ACM/ICPC Template Manaual

QUST

hxk

September 7, 2018

Contents

0 Include

```
//#include <bits/stdc++.h>
#include <algorithm>
#include  <iostream>
#include   <cstring>
#include    <string>
#include    <cstdio>
#include    <vector>
#include     <stack>
#include     <queue>
#include     <cmath>
#include       <set>
#include       <map>
using namespace std;
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define per(i,a,b) for(int i=a;i>=b;i--)
#define clr(a,x) memset(a,x,sizeof(a))
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef unsigned long long ull;
typedef long long ll;
typedef vector<int> vi;
typedef pair<int,int> pii;
/************head****************/
int work(){

    return 0;
}
int main(){
#ifdef superkunn
    freopen("input.txt","rt",stdin);
#endif
    work();
    return 0;
}
```

1 String Processing

```
//hihocoder 1014
const int maxnode=2600000+10;
const int sigma_size=26;
struct Trie{
    int ch[maxnode][sigma_size];
    int val[maxnode];
    int sz;
    void init(){sz=0;clr(ch[0],0);}
    int idx(char c){return c-'a';}
    void insert(char *s){
        int u=0,n=strlen(s);
        rep(i,0,n-1){
            int x=idx(s[i]);
            if(!ch[u][x]){
                ++sz;
                clr(ch[sz],0);
                val[sz]=0;
                ch[u][x]=sz;
            }
            u=ch[u][x];
            val[u]++;
        }
    }
    int query(char *s){
        int u=0,n=strlen(s),res=0;
        rep(i,0,n-1){
            int x=idx(s[i]);
            if(!ch[u][x])break;
            u=ch[u][x];
            if(i==n-1)res=val[u];
        }
        return res;
    }
}trie;
char s[30];
int work(){
    trie.init();
    int n,m;
    scanf("%d",&n);
    while(n--){
        scanf("%s",s);
        trie.insert(s);
    }
    scanf("%d",&m);
    while(m--){
        scanf("%s",s);
        printf("%d\n",trie.query(s));
    }
    return 0;
}
```

1.1 KMP

```
//MAXN
int nxt[MAXN];
void initkmp(char x[],int m){
    int i=0,j=nxt[0]=-1;
    while(i<m){
        while(j!=-1&&x[i]!=x[j])j=nxt[j];
        nxt[++i]=++j;
    }
}
//x:pa y:tx
int kmp(char x[],int m,char y[],int n){
    int i,j,ans;
    i=j=ans=0;
    initkmp(x,m);
    while(i<n){
        while(j!=-1&&y[i]!=x[j])j=nxt[j];
        i++,j++;
        if(j>=m){
            ans++;
            j=nxt[j];
            //pos:i-m
        }
    }
    return ans;
}
```

1.2 Manacher

```
//hihocoder 1032
const int MAXN=2e6+10;//more than 2 times !
char s[MAXN],str[MAXN];
int len1,len2,p[MAXN];
void init(){
    str[0]='$';
    str[1]='#';
    rep(i,0,len1){
        str[i*2+2]=s[i];
        str[i*2+3]='#';
    }
    len2=len1*2+2;
    str[len2]='*';
}
int manacher(){
    int id=0,mx=0,ans=0;
    rep(i,1,len2-1){
        if(mx>i)p[i]=min(p[2*id-i],mx-i);
        else p[i]=1;
        while(str[i+p[i]]==str[i-p[i]])p[i]++;
        if(i+p[i]>mx){
            mx=i+p[i];
            id=i;
        }
        ans=max(ans,p[i]);
    }
    return ans-1;
}
int work(){
    int T;
    scanf("%d",&T);
    while(T--){
        scanf("%s",s);
        len1=strlen(s);
        init();
        printf("%d\n",manacher());
    }
    return 0;
}
```

1.3 HashString

```
const ll B1=1e7+7;
const ll B2=1e9+7;
char pa[10004];
char tx[1000006];
int work(){
    int T;
    scanf("%d",&T);
    while(T--){
        scanf("%s%s",pa,tx);
        int pl=strlen(pa);
        int tl=strlen(tx);
        ll w=1;
        rep(i,1,pl)w=(w*B1)%B2;
        ll ph=0,th=0;
        rep(i,0,pl-1){
            ph=(ph*B1+pa[i])%B2;
            th=(th*B1+tx[i])%B2;
        }
        int ans=0;
        for(int i=0;i+pl<=tl;i++){
            if(ph==th)ans++;
            if(i+pl<tl)th=(th*B1+tx[i+pl]-tx[i]*w)%B2;
        }
        printf("%d\n",ans);
    }
    return 0;
}
```

2 Data Structure

2.1 other

```
//hdu 1394
const int MAXN=5005;
int n;
vi A;
int x[MAXN];
int merging(vi &a){
    int n=SZ(a);
    if(n<=1)return 0;
    int cnt=0;
    vi b(a.begin(),a.begin()+n/2);
    vi c(a.begin()+n/2,a.end());
    cnt+=merging(b);
    cnt+=merging(c);
    int ai=0,bi=0,ci=0;
    while(ai<n){
        if(bi<SZ(b)&&(ci==SZ(c)||b[bi]<=c[ci])){
            a[ai++]=b[bi++];
        }else{
            cnt+=n/2-bi;
            a[ai++]=c[ci++];
        }
    }
    return cnt;
}
int work(){
    while(~scanf("%d",&n)){
        A.clear();
        rep(i,1,n)scanf("%d",&x[i]),A.pb(x[i]);
        int sum=merging(A);
        int res=sum;
        rep(i,1,n){
            sum=sum-x[i]+(n-1-x[i]);
            res=min(res,sum);
        }
        printf("%d\n",res);
    }
    return 0;
}
```

2.1.1 QuickSelect

```
anytype QuickSelect(anytype arr[],int l,int r,int k){
    int i=l,j=r,mid=arr[(i+j)>>1];
    while(i<=j){
        while(arr[i]<mid)i++;
        while(arr[j]>mid)j--;
        if(i<=j){
            swap(arr[i],arr[j]);
            i++;
            j--;
        }
    }
    if(l<j&&k<=j)return QuickSelect(arr,l,j,k);
    if(i<r&&k>=i)return QuickSelect(arr,i,r,k);
    return arr[k];
}
```

