Code library for DU_On_a_Rampage

Max area rectangle in histogram with stack in O(n)

```
int getMaxArea(int n){
    a[n] = 0;    n++;
    stack<int> s;
    int res = 0, tp, tmp, i = 0;
    while (i < n){
        if (s.empty() || a[s.top()] <= a[i])    s.push(i++);
        else{
            tp = s.top();
            s.pop();
            tmp = a[tp] * (s.empty() ? i : i - s.top() - 1);
            if (res < tmp)  res = tmp;
        }
    }  return res;}
```

//Dinic's Max flow in O(V^2E)

```
int dist[MXX], p[MXX],start[MXX];
struct edge{
   int v,rc,rev;
};
vector<edge> adj[MXX];
void add_edge(int u,int v,int c)
{
   edge e1{v,c,(int)adj[v].size()};
   edge e2{u,0,(int)adj[u].size()};

      adj[u].pb(e1);
      adj[v].pb(e2);
  }
  void clr()
  {
      for(int i=0;i<MXX;i++)
         adj[i].clear();
  }
  bool bfs(int s,int t)
  {
      queue<int> q;
      memset(dist,-1,sizeof dist);
       dist[s]=0;
```

```
24.      q.push(s);
25.      while(!q.empty())
26.      {
27.          int u=q.front();
28.          q.pop();
29.          for(int i=0;i<adj[u].size();i++)
30.          {
31.              int v=adj[u][i].v;
32.              int rc=adj[u][i].rc;
33.              if(dist[v]==-1 && rc>0)
34.              {
35.                  dist[v]=dist[u]+1;
36.                  q.push(v);
37.              }
38.          }
39.      }
40.      return dist[t]!=-1;
41.  }
42.
43.  int dfs(int s,int t,int min_cap)
44.  {
45.      if(s==t)
46.          return min_cap;
47.
48.      for(int &i=start[s];i<adj[s].size();i++)
49.      {
50.          edge &e=adj[s][i];
51.          if(dist[e.v]!=dist[s]+1 || e.rc<1)
52.              continue;
53.
54.          int pushed=dfs(e.v,t,min(min_cap,e.rc));
55.
56.          if(pushed){
57.              e.rc-=pushed;
58.              adj[e.v][e.rev].rc+=pushed;
59.              return pushed;
60.          }
61.      }
62.      return 0;
63.  }
64.   int dinic(int s,int t){
```

```
65.     int max_flow=0;
66.     while(bfs(s,t)){
67.         memset(start,0,sizeof start);
68.         while(int flow=dfs(s,t,INT_MAX))
69.             max_flow+=flow;
70.     }
71.     return max_flow;
72. }
```

Hopcroft Carp MBP in O(root(v)*E)

```
vector<int> adj[MXX];
int matchL[MXX],matchR[MXX],dist[MXX];
void add_edge(int u,int v)
{
    adj[u].pb(v);
    //adj[v].pb(u);
}
bool bfs(int n)
{
    queue<int> q;
    for(int i=1; i<=n; i++)
    {
        if(!matchL[i])
        {
            dist[i]=0;
            q.push(i);
        }
        else
            dist[i]=INF;
    }
    dist[0]=INF;

    while(!q.empty())
    {
        int u=q.front();
        q.pop();
        if(u)
        {

            for(int i=0; i<adj[u].size(); i++)
```

```
            {
                int v=adj[u][i];
                if(dist[matchR[v]]==INF)
                {
                    dist[matchR[v]]=dist[u]+1;
                    q.push(matchR[v]);
                }
            }
        }
    }
    return dist[0]!=INF;
}

bool dfs(int u)
{
    if(u)
    {
        for(int i=0; i<adj[u].size(); i++)
        {
            int v=adj[u][i];
            if(dist[matchR[v]]==dist[u]+1)
            {
                if(dfs(matchR[v]))
                {
                    matchL[u]=v;
                    matchR[v]=u;
                    return true;
                }
            }
        }
        dist[u]=INF;
        return false;
    }
    return true;
}

int hopcroft_karp(int n,int m)
{
    int maximum_matching=0;
    memset(matchL,0,sizeof matchL);
    memset(matchR,0,sizeof matchR);
    while(bfs(n))
    {
```

```
        for(int i=1; i<=n; i++)
            if(!matchL[i] && dfs(i))
            {
                maximum_matching++;
            }
    }

    return maximum_matching;
}



//Lucas,CRT and Lucas for power of prime


ll combination(int n,int k,int m)
{
    if(n<k)
        return 0;
    k=min(k,n-k);
    ll ans=1;
    for(int i=0; i<k; i++)
    {
        ans=(ans*(n-i));
        ans/=(i+1);
    }
    return ans%m;
}

ll bigmod(ll a,ll b,ll m)
{
    if(b==0)
        return 1;
    ll x=bigmod(a,b/2,m);
    if(b%2==0) return (x*x)%m;
    return (a*((x*x)%m))%m;
}

int nCrModpLucas(int n, int r, int p)
{
    if (r==0)
        return 1;
```

```
    // Compute last digits of n and r in base p
    int ni = n%p, ri = r%p;

    return (nCrModpLucas(n/p, r/p, p)*combination(ni, ri, p)) % p;
}

int phi[]= {18,10,12,36};
int CRT(int num[], int rem[], int k)
{
    // Compute product of all numbers
    int prod = 1;
    for (int i = 0; i < k; i++)
        prod *= num[i];//here num[i] holds prime factors of original number

    int result = 0,inv;

    for (int i = 0; i < k; i++)
    {
        int pp = prod / num[i];
        inv=bigmod(pp,phi[i]-1,num[i]);//modular inverse of pp
        result += (rem[i] * pp * inv)%prod;
    }

    return result % prod;
}

int factorial[40];
int arr[40];
void precal(ll p, ll mod)//here mod=p^e
{
    arr[0]=arr[1]=1;
    factorial[0]=factorial[1]=1;
    int x=1;
    for(ll i=2; i<=mod; i++)
    {
        if(i%p)
            x=i;
        else
            x=1;
        arr[i]=(arr[i-1]*x)%mod;

        factorial[i]=(factorial[i-1]*i)%mod;
    }
```

```
}

ll f(ll n)
{
    //product of numbers <=n, and coprime to 3.
    return bigmod(arr[27],n/27,27)*arr[n%27];
}

ll F(ll n, ll mod, ll p)
{
    ll ret=1;
    ll i=1;
    while(i<=n)
    {
        ret=(ret*f(n/i))%mod;
        i=i*p;
    }
    return ret;
}

ll E(ll n, ll p)
{
    ll ret=0;
    while(n)
    {
        ret+=n/p;
        n=n/p;
    }
    return ret;
}


ll mod_27(ll n,ll r)
{
    ll pow3 = E(n,3) - E(r,3) - E(n-r,3);
    ll mod1=F(n,27,3);
    ll mod2=(F(r,27,3)*F(n-r,27,3))%27;
    return (bigmod(3,pow3,27)*mod1*bigmod(mod2,17,27))%27;
}
```

