The next code will be directly imported from a file

# 1 MISC

## 1.1 Template

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;
typedef pair<int, ll> pil;
typedef pair<ll, int> pli;
typedef pair<double,double> pdd;
#define SQ(i) ((i)*(i))
#define MEM(a, b) memset(a, (b), sizeof(a))
#define SZ(i) int(i.size())
#define FOR(i, j, k, in) for (int i=j ; i<k ; i+=in)
#define FOR3(i, a, b) for (int i = (a); i<(b); i++)
#define RFOR(i, j, k, in) for (int i=j ; i>=k ; i-=
    in)
#define REP(i, j) FOR(i, 0, j, 1)
#define REP1(i,j) FOR(i, 1, j+1, 1)
#define RREP(i, j) RFOR(i, j, 0, 1)
#define ALL(_a) _a.begin(),_a.end()
#define mp make_pair
#define pb push_back
#define eb emplace_back
#define X first
#define Y second
#define f first
#define s second
#define MN(a,b) a = min(a,(__typeof__(a))(b))
#define MX(a,b) a = max(a,(__typeof__(a))(b))
#define SORT_UNIQUE(c) (sort(c.begin(),c.end()), c.
    resize(distance(c.begin(),unique(c.begin(),c.end
    ()))))
#ifdef BTC
#define TIME(i) Timer i(#i)
#define debug(...) do{\
    fprintf(stderr,"%s - %d (%s) = ",
    __PRETTY_FUNCTION__,__LINE__,#__VA_ARGS__);\
    _do(__VA_ARGS__);\
}while(0)
template<typename T>void _do(T &&_x){cerr<<_x<<endl
    ;}
template<typename T,typename ...S> void _do(T &&_x,S
     &&..._t){cerr<<_x<<" ,";_do(_t...);}
template<typename _a,typename _b> ostream& operator
    << (ostream &_s,const pair<_a,_b> &_p){return _s
    <<"("<<_p.X<<","<<_p.Y<<")";}
template<typename It> ostream& _OUTC(ostream &_s,It
    _ita,It _itb)
{
    _s<<"{";
    for(It _it=_ita;_it!=_itb;_it++)
    {
        _s<<(_it==_ita?"":",")<<*_it;
    }
    _s<<"}";
    return _s;
}
template<typename _a> ostream &operator << (ostream
    &_s,vector<_a> &_c){return _OUTC(_s,ALL(_c));}
template<typename _a> ostream &operator << (ostream
    &_s,set<_a> &_c){return _OUTC(_s,ALL(_c));}
template<typename _a> ostream &operator << (ostream
    &_s,deque<_a> &_c){return _OUTC(_s,ALL(_c));}
template<typename _a,typename _b> ostream &operator
    << (ostream &_s,map<_a,_b> &_c){return _OUTC(_s,
    ALL(_c));}
template<typename _t> void pary(_t _a,_t _b){_OUTC(
    cerr,_a,_b);cerr<<endl;}
#define IOS()
class Timer {
private:
    string scope_name;
    chrono::high_resolution_clock::time_point
    start_time;
public:
    Timer (string name) : scope_name(name) {
        start_time = chrono::high_resolution_clock::
    now();
    }
    ~Timer () {
        auto stop_time = chrono::
    high_resolution_clock::now();
        auto length = chrono::duration_cast<chrono::
    microseconds>(stop_time - start_time).count();
        double mlength = double(length) * 0.001;
        debug(scope_name, mlength);
    }
};
#else
#define TIME(i)
#define debug(...)
#define pary(...)
#define endl '\n'
#define IOS() ios_base::sync_with_stdio(0);cin.tie
    (0)
#endif

const ll MOD = 1000000007;
const ll INF = 0x3f3f3f3f3f3f3f3f;
const int iNF = 0x3f3f3f3f;
// const int MAXN =

void GG(){cout<<"-1\n"; exit(0);}

/********** Good Luck :) **********/
int main () {
    TIME(main);
    IOS();

    return 0;
}
```

## 1.2 raw string

```
#include <bits/stdc++.h>
using namespace std;
int main () {
    string str1 = R"(\"'"^&*()))";
    cout << str1 << endl;
}
```

## 1.3 Random

```


main(){
    IOS();
    mt19937 rng(chrono::steady_clock::now().
    time_since_epoch().count());
    // Basically the same as rand()
    vector<int> v(10); iota(ALL(v),1);
    shuffle(ALL(v), rng); // Use instead of
    random_shuffle
    for (int x : v) cout<<x<<' ';
    cout<<"Random number [0,100): "<<rng()%100<<endl
    ;
}
```

# 2 Graph

## 2.1 centroid decomp

## 2.2 clique

## 2.3 hld

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

const int MAXN = 10003;

struct edge{
  int u,v,w,n;
}e[MAXN*2];

int t,n,a,b,c;
int dep[MAXN],sz[MAXN],fat[MAXN],son[MAXN],top[MAXN
    ];
int in[MAXN],cnt,idx,head[MAXN];
int sg[MAXN*2];
char cmd[10];

void add_edge(int u,int v,int w){
  e[cnt].u = u;
  e[cnt].v = v;
  e[cnt].w = w;
  e[cnt].n = head[u];
  head[u] = cnt++;
}

void dfs1 (int nd,int par) {
  dep[nd] = dep[par] + 1;
  sz[nd] = 1;
  fat[nd] = par;
  son[nd] = 0;
  for (int i=head[nd];i!=-1;i=e[i].n) {
    if (e[i].v==par) continue;
    dfs1(e[i].v,nd);
    sz[nd] += sz[e[i].v];
    if(sz[e[i].v] > sz[son[nd]]) son[nd] = e[i].v;
  }
}

void dfs2 (int nd,int tp) {
  in[nd] = idx++;
  top[nd] = tp;
  if (son[nd]) dfs2(son[nd],tp);
  for (int i=head[nd];i!=-1;i=e[i].n) {
    if (e[i].v==fat[nd] || e[i].v==son[nd]) continue
    ;
    dfs2(e[i].v,e[i].v);
  }
}

int qpath (int x,int y) {
  int ret = 0;
  while (top[x] != top[y]) {
    if (dep[top[x]] < dep[top[y]]) swap(x,y);
    // ret = max(ret,query(in[top[x]],in[x]+1));
    x = fat[top[x]];
  }
  if(x==y)return ret;
  if (dep[x] < dep[y]) swap(x,y);
//    ret = max(ret,query(in[son[y]],in[x]+1));
  return ret;
}
```