2.2 Binary Indexed Tree

```
//add(pos,a) sum(r)-sum(l-1)
//add(l,a) add(r+1,-a) sum(pos)
const int MAXN=100000;
struct BIT{
    int n,c[MAXN<<1];
    void init(int _n){
        n=_n;
        rep(i,0,n)c[i]=0;
    }
    void update(int i,int v){
        for(;i<=n;i+=i&-i)c[i]+=v;
    }
    int query(int i){
        int s=0;
        for(;i;i-=i&-i)s+=c[i];
        return s;
    }
    int findpos(int v){
        int sum=0;
        int pos=0;
        int i=1;
        for(;i<n;i<<=1);
        for(;i;i>>=1){
            if(pos+i<=n&&sum+c[pos+i]<v){
                sum+=c[pos+i];
                pos+=i;
            }
        }
        return pos+1;
    }
}bit;
```

2.2.1 poj3468

$a_i = \sum_{i=1}^{x} d_i$
$\sum_{i=1}^{x} a_i = \sum_{i=1}^{x} \sum_{j=1}^{i} d_j = \sum_{i=1}^{x} (x-i+1) d_i$
$\sum_{i=1}^{x} a_i = (x+1) \sum_{i=1}^{x} d_i - \sum_{i=1}^{x} d_i \times i$

```
const int MAXN=1e5+5;
int n,q,x,y,z;
long long c1[MAXN],c2[MAXN];
void add(int x,int y){
    for(int i=x;i<=n;i+=i&(-i))c1[i]+=y,c2[i]+=1LL*x*y;
}
ll sum(int x){
    ll ans(0);
    for(int i=x;i;i-=i&(-i))ans+=1LL*(x+1)*c1[i]-c2[i];
    return ans;
}
char op[5];
int work(){
    scanf("%d%d",&n,&q);
    int a1,a2;
    a1=0;
    rep(i,1,n){
        scanf("%d",&a2);
        add(i,a2-a1);
        a1=a2;
    }
    while(q--){
        scanf("%s",op);
        if(op[0]=='Q'){
            scanf("%d%d%d",&x,&y,&z);
            printf("%lld\n",sum(y)-sum(x-1));
        }else{
            scanf("%d%d%d",&x,&y,&z);
            add(x,z);
            add(y+1,-z);
        }
    }
    return 0;
}
```

2.3 Segment Tree

```
#define lson rt<<1
#define rson rt<<1|1
#define le l,m,lson
#define ri m+1,r,rson
#define mid m=(l+r)>>1
```

2.3.1 Single-point Update

```
const int MAXN=5e4+5;
int sum[MAXN<<2];
void push_up(int rt){
    sum[rt]=sum[lson]+sum[rson];
}
void build(int l,int r,int rt){
    if(l==r){
        scanf("%d",&sum[rt]);
        return;
    }
    int mid;
    build(le);
    build(ri);
    push_up(rt);
}
void update(int p,int v,int l,int r,int rt){
    if(l==r){
        sum[rt]+=v;
        return;
    }
    int mid;
    if(p<=m)update(p,v,le);
    else update(p,v,ri);
    push_up(rt);
}
```

```
int query(int L,int R,int l,int r,int rt){
    if(L<=l&&r<=R){
        return sum[rt];
    }
    int mid;
    int ret=0;
    if(L<=m)ret+=query(L,R,le);
    if(R>m)ret+=query(L,R,ri);
    return ret;
}
```

2.3.2 Interval Update

```
const int MAXN=1e5+5;
ll lazy[MAXN<<2];
ll tree[MAXN<<2];
void push_up(int rt){
    tree[rt]=tree[lson]+tree[rson];
}
void push_down(int rt,int m){
    ll w=lazy[rt];
    if(w){
        lazy[lson]+=w;
        lazy[rson]+=w;
        tree[lson]+=w*(m-(m>>1));
        tree[rson]+=w*(m>>1);
        lazy[rt]=0;
    }
}
void build(int l,int r,int rt){
    lazy[rt]=0;
    if(l==r){
        scanf("%lld",&tree[rt]);
        return;
    }
    int mid;
    build(le);
    build(ri);
    push_up(rt);
}
void update(int L,int R,int v,int l,int r,int rt){
    if(L<=l&&r<=R){
        lazy[rt]+=v;
        tree[rt]+=1ll*v*(r-l+1);
        return;
    }
    push_down(rt,r-l+1);
    int mid;
    if(L<=m)update(L,R,v,le);
    if(R>m)update(L,R,v,ri);
    push_up(rt);
}
ll query(int L,int R,int l,int r,int rt){
    if(L<=l&&r<=R){
        return tree[rt];
    }
    push_down(rt,r-l+1);
    int mid;
    ll ret=0;
    if(L<=m)ret+=query(L,R,le);
    if(R>m)ret+=query(L,R,ri);
    return ret;
}
```

2.4 Splay Tree

```
#define key_value ch[ch[rt][1]][0]
const int MAXN=1e5;
struct Splay{
    int a[MAXN];//0 base
    int sz[MAXN],ch[MAXN][2],fa[MAXN];
    int key[MAXN],rev[MAXN];
    int rt,tot;
    int stk[MAXN],top;
    void push_up(int x){
        sz[x]=sz[ch[x][0]]+sz[ch[x][1]]+1;
    }
    void push_down(int x){
        if(rev[x]){
            swap(ch[x][0],ch[x][1]);
            if(ch[x][0])rev[ch[x][0]]^=1;
            if(ch[x][1])rev[ch[x][1]]^=1;
            rev[x]=0;
        }
    }
    int newnode(int p=0,int k=0){
        int x=top?stk[top--]:++tot;
        fa[x]=p;
        sz[x]=1;
        ch[x][0]=ch[x][1]=0;
        key[x]=k;
        rev[x]=0;
        return x;
    }
    int build(int l,int r,int p){
        if(l>r)return 0;
        int mid=(l+r)>>1;
        int x=newnode(p,a[mid]);
        ch[x][0]=build(l,mid-1,x);
        ch[x][1]=build(mid+1,r,x);
        push_up(x);
        return x;
    }
    void init(int n){
        tot=0,top=0;
        rt=newnode(0,-1);
        ch[rt][1]=newnode(rt,-1);
        rep(i,0,n-1)a[i]=i+1;
        key_value=build(0,n-1,ch[rt][1]);
        push_up(ch[rt][1]);
        push_up(rt);
    }
    void rotate(int x,int d){
        int y=fa[x];
        push_down(y);
        push_down(x);
        ch[y][d^1]=ch[x][d];
        fa[ch[x][d]]=y;
        if(fa[y])ch[fa[y]][ch[fa[y]][1]==y]=x;
        fa[x]=fa[y];
        ch[x][d]=y;
        fa[y]=x;
        push_up(y);
    }
    void splay(int x,int goal=0){
        push_down(x);
        while(fa[x]!=goal){
            if(fa[fa[x]]==goal){
                rotate(x,ch[fa[x]][0]==x);
            }else{
                int y=fa[x];
                int d=ch[fa[y]][0]==y;
                ch[y][d]==x?rotate(x,d^1):rotate(y,d);
                rotate(x,d);
            }
        }
        push_up(x);
        if(goal==0)rt=x;
    }
    int kth(int r,int k){
        push_down(r);
        int t=sz[ch[r][0]]+1;
        if(t==k)return r;
        return t>k?kth(ch[r][0],k):kth(ch[r][1],k-t);
    }
    void select(int l,int r){
        splay(kth(rt,1),0);
        splay(kth(ch[rt][1],r-l+2),rt);
    }
};
```