Min cost Max flow

```
struct edge
{
    int v,rc,cpuf,rev;
};

vector<edge> adj[MXX];
int cost[MXX],parent[MXX],idx[MXX];
int f,mf,maxCost;

void add_edge(int u,int v,int rc,int cpuf)
{
    edge e1={v,rc,cpuf,adj[v].size()};
    edge e2={u,0,-cpuf,adj[u].size()};
    adj[u].pb(e1);
    adj[v].pb(e2);
}
void augment(int t,int k,int min_cap)
{
    if(parent[t]==-1)
    {
        f=min(k-mf,min_cap);
        return;
    }
    edge &e=adj[parent[t]][idx[t]];
    augment(parent[t],k,min(min_cap,e.rc));
    e.rc-=f;
    maxCost+=f*e.cpuf;
    adj[t][e.rev].rc+=f;

}
bool dijsktra(int s,int t)
{
    priority_queue<ii > pq;
    for(int i=0;i<MXX;i++)
        cost[i]=-INF;
    memset(parent,-1,sizeof parent);
    cost[s]=0;
    pq.push(ii(0,s));

    while(!pq.empty())
    {
```

```
        ii top=pq.top();
        pq.pop();

        int u=top.second;
        int c=top.first;

        if(c<cost[u])
        continue;

        for(int i=0;i<adj[u].size();i++)
        {
            edge e=adj[u][i];

            if(cost[e.v]<cost[u]+e.cpuf && e.rc>0)
            {
                cost[e.v]=cost[u]+e.cpuf;
                parent[e.v]=u;
                idx[e.v]=i;
                pq.push(ii(cost[e.v],e.v));
            }
        }
    }
    //cout<<cost[t]<<endl;
    return cost[t]!=-INF;
}

int min_cost_flow(int s,int t,int k)
{
    mf=0,maxCost=0;
    while(mf<k)
    {
        if(!dijsktra(s,t))
            break;
        augment( t,k,INT_MAX);
        mf+=f;
    }

}
```

Trie

```
struct trie
{
    trie* children[26]={};
    int words=0;
    int No_of_child=0;
    bool isCompleteWord=0;

};
trie *root=new trie();
bool already_added=false;
void addWord(trie *cur,string str,int idx)
{
    if(str.size()==idx)
    {
        if(cur->isCompleteWord)
            //already_added=true;
        cur->isCompleteWord=true;
        //cur->words+=1;
        return;
    }
    cur->No_of_child++;
    if(cur->children[str[idx]-97]==NULL)
    {
        cur->children[str[idx]-97]=new trie();
    }
    addWord(cur->children[str[idx]-97],str,idx+1);
    //if(already_added)
       // cur->No_of_child--;
}

int wordCount(trie *cur,string str,int idx)
{
    if(str.size()==idx)
        return cur->words;
    if(cur->children[str[idx]-97]==NULL)
        return 0;
    return wordCount(cur->children[str[idx]-97],str,idx+1);

}

int prefixCount(trie *cur,string str,int idx)
{
```

```
    if(str.size()==idx)
        return cur->No_of_child;
    if(cur->children[str[idx]-97]==NULL)
        return 0;
    return prefixCount(cur->children[str[idx]-97],str,idx+1);

}
bool flag=false;
bool removeWord(trie *cur,string str,int idx)
{
    if(str.size()==idx)
    {
        if(!cur->isCompleteWord)
        {
            return false;
        }
        flag=true;
        cur->isCompleteWord=false;
        cur->words--;
        return cur->No_of_child==0;
    }

    if(cur->children[str[idx]-97]==NULL)
        return false;

    bool should_remove_cur_child=
removeWord(cur->children[str[idx]-97],str,idx+1);

    if(flag) cur->No_of_child--;

    if(should_remove_cur_child)
    {
        free(cur->children[str[idx]-97]);
        cur->children[str[idx]-97]=NULL;
        return !cur->isCompleteWord && !cur->No_of_child;
    }
    return false;
}


Edmond blossom Matching for general graph in O(V^3)

/*
```

```
GETS:
V->number of vertices
E->number of edges
pair of vertices as edges (vertices are 1..V)

GIVES:
output of edmonds() is the maximum matching
match[i] is matched pair of i (-1 if there isn't a matched pair)
 */

#include <bits/stdc++.h>
using namespace std;
const int M=500;
struct struct_edge{int v;struct_edge* n;};
typedef struct_edge* edge;
struct_edge pool[M*M*2];
edge top=pool,adj[M];
int V,E,match[M],qh,qt,q[M],father[M],base[M];
bool inq[M],inb[M],ed[M][M];
void add_edge(int u,int v)
{
  top->v=v,top->n=adj[u],adj[u]=top++;
  top->v=u,top->n=adj[v],adj[v]=top++;
}
int LCA(int root,int u,int v)
{
  static bool inp[M];
  memset(inp,0,sizeof(inp));
  while(1)
    {
      inp[u=base[u]]=true;
      if (u==root) break;
      u=father[match[u]];
    }
  while(1)
    {
      if (inp[v=base[v]]) return v;
      else v=father[match[v]];
    }
}
void mark_blossom(int lca,int u)
{
  while (base[u]!=lca)
```

```
    {
      int v=match[u];
      inb[base[u]]=inb[base[v]]=true;
      u=father[v];
      if (base[u]!=lca) father[u]=v;
    }
}
void blossom_contraction(int s,int u,int v)
{
  int lca=LCA(s,u,v);
  memset(inb,0,sizeof(inb));
  mark_blossom(lca,u);
  mark_blossom(lca,v);
  if (base[u]!=lca)
    father[u]=v;
  if (base[v]!=lca)
    father[v]=u;
  for (int u=0;u<V;u++)
    if (inb[base[u]])
      {
      base[u]=lca;
      if (!inq[u])
        inq[q[++qt]=u]=true;
      }
}
int find_augmenting_path(int s)
{
  memset(inq,0,sizeof(inq));
  memset(father,-1,sizeof(father));
  for (int i=0;i<V;i++) base[i]=i;
  inq[q[qh=qt=0]=s]=true;
  while (qh<=qt)
    {
      int u=q[qh++];
      for (edge e=adj[u];e;e=e->n)
        {
        int v=e->v;
        if (base[u]!=base[v]&&match[u]!=v)
          if ((v==s)||(match[v]!=-1 && father[match[v]]!=-1))
            blossom_contraction(s,u,v);
          else if (father[v]==-1)
            {
            father[v]=u;
```

```
            if (match[v]==-1)
              return v;
            else if (!inq[match[v]])
              inq[q[++qt]=match[v]]=true;
            }
        }
    }
  return -1;
}
int augment_path(int s,int t)
{
  int u=t,v,w;
  while (u!=-1)
    {
      v=father[u];
      w=match[v];
      match[v]=u;
      match[u]=v;
      u=w;
    }
  return t!=-1;
}
int edmonds()
{
  int matchc=0;
  memset(match,-1,sizeof(match));
  for (int u=0;u<V;u++)
    if (match[u]==-1)
      matchc+=augment_path(u,find_augmenting_path(u));
  return matchc;
}
int main()
{
  int u,v;
  cin>>V>>E;
  while(E--)
    {
      cin>>u>>v;
      if (!ed[u-1][v-1])
      {
        add_edge(u-1,v-1);
        ed[u-1][v-1]=ed[v-1][u-1]=true;
      }
```

```
    }
  cout<<edmonds()<<endl;
  for (int i=0;i<V;i++)
    if (i<match[i])
      cout<<i+1<<" "<<match[i]+1<<endl;
}

//Trie using array
```

```
#include<bits/stdc++.h>
    using namespace std;
    #define MAX 100000
    int node[MAX][55],rote,nnode,root,isWord[MAX];
    int gid(const char &c){
        if(c >= 'A' and c <= 'Z') return c - 'A';
        return (c - 'a') + 26;
    }
    void initialize()
       {
           root=nnode=0;
           memset(node,-1,sizeof(node));
           memset(isWord,0,sizeof(isWord));
       }
       void insert(char* str, int len)
       {
           int now=root;
           for(int i=0;i<len;i++)
           {
               if(node[now][gid(str[i])]==-1){
                   node[now][gid(str[i])]=++nnode;

               }
               now=node[now][gid(str[i])];
           }
           isWord[now]++;
       }
       int search(char* str, int len)
       {
           int now = root;
           for (int i = 0; i < len; i++) {
               int id = gid(str[i]);
               if(node[now][gid(str[i])]==-1)
                   return -1;
```

```
            now=node[now][gid(str[i])];
        }
        return isWord[now];
    }
```

