目次

1 環境

1.1 .Xmodmap

［変換］と［半角/全角］,［無変換］と［Escape］を入れ替え,［CapsLock］を［Ctrl］に割り当てる.

```
keysym Muhenkan = Escape
keysym Henkan_Mode = Zenkaku_Hankaku
keysym Escape = Muhenkan
keysym Zenkaku_Hankaku = Henkan_Mode

remove Lock = Caps_Lock
keysym Caps_Lock = Control_L
add Control = Control_L
```

1.2 .vimrc

不要そうなものは適当に削る.

```
set number
set tabstop=4 shiftwidth=4
set textwidth=0
set nobackup noswapfile

set hlsearch noincsearch ignorecase

set cindent
set cinoptions=L0,:0,l1,g0,cs,C1,(s,m1
set cinkeys-=0#
set noshowmatch

set laststatus=2
set tabpagemax=32

set list listchars=tab:^\ ,trail:_

noremap <Up> <Nop>
noremap <Down> <Nop>
noremap <Left> :<C-u>tabprev<Return>
noremap <Right> :<C-u>tabnext<Return>
inoremap <Return> <Return>x<BS>
nnoremap o ox<BS>
nnoremap O Ox<BS>

filetype plugin indent on
syntax enable

augroup misc
  autocmd filetype * setlocal comments=
augroup END
```

1.3 Makefile

```
MAIN=

CXXFLAGS=-Wall -Wextra -Wno-sign-compare -Wno-unused-result -std=c++11 -g -D_GLIBCXX_DEBUG
#CXXFLAGS=-Wall -Wextra -Wno-sign-compare -Wno-unused-result -std=c++11 -O2

all: $(MAIN)
clean:
    find . -maxdepth 1 -type f -perm -111 -delete
run: $(MAIN)
    ./$< <$(MAIN).in
```

2 テンプレート

2.1 C++

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

#define dump(n) cout<<"# "<<#n<<'='<<(n)<<endl
#define repi(i,a,b) for(int i=int(a);i<int(b);i++)
#define peri(i,a,b) for(int i=int(b);i-->int(a);)
#define rep(i,n) repi(i,0,n)
#define per(i,n) peri(i,0,n)
#define all(c) begin(c),end(c)
#define mp make_pair
#define mt make_tuple

typedef unsigned int uint;
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int,int> pii;
typedef vector<int> vi;
typedef vector<vi> vvi;
typedef vector<ll> vl;
typedef vector<vl> vvl;
typedef vector<double> vd;
typedef vector<vd> vvd;
typedef vector<string> vs;

const int INF=1e9;
const int MOD=1e9+7;
const double EPS=1e-9;

template<typename T1,typename T2>
ostream& operator<<(ostream& os,const pair<T1,T2>& p){
    return os<<'('<<p.first<<','<<p.second<<')';
}
template<typename T>
ostream& operator<<(ostream& os,const vector<T>& a){
    os<<'[';
    rep(i,a.size()) os<<(i?" ":"")<<a[i];
    return os<<']';
}

int main()
{
}
```

2.2 Java

Solver#main に書く.

```
import java.io.*;
import java.lang.*;
import java.math.*;
import java.util.*;

class Main{public static void main(String[] args){new Solver();}}

class Solver{
    ScanReader reader=new ScanReader(System.in);
    PrintWriter writer=new PrintWriter(System.out);
    Solver(){main(); writer.flush();}

    void main(){
```

```
    }
}

class ScanReader{
    BufferedReader br;
    StringTokenizer st;
    public ScanReader(InputStream in){
        br=new BufferedReader(new InputStreamReader(in));
        st=null;
    }
    public boolean hasNext(){
        while(st==null || !st.hasMoreTokens()){
            try{
                if(!br.ready())
                    return false;
                st=new StringTokenizer(br.readLine());
            }
            catch(IOException e){
                throw new RuntimeException(e);
            }
        }
        return true;
    }
    public String nextLine(){
        hasNext();
        return st.nextToken("");
    }
    public String next(){
        hasNext();
        return st.nextToken();
    }
}
```

3 グラフ

3.1 定義

マッチング	端点を共有しない辺集合
独立集合	どの 2 点も隣接しない頂点集合
クリーク	どの 2 点も隣接している頂点集合
支配集合	自身とその隣接頂点のみで全頂点を被覆する頂点集合
辺被覆	どの頂点も少なくとも 1 つの辺に接続している辺集合
点被覆	どの辺も少なくとも 1 つの頂点に接続している頂点集合

3.2 定理

グラフ $G=(V,E)$ について,

$$|\text{最大マッチング}|+|\text{最小辺被覆}|=|V|$$
$$|\text{最大独立集合}|+|\text{最小点被覆}|=|V|$$

二部グラフであるとき,

$$|\text{最大マッチング}|=|\text{最小点被覆}|$$
$$|\text{最大独立集合}|=|\text{最小辺被覆}|$$

■Dilworth の定理・Mirsky の定理

推移性をみたす DAG であるとき,

Dilworth の定理：濃度最大の反鎖の濃度=個数最小の鎖被覆の個数

Mirsky の定理：濃度最大の鎖の濃度=個数最小の反鎖被覆の個数

■DAG の最小独立パス被覆

（最小独立パス被覆）$=|V|-$（$V+V'$ を頂点とする二部グラフの最大マッチング）

DAG が推移性をもつとき,

（最小独立パス被覆）=（最小パス被覆）

3.3 基本要素

```
struct Edge{
    int src,dst,weight;
    Edge(){}
    Edge(int s,int d,int w):src(s),dst(d),weight(w){}
};
typedef vector<vector<Edge> > Graph;
bool operator<(const Edge& a,const Edge& b)const{return a.weight<b.weight;}
bool operator>(const Edge& a,const Edge& b)const{return a.weight>b.weight;}
```

3.4 単一始点最短路（Dijkstra）

計算量：$O(E\log V)$

```
// Verify: UVA 341, AOJ 0155

void Dijkstra(const Graph& g,int v,vi& dist,vi& prev)
{
    priority_queue<Edge,vector<Edge>,greater<Edge>> pq;
    pq.emplace(-1,v,0);
    while(pq.size()){
        Edge cur=pq.top(); pq.pop();
        if(dist[cur.dst]!=INF) continue;
        dist[cur.dst]=cur.weight;
        prev[cur.dst]=cur.src;
        for(Edge e:g[cur.dst])
            pq.emplace(e.src,e.dst,cur.weight+e.weight);
    }
}

void BuildPath(const vi& prev,int v,vi& path)
{
    for(int u=v;u!=-1;u=prev[u])
        path.push_back(u);
    reverse(all(path));
}
```

3.5 単一始点最短路（Bellman-Ford）

計算量：$O(VE)$

```
// Verify: UVA 341

bool BellmanFord(const Graph& g,int begin,vi& cost,vi& prev)
{
    int size=g.size();
    cost.assign(size,INFTY);
    prev.assign(size,-1);

```

```
    cost[begin]=0;
    rep(k,size-1){
        rep(i,size){
            rep(j,g[i].size()){
                Edge cand=g[i][j]; // candidate
                if(cost[cand.dst]>cost[i]+cand.weight){
                    cost[cand.dst]=cost[i]+cand.weight;
                    prev[cand.dst]=cand.src;
                }
            }
        }
    }
    rep(i,size){
        rep(j,g[i].size()){
            Edge cand=g[i][j]; // candidate
            if(cost[cand.dst]>cost[j]+cand.weight)
                return false;
        }
    }
    return true;
}

void BuildPath(const vi& prev,int begin,int end,vi& path)
{
    path.clear();
    for(int i=end;;i=prev[i]){
        path.push_back(i);
        if(i==begin)
            break;
    }
    reverse(all(path));
}
```

3.6 全点間最短路（Warshall-Floyd）

計算量：$O(V^3)$

```
// Verify: PKU 0155

void WarshallFloyd(vvd& dist,vvi& next)
{
    int n=dist.size();

    rep(i,n) rep(j,n)
        next[i][j]=j;

    rep(k,n) rep(i,n) rep(j,n){
        if(dist[i][j]>dist[i][k]+dist[k][j]){
            dist[i][j]=dist[i][k]+dist[k][j];
            next[i][j]=next[i][k];
        }
    }
}
```

3.7 最小全域木（Prim）

root を含む最小全域木を計算する．tree に構成辺を保存し，重みの総和を返す．

計算量：$O(E \log V)$

```
// Verify: AOJ 0180, POJ 1287, POJ 1861

int Prim(const Graph& g,vector<Edge>& tree,int root=0)
{
    tree.clear();

    vi vis(g.size());
    priority_queue<Edge,vector<Edge>,greater<Edge> > pq;
    pq.push(Edge(-1,root,0));
    int res=0;
    while(pq.size()){
        Edge cur=pq.top(); pq.pop();
        if(vis[cur.dst])
            continue;
        vis[cur.dst]=true;
        res+=cur.weight;
        if(cur.src!=-1)
            tree.push_back(cur);
        foreach(e,g[cur.dst])
            pq.push(*e);
    }
    return res;
}
```

3.8 最小全域森（Kruskal）

forest に構成辺を保存し，重みの総和を返す．

計算量：$O(E \log V)$

