continue;

        if (isOn(mask,j) == 0) {
            int result = f(j, turnOn(mask, j)) + w[i][j];
            ans = min(ans, result);
        }
    }

    return mem[i][mask] = ans;
}
```

8.5 Mo's Algorithm

```
const int N = 2e5 + 5;
const int Q = 2e5 + 5;
const int SZ = sqrt(N) + 1;
struct qry {
    int l, r, id, blk;
    bool operator<(const qry& p) const {
        return blk == p.blk ? r < p.r : blk < p.blk;
    }
};
qry query[Q];
ll ans[Q];
void add(int id) {}
void remove(int id) {}
ll get() {}
int n, q;
void MO() {
    sort(query, query + q);
    int cur_l = 0, cur_r = -1;
    for (int i = 0; i < q; i++) {
        qry q = query[i];
        while (cur_l > q.l) add(--cur_l);
        while (cur_r < q.r) add(++cur_r);
        while (cur_l < q.l) remove(cur_l++);
        while (cur_r > q.r) remove(cur_r--);
        ans[q.id] = get();
    }
}
// O((N + Q) * sqrt(N))
/* 0 indexed. */
```

9 Big Int Library

```
#include   <bits/stdc++.h>
#include   <stdio.h>

using namespace std;
#define pb        push_back
#define eb        emplace_back
#define mem(x,i)  memset(x,i,sizeof(x))
#define ff        first
#define ss        second
#define all(x)    x.begin(),x.end()
#define Q         int t; scanf("%d", &t); for(int q=1;
    q<=t; q++)

typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;//%Lf
typedef pair<ll, ll> pi;

                        /* Debug Tools */
#define error(args...) \
{ \
    string _s = #args; replace(_s.begin(), _s.end(), ',',
        ' ');\
    stringstream _ss(_s); istream_iterator<string>
        _it(_ss); err(_it, args);\
}
void err(istream_iterator<string> it) {}
template<typename T, typename... Args>
void err(istream_iterator<string> it, T a, Args... args) {
    cerr<< *it << " = " << a
        <<",\n"[++it==istream_iterator<string>()];
    err(it, args...);
}

const int MOD = 1e9+7 ; //For big mod
template<typename T>inline T gcd(T a, T b){T c;while
     (b){c = b;b = a % b;a = c;}return a;} // better than
     __gcd
template<typename T>inline T lcm(T a, T b){return
     (a/gcd(a, b))*b;}
ll powmod(ll a,ll b){ll res=1;a%=MOD;if(b<0) return
     0;for(; b;
     b>>=1){if(b&1)res=res*a%MOD;a=a*a%MOD;}return res;}

const int xx[] = {+1, -1, +0, +0};//, +1, +1, -1, -1};//
     exclude last four when side adjacent
const int yy[] = {+0, +0, +1, -1};//, +1, -1, +1, -1};
const int INF  = 0x3f3f3f3f;// useful for memset
const ll LL_INF = 0x3f3f3f3f3f3f3f3f;
const double PI = acos(-1.0);
const double eps = 1e-9;
const int mod  = 1e9+7;
const int mxn  = 1e5+5;

typedef long long ll;
const int maxn = 1e2 + 14, lg = 15;

/*
  ################################################################
  ####################### THE  BIG  INT
      ##########################
*/
const int base = 1000000000;
const int base_digits = 9;
struct bigint {
        vector<int> a;
        int sign;
        /*<arpa>*/
        int size(){
                if(a.empty())return 0;
                int ans=(a.size()-1)*base_digits;
                int ca=a.back();
                while(ca)
                ans++,ca/=10;
        return ans;
}
bigint operator ^(const bigint &v){
        bigint ans=1,a=*this,b=v;
        while(!b.isZero()){
                if(b%2)
                        ans*=a;
                a*=a,b/=2;
        }
        return ans;
}
string to_string(){
        stringstream ss;
        ss << *this;
        string s;
        ss >> s;
        return s;
}
int sumof(){
        string s = to_string();
        int ans = 0;
        for(auto c : s) ans += c - '0';
        return ans;
}
/*</arpa>*/
bigint() :
        sign(1) {
}