//convex hull

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

#define MAX 100009
#define ld double
#define i64 long long

typedef struct { i64 x, y; } point;

point P[MAX], C[MAX], P0;
ld pi, eps = 1e-9;

inline i64 triArea2(const point &a, const point &b, const point &c) {
    return (a.x*(b.y-c.y) + b.x*(c.y-a.y) + c.x*(a.y-b.y));
}

inline i64 sqDist(const point &a, const point &b) {
    return ((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
}

inline bool comp(const point &a, const point &b) {
    i64 d = triArea2(P0, a, b);
    if(d < 0) return false;
    if(!d && sqDist(P0, a) > sqDist(P0, b)) return false;
    return true;
}

inline bool normal(const point &a, const point &b) {
    return ((a.x==b.x) ? a.y < b.y : a.x < b.x);
}

inline bool issame(const point &a, const point &b) {
    return (a.x == b.x && a.y == b.y);
}
```

```
37.
38.   inline void makeUnique(int &np) {
39.       sort(&P[0], &P[np], normal);
40.       np = unique(&P[0], &P[np], issame) - P;
41.   }
42.
43.   inline void convexHull(int &np, int &nc) {
44.       int i, j, pos = 0;
45.       for(i = 1; i < np; i++)
46.           if(P[i].y<P[pos].y || (P[i].y==P[pos].y && P[i].x<P[pos].x))
47.               pos = i;
48.       swap(P[0], P[pos]);
49.       P0 = P[0];
50.       sort(&P[1], &P[np], comp);
51.       for(i = 0; i < 3; i++) C[i] = P[i];
52.       for(i = j = 3; i < np; i++) {
53.           while(triArea2(C[j-2], C[j-1], P[i]) < 0) j--;
54.           C[j++] = P[i];
55.       }
56.       //cout<<j<<endl;
57.       nc = j;
58.   }
59.
60.   inline void compress(int &nc) {
61.       int i, j;
62.       C[nc] = C[0];
63.       for(i=j=1; i < nc; i++) {
64.           if(triArea2(C[j-1], C[i], C[i+1])) C[j++] = C[i];
65.       }
66.       nc = j;
67.   }
//2-sat
1.  #include<bits/stdc++.h>
2.  using namespace std;
3.  #define sfi(a) scanf("%d",&a)
4.  #define sfl(a) scanf("%lld",&a)
5.  #define sff(a) scanf("%lf",&a)
6.  #define sfs(a) scanf("%s",&a)
7.  #define pf printf
8.  #define MAX 110
9.  #define mymax(a,b,c) max(a,max(b,c))
10.    #define mymin(a,b,c) min(a,min(b,c))
11.#define mymiddle(a,b,c) a+b+c-max(a,max(b,c))-min(a,min(b,c))
```

```
#define check(n, pos) (n & (1<<pos))
#define set(n, pos) (n | (1<<pos))
#define sq(x) ((x)*(x))
   #define sf scanf
#define pb push_back
typedef long long int li;

int val[20100],vis[20100],com[20100],cnum,is[20100];
   vector<int>
  adj[20100],adji[20100],adjc[20100],adjcom[20100],lists;

void init(int n)
{
   for(int i=1;i<=n;i++)
      {
      adj[i].clear();
      adji[i].clear();
      adjc[i].clear();
      adjcom[i].clear();
      }
   memset(is,0,sizeof(is));
}

void dfs(int x)
   {
   vis[x]=1;

   for(int i=0;i<adj[x].size();i++)
   {
         int j=adj[x][i];
      if(!vis[j])
      {
         dfs(j);
      }
      }
   lists.push_back(x);
}