```
// Verify: AOJ 0180, POJ 1287, POJ 1861

int Kruskal(const Graph& g,vector<Edge>& forest)
{
    forest.clear();

    vector<Edge> es;
    rep(i,g.size()) rep(j,g[i].size())
        es.push_back(g[i][j]);
    sort(all(es));

    UnionFind uf(g.size());
    int res=0;
    rep(i,es.size())
        if(uf.Find(es[i].src)!=uf.Find(es[i].dst)){
            uf.Unite(es[i].src,es[i].dst);
            res+=es[i].weight;
            forest.push_back(es[i]);
        }
    return res;
}
```

3.9 最小シュタイナー木（Dreyfus-Wagner）

与えられた隣接行列とターミナルの集合に対し，最小シュタイナー木の重みの総和を返す．

計算量：$O(3^T V + 2^T V^2 + V^3)$

```
// Verify: AOJ 1040

int DreyfusWagner(const vvi& _dist,const vi& ts)
{
    int n=_dist.size();
    vvi dist=_dist;
    rep(k,n) rep(i,n) rep(j,n)
        dist[i][j]=min(dist[i][j],dist[i][k]+dist[k][j]);

    int tsize=ts.size();
    vvi dp(1<<tsize,vi(n,INFTY));
    rep(i,tsize) rep(j,n)
```

```
        dp[1<<i][j]=dist[ts[i]][j];
    rep(i,1<<tsize){
        for(int j=i;j;j=i&(j-1)) rep(k,n)
            dp[i][k]=min(dp[i][k],dp[j][k]+dp[i^j][k]);
        rep(j,n) rep(k,n)
            dp[i][j]=min(dp[i][j],dp[i][k]+dist[k][j]);
    }
    return *min_element(all(dp.back()));
}
```

3.10 最大独立集合

計算量：$O(1.466^n n)$

```
// Verify: POJ 1419, SRM 589 Medium

void DFS(const Graph& g,int u,vi& vis,int cur,int rem,int& res)
{
    int n=g.size();
    if(cur+rem<=res) return;
    res=max(res,cur);
    if(u==n) return;

    if(vis[u]){
        DFS(g,u+1,vis,cur,rem,res);
        return;
    }

    // use u
    vi tmp=vis;
    vis[u]=1;
    rep(i,g[u].size()){
        int v=g[u][i].dst;
        if(!vis[v]) vis[v]=1,rem--;
    }
    DFS(g,u+1,vis,cur+1,rem-1,res);
    swap(vis,tmp);
    rep(i,g[u].size()){
        int v=g[u][i].dst;
        if(!vis[v]) rem++;
    }

    // don't use u
    if(g[u].size()>1){
        vis[u]=1;
        DFS(g,u+1,vis,cur,rem-1,res);
        vis[u]=0;
    }
}

int MaxIndependentSet(const Graph& g)
{
    int n=g.size();
    vi vis(n);
    int res=0;
    DFS(g,0,vis,0,n,res);
    return res;
}
```

3.11 トポロジカルソート

計算量：$O(V+E)$

```
// Verify: UVA 10305

bool Visit(const Graph& g,int v,vi& color,vi& order)
{
    if(color[v]==1)
        return false;
    if(color[v]==2)
        return true;
    color[v]=1;
    rep(i,g[v].size())
        if(!Visit(g,g[v][i].dst,color,order))
            return false;
    color[v]=2;
    order.push_back(v);
    return true;
}

bool TopologicalSort(const Graph& g,vi& order)
{
    int size=g.size();
    order.clear();
    vi colors(size);
    rep(i,size){
        if(!Visit(g,i,colors,order)){
            order.clear();
            return false;
        }
    }
    reverse(all(order));
    return true;
}
```

3.12 強連結成分分解

計算量：$O(V+E)$

```
// Verify: POJ 2186, AOJ 2505

void DFS(const Graph& g,int v,vi& vis,vi& order)
{
    if(vis[v]) return;
    vis[v]=1;
    foreach(e,g[v])
        DFS(g,e->dst,vis,order);
    order.push_back(v);
}

vvi SCC(const Graph& g)
{
    int n=g.size();
    vi vis(n),order;
    rep(i,n) DFS(g,i,vis,order);
    Graph t(n);
    rep(i,n) foreach(e,g[i])
        t[e->dst].push_back(Edge(e->dst,e->src));
    fill(all(vis),0);
    vvi res;
    per(i,n) if(!vis[order[i]]){
        res.resize(res.size()+1);
        DFS(t,order[i],vis,res.back());
    }
    return res;
}
```

3.13 橋

計算量：$O(V+E)$

```cpp
// Verify: LOJ 1026

int DFS(const Graph& g,int u,int prev,int x,vi& ord,vi& low)
{
    ord[u]=low[u]=x++;
    rep(i,g[u].size()){
        int v=g[u][i].dst;
        if(ord[v]==-1){
            x=DFS(g,v,u,x,ord,low);
            low[u]=min(low[u],low[v]);
        }
        else if(v!=prev)
            low[u]=min(low[u],ord[v]);
    }
    return x;
}

void Bridge(const Graph& g,vector<Edge>& bs)
{
    bs.clear();
    int n=g.size();
    vi ord(n,-1),low(n);
    rep(i,n) if(ord[i]==-1)
        DFS(g,i,-1,0,ord,low);
    rep(i,n) foreach(e,g[i])
        if(ord[e->src]<low[e->dst])
            bs.push_back(*e);
}
```

3.14 関節点

計算量：$O(V+E)$

```cpp
// Verify: POJ 1144

int DFS(const Graph& g,int u,int prev,int x,vi& ord,vi& low,vi& as)
{
    ord[u]=low[u]=x++;
    int deg=0,isart=0;
    rep(i,g[u].size()){
        int v=g[u][i].dst;
        if(ord[v]==-1){
            x=DFS(g,v,u,x,ord,low,as);
            low[u]=min(low[u],low[v]);
            deg++;
            isart|=ord[u]<=low[v];
        }
        else if(v!=prev)
            low[u]=min(low[u],ord[v]);
    }
    if(prev==-1 && deg>=2 || prev!=-1 && isart)
        as.push_back(u);
    return x;
}

void Articulation(const Graph& g,vi& as)
{
    as.clear();
    int n=g.size();
    vi ord(n,-1),low(n);
    rep(i,n) if(ord[i]==-1)
        DFS(g,i,-1,i,ord,low,as);
}
```

3.15 2-SAT

$u \vee v \Leftrightarrow (\overline{u} \to v) \wedge (\overline{v} \to u)$ と式変形し，二項関係 $\to$ で辺を張り強連結成分分解する．強連結成分の真偽は一致するため，x と $\overline{x}$ が同じ強連結成分に属するならば充足不能．そうでないならば充足可能．

3.16 最小共通先祖

計算量：構築 $O(V \log V)$，クエリ $O(\log V)$

```cpp
// Verify: POJ 1330

struct LCA{
    vi ds;
    vvi ps;
    LCA(vvi& g,int root){
        ds.resize(g.size());
        ps.resize(33-__builtin_clz(g.size()-1),vi(g.size(),-1));
        DFS(g,root,-1,0);
        rep(i,ps.size()-1)
            rep(j,ps[i].size())
                ps[i+1][j]=ps[i][j]==-1?-1:ps[i][ps[i][j]];
    }
    void DFS(vvi& g,int cur,int prev,int depth){
        ds[cur]=depth;
        ps[0][cur]=prev;
        rep(i,g[cur].size())
            if(g[cur][i]!=prev)
                DFS(g,g[cur][i],cur,depth+1);
    }
    int Find(int a,int b){
        if(ds[a]>ds[b])
            swap(a,b);
        rep(i,ps.size())
            if(ds[b]-ds[a]&1<<i)
                b=ps[i][b];
        if(a==b)
            return a;
        per(i,ps.size())
            if(ps[i][a]!=ps[i][b]){
                a=ps[i][a];
                b=ps[i][b];
            }
        return ps[0][a];
    }
};
```

3.17 二部性判定

計算量：$O(V+E)$

```cpp
// Verify: SRM618 Family

bool DFS(const Graph& g,int u,int c,vi& color){
    if(color[u]!=-1)
        return color[u]==c;
    color[u]=c;
    rep(i,g[u].size())
        if(!DFS(g,g[u][i].dst,!c,color))
            return false;
    return true;
}

bool IsBipartite(const Graph& g){
    int n=g.size();
    vi color(n,-1);
```

```
    rep(i,n) if(color[i]==-1)
        if(!DFS(g,i,0,color))
            return false;
    return true;
}
```

4 フロー

4.1 最大流（Dinic）

計算量：$O(V^2E)$