## 2.4 lca

```cpp
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 15003;
const int MAXLG = __lg(MAXN) + 2;
int n,q,a,b;

int anc[MAXLG][MAXN];
int dep[MAXN];
vector<int> edge[MAXN];
void dfs(int nd,int par){
  anc[0][nd] = par;
  dep[nd] = dep[par] + 1;
  for(int v:edge[nd]){
    if(v!=par) dfs(v,nd);
  }
}
void build_lca(){
  for(int i=1;i<MAXLG;i++){
    for(int j=0;j<n;j++){
      anc[i][j] = anc[i-1][anc[i-1][j]];
    }
  }
}

int query(int u,int v){
  if(dep[u] < dep[v])swap(u,v);
  for(int i=MAXLG-1;i>=0;i--){
    if(dep[anc[i][u]] >= dep[v]) u = anc[i][u];
  }
  if(u==v)return u;

  for(int i=MAXLG-1;i>=0;i--){
    if(anc[i][u] != anc[i][v]) {
      u = anc[i][u];
      v = anc[i][v];
    }
  }

  return anc[0][u];
}
int main(){
  cin>>n>>q;
  for(int i=0;i<n-1;i++) cin>>a>>b,edge[a].
    emplace_back(b),edge[b].emplace_back(a);

  dfs(0,0);
  build_lca();
  for(int i=0;i<q;i++){
    cin>>a>>b;
    cout<<query(a,b)<<endl;
  }
}


// Doubling LCA
```

## 2.5 ap

```cpp
/*
from: http://sunmoon-template.blogspot.com
*/
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 100005;

std::vector<int> G[MAXN];// 1-base
std::vector<int> bcc[MAXN];
int low[MAXN],vis[MAXN],Time;
int bcc_id[MAXN],bcc_cnt;// 1-base
bool is_cut[MAXN];//bcc_id is ndef if is_cut
int st[MAXN],top;
void dfs(int u,int pa=-1){
  int v,child=0;
  low[u]=vis[u]=++Time;
  st[top++]=u;
  for(size_t i=0;i<G[u].size();++i){
    if(!vis[v=G[u][i]]){
      dfs(v,u),++child;
      low[u]=std::min(low[u],low[v]);
      if(vis[u]<=low[v]){
        is_cut[u]=1;
        bcc[++bcc_cnt].clear();
        int t;
        do{
          bcc_id[t=st[--top]]=bcc_cnt;
          bcc[bcc_cnt].push_back(t);
        }while(t!=v);
        bcc_id[u]=bcc_cnt;
        bcc[bcc_cnt].push_back(u);
      }
    }else if(vis[v]<vis[u]&&v!=pa)//reverse
      low[u]=std::min(low[u],vis[v]);
  }
  if(pa==-1&&child<2)is_cut[u]=0;//u for root
```

```
}
inline void bcc_init(int n){
  Time=bcc_cnt=top=0;
  for(int i=1;i<=n;++i){
    G[i].clear();
    vis[i]=0;
    is_cut[i]=0;
    bcc_id[i]=0;
  }
}

int main () {
    int n, m;
    cin >> n >> m;
    bcc_init(n);
    for (int i=0; i<m; i++) {
        int u, v;
        cin >> u >> v;
        G[u].emplace_back(v);
        G[v].emplace_back(u);
    }

    dfs(1);
    for (int i=1; i<=n; i++) {
        cout << (is_cut[i] ? -1 : bcc_id[i]) << " \n
    "[i==n];
    }
}
```

## 2.6 dijkstra

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int,int> pii;
#define REP(i,n) for(int i=0;i<n;i++)
#define REP1(i,n) for(int i=1;i<=n;i++)
#define X first
#define Y second
const int MAXN = 1000003;
const int INF = (int)0x3f3f3f3f;
int n,m,s,g,a,b,v;


int dis[MAXN];
bool vis[MAXN];
vector<pii> e[MAXN];

int dijkstra (int s, int t) {
    memset(dis,INF,(n+1)*4);
    memset(vis,0,(n+1)*4);

    dis[s] = 0;
    priority_queue<pii,vector<pii>,greater<pii>> pq;
    pq.emplace(0,s);
    REP(i,n){
      int found = -1;
      while(pq.size()&&vis[found=pq.top().Y])pq.pop
    ();
      if(found==-1)break;
      vis[found]=1;
      for(auto vp:e[found]){
        if(dis[vp.X]>dis[found]+vp.Y){
          dis[vp.X] = dis[found]+vp.Y;
          pq.emplace(dis[vp.X],vp.X);
        }
      }
    }
}

void add_edge (int f, int t, int w) {
    e[f].emplace_back(t, w);
}

int main(){
  ios_base::sync_with_stdio(0);cin.tie(0);
  while(cin>>n>>m>>s>>g){
    REP(i,m){
      cin>>a>>b>>v;
      add_edge(a, b, v);
    }

    cout<<(dis[g]==INF?-1:dis[g])<<'\n';
  }
}
```

## 2.7 bridge

```
/*
from: http://sunmoon-template.blogspot.com
*/
#include <bits/stdc++.h>
using namespace std;