void dfsscc(int x)
   {
   //cout<<x<<" Reach"<<endl;
   vis[x]=1;

```

```
54.    for(int i=0;i<adjcom[x].size();i++)
55.        {
56.        int j=adjcom[x][i];
57.        if(!vis[j])
58.        {
59.            dfsscc(j);
60.            }
61.    }
62.    lists.push_back(x);
63.}
64.
65.     void dfs1(int x,int c)
66.{
67.    vis[x]=1;
68.    com[x]=c;
69.    adjc[c].pb(x);
70.        for(int i=0;i<adji[x].size();i++)
71.    {
72.        int j=adji[x][i];
73.        if(!vis[j])
74.        {
75.                dfs1(j,c);
76.        }
77.    }
78.}
79.
80.     void dfssort(int n)
81.{
82.    lists.clear();
83.    memset(vis,0,sizeof(vis));
84.    for(int i=1;i<=n;i++)
85.        {
86.        if(!vis[i]) dfs(i);
87.    }
88.}
89.void scc(int n)
90.     {
91.
92.    dfssort(n);
93.    memset(vis,0,sizeof(vis));
94.    int c=0,i,j;
95.        for(i=lists.size()-1;i>=0;i--)
96.    {
```

```
97.         if(!vis[lists[i]]) dfs1(lists[i],++c);
98.     }
99.     cnum=c;
100.    //cout<<"Compo "<<cnum<<endl;
101.    //for(int i=1;i<=n;i++) cout<<com[i]<<endl;
102. }
103.
104. void conversion(int n)
105. {
106.     map<pair<int,int>,int> mp;
107.     for(int i=1;i<=n;i++)
108.     {
109.         int u=i,v;
110.         for(int j=0;j<adj[i].size();j++)
111.         {
112.             v=adj[i][j];
113.             //cout<<com[u]<<" "<<com[v]<<endl;
114.             if(!mp[make_pair(com[u],com[v])])
115.             {
116.                 adjcom[com[u]].pb(com[v]);
117.                 mp[make_pair(com[u],com[v])]=1;
118.             }
119.         }
120.     }
121.     lists.clear();
122.     memset(vis,0,sizeof(vis));
123.     //cout<<"Compo "<<cnum<<endl;
124.     for(int i=1;i<=cnum;i++)
125.     {
126.         //cout<<i<<endl;
127.         if(!vis[i]) dfsscc(i);
128.     }
129.
130.     memset(val,-1,sizeof(val));
131.     for(int i=0;i<lists.size();i++)
132.     {
133.         int setvalue=1;
134.         int node=lists[i];
135.         //cout<<node<<endl;
136.         for(int j=0;j<adjc[node].size();j++)
137.         {
138.             int u=adjc[node][j],v;
139.             if(u<=n/2)
```

```
            {
                v=u+n/2;
            }
            else
            {
                v=u-n/2;
            }
            if(val[com[v]]==1)
            {
                setvalue=0;
                break;
            }
        }
        val[node]=setvalue;
    }
}

void solve(int t){
    int n,m,u,v;
    sf("%d %d",&m,&n);
    init(2*n);
    for(int i=1;i<=m;i++)
    {
        sf("%d %d",&u,&v);
        if(u>0) is[u]=1;
        if(v>0) is[v]=1;
        if(u>0 && v>0)
        {
            adj[n+u].pb(v);
            adj[n+v].pb(u);
            adji[u].pb(n+v);
            adji[v].pb(n+u);
        }
        else if(u>0 && v<0){
            adj[n+u].pb(n-v);
            adj[-v].pb(u);
            adji[n-v].pb(n+u);
            adji[u].pb(-v);
        }
        else if(u<0 && v>0) {
            swap(u,v);
            adj[n+u].pb(n-v);
            adj[-v].pb(u);
```

```
183.            adji[n-v].pb(n+u);
184.            adji[u].pb(-v);
185.        }
186.        else if(u<0 && v<0)
187.        {
188.            adj[-u].pb(n-v);
189.            adj[-v].pb(n-u);
190.            adji[n-v].pb(-u);
191.            adji[n-u].pb(-v);
192.        }
193.    }
194.    scc(2*n);
195.    char *str="Yes";
196.    for(int i=1;i<=n;i++)
197.    {
198.        if(com[i]==com[n+i]) str="No";
199.    }
200.
201.    pf("Case %d: %s\n",t,str);
202.    if(str=="Yes")
203.    {
204.        conversion(2*n);
205.    }
206.    else return;
207.    vector<int> vs;
208.    for(int i=1;i<=n;i++)
209.    {
210.        if(val[com[i]]==1)
211.        {
212.            vs.pb(i);
213.        }
214.    }
215.    pf("%d",vs.size());
216.    for(int i=0;i<vs.size();i++)
217.    {
218.        pf(" %d",vs[i]);
219.    }
220.    puts("");
221. }
222.
223. int main()
224. {
225.    int t;
```