```
// headはvector<int>(頂点数,-1)で初期化されていること．
// 典型的な問題では，グラフの各辺について
//   AddEdge(es,head,src,dst,0,cap);
//   AddEdge(es,head,src,dst,0,0);
// と書く．
// Verify: POJ 1459(141ms), POJ 3498(2625ms), SPOJ 4110(int->ll,2.81s), AOJ 2076(0.025s)
//         UVa 10249(0.588s), COJ 1644(421ms)

struct Edge{
    int src,dst,flow,capacity,next;
    Edge(){}
    Edge(int s,int d,int f,int c,int n):src(s),dst(d),flow(f),capacity(c),next(n){}
};

void AddEdge(vector<Edge>& es,vi& head,int src,int dst,int flow,int capacity)
{
    es.push_back(Edge(src,dst,flow,capacity,head[src]));
    head[src]=es.size()-1;
}

vi BFS(const vector<Edge>& es,const vi& head,int begin)
{
    int size=head.size();
    vi label(size,size);
    queue<pii> q;
    q.push(mp(begin,0));
    while(q.size()){
        pii cur=q.front(); q.pop();
        if(label[cur.first]<=cur.second)
            continue;
        label[cur.first]=cur.second;
        for(int i=head[cur.first];i!=-1;i=es[i].next)
            if(es[i].capacity-es[i].flow>0)
                q.push(mp(es[i].dst,cur.second+1));
    }
    return label;
}

int DFS(vector<Edge>& es,vi& head,int v,int end,const vi& label,int f)
{
    if(v==end)
        return f;
    for(int& i=head[v];i!=-1;i=es[i].next){
        Edge& e=es[i];
        if(label[e.src]>=label[e.dst])
            continue;
        int residue=e.capacity-e.flow;
        if(residue<=0)
            continue;
        int augment=DFS(es,head,e.dst,end,label,min(residue,f));
        if(augment>0){
            e.flow+=augment;
            es[i^1].flow-=augment;
            return augment;
        }
    }
    return 0;
}

int Dinic(vector<Edge>& es,const vi& head,int begin,int end)
{
    int size=head.size();
    for(;;){
        vi label=BFS(es,head,begin);
        if(label[end]==size)
            break;
        vi temp=head;
        while(DFS(es,temp,begin,end,label,INFTY))
            ;
    }
    int res=0;
    for(int i=head[begin];i!=-1;i=es[i].next)
        res+=es[i].flow;
    return res;
}
```

4.2 最小費用流

計算量：$O(FE\log V)$

```
// Verify: PKU 2135(297ms), PKU 2195(157ms), UVA 10594(int→ll,344ms),
//         SRM506Div1 Medium, SRM611 GoodCompanyDivOne

struct Edge{
    int src,dst,cap,cost;
    Edge(){}
    Edge(int s,int d,int ca,int co):src(s),dst(d),cap(ca),cost(co){}
};
typedef vector<vector<Edge> > Graph;

int MinCostFlow(const Graph& _g,int begin,int end,int flow)
{
    int size=_g.size();
    Graph g(size);
    vvi rev(size);
    rep(i,size) foreach(e,_g[i]){
        g[i].push_back(*e);
        g[e->dst].push_back(Edge(e->dst,e->src,0,-e->cost));
        rev[i].push_back(g[e->dst].size()-1);
        rev[e->dst].push_back(g[i].size()-1);
    }

    int res=0;
    vi pots(size);
    while(flow){
        vi dist(size,INFTY);
        vector<pii> prev(size,mp(-1,-1));
        priority_queue<pii,vector<pii>,greater<pii> > pq;
        pq.push(mp(0,begin));
        dist[begin]=0;
        while(!pq.empty()){
            int curd=pq.top().first,curv=pq.top().second; pq.pop();
            if(dist[curv]<curd)
                continue;
            rep(i,g[curv].size()){
                Edge e=g[curv][i];
                if(e.cap==0)
```

```
                    continue;
                int newdist=dist[e.src]+e.cost+pots[e.src]-pots[e.dst];
                if(dist[e.dst]>newdist){
                    dist[e.dst]=newdist;
                    prev[e.dst]=mp(e.src,i);
                    pq.push(mp(newdist,e.dst));
                }
            }
        }

        if(prev[end].first==-1)
            return -1;
        rep(i,size)
            pots[i]+=dist[i];

        int f=flow;
        for(int v=end;v!=begin;v=prev[v].first){
            Edge e=g[prev[v].first][prev[v].second];
            f=min(f,e.cap);
        }
        for(int v=end;v!=begin;v=prev[v].first){
            int i=prev[v].first,j=prev[v].second;
            g[i][j].cap-=f;
            g[g[i][j].dst][rev[i][j]].cap+=f;
        }
        flow-=f;
        res+=f*pots[end];
    }

    return res;
}
```

4.3 2 部グラフの最大マッチング（naïve）

計算量：$O(VE)$

```
// Verify: AOJ 1163, LOJ 1149

struct Edge{
    int src,dst;
    Edge(){}
    Edge(int s,int d):src(s),dst(d){}
};
typedef vector<vector<Edge> > Graph;

bool Augment(const Graph& g,int u,vi& vis,vi& match)
{
    rep(i,g[u].size()){
        int v=g[u][i].dst;
        if(vis[v]) continue;
        vis[v]=1;
        if(match[v]==-1 || Augment(g,match[v],vis,match)){
            match[u]=v;
            match[v]=u;
            return true;
        }
    }
    return false;
}

int BipartiteMatching(const Graph& g,vi& match)
{
    int n=g.size();
    match.assign(n,-1);
    int res=0;
    rep(i,n) if(match[i]==-1){
        vi vis(n);
        res+=Augment(g,i,vis,match);
    }
    return res;
}
```

4.4 2 部グラフの最大マッチング（Hopcroft-Karp）

計算量：$O(\sqrt{V}E)$

```
// Verify: AOJ 1163, LOJ 1171, LOJ 1356, POJ 1466, POJ 2446, SPOJ 4206 (2.31s)

struct Edge{
    int src,dst;
    Edge(){}
    Edge(int s,int d):src(s),dst(d){}
};
typedef vector<vector<Edge> > Graph;

bool BFS(const Graph& g,const vi& side,const vi& match,vi& level)
{
    int n=g.size();
    level.assign(n,n);
    queue<pii> q;
    rep(i,n) if(side[i]==0 && match[i]==-1)
        q.push(mp(i,0));
    bool res=false;
    while(!q.empty()){
        pii cur=q.front(); q.pop();
        int u=cur.first,l=cur.second;
        if(level[u]<=l) continue;
        level[u]=l;
        rep(i,g[u].size()){
            int v=g[u][i].dst;
            if(match[v]==-1)
                res=true;
            else
                q.push(mp(match[v],l+2));
        }
    }
    return res;
}

bool DFS(const Graph& g,const vi& side,int u,vi& match,vi& level)
{
    rep(i,g[u].size()){
        int v=g[u][i].dst;
        if(match[v]==-1 || level[match[v]]>level[u] && DFS(g,side,match[v],match,level)){
            match[u]=v;
            match[v]=u;
            return true;
        }
    }
    level[u]=-1;
    return false;
}

int HopcroftKarp(const Graph& g,const vi& side,vi& match)
{
    int n=g.size();
    match.assign(n,-1);
    int res=0;
    for(vi level;BFS(g,side,match,level);)
        rep(i,n) if(side[i]==0 && match[i]==-1)
```

```
            res+=DFS(g,side,i,match,level);
    return res;
}
```

4.5 DAG の最小独立パス被覆

```
// Verify: AOJ 2251

int MinPathCover(const Graph& g)
{
    int n=g.size();
    Graph bg(2*n);
    rep(i,n) foreach(e,g[i])
        bg[e->src].push_back(Edge(e->src,n+e->dst));
    vi side(2*n),match(2*n);
    fill(n+all(side),1);
    return n-HopcroftKarp(bg,side,match);
}
```

4.6 無向グラフの全域最小カット（Nagamochi-Ibaraki）

グラフが連結でないとき 0 を返す．計算量：$O(V^3)$

```
// Verify: POJ 2914(5657ms)

int NagamochiIbaraki(const Graph& g)
{
    int size=g.size();
    vvi a(size,vi(size));
    rep(i,size) foreach(e,g[i])
        a[e->src][e->dst]+=e->weight;
    vi toi(size);
    rep(i,size) toi[i]=i;

    int res=INFTY;
    for(int n=size;n>=2;n--){
        vi ws(n);
        int u,v,w;
        rep(i,n){
            u=v; v=max_element(all(ws))-ws.begin();
            w=ws[v]; ws[v]=-1;
            rep(j,n) if(ws[j]>=0)
                ws[j]+=a[toi[v]][toi[j]];
        }
        res=min(res,w);
        rep(i,n){
            a[toi[u]][toi[i]]+=a[toi[v]][toi[i]];
            a[toi[i]][toi[u]]+=a[toi[i]][toi[v]];
        }
        toi.erase(toi.begin()+v);
    }
    return res;
}
```

5 文字列

5.1 Knuth-Morris-Pratt

最短周期は `n-fail[n]` で求まる．

計算量：テキスト長 N，パターン長 M として，初期化 $O(M)$，検索 $O(M+N)$

```
// Verify: [ctor] POJ 1961, POJ 2406, [MatchAll] POJ 3461, LOJ 1255, SPOJ 32

struct KMP{
    string pat;
    vi fail;
    KMP(const string& p):pat(p),fail(p.size()+1){
        fail[0]=-1;
        for(int i=0,j=-1;i<pat.size();){
            while(j>=0 && pat[i]!=pat[j])
                j=fail[j];
            i++,j++;
            fail[i]=j;
        }
    }
    int Match(const string& text){
        for(int i=0,j=0;i<text.size();){
            while(j>=0 && text[i]!=pat[j])
                j=fail[j];
            i++,j++;
            if(j==pat.size())
                return i-j;
        }
        return -1;
    }
    vi MatchAll(const string& text){
        vi res;
        for(int i=0,j=0;i<text.size();){
            while(j>=0 && text[i]!=pat[j])
                j=fail[j];
            i++,j++;
            if(j==pat.size()){
                res.push_back(i-j);
                j=fail[j];
            }
        }
        return res;
    }
};
```

5.2 最長回文（Manacher）

計算量：$O(N)$

```
// Verify: PKU 3974

int LongestPalindrome(const string& _s)
{
    int n=_s.size();
    string s(2*n+1,'.');
    rep(i,n)
        s[i*2+1]=_s[i];
    n=s.size();

    vi rad(n);
    for(int i=0,j=0;i<n;){
        for(;0<=i-j && i+j<n && s[i-j]==s[i+j];j++)
            ;
        rad[i]=j;

        int k=1;
        for(;0<=i-k && i+k<n && rad[i-k]<rad[i]-k;k++)
            rad[i+k]=rad[i-k];

        j=max(rad[i]-k,0);
        i+=k;
    }
```

```
    return *max_element(all(rad))-1;
}
```

5.3 Suffix Array（Kärkkäinen-Sanders）

計算量：構築 $O(N)$，LCP の計算 $O(N)$.