const int MAXN = 100005;
struct edge{
  int u,v;
  bool is_bridge;
  edge(int u=0,int v=0):u(u),v(v),is_bridge(0){}
};
std::vector<edge> E;
std::vector<int> G[MAXN];// 1-base
int low[MAXN],vis[MAXN],Time;
int bcc_id[MAXN],bridge_cnt,bcc_cnt;// 1-base
int st[MAXN],top;// for bcc
inline void add_edge(int u,int v){
  G[u].push_back(E.size());
  E.push_back(edge(u,v));
  G[v].push_back(E.size());
  E.push_back(edge(v,u));
}
void dfs(int u,int re=-1){// re is last edge
  int v;
  low[u]=vis[u]=++Time;
  st[top++]=u;
  for(size_t i=0;i<G[u].size();++i){
    int e=G[u][i];v=E[e].v;
    if(!vis[v]){
      dfs(v,e^1);//e^1 reverse
      low[u]=std::min(low[u],low[v]);
      if(vis[u]<low[v]){
        E[e].is_bridge=E[e^1].is_bridge=1;
        ++bridge_cnt;
      }
    }else if(vis[v]<vis[u]&&e!=re)
      low[u]=std::min(low[u],vis[v]);
  }
  if(vis[u]==low[u]){// build bcc
    ++bcc_cnt;// 1-base
    do bcc_id[v=st[--top]]=bcc_cnt;
    while(v!=u);
  }
}
inline void bcc_init(int n){
  Time=bcc_cnt=bridge_cnt=top=0;
  E.clear();
  for(int i=1;i<=n;++i){
    G[i].clear();
    vis[i]=0;
    bcc_id[i]=0;
  }
}

int main () {
    int n, m;
    cin >> n >> m;
    bcc_init(n);
    for (int i=0; i<m; i++) {
        int u, v;
        cin >> u >> v;
        add_edge(u, v);
    }

    dfs(1);
    for (int i=1; i<=n; i++) {
        cout << bcc_id[i] << " \n"[i==n];
    }
}
```

### 2.8 scc

# 3 Math

## 3.1 FFT-precision

```
#include <bits/stdc++.h>
using namespace std;
#define SZ(v) int(v.size())
#define REP(i,n) for(int i=0;i<n;i++)
#define REP1(i,n) for(int i=1;i<=n;i++)

const int MAXN = 1<<20;
typedef complex<double> cd;

const double pi = acos(-1);
vector<int> bs;
cd omg[MAXN+3];

void FFT (vector<cd> &v, int d) {
    for (int i=1,j=SZ(v)>>1; i<SZ(v)-1; i++) {
        if (i < j) {
            swap(v[i], v[j]);
        }
        int k = SZ(v)>>1;
        while (k <= j) {
            j -= k;
            k >>= 1;
        }
        if (k > j) {
            j += k;
        }
    }

    for (int h=2; h<=SZ(v); h<<=1) {
        for (int i=0; i<SZ(v); i+=h) {
            for (int k=i; k<i+h/2; k++) {
                int idx = k-i;
                int r = k+h/2;
                cd x = v[k] - omg[d > 0 ? idx*(MAXN/
    h) : MAXN-idx*(MAXN/h)] * v[r];
                v[k] = v[k] + omg[d > 0 ? idx*(MAXN/
    h) : MAXN-idx*(MAXN/h)] * v[r];
                v[r] = x;
            }
        }
    }

    if (d < 0) {
        REP (i, SZ(v)) {
            v[i] /= SZ(v);
        }
    }
}

void build_omg() {
    omg[0] = omg[MAXN] = 1;
    REP1 (i, MAXN-1) {
        omg[i] = polar(1.0, i*pi*2/MAXN);
    }
}

vector<int> mul (vector<int> &v1, vector<int> &v2) {
    int n = 1;
    while (n < SZ(v1) + SZ(v2)) {
        n <<= 1;
    }
    vector<cd> x(n), y(n);
    REP (i, SZ(v1)) {
        x[i] = v1[i];
    }
    REP (i, SZ(v2)) {
        y[i] = v2[i];
    }
    FFT(x, 1);
    FFT(y, 1);
    REP (i, n) {
        x[i] *= y[i];
    }
    FFT(x, -1);
    vector<int> ret(n);
    REP (i, n) {
        ret[i] = min(1, (int)round(x[i].real()));
    }
    while (SZ(ret)>1 && ret.back() == 0) {
        ret.pop_back();
    }
    return ret;
}