2.5 Functional Segment Tree

```
//poj 2104
const int MAXN=1e5+6;
int n,m,cnt,x,y,k,root[MAXN],a[MAXN];
struct node{int l,r,sum;}T[MAXN*40];
vi v;
int getid(int x){return lower_bound(all(v),x)-v.begin()+1;}
void update(int l,int r,int &x,int y,int pos){
    x=++cnt;
    T[x]=T[y];
    T[x].sum++;
    if(l==r)return;
    int mid=(l+r)>>1;
    if(mid>=pos)update(l,mid,T[x].l,T[y].l,pos);
    else update(mid+1,r,T[x].r,T[y].r,pos);
}
int query(int l,int r,int x,int y,int k){
    if(l==r)return l;
    int sum=T[T[y].l].sum-T[T[x].l].sum;
    int mid=(l+r)>>1;
    if(sum>=k)return query(l,mid,T[x].l,T[y].l,k);
    else return query(mid+1,r,T[x].r,T[y].r,k-sum);
```

```cpp
}
int work(){
    scanf("%d%d",&n,&m);
    v.clear();
    rep(i,1,n)scanf("%d",&a[i]),v.pb(a[i]);
    sort(all(v)),v.erase(unique(all(v)),v.end());
    cnt=0;
    rep(i,1,n)update(1,n,root[i],root[i-1],getid(a[i]));
    rep(i,1,m)scanf("%d%d%d",&x,&y,&k),printf("%d\n",v[query(1,n,root[x-1],root[y],k)
    -1]);
    return 0;
}
```

2.6 Sparse Table

```cpp
//Frequent values UVA - 11235
#include<bits/stdc++.h>
using namespace std;
const int MAXN=1e5+10;
int dp[MAXN][33];
int a[MAXN],b[MAXN],Belong[MAXN];
int rmq(int l,int r){
    int k=31-__builtin_clz(r-l+1);
    return max(dp[l][k],dp[r-(1<<k)+1][k]);
}
int main(){
    int n;
    while(scanf("%d",&n),n){
        int q;
        scanf("%d",&q);
        int index=0;
        int now=-111111;
        for(int i=1;i<=n;i++){
            int x;
            scanf("%d",&x);
            if(now!=x){
                index++;
                now=x;
                a[index]=i;
            }
            Belong[i]=index;
            b[index]=i;
        }
        for(int i=1;i<=index;i++){
            dp[i][0]=b[i]-a[i]+1;
        }
        for (int j = 1; (1 << j) <= index; j++){
            for (int i = 1; i + (1 << j) - 1 <= index; i++){
                dp[i][j] = max(dp[i][j - 1], dp[i + (1 << (j - 1))][j - 1]);
            }
        }
        while(q--){
            int l,r;
            scanf("%d%d",&l,&r);
            if(Belong[l]==Belong[r]){
                printf("%d\n",r-l+1);
            }else{
                int pos1=Belong[l];
                int ans=b[pos1]-l+1;
                int pos2=Belong[r];
                ans=max(ans,r-a[pos2]+1);
                pos1++;
                pos2--;
                if(pos1<=pos2){
                    ans=max(ans,rmq(pos1,pos2));
                }
                printf("%d\n",ans);
            }
        }

    }
    return 0;
}
```

3 Graph Theory

3.1 Union-Find Set

```
const int MAXN=1e6+5;
struct DSU{
    int p[MAXN];
    void init(int n){rep(i,0,n)p[i]=i;}
    int findp(int x){return x==p[x]?x:p[x]=findp(p[x]);}
    void unite(int x,int y){x=findp(x);y=findp(y);if(x==y)return;p[y]=x;}
    bool same(int x,int y){return findp(x)==findp(y);}
}dsu;
```

3.2 Minimal Spanning Tree

3.2.1 Kruskal

```
//poj 1258
#include<cstdio>
#include<algorithm>
using namespace std;
const int MAXE=1e5+5;
const int MAXN=1e5+5;
struct DSU{
    int p[MAXN];
    void init(int n){for(int i=0;i<=n;i++)p[i]=i;}
    int findp(int x){return x==p[x]?x:p[x]=findp(p[x]);}
    void unite(int x,int y){x=findp(x);y=findp(y);if(x==y)return;p[y]=x;}
    bool same(int x,int y){return findp(x)==findp(y);}
}dsu;
struct edge{int u,v,cost;}es[MAXE];
bool cmp(const edge &x,const edge &y){return x.cost<y.cost;}
int V,E;
int kruskal(){
    sort(es,es+E,cmp);
    dsu.init(V);
    int res=0;
    for(int i=0;i<E;i++){
        if(!dsu.same(es[i].u,es[i].v)){
            dsu.unite(es[i].u,es[i].v);
            res+=es[i].cost;
        }
    }
    return res;
}
int main(){
    while(~scanf("%d",&V)){
        E=0;
        for(int i=1;i<=V;i++){
            for(int j=1;j<=V;j++){
                int w;
                scanf("%d",&w);
                if(i==j)continue;
                es[E].u=i;
                es[E].v=j;
                es[E].cost=w;
                E++;
            }
        }
        printf("%d\n",kruskal());
    }
    return 0;
}
```