```
// Verify: [構築] LiveArchive 3451, POJ 1509, POJ 2774, POJ 3080, POJ 3261, POJ 3450,
//                SPOJ 6409 (0.76s), UVA 1223, UVA 1227, UVA 11512
//         [LCP]  LiveArchive 3451, POJ 2774, POJ 3080, POJ 3261, POJ 3450,
//                UVA 1223, UVA 1227, UVA 11512
//         [LCS]  POJ 3080, UVA 1227

template<typename C>
void RadixSort(const vi& src,vi& dst,const C& s,int ofs,int n,int asize)
{
    vi hist(asize+1);
    rep(i,n) hist[s[ofs+src[i]]]++;
    rep(i,asize) hist[i+1]+=hist[i];
    per(i,n) dst[--hist[s[ofs+src[i]]]]=src[i];
}

bool Less(int a1,int a2,int b1,int b2)
{
    return a1!=b1?a1<b1:a2<b2;
}
bool Less(int a1,int a2,int a3,int b1,int b2,int b3)
{
    return a1!=b1?a1<b1:Less(a2,a3,b2,b3);
}

// s[0..n-1]: 入力文字列．末尾に'\0'が3つ必要(s[n]=s[n+1]=s[n+2]=0)．
// sa[0..n-1]: 接尾辞配列．
// asize: アルファベットサイズ，s[i] \in [1,asize]
template<typename C>
void KaerkkaeinenSanders(const C& s,vi& sa,int asize)
{
    int n=sa.size();
    int n0=(n+2)/3,n1=(n+1)/3,n2=n/3,n02=n0+n2;
    vi s12(n02+3),sa12(n02);

    for(int i=0,j=0;i<n+(n0-n1);i++)
        if(i%3) s12[j++]=i;
    RadixSort(s12,sa12,s,2,n02,asize);
    RadixSort(sa12,s12,s,1,n02,asize);
    RadixSort(s12,sa12,s,0,n02,asize);

    int name=0,x=-1,y=-1,z=-1;
    rep(i,n02){
        int j=sa12[i];
        if(s[j]!=x || s[j+1]!=y || s[j+2]!=z)
            x=s[j],y=s[j+1],z=s[j+2],name++;
        if(j%3==1) s12[j/3]=name;
        else       s12[n0+j/3]=name;
    }

    if(name==n02) // unique
        rep(i,n02) sa12[s12[i]-1]=i;
    else{
        KaerkkaeinenSanders(s12,sa12,name);
        rep(i,n02) s12[sa12[i]]=i+1;
    }

    vi s0(n0),sa0(n0);
    for(int i=0,j=0;i<n02;i++)
        if(sa12[i]<n0) s0[j++]=3*sa12[i];
    RadixSort(s0,sa0,s,0,n0,asize);

    int i=0,j=n0-n1,k=0;
    while(i<n0 && j<n02){
        int p=sa0[i],q=sa12[j]<n0?sa12[j]*3+1:(sa12[j]-n0)*3+2;
        if(sa12[j]<n0?
            Less(s[p],s12[p/3],s[q],s12[n0+sa12[j]]):
            Less(s[p],s[p+1],s12[n0+p/3],s[q],s[q+1],s12[sa12[j]+1-n0]))
            sa[k++]=p,i++;
        else
            sa[k++]=q,j++;
    }
    for(;i<n0;k++,i++)
        sa[k]=sa0[i];
    for(;j<n02;k++,j++)
        sa[k]=sa12[j]<n0?sa12[j]*3+1:(sa12[j]-n0)*3+2;
}
void KaerkkaeinenSanders(const string& s,vi& sa)
{
    KaerkkaeinenSanders<string>(s+string(3,0),sa,127);
}
void KaerkkaeinenSanders(const char* s,vi& sa)
{
    KaerkkaeinenSanders<string>(s+string(3,0),sa,127);
}

// s[0..n-1] (s[n]=0), sa[0..n-1]
// lcp[0..n-1] (lcp[i]:s[sa[i-1]..]とs[sa[i]..]のLCP.lcp[0]=0)
template<typename C>
void LCP(const C& s,const vi& sa,vi& lcp)
{
    int n=sa.size();
    vi rank(n);
    rep(i,n) rank[sa[i]]=i;
    for(int i=0,h=0;i<n;i++){
        if(h>0) h--;
        if(rank[i]>0)
            for(int j=sa[rank[i]-1];;h++)
                if(s[j+h]!=s[i+h]) break;
        lcp[rank[i]]=h;
    }
}

// n: 文字列の数
// ls[0..n-1]: 各文字列の長さ
// LCSはs.substr(sa[res],lcp[res])で得られる．
// LCSが空の時res=0．lcp[res]=0でもある．
int LCS(int n,const vi& ls,const vi& sa,const vi& lcp)
{
    int m=sa.size();
    vi is(m);
    for(int i=0,j=0;i<n;i++){
        rep(k,ls[i]) is[j+k]=i;
        j+=ls[i];
        if(i<n-1) is[j++]=i;
    }
    int cnt=0,res=0;
    vi occ(n);
    deque<int> q;
    for(int i=n-1,j=n-1;i<m;i++){
        for(;j<m && cnt<n;j++){
            if(++occ[is[sa[j]]]==1) cnt++;
            while(!q.empty() && lcp[q.back()]>lcp[j]) q.pop_back();
```

```
            q.push_back(j);
        }
        if(cnt<n) break;
        if(q.front()==i) q.pop_front();
        assert(q.size());
        if(lcp[res]<lcp[q.front()])
            res=q.front();
        if(--occ[is[sa[i]]]==0) cnt--;
    }
    return res;
}
```

5.4 Suffix Array（SA-IS）

計算量：構築 $O(N)$，LCP の計算 $O(N)$．ただし $|\Sigma| = O(1)$ とする．

```
// Verify: UVA 1227(5.436s), SPOJ 6409(0.51s), POJ 1743, POJ 3261

enum{LTYPE,STYPE};

inline bool IsLMS(const vector<bool>& ts,int i)
{
    return i>0 && ts[i]==STYPE && ts[i-1]==LTYPE;
}

template<typename T>
void InitBucket(const T* s,int n,int* bucket,int bsize,bool inclusive)
{
    fill(bucket,bucket+bsize,0);
    rep(i,n)
        bucket[s[i]]++;
    for(int i=0,sum=0;i<bsize;i++){
        sum+=bucket[i];
        bucket[i]=inclusive?sum:sum-bucket[i];
    }
}

template<typename T>
void InduceSAL(const T* s,int* sa,int n,int* bucket,int bsize)
{
    InitBucket(s,n,bucket,bsize,false);
    rep(i,n){
        int j=sa[i]-1;
        if(j>=0 && s[j]>=s[j+1])
            sa[bucket[s[j]]++]=j;
    }
}

template<typename T>
void InduceSAS(const T* s,int* sa,int n,int* bucket,int bsize)
{
    InitBucket(s,n,bucket,bsize,true);
    per(i,n){
        int j=sa[i]-1;
        if(j>=0 && s[j]<=s[j+1])
            sa[--bucket[s[j]]]=j;
    }
}