int main () {

}
```

## 3.2 CRT

## 3.3 rho

```
#include <bits/stdc++.h>
using namespace std;
#define ll long long
#define pii pair<int, int>
#define ull unsigned ll
#define f first
#define s second
#define FOR(i,a,b) for (int i=(a); i<(b); i++)
#define REP(i,n) for (int i=0; i<(n); i++)
#define RREP(i,n) for (int i=(n-1); i>=0; i--)
#define ALL(x) x.begin(),x.end()
#define SZ(x) (int)x.size()
#define SQ(x) (x)*(x)
#define MN(a,b) a = min(a,(__typeof__(a))(b))
#define MX(a,b) a = max(a,(__typeof__(a))(b))
#define pb push_back
#define SORT_UNIQUE(c) (sort(c.begin(),c.end()), c.
    resize(distance(c.begin(),unique(c.begin(),c.end
    ()))))
#ifdef BALBIT
#define IOS()
#define debug(...) do{\
    fprintf(stderr,"%s - %d (%s) = ",
    __PRETTY_FUNCTION__,__LINE__,#__VA_ARGS__);\
    _do(__VA_ARGS__);\
}while(0)
template<typename T>void _do(T &&_x){cerr<<_x<<endl
    ;}
template<typename T,typename ...S> void _do(T &&_x,S
     &&..._t){cerr<<_x<<" ,";_do(_t...);}
template<typename _a,typename _b> ostream& operator
    << (ostream &_s,const pair<_a,_b> &_p){return _s
    <<"("<<_p.X<<","<<_p.Y<<")";}
template<typename It> ostream& _OUTC(ostream &_s,It
    _ita,It _itb)
{
    _s<<"{";
    for(It _it=_ita;_it!=_itb;_it++)
    {
        _s<<(_it==_ita?"":",")<<*_it;
    }
    _s<<"}";
    return _s;
}
template<typename _a> ostream &operator << (ostream
    &_s,vector<_a> &_c){return _OUTC(_s,ALL(_c));}
template<typename _a> ostream &operator << (ostream
    &_s,set<_a> &_c){return _OUTC(_s,ALL(_c));}
template<typename _a> ostream &operator << (ostream
    &_s,deque<_a> &_c){return _OUTC(_s,ALL(_c));}
template<typename _a,typename _b> ostream &operator
    << (ostream &_s,map<_a,_b> &_c){return _OUTC(_s,
    ALL(_c));}
template<typename _t> void pary(_t _a,_t _b){_OUTC(
    cerr,_a,_b);cerr<<endl;}
#else
```

```
#define IOS() ios_base::sync_with_stdio(0);cin.tie
    (0);
#define endl '\n'
#define debug(...)
#define pary(...)
#endif

// #define int ll

const int iinf = 1<<29;
const ll inf = 1ll<<60;
const ll mod = 1e9+7;


void GG(){cout<<"-1\n"; exit(0);}

ll mpow(ll a, ll n, ll mo = mod){ // a^n % mod
    ll re=1;
    while (n>0){
        if (n&1) re = re*a %mo;
        a = a*a %mo;
        n>>=1;
    }
    return re;
}

ll inv (ll b, ll mo = mod){
    if (b==1) return b;
    return (mo-mo/b) * inv(mo%b) % mo;
}

const int maxn = 1e5+5;

#define lll __int128

lll c = 1;
lll g(lll x, lll n){
    return (x*x+c)%n;
}

lll gcd(lll a, lll b){
    if (b==0) return a;
    return gcd(b,a%b);
}

lll po(lll n){
    lll x = 2, y = 2, d = 1;
    while (d==1){
        x = g(x,n); y = g(g(y,n),n);
        d = gcd(x>y?x-y:y-x,n);
    }
    if (d==n) return -1;
    return d;
}

ll fac(ll n){
    if (n%2==0) return 2;
    lll ans = -1;
    for (int i = 0; i<5 && ans==-1; i++) {
        c++; if (c==2) c++;
        ans = po(n);
    }
    return ans;
}

main(){
    ll test = 1709049187;
    lll moo = test;
    ll ans = fac(moo);
    cout<<ans<<endl;

}
```

## 3.4 FFT

```
const double PI = acos(-1.0);
#define cd complex<double>

void FFT(vector<cd> &a, bool rev=0){
    int n = SZ(a);
    for (int i = 1, j = 0; i<n; i++){
        int bit = n>>1;
        while (j>=bit) j-=bit, bit>>=1; j+=bit;
        if (i<j) swap(a[i], a[j]);
    }
    for (int B = 2; B<=n; B*=2){
        double ang = 2 * PI / B * (rev?-1:1);
        cd w0 (cos(ang), sin(ang));
        for (int i = 0; i<n; i+=B){
            cd w (1,0);
            for (int j = 0; j<B/2; j++){
                cd u = a[i+j], v = w*a[i+j+B/2];
                a[i+j] = u+v, a[i+j+B/2] = u-v;
                w *= w0;
            }
        }
    }
    if (rev) REP(i,n) a[i] /= n;
}

vector<ll> mul (vector<ll> a, vector<ll> b){
    int n = 1; while (n < SZ(a) + SZ(b)) n*=2;
    vector<cd> x(n), y(n);
    REP(i, SZ(a)) x[i] = cd(a[i],0); REP(j, SZ(b)) y
    [j] = cd(b[j],0);
    FFT(x); FFT(y);
    REP(i, n) x[i] *= y[i];
    FFT(x,1);
    vector<ll> re(n);
    REP(i,n) re[i] = min((ll)(round(x[i].real())),1
    ll);
    while (re.size()>1 && re.back()==0) re.pop_back
    (); return re;
}
```

## 3.5 NTT

```
void NTT(vector<ll> &a, ll mo, bool rev=0){
    // mo has to be 2^k * c + 1
    int n = SZ(a);
    while ((n&(-n))!=n) {
        a.pb(0); n++;
    }
    for (int i = 1, j = 0; i<n; i++){
        int bit = n>>1;
        while (j>=bit) j-=bit, bit>>=1; j+=bit;
        if (i<j) swap(a[i], a[j]);
    }
    for (int B = 2; B<=n; B*=2){
        ll w0 = mpow(3,(mo-1)/(B),mo);
        for (int i = 0; i<n; i+=B){
            ll w = 1;
            for (int j = 0; j<B/2; j++){
                ll u = a[i+j], v = w*a[i+j+B/2]%mo;
                a[i+j] = u+v, a[i+j+B/2] = u-v;
                if (a[i+j]>=mo) a[i+j]-=mo; if (a[i+
    j+B/2]<0) a[i+j+B/2]+=mo;
                w = w*w0%mod;
            }
        }
    }
    if (rev) {
        reverse(next(a.begin()),a.end());
        ll invn = inv(n,mo);
        REP(i,n) a[i] = a[i]*invn%mod;
    }
}

vector<ll> mul (vector<ll> a, vector<ll> b, ll mo =
    mod){
    int n = 1; while (n < SZ(a) + SZ(b)) n*=2;
    vector<ll> x(n), y(n);
    REP(i, SZ(a)) x[i] = a[i]; REP(j, SZ(b)) y[j] =
    b[j];
    NTT(x,mo); NTT(y,mo);
    REP(i, n) x[i] = x[i] * y[i] % mo;
```