```
    sfi(t);
    for(int i=1;i<=t;i++)
    {
       solve(i);
    }
    return 0;
}
```

//**Geometry**

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

#define pi      acos(-1.00)
#define eps     1e-9
#define D(x)    cout << #x " = " << (x) << endl

const int inf = numeric_limits<int>::max();
bool eq(double a, double b) { return fabs( a - b ) < eps; } //two numbers are equal

   struct point{
       double x, y;
       point(){}

       point(double xx, double yy) {x = xx, y = yy;} // NEVER USE xx = 0 or yy = 0 HERE
   } origin = point(0, 0);


   point operator+(const point &u, const point &v) {return point(u.x + v.x, u.y + v.y);} //OK
   point operator-(const point &u, const point &v) {return point(u.x - v.x, u.y - v.y);} //OK
   point operator*(const point &u, double v) {return point(u.x*v, u.y*v);} //OK
   point operator*(double v, const point &u) {return point(u.x*v, u.y*v);} //OK
   point operator*(const point &u, const point &v) {return point(u.x * v.x - u.y * v.y, u.x * v.y + v.x * u.y);} // multiplying two complex numbers
```

```
25.   point operator/(const point &u, double v) {assert(abs(v) > eps);
   return point(u.x/v, u.y/v);} //OK
26.   bool operator != (const point &u, const point &v) {return !(eq(u.x,
   v.x) && eq(u.y, v.y));} //OK
27.
28.   ostream &operator <<(ostream &os, const point &p) {
29.     os << "(" << p.x << "," << p.y << ")";
30.   } //OK
31.
32.   bool operator <(const point &u, const point &v){
33.       if(fabs(u.x - v.x ) < eps) return u.y + eps < v.y;
34.       return u.x + eps < v.x;
35.   }
36.
37.   double norm(point u){return sqrt(u.x * u.x + u.y * u.y);} //OK
38.   double arg(point u){ assert(u != origin); return atan2(u.y, u.x);}
   //OK
39.   point polar(double r, double theta) {return point(r * cos(theta), r
   * sin(theta));} //OK
40.
41.   double dotp(point u, point v) {return u.x * v.x + u.y * v.y;} //OK
42.   double crsp(point u, point v) {return u.x * v.y - u.y * v.x;} //OK
43.
44.
45.   point unit_vector(point u) { return u / norm(u); } //OK
46.   point rtt(point piv, point u, double theta) {return (u - piv) *
   polar(1.00, theta) + piv;} //OK
47.   point projection(point p, point st, point ed) { return dotp(ed - st,
   p - st) / norm(ed - st) * unit_vector(ed - st) + st;} //OK
48.   point extend(point st, point ed, double len) { return ed +
   unit_vector(ed-st) * len;} //OK
49.
50.   point segmentProjection(point p, point st, point ed)
51.   {
52.       double d = dotp(p - st, ed - st) / norm(ed - st);
53.       if(d < 0) return st;
54.       if(d > norm(ed - st) + eps) return ed;
55.       return st + unit_vector(ed - st) * d;
56.   } //OK
57.
58.   double distancePointSegment(point p, point st, point ed) {return
   norm(p - segmentProjection(p, st, ed)); } //OK
```

```
   double distancePointLine( point P, point st, point ed) { return
   norm( projection(P, st, ed) - P ); } //OK

   point reflection(point p, point st, point ed){
       point proj = projection(p, st, ed);
       if(p != proj) return extend(p, proj, norm(p - proj));
       return proj;
   } //OK

//sieve

void sieve(){
      mark[1] = true;

      for (int i=4; i<sz; i+=2)     mark[i] = true;

      for (int p=3; p<=sz; p+=2)
      {
      if (mark[p] == false)
      {
            for (int i = 2*p; i<sz; i += p)
            {
                  mark[i] = true;
            }
      }
      }
}

//stable marriage

#include<bits/stdc++.h>
using namespace std;
#define MAX 210
#define light(x) (printf("Case %d:",x))
int pref[MAX][MAX],wpart[MAX],n,tot,po[MAX][MAX];
int fre[MAX];
//w refered to woman , pm previous man , current man
bool preferance(int w,int pm,int cm)
{
      return po[w][cm]<po[w][pm];
   }
   void stable_marriage()
   {
```

```
    int fcount=n;
    memset(fre,-1,sizeof(fre));
    memset(wpart,-1,sizeof(wpart));
    while(fcount)
    {
        //cout<<fcount<<endl;
        int m;
        for(m=0;m<n;m++)
        {
            if(fre[m]==-1) break;
        }
        for(int i=0;i<n && fre[m]==-1;i++)
        {
            int w=pref[m][i];//women to check
            if(wpart[w]==-1)
            {
                wpart[w]=m;
                fre[m]=1;
                fcount--;
            }
            else
            {
                int m1=wpart[w];
                if(preferance(w,m1,m))
                {
                    wpart[w]=m;
                    fre[m]=1;
                    fre[m1]=-1;
                }
            }
        }
    }
    for(int i=0;i<n;i++)
    {
        printf(" \(%d %d\)",wpart[i]+1,i+n+1);
    }
    puts("");
}

//String

template < class T > string ToString ( T n )
```

```
{
       ostringstream ss;
       ss << n;
       return ss.str();
}



//    String multiplication

string multiply( string a, long long b ) {
      int carry = 0, i;

      for( i = 0; i < a.size(); i++ ) {
            carry += (a[i] - 48) * b;
            a[i] = ( carry % 10 + 48 );
            carry /= 10;
      }

      while( carry ) {
            a += ( carry % 10 + 48 );
            carry /= 10;
      }

      return a;
}

#define d(x) cerr << #x " = " << (x) << endl

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

int lps[1000005];
char a[1000005], b[1000005];

void computeLPSArray(char *pat, int M)
{

      int len = 0, i = 1;

      lps[0] = 0;
```

```
	while (i < M)
	{
	if (pat[i] == pat[len])
	{
		len++;
		lps[i] = len;
		i++;
	}
	else
	{
		if (len != 0)
		{
			len = lps[len-1];
		}
		else
		{
			lps[i] = 0;
			i++;
		}
	}
	}
}


int KMPSearch(char *txt, char *pat)
{
	int M = strlen(pat);
	int N = strlen(txt);
	int res = 0;

	computeLPSArray(pat, M);

	int i = 0;
	int j  = 0;
	while (i < N)
	{
	if (pat[j] == txt[i])
	{
		j++;
		i++;
	}

	if (j == M)
```

```
	{
	//   printf("Found pattern at index %d \n", i-j);
		++res;
		j = lps[j-1];
	}

	else if (i < N && pat[j] != txt[i])
	{
		if (j != 0) j = lps[j-1];

		else i = i+1;
	}
	}

	return res;
}

//Matrix Expo

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

typedef unsigned long long i64;

struct Matrix
{
	i64 r, c, a[20][20];

	Matrix()
	{
	memset(a, 0,sizeof(a));
	}

	Matrix(i64 n, i64 m)
	{
	r = n;
	c = m;
	memset(a, 0,sizeof(a));
	}

	Matrix(i64 n)
	{
	r = c = n;
```

```
memset(a, 0, sizeof(a));
}

Matrix operator * (Matrix b)
{
Matrix res(r, b.c);

for(int i=0; i<r; i++)
{
      for(int j=0; j<b.c; j++)
      {
            i64 temp = 0;

            for(int k=0; k<c; k++)
            {
            temp = (temp + (a[i][k]*b.a[k][j]));
            }
            res.a[i][j] = temp;
      }
}
return res;
}

Matrix operator ^(i64 n)
{
Matrix res(r);
res.identity();
Matrix p(r, c);
for(int i = 0; i<r; i++)
      for(int j=0; j<c; j++)  p.a[i][j] = a[i][j];

while(n)
{
      if( n & 1 ) res = res * p;
      p = p * p;
      n >>= 1;
}

return res;
}

void identity()
{
```

```
	memset(a, 0, sizeof(a));
	for(int i=0; i<20; i++) a[i][i] = 1;
	}

};

int main()
{

	int test, cs = 0;

	scanf("%d", &test);

	while(test--)
	{
	i64 p, q, n;

	cin>>p>>q>>n;

	Matrix a(2);

	a.a[0][0] = p;
	a.a[0][1] = -q;
	a.a[1][0] = 1;
	a.a[1][1] = 0;

	a = a^(n-1);

	i64 ff = a.a[0][0] * p;
	i64 ss = a.a[0][1] * 2;

	i64 res = ff+ss;
	if(n==0)    res = 2;

	cout<<"Case "<<++cs<<": "<<res<<endl;
	}
	return 0;

}

//Segment tree////////Normal/////////////////////////////////////////////

struct data {
```

```
	int minimum, maximum;