bigint(long long v) {
        *this = v;
}

bigint(const string &s) {
        read(s);
}

void operator=(const bigint &v) {
        sign = v.sign;
        a = v.a;
}

void operator=(long long v) {
        sign = 1;
        a.clear();
        if (v < 0)
                sign = -1, v = -v;
        for (; v > 0; v = v / base)
                a.push_back(v % base);
}

bigint operator+(const bigint &v) const {
        if (sign == v.sign) {
                bigint res = v;

                for (int i = 0, carry = 0; i < (int)
                    max(a.size(), v.a.size()) ||
                    carry; ++i) {
                        if (i == (int) res.a.size())
                                res.a.push_back(0);
                        res.a[i] += carry + (i <
                            (int) a.size() ? a[i] :
                            0);
                        carry = res.a[i] >= base;
                        if (carry)
                                res.a[i] -= base;
                }
                return res;
        }
        return *this - (-v);
}

bigint operator-(const bigint &v) const {
        if (sign == v.sign) {
                if (abs() >= v.abs()) {
                        bigint res = *this;
                        for (int i = 0, carry = 0; i
                            < (int) v.a.size() ||
                            carry; ++i) {
                                res.a[i] -= carry +
                                    (i < (int)
                                    v.a.size() ?
                                    v.a[i] : 0);
                                carry = res.a[i] < 0;
                                if (carry)
                                        res.a[i] +=
                                            base;
                        }
                        res.trim();
                        return res;
                }
                return -(v - *this);
        }
        return *this + (-v);
}

void operator*=(int v) {
        if (v < 0)
                sign = -sign, v = -v;
        for (int i = 0, carry = 0; i < (int)
            a.size() || carry; ++i) {
                if (i == (int) a.size())
                        a.push_back(0);
                long long cur = a[i] * (long long) v
                    + carry;
                carry = (int) (cur / base);
                a[i] = (int) (cur % base);
                //asm("divl %%ecx" : "=a"(carry),
                    "=d"(a[i]) : "A"(cur),
                    "c"(base));
        }
        trim();
}

bigint operator*(int v) const {
        bigint res = *this;
        res *= v;
        return res;
}

void operator*=(long long v) {
        if (v < 0)
                sign = -sign, v = -v;
        if(v > base){
                *this = *this * (v / base) * base +
                    *this * (v % base);
                return ;
        }
        for (int i = 0, carry = 0; i < (int)
            a.size() || carry; ++i) {
                if (i == (int) a.size())
                        a.push_back(0);
                long long cur = a[i] * (long long) v
                    + carry;
                carry = (int) (cur / base);
                a[i] = (int) (cur % base);
                //asm("divl %%ecx" : "=a"(carry),
                    "=d"(a[i]) : "A"(cur),
                    "c"(base));
        }
        trim();
}

bigint operator*(long long v) const {
        bigint res = *this;
        res *= v;
        return res;
}

friend pair<bigint, bigint> divmod(const bigint
    &a1, const bigint &b1) {
        int norm = base / (b1.a.back() + 1);
        bigint a = a1.abs() * norm;
        bigint b = b1.abs() * norm;
        bigint q, r;
        q.a.resize(a.a.size());

        for (int i = a.a.size() - 1; i >= 0; i--) {
                r *= base;
                r += a.a[i];
                int s1 = r.a.size() <= b.a.size() ?
                    0 : r.a[b.a.size()];
                int s2 = r.a.size() <= b.a.size() -
                    1 ? 0 : r.a[b.a.size() - 1];
                int d = ((long long) base * s1 + s2)
                    / b.a.back();
                r -= b * d;
                while (r < 0)
                        r += b, --d;
                q.a[i] = d;
        }

        q.sign = a1.sign * b1.sign;
        r.sign = a1.sign;
        q.trim();
        r.trim();
        return make_pair(q, r / norm);
}

bigint operator/(const bigint &v) const {
        return divmod(*this, v).first;
}

bigint operator%(const bigint &v) const {
        return divmod(*this, v).second;
}

void operator/=(int v) {
        if (v < 0)
                sign = -sign, v = -v;
        for (int i = (int) a.size() - 1, rem = 0; i
            >= 0; --i) {
                long long cur = a[i] + rem * (long
                    long) base;
                a[i] = (int) (cur / v);
                rem = (int) (cur % v);
        }
        trim();
}

bigint operator/(int v) const {
        bigint res = *this;
        res /= v;
        return res;
}
```