// s[0..n-1]: 入力文字列，n>=2かつs[n-1]=0であること
// sa[0..n-1]: 接尾辞配列．
// asize: アルファベットサイズ，s[n-1]を除き1以上asize以下
template<typename T>
void SAIS(const T* s,int* sa,int n,int asize)
{
    vector<bool> ts(n,STYPE);
    per(i,n-1)
        if(s[i]>s[i+1] || (s[i]==s[i+1] && ts[i+1]==LTYPE))
            ts[i]=LTYPE;

    fill(sa,sa+n,-1);

    int* bucket=new int[asize+1];
    InitBucket(s,n,bucket,asize+1,true);
    per(i,n) if(IsLMS(ts,i))
        sa[--bucket[s[i]]]=i;
    InduceSAL(s,sa,n,bucket,asize+1);
    InduceSAS(s,sa,n,bucket,asize+1);
    delete[] bucket;

    int n1=0;
    rep(i,n) if(IsLMS(ts,sa[i]))
        sa[n1++]=sa[i];
    fill(sa+n1,sa+n,-1);

    int name=0,prev=-1;
    rep(i,n1){
        int cur=sa[i];
        bool diff=prev==-1;
        for(int j=0;!diff;j++){
            if(j>0 && IsLMS(ts,cur+j))
                break;
            if(s[cur+j]!=s[prev+j])
                diff=true;
        }
        if(diff)
            name++;
        sa[n1+cur/2]=name-1;
        prev=cur;
    }
    for(int i=n1,j=n1;i<n;i++)
        if(sa[i]>=0)
            sa[j++]=sa[i];
    fill(sa+n1+n1,sa+n,-1);

    int *sa1=sa,*s1=sa+n1;
    if(name==n1)
        rep(i,n1) sa1[s1[i]]=i;
    else
        SAIS(s1,sa1,n1,name);

    n1=0;
    rep(i,n) if(IsLMS(ts,i))
        s1[n1++]=i;
    rep(i,n1)
        sa1[i]=s1[sa1[i]];
    fill(sa+n1,sa+n,-1);

    bucket=new int[asize+1];
    InitBucket(s,n,bucket,asize+1,true);
    per(i,n1){
        int j=sa1[i]; sa1[i]=-1;
        sa[--bucket[s[j]]]=j;
    }
    InduceSAL(s,sa,n,bucket,asize+1);
    InduceSAS(s,sa,n,bucket,asize+1);
    delete[] bucket;
}

// nはsとsaのサイズ，lcpのサイズはn-1
```

```
template<typename T>
void LCP(const T* s,const int* sa,int n,int* lcp)
{
    vi rank(n);
    rep(i,n) rank[sa[i]]=i;
    for(int i=0,h=0;i<n-1;i++){
        if(h>0) h--;
        for(int j=sa[rank[i]-1];;h++)
            if(s[j+h]!=s[i+h])
                break;
        lcp[rank[i]-1]=h;
    }
}
```

6 動的計画法

6.1 最長共通部分列

計算量：$O(MN)$

```
// Verify: PKU 1458

int LCS(const string& s1,const string& s2)
{
    vvi dp(2,vi(s2.size()+1));
    rep(i,s1.size()){
        rep(j,s2.size()){
            int ii=i+1,jj=j+1;
            if(s1[i]==s2[j])
                dp[ii&1][jj]=dp[(ii-1)&1][jj-1]+1;
            else
                dp[ii&1][jj]=max(dp[(ii-1)&1][jj-1],max(dp[(ii-1)&1][jj],dp[ii&1][jj-1]));
        }
    }
    return dp[s1.size()&1][s2.size()];
}
```

6.2 最長増加部分列

計算量：$O(N \log N)$.

```
// Verify: PKU 3903

int LIS(const vi& a)
{
    vi dp;
    rep(i,a.size()){
        int j=lower_bound(all(dp),a[i])-dp.begin();
        if(j==dp.size())
            dp.push_back(a[i]);
        else
            dp[j]=a[i];
    }
    return dp.size();
}
```

7 データ構造

7.1 Union-Find Tree

計算量：各操作に関して $O(\alpha(N))$. ただし $\alpha(N)$ はアッカーマン関数の逆関数.

```
// Verify: PKU 2524, PKU 1182

struct UnionFind{
    vi data;
    UnionFind(int size):data(size,-1){}
    int Find(int n){
        return data[n]<0?n:(data[n]=Find(data[n]));
    }
    void Unite(int a,int b){
        int ra=Find(a),rb=Find(b);
        if(ra!=rb){
            if(-data[ra]<-data[rb])
                swap(ra,rb);
            data[ra]+=data[rb];
            data[rb]=ra;
        }
    }
    int Size(int n){
        return -data[Find(n)];
    }
};
```

7.2 Fenwick Tree

計算量：各操作に関して $O(\log N)$

```
// 点更新/区間質問
// Verify: POJ 3067, POJ 3928

struct FenwickTree{
    vector<ll> data;
    FenwickTree(int n):data(n+1){}
    void Add(int i,int x){
        for(i++;i<data.size();i+=i&-i)
            data[i]+=x;
    }
    ll Sum(int i){
        ll res=0;
        for(;i;i-=i&-i)
            res+=data[i];
        return res;
    }
    ll Sum(int i,int j){
        return Sum(j)-Sum(i);
    }
};

// 区間更新/点質問
// Verify: AOJ 2412, LOJ 1080, POJ 2182

struct FenwickTree{
    vector<ll> data;
    FenwickTree(int n):data(n+1){}
    void AddRange(int i,int x){
        for(;i;i-=i&-i)
            data[i]+=x;
    }
    void AddRange(int i,int j,int x){
        AddRange(i,-x);
        AddRange(j,x);
    }
    int Get(int i){
        ll res=0;
        for(i++;i<data.size();i+=i&-i)
```

```
            res+=data[i];
        return res;
    }
};

// 2次元版，点更新/区間質問
// Verify: POJ 1195

struct FenwickTree2D{
    vector<FenwickTree> data;
    FenwickTree2D(int h,int w):data(h+1,FenwickTree(w)){}
    void Add(int i,int j,int x){
        for(i++;i<data.size();i+=i&-i)
            data[i].Add(j,x);
    }
    ll Sum(int i,int j){
        ll res=0;
        for(;i;i-=i&-i)
            res+=data[i].Sum(j);
        return res;
    }
    ll Sum(int i,int j,int k,int l){
        return Sum(k,l)-Sum(k,j)-Sum(i,l)+Sum(i,j);
    }
};
```

7.3 Segment Tree

計算量：初期化 $O(N)$，更新 $O(\log N)$，クエリ $O(\log N)$

```
// Verify: UVa 12299, POJ 3264, AOJ 1068

int Need(int x)
{
    x--;
    rep(i,5) x|=x>>(1<<i);
    return x+1;
}

struct SegmentTree{
    int size;
    vi data;
    SegmentTree(int s):size(Need(s)),data(size*2,INFTY){}
    SegmentTree(const vi& a):size(Need(a.size())),data(size*2,INFTY){
        copy(all(a),data.begin()+size);
        for(int i=size;--i;)
            data[i]=min(data[i*2],data[i*2+1]);
    }
    int Get(int i){
        return data[size+i];
    }
    void Update(int i,int x){
        data[size+i]=x;
        for(i=(i+size)/2;i;i/=2)
            data[i]=min(data[i*2],data[i*2+1]);
    }
    int Query(int a,int b,int i,int l,int r){
        if(b<=l || r<=a) return INFTY;
        if(a<=l && r<=b) return data[i];
        return min(Query(a,b,i*2,l,(l+r)/2),Query(a,b,i*2+1,(l+r)/2,r));
    }
    int Query(int a,int b){
        return Query(a,b,1,0,size);
    }
};

// 2次元版
// Verify: AOJ 1068

struct SegmentTree2D{
    int size;
    vector<SegmentTree> data;
    SegmentTree2D(int h,int w):size(Need(h)),data(size*2,SegmentTree(w)){}
    SegmentTree2D(const vvi& a):size(Need(a.size())),data(size*2,SegmentTree(a[0].size())){
        copy(all(a),data.begin()+size);
        for(int i=size;--i;)
            repi(j,1,data[i].data.size())
                data[i].data[j]=min(data[i*2].data[j],data[i*2+1].data[j]);
    }
    int Get(int i,int j){
        return data[size+i].Get(j);
    }
    void Update(int i,int j,int x){
        data[size+i].Update(j,x);
        for(i=(i+size)/2;i;i/=2)
            data[i].Update(j,min(data[i*2].Get(j),data[i*2+1].Get(j)));
    }
    int Query(int a,int b,int c,int d,int i,int to,int bo){
        if(c<=to || bo<=a) return INFTY;
        if(a<=to && bo<=c) return data[i].Query(b,d);
        return min(Query(a,b,c,d,i*2,to,(to+bo)/2),Query(a,b,c,d,i*2+1,(to+bo)/2,bo));
    }
    int Query(int a,int b,int c,int d){
        return Query(a,b,c,d,1,0,size);
    }
};

// 区間更新/区間質問．PropagateとMergeを適切に書き換える
// Verify: SPOJ 7259

struct SegmentTree{
    int size;
    vi data,prop;
    SegmentTree(int s):size(Need(s)),data(size*2),prop(size*2){}
    SegmentTree(const vi& a):size(Need(a.size())),data(size*2),prop(size*2){
        copy(all(a),data.begin()+size);
        for(int i=size;--i;)
            data[i]=Merge(data[i*2],data[i*2+1]);
    }
    void Update(int a,int b,int x,int i,int l,int r){
        Propagate(i,l,r);
        if(b<=l || r<=a)
            return;
        if(a<=l && r<=b){
            prop[i]=x;
            Propagate(i,l,r);
            return;
        }
        Update(a,b,x,i*2+0,l,(l+r)/2);
        Update(a,b,x,i*2+1,(l+r)/2,r);
        data[i]=Merge(data[i*2],data[i*2+1]);
    }
    ll Query(int a,int b,int i,int l,int r){
        Propagate(i,l,r);
        if(b<=l || r<=a) return 0;
        if(a<=l && r<=b) return data[i];
        ll x=Query(a,b,i*2+0,l,(l+r)/2);
        ll y=Query(a,b,i*2+1,(l+r)/2,r);
        return Merge(x,y);
    }
```