} tree[MAX*4];

data Merge(data l, data r) {
	data ret;
	ret.minimum = min(l.minimum, r.minimum);
	ret.maximum = max(l.maximum, r.maximum);
	return ret;
}

void init(int node, int beg, int endd) {
	if(beg == endd) {
	tree[node] = { arr[beg], arr[beg] };
	return;
	}

	int left = node*2;
	int right = node*2+1;
	int mid = (beg+endd) / 2;

	init(left, beg, mid);
	init(right, mid+1, endd);

	tree[node] = Merge(tree[left], tree[right]);
}

data query(int node, int beg, int endd, int x, int y) {
	if(x > y) return { INT_MAX, INT_MIN }; // dummy
	if(beg == x && endd == y) return tree[node];

	int left = node*2;
	int right = node*2+1;
	int mid = (beg+endd) / 2;

	data l = query(left, beg, mid, x, min(y, mid));
	data r = query(right, mid+1, endd, max(x, mid+1), y);
	return Merge(l, r);
}

void update(int node, int beg, int endd, int x, int val) {
	if(beg == x && endd == x) {
	tree[node] = { val, val };
	return;
```

```
      }
      int left = node*2;
      int right = node*2+1;
      int mid = (beg+endd) / 2;

      if(x <= mid) update(left, beg, mid, x, val);
      else update(right, mid+1, endd, x, val);

      tree[node] = Merge(tree[left], tree[right]);
}

//////////////////////////Lazy without propagation/////////////////////////////

void update(int at,int L,int R,int l,int r)
{
    if(r<L || R<l) return;
    if(l<=L && R<=r) { toggle[at]^=1; return; }
    int mid=(L+R)/2;
    update(at*2,L,mid,l,r);
    update(at*2+1,mid+1,R,l,r);
}
int query(int at,int L,int R,int pos)
{
    if(pos<L || R<pos) return 0;
    if(L==pos && pos==R) {
    return toggle[at];}
    int mid=(L+R)/2;
    if(pos<=mid) return query(at*2,L,mid,pos)^toggle[at];
    else return query(at*2+1,mid+1,R,pos)^toggle[at];
}

//////////////////////////Lazy With
propagation//////////////////////////////
struct node{
    int sum;
}tree[4*MAX];
int lazy[4*MAX];
node merge(node a,node b)
{
    node ret;
    ret.sum=a.sum+b.sum;
    return ret;
}
```

```
void lazyUpdate(int n,int st,int ed)
{
    if(lazy[n]==0) return;
    tree[n].sum+=(ed-st+1)*lazy[n];
    if(st!=ed)
    {
       lazy[2*n]+=lazy[n];
       lazy[2*n+1]+=lazy[n];
    }
    lazy[n]=0;
}
void build(int n,int st,int ed)
{
    lazy[n]=0;
    if(st==ed) {
       tree[n].sum=ara[st];return;
    }
    int mid=(st+ed)/2;
    build(2*n,st,mid);
    build(2*n+1,mid+1,ed);
    tree[n]=merge(tree[2*n],tree[2*n+1]);
}
node query(int n,int st,int ed,int i,int j)
{
    lazyUpdate(n,st,ed);
    if(st>=i && ed<=j) return tree[n];
    int mid=(st+ed)/2;
    if(mid<i) return query(2*n+1,mid+1,ed,i,j);
    else if(mid>=j) return query(2*n,st,mid,i,j);
    else return  merge(query(2*n+1,mid+1,ed,i,j),query(2*n,st,mid,i,j));
}
void update(int n,int st,int ed,int i,int j,int v)
{
    lazyUpdate(n,st,ed);
    if(st>j || ed<i) return;
    if(st>=i && ed<=j)
    {
       lazy[n]+=v;
       lazyUpdate(n,st,ed);
       return;
    }
    int mid=(st+ed)/2;
    update(2*n,st,mid,i,j,v);
```

```
    update(2*n+1,mid+1,ed,i,j,v);
    tree[n]=merge(tree[2*n],tree[2*n+1]);
}

////////////////////segment tree(indexing)//////////////////////////////////
      #include <bits/stdc++.h>
      using namespace std;

      const int MAX = 200003;
      int n, arr[MAX] ;
      int tree[MAX*4];
      vector<int> v;

      void init(int node, int beg, int endd) {
      if(beg == endd) {
            tree[node] = 1;
            return;
      }

      int left = node*2;
      int right = node*2+1;
      int mid = (beg+endd) / 2;

      init(left, beg, mid);
      init(right, mid+1, endd);

      tree[node] = tree[left]+tree[right];
      }

      int query(int node, int beg, int endd,int pos) {
      if(beg == endd) return beg;

      int left = node*2;
      int right = node*2+1;
      int mid = (beg+endd) / 2;

      if(tree[left]>=pos) return query(left, beg, mid,pos);
      else query(right, mid+1, endd ,pos-tree[left]);
      }

      void update(int node, int beg, int endd, int x, int val) {
      if(beg == x && endd == x) {
            tree[node] = val;
```

```
	return;
}

int left = node*2;
int right = node*2+1;
int mid = (beg+endd) / 2;

if(x <= mid) update(left, beg, mid, x, val);
else update(right, mid+1, endd, x, val);

tree[node] = tree[left] + tree[right];
}
int main()
{
int t,n,q,a,l,i,m,sizes;
char ch[3];
scanf("%d",&t);
for(int k=1;k<=t;k++)
{
	init(1,1,MAX-1);
	printf("Case %d:\n", k);
	scanf("%d %d",&n,&q);
	memset(arr,0,sizeof(arr));
	v.clear();
	v.push_back(1);
	for(i=1;i<=n;i++)
	{
		scanf("%d",&a);
		v.push_back(a);
	}
	for(i=0;i<q;i++){
		scanf("%s %d",&ch,&a);
		if(ch[0]=='c')
		{
		if(a>n)
		{
			printf("none\n");
		}
		else
		{
			l=query(1,1,MAX-1,a);
			update(1,1,MAX-1,l,0);
			printf("%d\n",v[l]);
```

```
                    n--;
                }
                }
                else
                {
                v.push_back(a);
                n++;
                }
            }
        }
        return 0;
        }

//stable marriage
#include<bits/stdc++.h>
using namespace std;
#define MAX 210
#define light(x) (printf("Case %d:",x))
int pref[MAX][MAX],wpart[MAX],n,tot,po[MAX][MAX];
int fre[MAX];
//w refered to woman , pm previous man , current man
bool preferance(int w,int pm,int cm)
{
    return po[w][cm]<po[w][pm];
  }
  void stable_marriage()
  {
      int fcount=n;
      memset(fre,-1,sizeof(fre));
      memset(wpart,-1,sizeof(wpart));
      while(fcount)
      {
          int m;
          for(m=0;m<n;m++)
          {
              if(fre[m]==-1) break;
          }
          for(int i=0;i<n && fre[m]==-1;i++)
          {
              int w=pref[m][i];//women to check
              if(wpart[w]==-1)
              {
                  wpart[w]=m;
```

```
                    fre[m]=1;
                    fcount--;
                }
                else
                {
                    int m1=wpart[w];
                    if(preferance(w,m1,m)){
                        wpart[w]=m;
                        fre[m]=1;
                        fre[m1]=-1;
                    }
                }
            }
        }
        for(int i=0;i<n;i++){
            printf(" \(%d %d\)",wpart[i]+1,i+n+1);
        }
        puts("");
    }
    void solve(int t)
    {    scanf("%d",&n);
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
            {
                scanf("%d",&pref[i][j]);
                pref[i][j]-=(n+1);
            }
        }
        int pp;
        for(int i=0;i<n;i++)
        {
            for(int j=0;j<n;j++)
            {
                scanf("%d",&pp);
                po[i][pp-1]=j;
            }
        }
        light(t);
        stable_marriage();
    }

//Ternary search
```

```cpp
#include <bits/stdc++.h>
using namespace std;
#define sq(x) ((x)*(x))
double prec=1e-8;
#define x first
#define y second
typedef pair<double,double> Point;
Point a, b, c, d, e, f;

   double calc(double k) {
       e.x = a.x + k * (b.x - a.x);
       e.y = a.y + k * (b.y - a.y);
       f.x = c.x + k * (d.x - c.x);
       f.y = c.y + k * (d.y - c.y);
       return sqrt(sq(e.x - f.x) + sq(e.y - f.y));
   }

   double ternary(double lo, double hi) {
       double lt, rt;
       lt = (2.0 * lo + hi) / 3.0;
       rt = (lo + 2.0 * hi) / 3.0;
       if(abs(calc(lt) - calc(rt))<prec) return (lo + hi) / 2.0;
       if(calc(lt) < calc(rt)) return ternary(lo,rt);
       else return ternary(lt,hi);
   }