3.3 Shortest Path

3.3.1 Dijkstra

```
#include<bits/stdc++.h>
using namespace std;
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define clr(a,x) memset(a,x,sizeof(a))
#define mp make_pair
const int MAXV=2e6;
const int MAXE=5e6+10;
typedef long long anytype;
typedef pair<anytype,int> P;
int tot=0;
int head[MAXV];
struct Edge{
    int v,c,nxt;
    Edge(){}
    Edge(int v,int c,int nxt):v(v),c(c),nxt(nxt){}
}edge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v,int c){
    edge[tot]=Edge(v,c,head[u]);
    head[u]=tot++;
}
anytype d[MAXV];
void dij(int s){
    priority_queue<P,vector<P>,greater<P> > que;
    clr(d,-1);
    d[s]=0;
    que.push(P(0,s));
    while(!que.empty()){
        P t=que.top();
        que.pop();
        int v=t.second;
        if(d[v]!=-1&&d[v]<t.first)continue;
        for(int i=head[v];~i;i=edge[i].nxt){
            Edge e=edge[i];
            if(d[e.v]==-1||d[e.v]>d[v]+e.c){
                d[e.v]=d[v]+e.c;
                que.push(mp(d[e.v],e.v));
            }
        }
    }
}
int main(){
    int T;
    scanf("%d",&T);
    while(T--){
        int n,m,k;
        scanf("%d%d%d",&n,&m,&k);
        init();
        rep(i,1,m){
            int u,v,c;
            scanf("%d%d%d",&u,&v,&c);
            rep(j,0,k){
                add_edge(u+j*n,v+j*n,c);
                if(j!=k)add_edge(u+j*n,v+(j+1)*n,0);
            }
        }
        dij(1);
        printf("%lld\n",d[n+k*n]);
    }
    return 0;
}
```

3.3.2 Spfa

```
//hdu3592
const int MAXN=1e3+5;
const int MAXE=3e4+5;
const int INF=0x3f3f3f3f;
int N,X,Y;
int tot;
int head[MAXN];
struct Edge{
    int v,w,nxt;
    Edge(){}
    Edge(int v,int w,int nxt):v(v),w(w),nxt(nxt){}
}edge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v,int w){
    edge[tot]=Edge(v,w,head[u]);
    head[u]=tot++;
}
queue<int> que;
bool inq[MAXN];
int qtime[MAXN];
int d[MAXN];
int spfa(){
    while(!que.empty())que.pop();
    clr(qtime,0);
    clr(inq,0);
    rep(i,1,N)d[i]=INF;
    d[1]=0;
    que.push(1);
    inq[1]=1;
    qtime[1]++;
    while(!que.empty()){
        int u=que.front();
        que.pop();
        inq[u]=0;
        for(int i=head[u];i!=-1;i=edge[i].nxt){
            int v=edge[i].v;
            int w=edge[i].w;
```

```
                if(d[v]>d[u]+w){
                    d[v]=d[u]+w;
                    if(!inq[v]){
                        que.push(v);
                        inq[v]=1;
                        qtime[v]++;
                        if(qtime[v]>N)return -1;
                    }
                }
            }
        }
    if(d[N]==INF)return -2;
    else return d[N];
}
int work(){
    int T;
    scanf("%d",&T);
    while(T--){
        scanf("%d%d%d",&N,&X,&Y);
        init();
        rep(i,1,N-1){
            add_edge(i+1,i,0);
        }
        while(X--){
            int x,y,z;
            scanf("%d%d%d",&x,&y,&z);
            add_edge(x,y,z);
        }
        while(Y--){
            int x,y,z;
            scanf("%d%d%d",&x,&y,&z);
            add_edge(y,x,-z);
        }
        printf("%d\n",spfa());
    }
    return 0;
}
```

3.4 Topo Sort

```
//cf 915D
const int MAXN=505;
const int MAXM=1e5+5;
int n,m;
int tot;
int head[MAXN],cur[MAXN],idec[MAXN];
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXM];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    head[u]=tot++;
}
int que[MAXN];
int st,ed;
bool topsort(int x){
    int nst=1,ned=0;
    rep(i,1,n)cur[i]=idec[i];
    cur[x]--;
    que[++ned]=x;
    while(nst<=ned){
        int u=que[nst++];
        for(int i=head[u];i!=-1;i=edge[i].nxt){
            int v=edge[i].v;
            if(--cur[v]==0)que[++ned]=v;
        }
    }
    if(ned+ed==n)return true;
    else return false;
}
int work(){
    scanf("%d%d",&n,&m);
    init();
    while(m--){
        int u,v;
        scanf("%d%d",&u,&v);
        add_edge(u,v);
        idec[v]++;
    }
    st=1,ed=0;
    rep(i,1,n){
        if(idec[i]==0)que[++ed]=i;
    }
    while(st<=ed){
        int u=que[st++];
        for(int i=head[u];i!=-1;i=edge[i].nxt){
            int v=edge[i].v;
            if(--idec[v]==0)que[++ed]=v;
        }
    }
    if(ed==n){
        puts("YES");
        return 0;
    }
    rep(i,1,n){
        if(idec[i]==1){
            if(topsort(i)){
                puts("YES");
                return 0;
            }
        }
    }
    puts("NO");
    return 0;
}
```

3.5 LCA

3.5.1 LCArmq

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

```
#define MAXV 100005
#define MAXLOGV 32
using namespace std;
int N,M,Q;
int st[MAXLOGV][MAXV];
vector<int> G[MAXV];
int root;
int vs[MAXV*2];
int depth[MAXV*2];
int id[MAXV];
void dfs(int v,int p,int d,int &k){
    id[v]=k;
    vs[k]=v;
    depth[k++]=d;
    for(int i=0;i<G[v].size();i++){
        if(G[v][i]!=p){
            dfs(G[v][i],v,d+1,k);
            vs[k]=v;
            depth[k++]=d;
        }
    }
}
int getMin(int x, int y){
    return depth[x]<depth[y]?x:y;
}

void rmq_init(int n){
    for(int i=0;i<n;++i) st[0][i]=i;
    for(int i=1;1<<i<n;++i)
        for(int j=0;j+(1<<i)-1<n;++j)
            st[i][j]=getMin(st[i-1][j],st[i-1][j+(1<<(i-1))]);
}
void init(int V){
    int k=0;
    dfs(root,-1,0,k);
    rmq_init(V*2-1);
}
int query(int l, int r){
    int k=31-__builtin_clz(r-l+1);
    return getMin(st[k][l],st[k][r-(1<<k)+1]);
}
int lca(int u,int v){
    if(u==v) return u;
    return vs[query(min(id[u],id[v]),max(id[u],id[v]))];
}
int dis(int u,int v){
    return depth[id[u]]+depth[id[v]]-2*depth[id[lca(u,v)]];
}
int main()
{
    scanf("%d%d",&N,&M);
    for(int i=0;i<M;i++){
        int x,y;
        scanf("%d%d",&x,&y);
        G[x].push_back(y);
        G[y].push_back(x);
    }
    root=0;
    init(N);
    scanf("%d",&Q);
    while(Q--){
        int x,y;
        scanf("%d%d",&x,&y);
        printf("%d\n",lca(x,y));
    }
    return 0;
}
```