```
int operator%(int v) const {
        if (v < 0)
                v = -v;
        int m = 0;
        for (int i = a.size() - 1; i >= 0; --i)
                m = (a[i] + m * (long long) base) %
                    v;
        return m * sign;
}

void operator+=(const bigint &v) {
        *this = *this + v;
}
void operator-=(const bigint &v) {
        *this = *this - v;
}
void operator*=(const bigint &v) {
        *this = *this * v;
}
void operator/=(const bigint &v) {
        *this = *this / v;
}

bool operator<(const bigint &v) const {
        if (sign != v.sign)
                return sign < v.sign;
        if (a.size() != v.a.size())
                return a.size() * sign < v.a.size()
                    * v.sign;
        for (int i = a.size() - 1; i >= 0; i--)
                if (a[i] != v.a[i])
                        return a[i] * sign < v.a[i]
                            * sign;
        return false;
}

bool operator>(const bigint &v) const {
        return v < *this;
}
bool operator<=(const bigint &v) const {
        return !(v < *this);
}
bool operator>=(const bigint &v) const {
        return !(*this < v);
}
bool operator==(const bigint &v) const {
        return !(*this < v) && !(v < *this);
}
bool operator!=(const bigint &v) const {
        return *this < v || v < *this;
}

void trim() {
        while (!a.empty() && !a.back())
                a.pop_back();
        if (a.empty())
                sign = 1;
}

bool isZero() const {
        return a.empty() || (a.size() == 1 &&
            !a[0]);
}

bigint operator-() const {
        bigint res = *this;
        res.sign = -sign;
        return res;
```

```
    }

    bigint abs() const {
            bigint res = *this;
            res.sign *= res.sign;
            return res;
    }

    long long longValue() const {
            long long res = 0;
            for (int i = a.size() - 1; i >= 0; i--)
                    res = res * base + a[i];
            return res * sign;
    }

    friend bigint gcd(const bigint &a, const bigint
        &b) {
            return b.isZero() ? a : gcd(b, a % b);
    }
    friend bigint lcm(const bigint &a, const bigint
        &b) {
            return a / gcd(a, b) * b;
    }

    void read(const string &s) {
            sign = 1;
            a.clear();
            int pos = 0;
            while (pos < (int) s.size() && (s[pos] ==
                '-' || s[pos] == '+')) {
                    if (s[pos] == '-')
                            sign = -sign;
                    ++pos;
            }
            for (int i = s.size() - 1; i >= pos; i -=
                base_digits) {
                    int x = 0;
                    for (int j = max(pos, i -
                        base_digits + 1); j <= i; j++)
                            x = x * 10 + s[j] - '0';
                    a.push_back(x);
            }
            trim();
    }

    friend istream& operator>>(istream &stream, bigint
        &v) {
            string s;
            stream >> s;
            v.read(s);
            return stream;
    }

    friend ostream& operator<<(ostream &stream, const
        bigint &v) {
            if (v.sign == -1)
                    stream << '-';
            stream << (v.a.empty() ? 0 : v.a.back());
            for (int i = (int) v.a.size() - 2; i >= 0;
                --i)
                    stream << setw(base_digits) <<
                        setfill('0') << v.a[i];
            return stream;
    }

    static vector<int> convert_base(const vector<int>
        &a, int old_digits, int new_digits) {
            vector<long long> p(max(old_digits,
                new_digits) + 1);
            p[0] = 1;
            for (int i = 1; i < (int) p.size(); i++)
                    p[i] = p[i - 1] * 10;
            vector<int> res;
            long long cur = 0;
            int cur_digits = 0;
            for (int i = 0; i < (int) a.size(); i++) {
                    cur += a[i] * p[cur_digits];
                    cur_digits += old_digits;
                    while (cur_digits >= new_digits) {
                            res.push_back(int(cur %
                                p[new_digits]));
                            cur /= p[new_digits];
                            cur_digits -= new_digits;
                    }
            }
            res.push_back((int) cur);
            while (!res.empty() && !res.back())
                    res.pop_back();
            return res;
    }