```
    void Update(int a,int b,int x){
        Update(a,b,x,1,0,size);
    }
    int Query(int a,int b){
        return Query(a,b,1,0,size);
    }
    void Propagate(int i,int l,int r){
        if(i<size){
            prop[i*2+0]^=prop[i];
            prop[i*2+1]^=prop[i];
        }
        if(prop[i])
            data[i]=r-l-data[i];
        prop[i]=0;
    }
    int Merge(ll x,ll y){
        return x+y;
    }
};

// 区間Set,Reset,Flip,Count
// Verify: UVa 11402, Codeforces 242E

struct SegmentTree{
    int size;
    vi data,prop;
    SegmentTree(int s):size(Need(s)),data(size*2),prop(size*2){}
    SegmentTree(const vi& a):size(Need(a.size())),data(size*2),prop(size*2){
        copy(all(a),data.begin()+size);
        for(int i=size;--i;)
            data[i]=Merge(data[i*2],data[i*2+1]);
    }
    void Update(int a,int b,int x,int i,int l,int r){
        Propagate(i,l,r);
        if(b<=l || r<=a)
            return;
        if(a<=l && r<=b){
            prop[i]=x;
            Propagate(i,l,r);
            return;
        }
        Update(a,b,x,i*2+0,l,(l+r)/2);
        Update(a,b,x,i*2+1,(l+r)/2,r);
        data[i]=Merge(data[i*2],data[i*2+1]);
    }
    ll Query(int a,int b,int i,int l,int r){
        Propagate(i,l,r);
        if(b<=l || r<=a) return 0;
        if(a<=l && r<=b) return data[i];
        ll x=Query(a,b,i*2+0,l,(l+r)/2);
        ll y=Query(a,b,i*2+1,(l+r)/2,r);
        return Merge(x,y);
    }
    void Propagate(int i,int l,int r){
        if(i<size){
            if(prop[i]==1) prop[i*2+0]=prop[i*2+1]=1;
            if(prop[i]==2) prop[i*2+0]=prop[i*2+1]=2;
            if(prop[i]==3){
                prop[i*2+0]=3-prop[i*2+0];
                prop[i*2+1]=3-prop[i*2+1];
            }
        }
        if(prop[i]==1) data[i]=r-l;
        if(prop[i]==2) data[i]=0;
        if(prop[i]==3) data[i]=r-l-data[i];
        prop[i]=0;
    }
    int Merge(ll x,ll y){
        return x+y;
    }
    void Set(int a,int b){
        Update(a,b,1,1,0,size);
    }
    void Reset(int a,int b){
        Update(a,b,2,1,0,size);
    }
    void Flip(int a,int b){
        Update(a,b,3,1,0,size);
    }
    int Count(int a,int b){
        return Query(a,b,1,0,size);
    }
};
```

7.4 Range Tree

計算量：初期化 ($O(N\log^2 N)$)，クエリ $O(N\log^2 N)$

```
// Verify: LiveArchive 5821, ACPC2012 F

struct Point{
    int x,y;
    Point(){}
    Point(int x,int y):x(x),y(y){}
};
bool LessX(Point a,Point b){
    return a.x<b.x;
}
bool LessY(Point a,Point b){
    return a.y<b.y;
}

struct RangeTree2D{
    vector<Point> yps;
    struct RangeNode2D{
        vector<Point> xps;
        RangeNode2D *left,*right;
        RangeNode2D(const vector<Point>& ps):xps(ps),left(0),right(0){
            sort(all(xps),LessX);
        }
        ~RangeNode2D(){
            delete left;
            delete right;
        }
    }*root;
    RangeTree2D(const vector<Point>& ps):yps(ps){
        sort(all(yps),LessY);
        root=Build(0,yps.size());
    }
    ~RangeTree2D(){
        delete root;
    }
    RangeNode2D* Build(int first,int last){
        RangeNode2D* node=new RangeNode2D(vector<Point>(yps.begin()+first,yps.begin()+last));
        if(last-first>=2){
            int middle=(first+last)/2;
            node->left=Build(first,middle);
            node->right=Build(middle,last);
        }
        return node;
```

```
    }
    int Query(int to,int le,int bo,int ri,RangeNode2D* root,int first,int last){
        if(first==last || yps[last-1].y<to || bo<=yps[first].y)
            return 0;
        if(to<=yps[first].y && yps[last-1].y<bo){
            iter(root->xps) i=lower_bound(all(root->xps),Point(le,0),LessX);
            iter(root->xps) j=lower_bound(all(root->xps),Point(ri,0),LessX);
            return j-i;
        }
        int middle=(first+last)/2,res=0;
        res+=Query(to,le,bo,ri,root->left,first,middle);
        res+=Query(to,le,bo,ri,root->right,middle,last);
        return res;
    }
    int Query(int to,int le,int bo,int ri){
        return Query(to,le,bo,ri,root,0,yps.size());
    }
};
```

8 数学

8.1 数学定数

■円周率 $\pi = 3.14159265358979323846264338327950288$

■自然対数の底 $e = 2.71828182845904523536028747135266257$

■オイラーの定数 $\gamma = 0.57721566490153286060651209008240243$

8.2 ユークリッドの互除法

ExtendedGCD は $ax + by = \gcd(a, b)$ をみたす x, y を求める．計算量：$O(\log \max(a, b))$

```
int GCD(int a,int b)
{
    return b==0?a:GCD(b,a%b);
}

int LCM(int a,int b)
{
    return a/GCD(a,b)*b;
}

int ExtendedGCD(int a,int b,int& x,int& y)
{
    if(b==0){
        x=1; y=0;
        return a;
    }
    else{
        int g=ExtendedGCD(b,a%b,y,x);
        y-=a/b*x;
        return g;
    }
}
```

8.3 逆元

$ax \equiv 1 \pmod m$ をみたす x を求める．逆元が存在しない場合は 0 を返す．

```
int ModInverse(int a,int m)
{
    int x,y;
    int g=ExtendedGCD(a,m,x,y);
    if(g==1)
        return (x+m)%m;
    else
        return 0; // invalid
}
```

8.4 篩

エラトステネスの篩と，それを用いた区間篩．

■計算量 エラトステネスの篩：$O(N \log \log N)$，区間篩：$O(\sqrt{b} \log \log b + (b - a) \log \log b)$

```
// Verify: LOJ 1197

// [0,n)
vi Sieve(int n)
{
    vi isp(n);
    repi(i,2,n) isp[i]=i;
    for(int i=2;i*i<n;i++) if(isp[i])
        for(int j=i*i;j<n;j+=i)
            isp[j]=0;
    return isp;
}

// [a,b)
vl SegmentedSieve(ll a,ll b)
{
    const vi isp=Sieve(sqrt(b)+1);
    vl isp2(b-a);
    rep(i,b-a) if(a+i>=2) isp2[i]=a+i;
    rep(i,isp.size()) if(isp[i])
        for(ll j=max((a+i-1)/i,2ll)*i;j<b;j+=i)
            isp2[j-a]=0;
    return isp2;
}
```

8.5 素数判定（Miller-Rabin）

問題によって実装を切り替える．

```
// ModMulとwsは問題に応じて使い分けること．
// witnessは以下のページを参考にした:
//   http://en.wikipedia.org/wiki/Miller%E2%80%93Rabin_primality_test
//   http://mathworld.wolfram.com/StrongPseudoprime.html
//   http://oeis.org/A014233
// Verify: SPOJ 288

// a*b mod m
ull ModMul(ull a,ull b,ull m)
{
    //return a*b%m;
    ull c=0;
    for(;b;b>>=1,(a<<=1)%=m)
        if(b&1)
            (c+=a)%=m;
    return c;
}
// a^r mod m
ull ModPow(ull a,ull r,ull m)
{
    if(r==0) return 1;
    ull b=ModPow(a,r/2,m);
    b=ModMul(b,b,m);
    return r&1?ModMul(b,a,m):b;
}
```

```
bool MillerTest(ull n,int a,ull d,int s)
{
    ull x=ModPow(a,d,n);
    if(x==1) return true;
    for(int i=0;i<s;i++){
        if(x==n-1) return true;
        x=ModMul(x,x,n);
    }
    return false;
}
bool MillerRabin(ull n)
{
    if(n<=1)   return false;
    if(n%2==0) return n==2;
    ull d=n-1; int s=0;
    while(d%2==0)
        d/=2,s++;
    //int ws[]={2,7,61,-1}; // n<4759123141
    int ws[]={2,3,5,7,11,13,17,19,23,-1}; // n<3825123056546413051 (conjectured)
    for(int i=0;~ws[i] && ws[i]<n;i++)
        if(!MillerTest(n,ws[i],d,s))
            return false;
    return true;
}
```

8.6 Gauss-Jordan の消去法

解が一意でないとき false を返す．計算量：$O(N^3)$