3.6 Depth-First Traversal

```
vector<int> G[MAXN];
int vis[MAXN];
void dfs(int u){
    vis[u]=1;
    PREVISIT(u);
    for(auto v:G[u]){
        if(!vis[v])dfs(v);
    }
    POSTVISIT(u);
}
```

3.6.1 Biconnected-Component

```
//UVALive - 3523
#include<bits/stdc++.h>
using namespace std;
#define clr(a,x) memset(a,x,sizeof(a))
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define mp make_pair
#define fi first
#define se second
#define pb push_back
typedef pair<int,int> pii;
typedef vector<int> vi;
const int MAXV=1e3+10;
const int MAXE=1e6+10;
int tot;
int head[MAXV];
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE<<1];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    head[u]=tot++;
}
int pre[MAXV],is_cut[MAXV],bccno[MAXV],dfs_clock,bcc_cnt;
vi bcc[MAXV];
stack<pii > st;
int dfs(int u,int fa){
    int lowu=pre[u]=++dfs_clock;
    int child=0;
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        pii e=mp(u,v);
        if(!pre[v]){
            st.push(e);
            child++;
            int lowv=dfs(v,u);
            lowu=min(lowu,lowv);
            if(lowv>=pre[u]){
                is_cut[u]=1;
                bcc_cnt++;
                bcc[bcc_cnt].clear();
                for(;;){
                    pii x=st.top();
                    st.pop();
                    if(bccno[x.fi]!=bcc_cnt){
                        bcc[bcc_cnt].pb(x.fi);
                        bccno[x.fi]=bcc_cnt;
                    }
                    if(bccno[x.se]!=bcc_cnt){
                        bcc[bcc_cnt].pb(x.se);
                        bccno[x.se]=bcc_cnt;
                    }
                    if(x.fi==u&&x.se==v)break;
                }
            }
        }else if(pre[v]<pre[u]&&v!=fa){
            st.push(e);
            lowu=min(lowu,pre[v]);
        }
    }
    if(fa<0&&child==1)is_cut[u]=0;
    return lowu;
}
void find_bcc(int n){
    clr(pre,0);
    clr(is_cut,0);
    clr(bccno,0);
    dfs_clock=bcc_cnt=0;
    rep(i,1,n){
        if(!pre[i])dfs(i,-1);
    }
}
int odd[MAXV],color[MAXV];
bool bipartite(int u,int b){
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        if(bccno[v]!=b)continue;
        if(color[v]==color[u])return false;
        if(!color[v]){
            color[v]=3-color[u];
            if(!bipartite(v,b))return false;
        }
    }
    return true;
}
bool mmp[MAXV][MAXV];
int main(){
    int n,m;
    while(scanf("%d%d",&n,&m),n+m){
        clr(mmp,0);
        rep(i,1,m){
            int x,y;
            scanf("%d%d",&x,&y);
            mmp[x][y]=1;
            mmp[y][x]=1;
        }
        init();
        rep(i,1,n){
            rep(j,i+1,n){
                if(!mmp[i][j]){
                    add_edge(i,j);
                    add_edge(j,i);
                }
            }
        }
        find_bcc(n);
        clr(odd,0);
        for(int i=1;i<=bcc_cnt;i++){
            clr(color,0);
            for(int j=0;j<bcc[i].size();j++){
                bccno[bcc[i][j]]=i;
            }
            int u=bcc[i][0];
            color[u]=1;
            if(!bipartite(u,i)){
                for(int j=0;j<bcc[i].size();j++){
                    odd[bcc[i][j]]=1;
                }
            }
        }
        int ans=n;
        rep(i,1,n)if(odd[i])ans--;
        printf("%d\n",ans);
    }
    return 0;
}
```

3.6.2 Strongly Connected Component

```
const int MAXV=1e4+10;
const int MAXE=1e5+10;
int tot,head[MAXV];
int low[MAXV],dfn[MAXV],stk[MAXV],Belong[MAXV];
int idx,top,scc;
bool instk[MAXV];
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    head[u]=tot++;
}
void Tarjan(int u){
    int v;
    low[u]=dfn[u]=++idx;
    stk[top++]=u;
    instk[u]=true;
    for(int i=head[u];~i;i=edge[i].nxt){
        v=edge[i].v;
        if(!dfn[v]){
            Tarjan(v);
            if(low[u]>low[v])low[u]=low[v];
        }else if(instk[v]&&low[u]>dfn[v])low[u]=dfn[v];
    }
    if(low[u]==dfn[u]){
        scc++;
        do{
            v=stk[--top];
            instk[v]=false;
            Belong[v]=scc;
        }while(v!=u);
    }
}
void tscc(int N){
    clr(dfn,0);
    clr(instk,0);
    idx=scc=top=0;
    rep(i,1,N)if(!dfn[i])Tarjan(i);
}
```

3.6.3 Kosaraju

```
const int MAXV=2e4+10;
const int MAXE=5e4+10;
int tot,scc,head[MAXV],rhead[MAXV],Belong[MAXV];
bool vis[MAXV];
int stk[MAXV],top;
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE],redge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
    clr(rhead,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    redge[tot]=Edge(u,rhead[v]);
    head[u]=rhead[v]=tot++;
}
void dfs(int u){
    vis[u]=true;
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        if(!vis[v])dfs(v);
    }
    stk[++top]=u;
}
void rdfs(int u,int k){
    vis[u]=true;
    Belong[u]=k;
    for(int i=rhead[u];~i;i=redge[i].nxt){
        int v=redge[i].v;
        if(!vis[v])rdfs(v,k);
    }
}
void kscc(int V){
    scc=top=0;
    clr(vis,0);
    rep(i,1,V)if(!vis[i])dfs(i);
    clr(vis,0);
    per(i,top,1){
        int v=stk[i];
        if(!vis[v])rdfs(v,++scc);
    }
}
```