```

BIT

```cpp
int update(int pos,int limit,int x){

    while(pos<=limit)   {
       tree[pos]+=x;
       pos+=(pos)&(-pos);
    }
}
int query(int pos){
    int sum = 0;
       while(pos>0)   {
          sum+=tree[pos];
          pos-=(pos)&(-pos);
       }
       return sum;
```

```
}
```

2D BIT

```
void update(int x, int y, int lmx, int lmy, int val){
    for(int i = x; i<=lmx; i+= i & (-i)) {
        for(int j = y; j<=lmy; j+= j & (-j))  {
            tree[i][j] += val;
        }
    }
}

int query(int x, int y){
    int sum = 0;
    for(int i = x; i>0; i-= i & (-i)) {
        for(int j = y; j>0; j -= j & (-j)) {
            sum += tree[i][j];
        }
    }
    return sum;}
```

Area of two circle's intersection

```
double solve(){
    double x1, y1, x2, y2, r1, r2;
    scanf("%lf %lf %lf %lf %lf %lf", &x1, &y1, &r1, &x2, &y2, &r2);
    double res = 0.0;
    double d = sqrt((x1-x2)*(x1-x2) + (y1-y2)*(y1-y2));
    double dsq = (x1-x2)*(x1-x2) + (y1-y2)*(y1-y2);
    if(d>=(r1+r2))  return res;
     if(d<=fabs(r1-r2)){
        res = min(r1, r2)*min(r1, r2)*acos(-1.0);
            return res;
      }
       double angle_a = 2.0 * acos((r1*r1 + dsq - r2*r2)/(2.0*r1*d));
       double angle_b = 2.0 * acos((r2*r2 + dsq - r1*r1)/(2.0*r2*d));
       double curve1 = 0.5 * angle_a * r1* r1;
       double curve2 = 0.5 * angle_b * r2* r2;
       res = 0.5 * (r1*r1*(angle_a - sin(angle_a))+ r2*r2*(angle_b -
    sin(angle_b)));
       return res;
}
```

```
# Point inside convex polygon in logN
```

```
#include<bits/stdc++.h>
using namespace std;
typedef long long i64;

struct point{
    i64 x, y;
    point() {}
    point(int x, int y) : x(x), y(y) {}
} a[100005];

i64 orientation(point a, point b, point c) // triangle area
{
    return a.x*(b.y - c.y) + b.x*(c.y - a.y) + c.x*(a.y - b.y);
}

bool binarySearch(point qp, int n){
    int lo = 1, hi = n-1, mid;
    while((hi-lo) > 1) {
        mid = (hi+lo)/2;
        if(orientation(a[0], a[mid], qp) < 0) hi = mid;
        else lo = mid;
    }
    if(orientation(a[0], a[lo], qp)  < 0) return false;
    if(orientation(a[lo], a[hi], qp) < 0) return false;
    if(orientation(a[hi], a[0], qp)  < 0) return false;
    return true;
}
```