```
// Verify: AOJ 1328, ZOJ 3645

bool GaussJordan(const vvd& _a,const vd& b,vd& x)
{
    int n=_a.size();
    vvd a(n,vd(n+1));
    rep(i,n){
        copy(all(_a[i]),begin(a[i]));
        a[i][n]=b[i];
    }

    rep(i,n){
        int p=i;
        repi(j,i+1,n) if(abs(a[p][i])<abs(a[j][i])) p=j;
        if(abs(a[p][i])<EPS) return false;
        swap(a[i],a[p]);
        peri(j,i,n+1) a[i][j]/=a[i][i];
        rep(j,n) if(j!=i) peri(k,i,n+1) a[j][k]-=a[j][i]*a[i][k];
    }

    rep(i,n) x[i]=a[i][n];
    return true;
}
```

9 幾何

9.1 基本要素

```
const double PI=acos(-1);

int Signum(double x){
    return x<-EPS?-1:x>EPS?1:0;
}

struct Point{
    double x,y;
    Point(){}
    Point(double x,double y):x(x),y(y){}
    Point& operator+=(Point p){
        x+=p.x,y+=p.y;
        return *this;
    }
    Point& operator-=(Point p){
        x-=p.x,y-=p.y;
        return *this;
    }
    Point& operator*=(double c){
        x*=c,y*=c;
        return *this;
    }
    Point& operator/=(double c){
        x/=c,y/=c;
        return *this;
    }
};
Point operator+(Point a,Point b){
    return a+=b;
}
Point operator-(Point a,Point b){
    return a-=b;
}
Point operator*(Point a,double c){
    return a*=c;
}
Point operator*(double c,Point a){
    return a*=c;
}
Point operator/(Point a,double c){
    return a/=c;
}
bool operator==(Point a,Point b){
    return abs(a.x-b.x)<EPS && abs(a.y-b.y)<EPS;
}
bool operator!=(Point a,Point b){
    return !(a==b);
}

double Abs(Point p){
    return sqrt(p.x*p.x+p.y*p.y);
}
double Abs2(Point p){
    return p.x*p.x+p.y*p.y;
}
double Arg(Point p){
    return atan2(p.y,p.x);
}
double Dot(Point a,Point b){
    return a.x*b.x+a.y*b.y;
}
double Cross(Point a,Point b){
    return a.x*b.y-a.y*b.x;
}
Point Rot(Point p,double t){
    return Point(cos(t)*p.x-sin(t)*p.y,sin(t)*p.x+cos(t)*p.y);
}

struct Line{
    Point pos,dir;
    Line(){}
```

```
    Line(Point p,Point d):pos(p),dir(d){}
    Line(double px,double py,double dx,double dy):pos(px,py),dir(dx,dy){}
};

Point Proj(Line l,Point p){
    Point a=p-l.pos,b=l.dir;
    return l.pos+Dot(a,b)/Abs2(b)*b;
}

struct Segment{
    Point pos,dir;
    Segment(){}
    Segment(Point p,Point d):pos(p),dir(d){}
    Segment(double px,double py,double dx,double dy):pos(px,py),dir(dx,dy){}
    explicit Segment(Line l):pos(l.pos),dir(l.dir){}
    explicit operator Line()const{return Line(pos,dir);}
};

struct Circle{
    Point center;
    double radius;
    Circle(){}
    Circle(Point c,double r):center(c),radius(r){}
    Circle(double x,double y,double r):center(x,y),radius(r){}
};
bool operator==(Circle a,Circle b){
    return a.center==b.center && abs(a.radius-b.radius)<EPS;
}
bool operator!=(Circle a,Circle b){
    return !(a==b);
}

// stream
ostream& operator<<(ostream& os,const Point& p){
    return os<<'('<<p.x<<','<<p.y<<')';
}
ostream& operator<<(ostream& os,const Line& l){
    return os<<'('<<l.pos<<','<<l.dir<<')';
}
ostream& operator<<(ostream& os,const Circle& c){
    return os<<'('<<c.center.x<<','<<c.center.y<<','<<c.radius<<')';
}

// comparer
struct LessX{
    bool operator()(Point a,Point b){
        return abs(a.x-b.x)>EPS?a.x<b.x-EPS:a.y<b.y-EPS;
    }
};
```

9.2 回転方向

```
int CCW(Point a,Point b,Point c){
    b-=a,c-=a;
    if(int sign=Signum(Cross(b,c)))
        return sign; // 1:ccw,-1:cw
    if(Dot(b,c)<-EPS)
        return -2;   // c-a-b
    if(Abs2(b)<Abs2(c)-EPS)
        return 2;    // a-b-c
    return 0;        // a-c-b (inclusive)
}
```

9.3 面積

```
double Area(const vector<Point>& ps){
    double res=0;
    repi(i,2,ps.size())
        res+=Cross(ps[i-1]-ps[0],ps[i]-ps[0])/2;
    return res;
}
```

9.4 交差判定

```
bool IntersectLL(Line a,Line b){
    return abs(Cross(a.dir,b.dir))>EPS || abs(Cross(a.dir,b.pos-a.pos))<EPS;
}
bool IntersectLS(Line l,Segment s){
    Point a=s.pos-l.pos,b=s.pos+s.dir-l.pos;
    return Signum(Cross(l.dir,a))*Signum(Cross(l.dir,b))<=0;
}
bool IntersectSS(Segment a,Segment b){
    int c1=CCW(a.pos,a.pos+a.dir,b.pos),c2=CCW(a.pos,a.pos+a.dir,b.pos+b.dir);
    int c3=CCW(b.pos,b.pos+b.dir,a.pos),c4=CCW(b.pos,b.pos+b.dir,a.pos+a.dir);
    return c1*c2<=0 && c3*c4<=0;
}
bool IntersectSP(Segment s,Point p){
    return CCW(s.pos,s.pos+s.dir,p)==0;
}
// 交点の個数を返す
int IntersectCC(Circle c1,Circle c2){
    if(c1==c2)
        return INFTY;
    double r1=c1.radius,r2=c2.radius,d=Abs(c1.center-c2.center);
    if(d>r1+r2+EPS || d<abs(r1-r2)-EPS)
        return 0;
    if(abs(d-(r1+r2))<EPS || abs(d-abs(r1-r2))<EPS)
        return 1;
    return 2;
}

// aとbは必ず交差していること
// 同一直線上にある場合，a.posを返す
Point InterPointLL(Line a,Line b){
    if(abs(Cross(a.dir,b.dir))<EPS) return a.pos;
    return a.pos+Cross(b.pos-a.pos,b.dir)/Cross(a.dir,b.dir)*a.dir;
}
// 同一直線上にある場合，s.posを返す
Point InterPointLS(Line l,Segment s){
    return InterPointLL(Line(s),l);
}
// 同一直線上にある場合，両端点を順に試し最初に交点であると判定されたものを返す
Point InterPointSS(Segment a,Segment b){
    if(abs(Cross(a.dir,b.dir))<EPS){
        if(IntersectSP(b,a.pos)) return a.pos;
        if(IntersectSP(b,a.pos+a.dir)) return a.pos+a.dir;
        if(IntersectSP(a,b.pos)) return b.pos;
        if(IntersectSP(a,b.pos+b.dir)) return b.pos+b.dir;
    }
    return InterPointLL(Line(a),Line(b));
}

// c1とc2は必ず2点で交わること
pair<Point,Point> InterPointCC(Circle c1,Circle c2){
    Point p1=c1.center,p2=c2.center;
    double r1=c1.radius,r2=c2.radius;
    double d=Abs(p1-p2);
    double a=(d*d+r1*r1-r2*r2)/(2*d);
    double s=sqrt(r1*r1-a*a);
```

```
    return mp(p1+a/d*(p2-p1)+s*Rot((p2-p1)/d,PI/2),
              p1+a/d*(p2-p1)-s*Rot((p2-p1)/d,PI/2));
}
```

9.5 距離

```
double DistLP(Line l,Point p){
    return Abs(Proj(l,p)-p);
}
double DistSP(Segment s,Point p){
    int ccw=CCW(s.pos,s.pos+s.dir,Proj(Line(s),p));
    if(ccw==-2) return Abs(s.pos-p);
    if(ccw== 2) return Abs(s.pos+s.dir-p);
    return DistLP(Line(s),p);
}
double DistLS(Line l,Segment s){
    if(IntersectLS(l,s)) return 0;
    return min(DistLP(l,s.pos),DistLP(l,s.pos+s.dir));
}
double DistSS(Segment a,Segment b){
    if(IntersectSS(a,b)) return 0;
    double d1=min(DistSP(a,b.pos),DistSP(a,b.pos+b.dir));
    double d2=min(DistSP(b,a.pos),DistSP(b,a.pos+a.dir));
    return min(d1,d2);
}
```

9.6 円の接線

```
vector<Line> Intangent(Circle c1,Circle c2)
{
    double r1=c1.radius,r2=c2.radius;
    double d=Abs(c1.center-c2.center);
    vector<Line> res;
    if(d>r1+r2+EPS){
        double t=acos((r1+r2)/d);
        rep(i,2){
            Point p1=c1.center+Rot(r1/d*(c2.center-c1.center),(i?1:-1)*t);
            Point p2=c2.center+Rot(r2/d*(c1.center-c2.center),(i?1:-1)*t);
            res.push_back(Line(p1,p2-p1));
        }
    }
    else if(d>r1+r2-EPS){
        Point p=c1.center+r1/d*(c2.center-c1.center);
        res.push_back(Line(p,Rot(p-c1.center,PI/2)));
    }
    return res;
}
vector<Line> Extangent(Circle c1,Circle c2)
{
    double r1=c1.radius,r2=c2.radius;
    double d=Abs(c1.center-c2.center);
    vector<Line> res;
    if(d>abs(r2-r1)+EPS){
        double t=acos((r1-r2)/d);
        rep(i,2){
            Point p1=c1.center+Rot(r1/d*(c2.center-c1.center),(i?1:-1)*t);
            Point p2=c2.center+Rot(r2/d*(c2.center-c1.center),(i?1:-1)*t);
            res.push_back(Line(p1,p2-p1));
        }
    }
    else if(d>abs(r2-r1)-EPS){
        Point p=c1.center+r1/d*(c2.center-c1.center);
        res.push_back(Line(p,Rot(p-c1.center,PI/2)));
    }
    return res;
}
```

9.7 凸包

計算量：$O(N \log N)$