3.6.4 TwoSAT

```
//poj3683
//0 base !
//if (x V (!y))then add_clause(1,x,0,y)
//if  x then add_var(1,x)
const int MAXV=1e5;
const int MAXE=3e6+5;
int tot,scc,head[MAXV],rhead[MAXV],Belong[MAXV];
bool vis[MAXV];
int stk[MAXV],top;
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE],redge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
    clr(rhead,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    redge[tot]=Edge(u,rhead[v]);
    head[u]=rhead[v]=tot++;
}
void dfs(int u){
    vis[u]=true;
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        if(!vis[v])dfs(v);
    }
    stk[++top]=u;
}
void rdfs(int u,int k){
    vis[u]=true;
    Belong[u]=k;
    for(int i=rhead[u];~i;i=redge[i].nxt){
        int v=redge[i].v;
        if(!vis[v])rdfs(v,k);
    }
}
void kscc(int V){
    scc=top=0;
    clr(vis,0);
    rep(i,0,V-1)if(!vis[i])dfs(i);
    clr(vis,0);
    per(i,top,1){
        int v=stk[i];
        if(!vis[v])rdfs(v,++scc);
    }
}
void add_clause(int xv,int x,int yv,int y){
    x=x<<1|xv;
    y=y<<1|yv;
    add_edge(x^1,y);
    add_edge(y^1,x);
}
void add_var(int xv,int x){
    x=x<<1|xv;
    add_edge(x^1,x);
}
int st[MAXV],ed[MAXV],d[MAXV];
char tm[10];
int fun(){
    int res=0;
    int h=(tm[0]-'0')*10+tm[1]-'0';
    res=h*60;
    res+=(tm[3]-'0')*10+tm[4]-'0';
    return res;
}
int work(){
    int n;
    scanf("%d",&n);
    rep(i,0,n-1){
        scanf("%s",tm);
        st[i]=fun();
        scanf("%s",tm);
        ed[i]=fun();
        scanf("%d",&d[i]);
    }
    init();
    rep(i,0,n-1){
        rep(j,0,i-1){
            if(min(st[i]+d[i],st[j]+d[j])>max(st[i],st[j])){
                add_clause(0,i,0,j);
            }
            if(min(st[i]+d[i],ed[j])>max(st[i],ed[j]-d[j])){
                add_clause(0,i,1,j);
            }
            if(min(ed[i],st[j]+d[j])>max(ed[i]-d[i],st[j])){
                add_clause(1,i,0,j);
            }
            if(min(ed[i],ed[j])>max(ed[i]-d[i],ed[j]-d[j])){
                add_clause(1,i,1,j);
            }
        }
    }
    kscc(2*n);
    rep(i,0,n-1){
        if(Belong[i<<1]==Belong[i<<1|1]){
            puts("NO");
            return 0;
        }
    }
    puts("YES");
    rep(i,0,n-1){
        if(Belong[i<<1|1]>Belong[i<<1]){
            printf("%02d:%02d %02d:%02d\n",st[i]/60,st[i]%60,(st[i]+d[i])/60,(st[i]+d[i
    ])%60);
        }else{
            printf("%02d:%02d %02d:%02d\n",(ed[i]-d[i])/60,(ed[i]-d[i])%60,ed[i]/60,ed[
    i]%60);
        }
    }
    return 0;
}
```

3.6.5 cut$_v$$ertex$

```
//poj 1144
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define clr(a,x) memset(a,x,sizeof(a))
const int MAXV=105;
const int MAXE=1e5;
int tot;
int head[MAXV];
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE<<1];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    head[u]=tot++;
}
int n;
bool is_cut[MAXV];
int low[MAXV],pre[MAXV];
int dfs_clock;
int dfs(int u,int fa){
    int lowu=pre[u]=++dfs_clock;
    int child=0;
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        if(!pre[v]){
            child++;
            int lowv=dfs(v,u);
            lowu=min(lowu,lowv);
            if(lowv>=pre[u]){
                is_cut[u]=true;
            }
        }else if(pre[v]<pre[u]&&v!=fa){
            lowu=min(lowu,pre[v]);
        }
    }
    if(fa<0&&child==1)is_cut[u]=false;
    low[u]=lowu;
    return lowu;
}
int main(){
    while(scanf("%d",&n),n){
        init();
        int x;
        while(scanf("%d",&x),x){
            int y;
            while(getchar()!='\n'){
                scanf("%d",&y);
                add_edge(x,y);
                add_edge(y,x);
            }
        }
        clr(is_cut,0);
        clr(low,0);
        clr(pre,0);
        dfs_clock=0;
        int cnt=0;
        dfs(1,-1);
        for(int i=1;i<=n;i++){
            if(is_cut[i])cnt++;
        }
        printf("%d\n",cnt);
    }
    return 0;
}
```

3.7 Bipartite Graph Matching

3.7.1 Hungry

```
//poj3041
const int MAXV=1e3+5;
struct BM{
    int V;
    vi G[MAXV];
    int match[MAXV];
    bool vis[MAXV];
    void init(int x){
        V=x;
        rep(i,1,V)G[i].clear();
    }
    void add_edge(int u,int v){
        G[u].pb(v);
        G[v].pb(u);
    }
    bool dfs(int u){
        vis[u]=true;
        for(int i=0;i<(int)G[u].size();i++){
            int v=G[u][i];
            int w=match[v];
            if(w==-1||(!vis[w]&&dfs(w))){
                match[u]=v;
                match[v]=u;
                return true;
            }
        }
        return false;
    }
    int matching(){
        int ret=0;
        clr(match,-1);
        rep(i,1,V){
            if(match[i]==-1){
                clr(vis,0);
                if(dfs(i))ret++;
            }
        }
        return ret;
    }
}bm;
int work(){
    int n,k;
    scanf("%d%d",&n,&k);
    bm.init(2*n);
    while(k--){
        int u,v;
        scanf("%d%d",&u,&v);
        bm.add_edge(u,n+v);
    }
    printf("%d",bm.matching());
    return 0;
}
```