Pick’s theorem

```
i64 polygonArea(int n){
    i64 area = 0;
    int j = n - 1;
    for (int i = 0; i < n; i++){
        area += (x[j] + x[i]) * (y[j] - y[i]);
        j = i;
    }
    return abs(area / 2);
}

i64 getBoundaryCount(int n){
    i64 sum = n;
    x[n] = x[0];   y[n] = y[0];
    for(int i = 0; i<n; i++){
```

```
        int p = i;
        int q = i+1;
        sum += __gcd(abs(x[p] - x[q]), abs(y[p] - y[q]))-1;
    }
    return sum;
}

int main(){
    int test, cs = 0;
    scanf("%d", &test);
    while(test--){
        int n;  scanf("%d", &n);
        for(int i = 0; i<n; i++) scanf("%lld %lld", &x[i], &y[i]);
        i64 A = polygonArea(n);
        i64 B = getBoundaryCount(n);
       i64 res = 1 + A - B/2;   printf("Case %d: %lld\n", ++cs, res); }

//Articulation point and bridge
1. #include<bits/stdc++.h>
2. using namespace std;
3. #define MAX 10100
4. #define pp pair<int,int>
5. #define mclear(a, x) ( memset(a,x,sizeof(a)) )
6. #define pii pair <int, int>
7. #define pb push_back
8. #define sf(a) scanf("%d",&a)
9. #define sff(a,b) scanf("%d %d",&a,&b)
10.   #define printcase(a,b) printf("Case %d: %d\n",a,b)
11.   #define mp(a,b) make_pair(a,b)
12.   vector<int> lists;
13.   vector<int> adj[MAX],adji[MAX],comp[MAX];
14.   int
      vis[MAX],t,start[MAX],end[MAX],dis[MAX],point[MAX],low[MAX],d[MAX],par[
      MAX],colour[MAX],co[MAX],f[MAX],counter[MAX];
15.   set<int> apoint;
16.   set<pp> abridge;
17.   map<pp,bool> bridge;
18.   void dfs(int s,int c)
19.   {
20.       if(vis[s]) return;
21.       co[s]=c;
22.       vis[s]=1;
23.       for(int i=0;i<adj[s].size();i++)
```

```
    {
        int v=adj[s][i];
        if(!bridge[mp(min(s,v),max(s,v))]) dfs(v,c);
    }
}
void Articulation(int u)
{
    int child=0;
    t++;
    low[u]=d[u]=t;
    vis[u]=1;
    for(int i=0;i<adj[u].size();i++)
    {
        int v=adj[u][i];
        if(v==par[u]) continue;
        if(vis[v]==1)
        {
            low[u]=min(low[u],d[v]);
        }
        else if(!vis[v])
        {
            child++;
            par[v]=u;
            Articulation(v);
            low[u]=min(low[u],low[v]);
            if(d[u]<low[v])
            {
                abridge.insert(make_pair(min(u,v),max(u,v)));
                bridge[make_pair(min(u,v),max(u,v))]=1;
            }
        }
    }
}
void printbridge()
{
    std::set<pp>::iterator it;
    for(it=abridge.begin();it!=abridge.end();it++)
    {
        pp x=*it;
        counter[co[x.first]]++;
        counter[co[x.second]]++;
    }
}
```

```
67.  int main()
68.  {
69.      int t,m,n,u,v,i,r;
70.      sf(r);
71.      for(int k=1;k<=r;k++)
72.      {
73.          mclear(par,-1);
74.          mclear(vis,0);
75.          mclear(counter,0);
76.          bridge.clear();
77.          abridge.clear();
78.          t=0;
79.          sff(n,m);
80.          for(i=1;i<=m;i++)
81.          {
82.              sff(u,v);
83.              adj[u].pb(v);
84.              adj[v].pb(u);
85.          }
86.          Articulation(0);
87.          mclear(vis,0);
88.          int comp=0,ans=0;
89.          for(i=0;i<n;i++)
90.          {
91.              if(!vis[i]) { comp++ ; dfs(i,comp); }
92.          }
93.          printbridge();
94.          for(i=1;i<=comp;i++)
95.          {
96.              if(counter[i]==1) ans++;
97.          }
98.          printcase(k,(ans+1)/2);
99.          for(i=0;i<n;i++) adj[i].clear();
100.     }
101. }
```

#Djakstsra

```
1. #include<bits/stdc++.h>
2. using namespace std;
3. struct Node
4. {
5.     int at,cost;
6.     Node(int _at,int _cost)
7.     {
```

```
        at= _at;
        cost= _cost;
    }
};
bool operator<(Node A,Node B)
{
    return A.cost>B.cost;
}
struct Edge
{
    int v,w;
    Edge(int x,int y)
    {
        v=x;
        w=y;
    }
};
vector<Edge> adj[600];
priority_queue<Node> PQ;
int dist[600];
int n;
void dijkastra(int s)
{
    for(int i=1;i<=n;i++)
    {
        dist[i]=1000000000;
    }
    dist[s]=0;
    PQ.push(Node(s,0));
    while(!PQ.empty())
    {
        Node u=PQ.top();
        PQ.pop();
        if(u.cost!=dist[u.at])
        {
            continue;
        }
        for(int i=0;i<adj[u.at].size();i++)
        {
            Edge e=adj[u.at][i];
            if(dist[e.v]>u.cost+e.w)
            {
                dist[e.v]=u.cost+e.w;
```

```
                    PQ.push(Node(e.v,dist[e.v]));
                }
            }
        }
    }
#MST prim
#include<bits/stdc++.h>
using namespace std;
#define MX 105
vector<int>v[MX],cost[MX];
int visited[MX];
int total;
struct data
{
    int vertex,weight,edge;
    bool operator < (data d) const
    {
        return  (weight>d.weight);
    }
    };

int minimum_spanning_tree_prims_algo(int s,int n)
{
    data d;
        int sum=0;
    bool flag=false;
    priority_queue<data>pq;
    for(int j=0; j<n-1; j++)
    {
            visited[s] = 1;
        for(int i=0; i<v[s].size(); i++)
        {
            if(visited[v[s][i]]==0)
            {
                    d.vertex = v[s][i];
                d.edge = s; // parent
                d.weight = cost[s][i];
                pq.push(d);
            }
            }
        while(visited[s])
        {
            if(pq.empty())
```

```
        {
                flag=true;
            break;
        }
        d = pq.top();
        pq.pop();
            s = d.vertex;
    }
    if(flag) break;
    sum += d.weight;
  }
  if(!flag) return (total-sum);
  return -1;
}
 void solve(int t)
{
  int i,n,e,x,y,w;total=0;
  scanf("%d",&n);
  for(x=1; x<=n; x++)
    {
    for(y=1;y<=n;y++)
    {
       scanf("%d",&w);
       if(w>0)
          {
          v[x].push_back(y);
          v[y].push_back(x);
          cost[x].push_back(w);
          cost[y].push_back(w);
             total+=w;
       }
    }
  }
  memset(visited,0,sizeof(visited));
  int ans=minimum_spanning_tree_prims_algo(1,n);
    printf("Case %d: %d\n",t,ans);
  for(int i=1;i<=n;i++)
  {
    v[i].clear();
    cost[i].clear();
    }
}
```