```
// 連続する3点が同一直線上にある場合はfalseを返す
bool IsConvex(const vector<Point>& ps){
    int n=ps.size(),res=true;
    rep(i,n) res&=CCW(ps[i],ps[(i+1)%n],ps[(i+2)%n])==1;
    return res;
}

// 同一直線上の点はとらない.
// もしとりたければ, while(n>=2 && CCW(cs[n-2],cs[n-1],ps[i])<=0)と書き換える.
vector<Point> ConvexHull(vector<Point> ps){
    if(ps.size()==1) return ps;
    sort(all(ps),LessX());
    vector<Point> res;
    rep(_,2){
        int n=0;
        vector<Point> half(ps.size());
        rep(i,ps.size()){
            while(n>=2 && CCW(half[n-2],half[n-1],ps[i])!=1)
                n--;
            half[n++]=ps[i];
        }
        res.insert(res.end(),half.begin(),half.begin()+n-1);
        reverse(all(ps));
    }
    return res;
}
```

9.8 凸多角形の包含判定

```
// 凸多角形psの内部に点pがあるときtrue(境界を含む)
// ps.size()==0のときtrue
// ps.size()==1のとき, ps[0]とpが一致していればtrue
// ps.size()==2のとき, ps[0],ps[1]上にpがあればtrue
bool CoverGP(const vector<Point>& ps,Point p){
    rep(i,ps.size()){
        int ccw=CCW(ps[i],ps[(i+1)%ps.size()],p);
        if(!(ccw==1 || ccw==0))
            return false;
    }
    return true;
}
// 凸多角形psが凸多角形qsを内部にもつときtrue(境界を含む)
// ps.size()==0またはqs.size()==0のときtrue
bool CoverGG(const vector<Point>& ps,vector<Point>& qs){
    rep(i,qs.size())
        if(!CoverGP(ps,qs[i]))
            return false;
    return true;
}
```

9.9 凸多角形の切断

```
vector<Point> ConvexCut(const vector<Point>& ps,Line l){
    int n=ps.size();
    vector<Point> res;
    rep(i,n){
        int c1=CCW(l.pos,l.pos+l.dir,ps[i]);
```

```
        int c2=CCW(l.pos,l.pos+l.dir,ps[(i+1)%n]);
        if(c1!=-1)
            res.push_back(ps[i]);
        if(c1*c2==-1)
            res.push_back(InterPointLS(l,Segment(ps[i],ps[(i+1)%n]-ps[i])));
    }
    return res;
}
```

9.10 線分アレンジメント

同一線分上の推移的な辺を省略している．計算量：$O(N^3)$

```
// オーバーラップする線分は取り除いておくこと．
struct Edge{
    int src,dst;
    double weight;
    Edge(){}
    Edge(int s,int d,double w):src(s),dst(d),weight(w){}
    bool operator<(const Edge& e)const{return Signum(weight-e.weight)<0;}
    bool operator>(const Edge& e)const{return Signum(weight-e.weight)>0;}
};
void SegmentArrangement(const vector<Segment>& ss,Graph& g,vector<Point>& ps){
    rep(i,ss.size()){
        ps.push_back(ss[i].pos);
        ps.push_back(ss[i].pos+ss[i].dir);
        repi(j,i+1,ss.size()) if(IntersectSS(ss[i],ss[j]))
            ps.push_back(InterPointSS(ss[i],ss[j]));
    }
    sort(all(ps),LessX());
    ps.erase(unique(all(ps)),ps.end());

    g.resize(ps.size());
    rep(i,ss.size()){
        vector<pair<double,int> > ds;
        rep(j,ps.size()) if(IntersectSP(ss[i],ps[j]))
            ds.push_back(mp(Abs(ps[j]-ss[i].pos),j));
        sort(all(ds));
        rep(j,ds.size()-1){
            int u=ds[j].second,v=ds[j+1].second;
            double w=ds[j+1].first-ds[j].first;
            g[u].push_back(Edge(u,v,w));
            g[v].push_back(Edge(v,u,w));
        }
    }
}
```

10 その他

10.1 ビット演算

clz，clzll，ctz，ctzll は，$x = 0$ のときビット長を返す．

```
// population count
inline int popcount(uint x)
{
    x=(x&0x55555555)+(x>>1&0x55555555);
    x=(x&0x33333333)+(x>>2&0x33333333);
    x=(x&0x0f0f0f0f)+(x>>4&0x0f0f0f0f);
    x=(x&0x00ff00ff)+(x>>8&0x00ff00ff);
    return (x&0x0000ffff)+(x>>16&0x0000ffff);
}
inline int popcountll(ull x)
{
    return popcount(x)+popcount(x>>32);
}

// count leading zero
inline int clz(uint x)
{
    int i=0;
    if(!(x&0xffff0000)) i+=16,x<<=16;
    if(!(x&0xff000000)) i+=8,x<<=8;
    if(!(x&0xf0000000)) i+=4,x<<=4;
    if(!(x&0xc0000000)) i+=2,x<<=2;
    if(!(x&0x80000000)) i+=1,x<<=1;
    return i+!x;
}
inline int clzll(ull x)
{
    int y=clz(x>>32);
    return y==32?y+clz(x):y;
}

// count trailing zero
inline int ctz(uint x)
{
    int i=0;
    if(!(x&0x0000ffff)) i+=16,x>>=16;
    if(!(x&0x000000ff)) i+=8,x>>=8;
    if(!(x&0x0000000f)) i+=4,x>>=4;
    if(!(x&0x00000003)) i+=2,x>>=2;
    if(!(x&0x00000001)) i+=1,x>>=1;
    return i+!x;
}
inline int ctzll(ull x)
{
    int y=ctz(x);
    return y==32?y+ctz(x>>32):y;
}

// find first set
inline int ffs(uint x)
{
    return x?clz(x)+1:0;
}
inline int ffs(ull x)
{
    return x?clzll(x)+1:0;
}

// bitwise reverse
inline uint bitrev(uint x)
{
    x=(x&0xaaaaaaaa)>>1|(x&0x55555555)<<1;
    x=(x&0xcccccccc)>>2|(x&0x33333333)<<2;
    x=(x&0xf0f0f0f0)>>4|(x&0x0f0f0f0f)<<4;
    x=(x&0xff00ff00)>>8|(x&0x00ff00ff)<<8;
    return (x&0xffff0000)>>16|(x&0x0000ffff)<<16;
}
inline ull bitrev(ull x)
{
    return bitrev(x>>32)|(ull)bitrev(x)<<32;
}

// 非零部分集合を列挙
for(int s=b;s;s=(s-1)&b){
    ...
```

```
}

// {0,1,...,n-1}の，サイズkの部分集合を列挙
// 最終行を s|=(lz>>ffs(lo))-1 とすれば除算をなくせる
// 注意: 0<k<=n でなければならない
for(int s=(1<<k)-1;!(s&1<<n);){
    /* ... */
    int lo=s&-s,lz=(s+lo)&~s;
    s=(s|lz)&~(lz-1);
    s|=(lz/lo/2-1);
}
```

10.2 ハッシュ関数

ユーザ定義型を unordered_set や unordered_map のキーとして使うためには，hash の特殊化を書く必要がある．以下は FNV-1a による実装例.

```
namespace std{
    template<>
    struct hash<tuple<int,int>>{
        size_t operator()(const tuple<int,int>& x)const{
            const char* p=(const char*)&x;
            size_t res=2166136261;
            rep(i,sizeof(x)) (res^=*p++)*=16777619;
            return res;
        }
    };
}
```

10.3 Fairfield の公式

西暦 1 年 1 月 0 日からの経過日数を求める．（経過日数）≡ 0 (mod 7) のとき日曜日.

```
int Fairfield(int y,int m,int d)
{
    if(m<=2) y--,m+=12;
    return 365*y+y/4-y/100+y/400+153*(m+1)/5+d-428;
}
```

10.4 さいころ

```
struct Dice{
    int face[6]; // top,bottom,left,right,front,back
    void Rotate(int a,int b,int c,int d){
        int temp=face[a];
        face[a]=face[b];
        face[b]=face[c];
        face[c]=face[d];
        face[d]=temp;
    }
    void TurnF(int n){rep(i,n)Rotate(0,2,1,3);} // x軸を回転軸として右回りに90n度回転
    void TurnR(int n){rep(i,n)Rotate(0,4,1,5);} // y軸を回転軸として右回りに90n度回転
    void TurnU(int n){rep(i,n)Rotate(4,3,5,2);} // z軸を回転軸として右回りに90n度回転
    void TurnF(){TurnF(1);}
    void TurnR(){TurnR(1);}
    void TurnU(){TurnU(1);}
};
```