    typedef vector<long long> vll;

    static vll karatsubaMultiply(const vll &a, const
        vll &b) {
            int n = a.size();
            vll res(n + n);
            if (n <= 32) {
                    for (int i = 0; i < n; i++)
                            for (int j = 0; j < n; j++)
                                    res[i + j] += a[i] *
                                        b[j];
                    return res;
            }

            int k = n >> 1;
            vll a1(a.begin(), a.begin() + k);
            vll a2(a.begin() + k, a.end());
            vll b1(b.begin(), b.begin() + k);
            vll b2(b.begin() + k, b.end());

            vll a1b1 = karatsubaMultiply(a1, b1);
            vll a2b2 = karatsubaMultiply(a2, b2);

            for (int i = 0; i < k; i++)
                    a2[i] += a1[i];
            for (int i = 0; i < k; i++)
                    b2[i] += b1[i];

            vll r = karatsubaMultiply(a2, b2);
            for (int i = 0; i < (int) a1b1.size(); i++)
                    r[i] -= a1b1[i];
            for (int i = 0; i < (int) a2b2.size(); i++)
                    r[i] -= a2b2[i];

            for (int i = 0; i < (int) r.size(); i++)
                    res[i + k] += r[i];
            for (int i = 0; i < (int) a1b1.size(); i++)
                    res[i] += a1b1[i];
            for (int i = 0; i < (int) a2b2.size(); i++)
                    res[i + n] += a2b2[i];
            return res;
    }

    bigint operator*(const bigint &v) const {
            vector<int> a6 = convert_base(this->a,
                base_digits, 6);
            vector<int> b6 = convert_base(v.a,
                base_digits, 6);
            vll a(a6.begin(), a6.end());
            vll b(b6.begin(), b6.end());
            while (a.size() < b.size())
                a.push_back(0);
            while (b.size() < a.size())
                b.push_back(0);
            while (a.size() & (a.size() - 1))
                a.push_back(0), b.push_back(0);
            vll c = karatsubaMultiply(a, b);
            bigint res;
            res.sign = sign * v.sign;
            for (int i = 0, carry = 0; i < (int)
                c.size(); i++) {
                long long cur = c[i] + carry;
                res.a.push_back((int) (cur %
                    1000000));
                carry = (int) (cur / 1000000);
            }
            res.a = convert_base(res.a, 6, base_digits);
            res.trim();
            return res;
        }
};
int main()
{
    Q{
        ll nn, kk, ll;
        cin >> nn >> kk >> ll;
        // error("hi")
        bigint n = nn;
        bigint k = kk;
        bigint l = ll;
        bigint r = l+n;
        bigint sm = (r*(r-1))/2;
        sm -= (l*(l-1))/2;
        sm *= 2;
        sm *= k;
        sm /= n;
        bigint mo = 1;
        for(int i=0; i<18; i++) mo *= 10;
        sm = sm%mo;
        cout << sm << "\n";
    }
}
```

10 Python$_list_input$

```
# -*- coding: utf-8 -*-
"""
Created on Fri Jun 3 13:25:16 2022

@author: PROME
"""

T = int(input())
for i in range(T):
    lst = list(map(int, input("").strip().split()))
    n = lst[0]
    k = lst[1]
    l = lst[2]
    L = l
    R = l+n-1
    mid = int((L+R)/2)
    ans = 0
    mod = int(1e18)
    if((R-L+1)&1):
        ans = (k%mod*(2*mid)%mod)%mod
    else:
        ans = (k%mod*(2*mid+1)%mod)%mod
    ans = int(ans)
    print(ans)
```

11 Python$_matrix_input$

```
l1 = []
check = []
l2 = list(map(int, input("").strip().split()))
l1.append(l2)
if all(v == 0 for v in l2):
    check.append(int(0))
length = len(l2)
# print(len(l1))
for i in range(length-1):
    l3 = list(map(int, input("").strip().split()))
    l1.append(l3)
    if all(v == 0 for v in l3):
        check.append(int(i+1))

p=-1
for i in check:
    p=i
    for j in range(length):
        if i!=j:
            if l1[j][i]!=1:
                print(i, j)
                p=0
                break
if p<=0:
    print("No celebrity")
else:
    print(p)
```