3.8 Network Flow

3.8.1 Dinic

```
//poj 3281
#include<cstdio>
#include<iostream>
#include<algorithm>
#include<cstring>
#include<queue>
using namespace std;
#define clr(a,x) memset(a,x,sizeof(a))
const int MAXV=400+5;
const int MAXE=1e5+5;
const int INF=0x3f3f3f3f;
int tot;
int head[MAXV],level[MAXV],iter[MAXV];
struct Edge{
    int v,cap,nxt;
    Edge(){}
    Edge(int v,int cap,int nxt):v(v),cap(cap),nxt(nxt){}
}edge[MAXE<<1];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v,int c){
    edge[tot]=Edge(v,c,head[u]);
    head[u]=tot++;
    edge[tot]=Edge(u,0,head[v]);
    head[v]=tot++;
}
void bfs(int s){
    clr(level,-1);
    level[s]=0;
    queue<int> que;
    que.push(s);
    while(!que.empty()){
        int u=que.front();
        que.pop();
        for(int i=head[u];~i;i=edge[i].nxt){
            int v=edge[i].v;
            int c=edge[i].cap;
            if(c>0&&level[v]<0){
                level[v]=level[u]+1;
                que.push(v);
            }
        }
    }
}
int dfs(int u,int t,int f){
    if(u==t)return f;
    for(int &i=iter[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        int c=edge[i].cap;
        if(c>0&&level[u]<level[v]){
            int d=dfs(v,t,min(f,c));
            if(d>0){
                edge[i].cap-=d;
                edge[i^1].cap+=d;
                return d;
            }
        }
    }
    return 0;
}
int max_flow(int s,int t){
    int flow=0;
    while(1){
        bfs(s);
        if(level[t]<0)return flow;
        int f;
        memcpy(iter,head,sizeof(head));
        while(f=dfs(s,t,INF))flow+=f;
    }
}
int main(){
    int n,f,d;
    scanf("%d%d%d",&n,&f,&d);
    int s=0,t=2*n+f+d;
    init();
    for(int i=1;i<=f;i++){
        add_edge(s,2*n+i,1);
    }
    for(int i=1;i<=d;i++){
        add_edge(2*n+f+i,t,1);
    }
    for(int i=1;i<=n;i++){
        add_edge(i,n+i,1);
        int ff,dd;
        scanf("%d%d",&ff,&dd);
        while(ff--){
            int x;
            scanf("%d",&x);
            add_edge(2*n+x,i,1);
        }
        while(dd--){
            int x;
            scanf("%d",&x);
            add_edge(n+i,2*n+f+x,1);
        }
    }
    printf("%d",max_flow(s,t));
    return 0;
}
```

3.8.2 MinCost MaxFlow

```
// poj2135
#include<cstdio>
#include<vector>
#include<algorithm>
#include<queue>
using namespace std;
const int MAXV=1005;
const int MAXE=50000;
const int INF=100000000;
typedef pair<int,int> P;
struct edge{int to,cap,cost,rev;};
int dist[MAXV],h[MAXV],prevv[MAXV],preve[MAXV];
int V;
vector<edge> G[MAXV];
void add_edge(int from,int to,int cap,int cost){
    G[from].push_back((edge){to,cap,cost,G[to].size()});
    G[to].push_back((edge){from,0,-cost,G[from].size()-1});
}
int min_cost_flow(int s,int t,int f){
    int res=0;
    fill(h,h+V,0);
    while(f>0){
        priority_queue<P,vector<P>,greater<P> >que;
        fill(dist,dist+V,INF);
        dist[s]=0;
        que.push(P(0,s));
        while(!que.empty()){
            P p=que.top(); que.pop();
            int v=p.second;
            if(dist[v]<p.first) continue;
            for(int i=0;i<G[v].size();i++){
                edge &e=G[v][i];
                if(e.cap>0&&dist[e.to]>dist[v]+e.cost+h[v]-h[e.to]){
                    dist[e.to]=dist[v]+e.cost+h[v]-h[e.to];
                    prevv[e.to]=v;
                    preve[e.to]=i;
                    que.push(P(dist[e.to],e.to));
                }
            }
        }
        if(dist[t]==INF){
            return -1;
        }
        for(int v=0;v<V;v++) h[v]+=dist[v];
        int d=f;
        for(int v=t;v!=s;v=prevv[v]){
            d=min(d,G[prevv[v]][preve[v]].cap);
        }
        f-=d;
        res+=d*h[t];
        for(int v=t;v!=s;v=prevv[v]){
            edge &e=G[prevv[v]][preve[v]];
            e.cap-=d;
            G[v][e.rev].cap+=d;
        }
    }
    return res;
}
int main(){
    int N,M;
    scanf("%d%d",&N,&M);
    V=N;
    for(int i=1;i<=M;i++){
        int x,y,z;
        scanf("%d%d%d",&x,&y,&z);
        add_edge(x-1,y-1,1,z);
        add_edge(y-1,x-1,1,z);
    }
    printf("%d",min_cost_flow(0,N-1,2));
    return 0;
}
```

4 Others

4.1 Matrix

4.1.1 Matrix FastPow

```
typedef vector<ll> vec;
typedef vector<vec> mat;
mat mul(mat& A, mat& B)
{
    mat C(A.size(), vec(B[0].size()));
    for (int i = 0; i < A.size(); i++)
        for (int k = 0; k < B.size(); k++)
            if (A[i][k]) // □□□□□□□□
                for (int j = 0; j < B[0].size(); j++)
                    C[i][j] = (C[i][j] + A[i][k] * B[k][j]) % mod;
    return C;
}
mat Pow(mat A, ll n)
{
    mat B(A.size(), vec(A.size()));
    for (int i = 0; i < A.size(); i++) B[i][i] = 1;
    for (; n; n >>= 1, A = mul(A, A))
        if (n & 1) B = mul(B, A);
    return B;
}
```

4.2 Tricks

4.2.1 Stack-Overflow

```
#pragma comment(linker, "/STACK:1024000000,1024000000")
```

4.2.2 Fast-Scanner

```
template <class T>
inline bool scan_d(T &ret){
    char c;
    int sgn;
    if (c = getchar(), c == EOF) return 0; //EOF
    while (c != '-' && (c < '0' || c > '9')) c = getchar();
    sgn = (c == '-') ? -1 : 1;
    ret = (c == '-') ? 0 : (c - '0');
    while (c = getchar(), c >= '0' && c <= '9') ret = ret * 10 + (c - '0');
    ret *= sgn;
    return 1;
}
inline void out(int x){
    if(x<0){
        putchar('-');
        x=-x;
    }
    if (x > 9) out(x / 10);
    putchar(x % 10 + '0');
}
```

4.2.3 Strok-Sscanf

```
// get some integers in a line
gets(buf);
int v;
char *p = strtok(buf, " ");
while (p){
    sscanf(p, "%d", &v);
    p = strtok(NULL," ");
}
```