```
    NTT(x,mo,1);
    while (x.size()>1 && x.back()==0) x.pop_back();
    return x;
}
```

### 3.6 miller rabin

### 3.7 inversion

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

const ll mod = 10000007;

ll inv (ll b, ll mo = mod){
    if (b==1) return b;
    return (mo-mo/b) * inv(mo%b) % mo;
}

void extGCD(ll A,ll B,ll &x,ll &y) { // A p coprime
    if (B == 0) {
        x = 1;
        y = 0;
        assert(A == 1);
        return;
    }
    ll xx,yy;
    extGCD(B,A%B,xx,yy);
    x = yy;
    y = xx - A/B*yy;
    return;
}

ll ext_inv (ll a, ll p) { // a, p co-prime
    ll x, y;
    extGCD(a,p, x, y);
    x %= p;
    if (x < 0) {
        x += p;
    }
    assert(a * x % p);
    return x;
}
int main () {
    ll a, p;
    cin >> a >> p;
    ll ainv = ext_inv(a, p);
    cout << ainv << endl;
}
```

### 3.8 geometry

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

struct Point{
  double x, y;

  bool operator < (const Point &b) const {
    return tie(x,y) < tie(b.x,b.y);
    //return atan2(y,x) < atan2(b.y,b.x);
  }
  Point operator + (const Point &b) const {
    return {x+b.x,y+b.y};
  }
  Point operator - (const Point &b) const {
    return {x-b.x,y-b.y};
  }
  Point operator * (const double d) const {
    return {x*d,y*d};
  }
  Point operator / (const double d) const {
    return {x/d,y/d};
  }
  double operator * (const Point &b) const {
    return x*b.x + y*b.y;
  }
  double operator % (const Point &b) const { //
    Cross!
    return x*b.y - y*b.x;
  }
  Point(double xx, double yy): x(xx), y(yy){  }
};

double Length( const Point &p ){
  return sqrt( p.x*p.x + p.y*p.y );
}

int ori(const Point &a, const Point &b, const Point
    &c){
  int tmp = (c-a)%(b-a);
  if (tmp==0) return 0; //Collinear
  return tmp>0? 1: -1;
}

bool collinear(const Point &a, const Point &b, const
     Point &c){
  return ori(a, b, c)==0;
}

bool btw(const Point &a, const Point &b, const Point
     &c){
  return(a-c)*(b-c)<=0;
}

typedef Point Vector;

double Angle( const Vector &a, const Vector &b ){
  double A = Length(a);
  double B = Length(b);
  double v = a*b;
  double theta = acos( v/A/B );
  return theta;
}

Vector rot(Vector vec, double a){
  return Vector(cos(a)*vec.x-sin(a)*vec.y, sin(a)*
    vec.x+cos(a)*vec.y);
}

Vector Normal(const Vector &v){
  return v / Length(v);
}

Point intersect_at(const Point &p, const Vector &v,
    const Point &q, const Vector &w){
  Vector u = q-p;
  return p+v*(u%w)/(q%w);
}

bool cmp(const Point&a, const Point &b){
  return a<b;
  //Sort by x first, then by y.
}

vector<Point> convex_hull(vector<Point>arr){
  sort (arr.begin(), arr.end(), cmp);
  vector<Point> p;
  int m = 0; // size of p
  for (int i=0; i<arr.size(); i++){ // Lower hull
    //cout<<"On the "<<i<<"-th one. "<<arr[i].x<<'
    '<<arr[i].y<<'\n';
    while (m>=2&&(p[m-1]-p[m-2])%(arr[i]-p[m-2])<0){
      //Get rid of a previous point
      //cout<<"Got rid of "<<p[m-1].x<<' '<<p[m-1].y
    <<'\n';
      p.pop_back(); m--;
    }
    p.push_back(arr[i]); m++;
  }
  //cout<<"Onto upper hull"<<'\n';
  int tmp = m+1; //the size of lower hull +1
  for (int i=arr.size()-2; i>=0; i--){
    //cout<<"On the "<<i<<"-th one. "<<arr[i].x<<'
    '<<arr[i].y<<'\n';
    while (m>=tmp&&(p[m-1]-p[m-2])%(arr[i]-p[m-2])
```

```
    <0){
      //cout<<"Got rid of "<<p[m-1].x<<' '<<p[m-1].y
    <<'\n';
      p.pop_back(); m--;
    }
    p.push_back(arr[i]); m++;
  }
  //cout<<m<<'\n';
  if (arr.size()>1) p.pop_back(); //Repeated
  return p;
}

//Segment banana

double signedArea(Point p[], int n){
  double re = 0.0;
  for (int i=0; i<n; i++){
    re+=p[i]%p[(i+1)%n];
  }
  return re/2.0; //Cross returns twice the triangle'
    s area
}

bool intersect(const Point a, const Point b, const
    Point c, const Point d){
  int abc = ori(a, b, c);
  int abd = ori(a, b, d);
  int cda = ori(c, d, a);
  int cdb = ori(c, d, b);
  if (abc==0&&abd==0){
    return btw(a,b,c)||btw(a,b,d)||btw(c,d,a)||btw(c
    ,d,b);
  }else return (abc*abd<=0&&cda*cdb<=0);
}
```

### 3.9 linear sieve

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

const int MAXC = 1000006;
bool sieve[MAXC];
vector<int> prime;

void linear_sieve() {
    for (int i=2; i<MAXC; i++) {
        if (!sieve[i]) prime.emplace_back(i);
        for (int j=0; i*prime[j]<MAXC; j++) {
            sieve[i*prime[j]] = true;
            if (i % prime[j] == 0) {
                break;
            }
        }
    }
}

int main () {
    linear_sieve();
    for (int i=0; i<20; i++) {
        cout << prime[i] << " \n"[i==19];
    }
}
```

## 4 String

### 4.1 zvalue

```
#include <bits/stdc++.h>
using namespace std;
const int MAXN = 2000006;

int z[MAXN];
string a;
void init(string x) {
    a = x;
    std::memset(z, 0, sizeof z);
}
void z_build() {
    z[0] = 0;
    for (int i = 1, bst = 0; a[i]; i++) {
        if (bst + z[bst] < i) {
            z[i] = 0;
        } else {
            z[i] = min(z[i - bst], bst + z[bst] - i)
    ;
        }
        while (a[z[i]] == a[z[i] + i]) {
            z[i]++;
        }
        if (i + z[i] > bst + z[bst]) {
            bst = i;
        }
    }
}

int mat(string x,string y) {
    int ret = 0;
    init(x+'$'+y);
    z_build();
    for (int i=int(x.size())+1;i<=int(x.size()+y.
    size());i++) {
        ret +=  (z[i] == int(x.size()));
    }
    return ret;
}

int main () {
    string a, b;
    cout << mat(a, b) << endl;
}
```

### 4.2 suffix array

```
//
// Suffix Array (Manbar and Myers' O(n (log n)^2))
//
// Description:
//   For a string s, tts suffix array is a
    lexicographically sorted
//   list of suffixes of s. For example, for s = "
    abbab", its SA is
//     0 ab
//     1 abbab
//     2 b
//     3 bab
//     4 bbab
//
// Algorithm:
//   Manbar and Myers' doubling algorithm.
//   Suppose that suffixes are sorted by its first h
    characters.
//   Then, the comparison of first 2h characters is
    computed by
//     suf(i) <_2h suf(j) == if (suf(i) !=_h suf(j))
     suf(i) <_h suf(j)
//                           else
     suf(i+h) <_h suf(j+h)
//
// Complexity:
//   O(n (log n)^2).
//   If we use radix sort instead of standard sort,
//   we obtain O(n log n) algorithm. However, it
    does not improve
//   practical performance so much.
//
// Verify:
//   SPOJ 6409: SARRAY (80 pt)
//
#include <bits/stdc++.h>

using namespace std;

struct suffix_array {
  int n;
  vector<int> x;
  suffix_array(const char *s) : n(strlen(s)), x(n) {
    vector<int> r(n), t(n);
```

```
    for (int i = 0; i < n; ++i) r[x[i] = i] = s[i];
    for (int h = 1; t[n-1] != n-1; h *= 2) {
      auto cmp = [&](int i, int j) {
        if (r[i] != r[j]) return r[i] < r[j];
        return i+h < n && j+h < n ? r[i+h] < r[j+h]
    : i > j;
      };
      sort(x.begin(), x.end(), cmp);
      for (int i = 0; i+1 < n; ++i) t[i+1] = t[i] +
    cmp(x[i], x[i+1]);
      for (int i = 0; i < n; ++i)   r[x[i]] = t[i];
    }
  }
  int operator[](int i) const { return x[i]; }
};

int main() {
  char s[100010];
  scanf("%s", s);
  suffix_array sary(s);
  for (int i = 0; i < sary.n; ++i)
    printf("%d\n", sary[i]);
}
```

### 4.3 kmp

```
int app(string s, string t){ // Returns number of
    times s appears in t
  int n = s.length(), m = t.length();
  if (n>m) return 0;
  vector<int> f(n); f[0]=-1;
  for (int i = 1; i<n; i++){
    f[i] = f[i-1];
    while (f[i]!=-1 && s[f[i]+1]!=s[i]) f[i] = f[f[i
    ]];
    if (s[f[i]+1]==s[i]) f[i]++;
  }
  int j = 0, re = 0;
  for (int i = 0; i<m; i++){
    if (t[i] == s[j]) j++;
    else if (j) j = f[j-1]+1, i--;
    if (j==n) re++, j = f[j-1]+1;
  }
  return re;
}
```

### 4.4 ac automation

## 5 Flow and Matching

### 5.1 km o(n4)

```
const int mxn = 100;

bool vx[mxn], vy[mxn]; // Visited x or y
int my[mxn]; // Match of y
ll slk[mxn], lx[mxn], ly[mxn]; // Slack (on y),
    value on x, value on y
int g[mxn][mxn]; // Adjacency matrix with weights
int n;

bool dfs(int v){
    vx[v] = 1;
    REP(i,n){
        if (vy[i]) continue;
        if (g[v][i] == lx[v] + ly[i]) {
            vy[i] = 1;
            if (my[i]==-1 || dfs(my[i])){
                my[i] = v; return 1;
            }
        }else{
            MN(slk[i], lx[v] + ly[i] - g[v][i]);
        }
    }
    return 0;
}

ll mxmch(){
    REP(i,n) REP(j,n) MX(lx[i], g[i][j]);
    fill(my, my+n, -1);
    REP(i,n){
        while (1){
            fill(vx, vx+n, 0); fill(vy, vy+n, 0);
    fill(slk, slk+n, inf);
            if (dfs(i)) break;
            ll hv = *min_element(slk, slk+n);
            REP(i,n) if (vx[i]) lx[i] -= hv;
            REP(i,n) if (vy[i]) ly[i] += hv;
        }
    }
    ll re= 0;
    REP(i,n) re += g[my[i]][i];
    return re;
}
```