4.3 Mo Algorithm

```
//cf 671 E
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int MAXN=1<<20;
struct node{
    int l,r,id;
}Q[MAXN];
int n,m,k;
int block;
int a[MAXN];
int pre[MAXN];
ll cnt[MAXN];
ll ANS,ans[MAXN];
bool cmp(node x,node y){
    if(x.l/block==y.l/block)return x.r<y.r;
    else return x.l/block<y.l/block;
}
void add(int x){
    ANS+=cnt[pre[x]^k];
    cnt[pre[x]]++;
}
void del(int x){
    cnt[pre[x]]--;
    ANS-=cnt[pre[x]^k];
}
int main(){
    scanf("%d%d%d",&n,&m,&k);
    block=(int)sqrt(n);
    pre[0]=0;
    for(int i=1;i<=n;i++){
        scanf("%d",&a[i]);
        pre[i]=a[i]^pre[i-1];
    }
    for(int i=1;i<=m;i++){
        scanf("%d%d",&Q[i].l,&Q[i].r);
        Q[i].id=i;
    }
    sort(Q+1,Q+1+m,cmp);
    ANS=0;
    memset(cnt,0,sizeof(cnt));
    cnt[0]=1;
    int L=1,R=0;
    for(int i=1;i<=m;i++){
        while(L>Q[i].l){L--;add(L-1);};
        while(L<Q[i].l){del(L-1);L++;}
        while(R<Q[i].r){R++;add(R);};
        while(R>Q[i].r){del(R);R--;};
        ans[Q[i].id]=ANS;
    }
    for(int i=1;i<=m;i++){
        printf("%lld\n",ans[i]);
    }
    return 0;
}
```

4.4 BigNum

4.4.1 High-precision

```
import java.io.*;
import java.math.BigInteger;

public class Main {
    public static void main(String args[]) throws IOException {
        StreamTokenizer in = new StreamTokenizer(new BufferedReader(new
    InputStreamReader(System.in)));
        PrintWriter out = new PrintWriter(new OutputStreamWriter(System.out));

        BigInteger a;
        BigInteger b;
        in.nextToken();
        int A=(int) in.nval;
        in.nextToken();
        int B=(int) in.nval;

        a=BigInteger.valueOf(A);
        b=BigInteger.valueOf(B);
        out.println(a.pow(B).subtract(b.pow(A)));
        out.flush();
    }
}
```

4.5 VIM

```
syntax on
set nu
set tabstop=4
set expandtab
set autoindent
set cin
set mouse=a

map<F2> :call SetTitle()<CR>
func SetTitle()
let l = 0
let l = l + 1 | call setline(l,'#include <algorithm>')
let l = l + 1 | call setline(l,'#include  <iostream>')
let l = l + 1 | call setline(l,'#include   <cstring>')
let l = l + 1 | call setline(l,'#include    <string>')
let l = l + 1 | call setline(l,'#include    <cstdio>')
let l = l + 1 | call setline(l,'#include    <vector>')
let l = l + 1 | call setline(l,'#include     <stack>')
let l = l + 1 | call setline(l,'#include     <queue>')
let l = l + 1 | call setline(l,'#include     <cmath>')
let l = l + 1 | call setline(l,'#include       <set>')
let l = l + 1 | call setline(l,'#include       <map>')
let l = l + 1 | call setline(l,'using namespace std;')
let l = l + 1 | call setline(l,'#define rep(i,a,b) for(int i=a;i<=b;i++)')
let l = l + 1 | call setline(l,'#define per(i,a,b) for(int i=a;i>=b;i--)')
let l = l + 1 | call setline(l,'#define clr(a,x) memset(a,x,sizeof(a))')
let l = l + 1 | call setline(l,'#define pb push_back')
let l = l + 1 | call setline(l,'#define mp make_pair')
let l = l + 1 | call setline(l,'#define all(x) (x).begin(),(x).end()')
let l = l + 1 | call setline(l,'#define fi first')
let l = l + 1 | call setline(l,'#define se second')
let l = l + 1 | call setline(l,'#define SZ(x) ((int)(x).size())')
let l = l + 1 | call setline(l,'typedef unsigned long long ull;')
let l = l + 1 | call setline(l,'typedef long long ll;')
let l = l + 1 | call setline(l,'typedef vector<int> vi;')
let l = l + 1 | call setline(l,'typedef pair<int,int> pii;')
let l = l + 1 | call setline(l,'/************head******************/')
let l = l + 1 | call setline(l,'int work(){')
let l = l + 1 | call setline(l,'')
let l = l + 1 | call setline(l,'    return 0;')
let l = l + 1 | call setline(l,'}')
let l = l + 1 | call setline(l,'int main(){')
let l = l + 1 | call setline(l,'#ifdef superkunn')
let l = l + 1 | call setline(l,'    freopen("input.txt","rt",stdin);')
let l = l + 1 | call setline(l,'#endif')
let l = l + 1 | call setline(l,'    work();')
let l = l + 1 | call setline(l,'    return 0;')
let l = l + 1 | call setline(l,'}')
endfunc
```

4.6 BASH

```
g++ -g -Wall -std=c++11 -Dsuperkunn main.cpp
./a.out
```

5 Geometry

```
struct Point{
    double x,y;
    Point(double x=0,double y=0):x(x),y(y){}
};
typedef Point Vector;
Vector operator + (Vector A,Vector B){return Vector(A.x+B.x,A.y+B.y);}
Vector operator - (Point A,Point B){return Vector(A.x-B.x,A.y-B.y);}
Vector operator * (Vector A,double p){return Vector(A.x*p,A.y*p);}
Vector operator / (Vector A,double p){return Vector(A.x/p,A.y/p);}
bool operator < (const Point& a,const Point &b){
    return a.x<b.x||(a.x==b.x&&a.y<b.y);
}
const double eps = 1e-10;
int dcmp(double x){
    if(fabs(x)<eps)return 0;else return x<0?-1:1;
}
bool operator == (const Point& a,const Point &b){
    return dcmp(a.x-b.x)==0&&dcmp(a.y-b.y)==0;
}
//(x,y)->  atan2(y,x)
double Dot(Vector A,Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Angle(Vector A,Vector B){return acos(Dot(A,B)/Length(A)/Length(B));}
double Cross(Vector A,Vector B){return A.x*B.y-A.y*B.x;}
double Area2(Point A,Point B,Point C){return Cross(B-A,C-A);}
Vector Rotate(Vector A,double rad){
    return Vector(A.x*cos(rad)-A.y*sin(rad),A.x*sin(rad)+A.y*cos(rad));
}
Vector Normal(Vector A){
    double L=Length(A);
    return Vector(-A.y/L,A.x/L);
}
```