### 5.2 mcmf

```
struct MCMF{
    int n, s, t;
    struct Edge{
        int to, rev;
        ll cost, cap, flow=0; // Can have negative
    flow!!!!!
        Edge(int to, int rev, ll cost, ll cap): to(
    to), rev(rev), cost(cost), cap(cap) {}
    };
    vector<int> par, id;
    vector<ll> dist;
    vector<vector<Edge> > g;
    MCMF(int n,int s,int t): n(n), s(s), t(t){
        par.resize(n); id.resize(n); dist.resize(n,
    inf);
        g.resize(n);
    }
    void add(int v, int u, ll f, ll c){
        g[v].pb({u,SZ(g[u]),c,f});
        g[u].pb({v,SZ(g[v])-1,-c,0});
    }
    bool spfa(){ // SPFA
        queue<int> q ({s});
        vector<int> vis(n,0);
        fill(ALL(dist), inf); dist[s] = 0;
        while (!q.empty()){
            int v = q.front(); q.pop();
            vis[v] = 0;
            for (int i = 0; i<SZ(g[v]); i++){
                Edge &e = g[v][i];
                if (e.cap - e.flow==0) continue;
                if (dist[e.to] > dist[v] + e.cost){
                    dist[e.to] = dist[v] + e.cost;
                    par[e.to] = v; id[e.to] = i;
                    if (!vis[e.to]){
                        q.push(e.to); vis[e.to] = 1;
                    }
                }
            }
        }
        return dist[t] != inf;
    }
    pair<ll, ll> mf(){
        pair<ll, ll> re = {0,0};
        while (spfa()){
            ll famt = inf;
            for (int v = t; v!=s; v = par[v]){
                Edge &e = g[par[v]][id[v]];
                MN(famt, e.cap - e.flow);
            }
            for (int v = t; v!=s; v = par[v]){
                Edge &e = g[par[v]][id[v]];
                e.flow += famt;
                g[e.to][e.rev].flow -= famt;
            }
            re.f += famt;
            re.s += dist[t] * famt;
        }
        return re;
```

```
    }
};
```

### 5.3 bipartite matching

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

const int MAXN = 1003;
int mx[MAXN],my[MAXN];
bool vy[MAXN];
vector<int> edge[MAXN];

int n, m;
int greedy_matching()
{
    int c = 0;
    for (int x=1; x<=n; ++x) {
        if (mx[x] == -1) {
            for (auto y : edge[x]) {
                if (my[y] == -1) {
                        mx[x] = y; my[y] = x;
                        c++;
                        break;
                }
            }
        }
    }
    return c;
}

bool DFS(int x)
{
    for (auto y : edge[x]) {
        if (!vy[y]) {
            vy[y] = true;
            if (my[y] == -1 || DFS(my[y]))
            {
                mx[x] = y; my[y] = x;
                return true;
            }
        }
    }
    return false;
}

int bipartite_matching()
{
    memset(mx, -1, sizeof(mx));
    memset(my, -1, sizeof(my));

    int c = greedy_matching();

    for (int x=1; x<=n; ++x)
        if (mx[x] == -1)
        {
            memset(vy, false, sizeof(vy));
            if (DFS(x)) c++;
        }
    return c;
}


int main () {
    cin >> n >> m;
    int ecnt;
    cin >> ecnt;
    while (ecnt--) {
        int f,t;
        cin >> f >> t;
        edge[f].emplace_back(t);
    }

    cout << bipartite_matching() << endl;
}
```

### 5.4 matching

### 5.5 dinic

```
struct Dinic{
    struct Edge{
        int to, rev; ll cap, flow=0;
        Edge(int to,int rev, ll cap) : to(to), rev(
    rev), cap(cap) {}
    };

    vector<vector<Edge> > g;
    int n;
    int s, t;
    vector<int> level, ptr;
    Dinic(int n, int s, int t):n(n),s(s),t(t){
        level.resize(n,-1); ptr.resize(n); g.resize(
    n);
    }
    void add(int v, int u, ll cap){
        g[v].pb({u,SZ(g[u]),cap});
        g[u].pb({v,SZ(g[v])-1,0});
    }
    bool bfs(){ // Build layers with edges on the
    residual graph that aren't full
        queue<int> q({s});
        level[s] = 0;
        while (!q.empty() && level[t] == -1){
            int v = q.front(); q.pop();
            for (auto &e : g[v]){
                if (e.cap - e.flow ==0) continue;
                int u = e.to;
                if (level[u]==-1){
                    level[u] = level[v]+1; q.push(u)
    ;
                }
            }
        } return level[t]!=-1;
    }
    ll dfs(int v, ll amt){ // Returns flow amount of
     any flow on bfs graph
        if (amt == 0 || v==t) return amt;
        for (; ptr[v] <SZ(g[v]); ptr[v]++){
            Edge &e = g[v][ptr[v]];
            int u = e.to;
            if (level[u] == level[v]+1){
                ll tt = dfs(u,min(amt, e.cap - e.
    flow));
                if (tt==0) continue;
                e.flow+=tt; g[e.to][e.rev].flow-=tt;
     return tt;
            }
        } return 0;
    }
    ll mf(){
        ll re = 0;
        while (bfs()){
            while (ll amt = dfs(s,inf)) re += amt;
    // Basically ford fulkerson, but on layered
    graph
            fill(ALL(level), -1); fill(ALL(ptr), 0);
        } return re;
    }
};

signed main(){
    int n = 100;
    int N = n+5; int s = N-1, t = N-2;
    Dinic dd (N,s,t);
    int mf = dd.mf();
}
```

## 6 DataStructure

### 6.1 zkw tree

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

const int MAXN = 100005;
```

```
int n, zkw[MAXN*2];

/*
    query: range max
    add: single change value
*/
void build () {
    for (int i=n-1; i>0; i--) {
        zkw[i] = max(zkw[i<<1], zkw[i<<1|1]);
    }
}

void chg (int x, int val) {
    for (zkw[x+=n]=val; x>1; x>>=1) {
        zkw[x>>1] = max(zkw[x], zkw[x^1]);
    }
}

int qry (int l, int r) {
    int ret = -0x3f3f3f3f;
    for (l+=n,r+=n; l<r; l>>=1, r>>=1) {
        if (l&1) {
            ret = max(ret, zkw[l++]);
        }
        if (r&1) {
            ret = max(ret, zkw[--r]);
        }
    }
    return ret;
}

int main () {
    cin >> n;
    for (int i=0; i<n; i++) {
        cin >> zkw[i+n];
    }

    build();
    int cmd;
    while (cin >> cmd) {
        int l, r, x, v;
        if (cmd == 1) {
            cin >> l >> r;
            cout << qry(l, r) << endl;
        } else {
            cin >> x >> v;
            chg(x, v);
        }
    }
}
```

## 6.2 segment tree dynamic

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

int main () {

}
```

## 6.3 2Dstructure

## 6.4 segment tree array

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
#define REP(i, n) for(int i=0; i<n;i++)

const int MAXN = 100005;

int n, m, a[MAXN], len[MAXN*4], dt[MAXN*4], tag[MAXN
    *4];

void push (int o) {
    if (len[o] > 1 && tag[o] != 0) {
        tag[o<<1] += tag[o];
        tag[o<<1|1] += tag[o];
        dt[o] += tag[o] * len[o];
        tag[o] = 0;
    }
}

ll sum (int o) {
    return tag[o]*len[o] + dt[o];
}

void pull (int o) {
    dt[o] = sum(o<<1) + sum(o<<1|1);
}

void build (int o=1, int l=0, int r=n) {
    if (l == r - 1) {
        dt[o] = tag[o] = 0;
        len[o] = 1;
    } else {
        int mid = (l + r) >> 1;
        build(o<<1, l, mid);
        build(o<<1|1, mid, r);
        len[o] = len[o<<1] + len[o<<1|1];
        pull(o);
    }
}


ll query(int qL, int qR, int o=1, int nL=0, int nR=n
    ) {
    if (qR <= nL || qL >= nR || qL >= qR) {
        return 0;
    } else if (nL >= qL && nR <= qR) {
        return sum(o);
    } else {
        push(o);
        int mid = (nL + nR) >> 1;
        return query(qL, qR, o<<1, nL, mid) + query(
    qL, qR, o<<1|1, mid, nR);
    }
}

void modify(int qL, int qR, int val, int o=1, int nL
    =0, int nR=n) {
    if (qR <= nL || qL >= nR || qL >= qR) {
        return;
    } else if (nL >= qL && nR <= qR) {
        tag[o] += val;
    } else {
        push(o);
        int mid = (nL + nR) >> 1;
        modify(qL, qR, val, o<<1, nL, mid);
        modify(qL, qR, val, o<<1|1, mid, nR);
        pull(o);
    }
}

int main () {
    cin >> n;
    build();
    int cmd;
    while (cin >> cmd) {
        int l, r, v;
        if (cmd == 1) {
            cin >> l >> r >> v;
            modify(l, r, v);
        } else {
            cin >> l >> r;
            cout << query(l, r) << endl;
        }
    }
}

/*
10
1 0 3 3
0 0 5
1 2 4 2
0 0 5
```

```
*/
```

## 6.5 treap

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

const ll ra = 880301,rb = 53, rm = 20020607;
ll rn = 97;
int random () {
    return rn = (rn*ra+rb) % rm;
}


struct Node {
    Node *l,*r;
    ll key,val,tag;
    int sz,pri;
    Node (ll k,ll v) {
        l = r = 0;
        pri = random();
        key = k;
        tag = val = v;
        sz = 1;
    }

    void pull() {
        sz = 1;
        tag = val;
        if (l) {
            tag = max(tag,l->tag);
            sz += l->sz;
        }
        if (r) {
            tag = max(tag,r->tag);
            sz += r->sz;
        }
    }
};
Node *root;

int SIZ(Node *nd) {
    return nd ? nd->sz : 0;
}

Node *mrg(Node *a,Node *b) {
    if (!a || !b) {
        return a ? a : b;
    }
    if (a->pri > b->pri) {
        a->r = mrg(a->r,b);
        a->pull();
        return a;
    } else {
        b->l = mrg(a,b->l);
        b->pull();
        return b;
    }
}

// max a key <= key
void split_key(Node *o,Node *&a,Node *&b,ll key) {
    if (!o) {
        a = b = 0;
        return;
    }
    if (o->key <= key) {
        a = o;
        split_key(o->r,a->r,b,key);
        a->pull();
    } else {
        b = o;
        split_key(o->l,a,b->l,key);
        b->pull();
    }
}

// size of a equals sz
void split_sz(Node *o,Node *&a,Node *&b,ll sz) {
    if (!o) {
        a = b = 0;
        return;
    }
    if (SIZ(o->l)+1 <= sz) {
        a = o;
        split_sz(o->r,a->r,b,sz-SIZ(o->l)-1);
        a->pull();
    } else {
        b = o;
        split_sz(o->l,a,b->l,sz);
        b->pull();
    }
}

void ins(ll key,ll val) {
    Node *nw = new Node(key,val);
    if (!root) {
        root = nw;
    } else {
        Node *l,*r;
        split_key(root,l,r,key);
        root = mrg(l,mrg(nw,r));
    }
}

// static rmq on treap lol
ll query(ll l,ll r) {
    Node *a,*b,*c;
    split_sz(root,a,b,l-1);
    split_sz(b,b,c,r-l+1);
    ll ret = b->tag;
    root = mrg(a,mrg(b,c));
    return ret;
}
```

## 6.6 sparse table