Team Note of 541

Compiled on October 31, 2025

Contents

ALL BELOW HERE ARE USELESS IF YOU READ THE STATEMENT WRONG

1 자료구조

1.1 레이지 펜윅

```
const int TSIZE = 100000;
int a_tr[TSIZE + 1]; //일차항 계수
int b_tr[TSIZE + 1]; //상수항
int sum(int tr[], int i) {
  int ans = 0;
  while (i > 0) {
    ans += tr[i];
    i -= (i & -i);
  }
  return ans;
}
void update(int tr[], int i, int val) {
  while (i <= TSIZE) {
    tr[i] += val;
    i += (i&-i);
  }
}
// Range update [L, R]
void range_update(int L, int R, int val){
  update(a_tr, L, val); update(a_tr, R + 1, -val); //일차항 계수 update
  update(b_tr, L, (-L + 1)*val); update(b_tr, R + 1, R*val); //상수항 update
}
// Range query [L, R]
int range_query(int L, int R){
  int res = 0;
  res += sum(a_tr, R)*R + sum(b_tr, R); //[1 , R]
  res -= sum(a_tr, L - 1)*(L - 1) + sum(b_tr, L - 1); //[1, L-1]
  return res;
}
```

1.2 HLD

```
int pv;
void dfs1(int v=1){
    sz[v]=1;
    for(auto &i:arr[v]){
        if(par[v]==i) continue;
        dep[i]=dep[v]+1;
        par[i]=v;
        dfs1(i);
        sz[v]+=sz[i];
        if(sz[i]>sz[arr[v][0]]||par[v]==arr[v][0])swap(i,arr[v][0]);
    }
}
void dfs2(int v=1){
    in[v]=++pv;num[pv]=v;bot[top[v]]=v;
    for(auto i:arr[v]){
        if(in[i]) continue;
        top[i]=i==arr[v][0]?top[v]:i;
        dfs2(i);
    }out[v]=pv;
}
void que(int a){
    int b=1;
    while(top[a] != top[b]){
        if(dep[top[a]] < dep[top[b]]) swap(a, b);
        int st = top[a];
        query(1,1,pv,in[st],in[a]);
        a = par[st];
    }if(dep[a] < dep[b]) swap(a, b);
    query(1,1,pv,in[b],in[a]);
}
```

1.3 suffixAutomata

```
struct sufAuto{
    struct node{
        int slink,len,slen;
        int nxt[26];
    };int pv,now;
    node vec[2'000'010];
    void init(){
        vec[0]={-1,0,{}};
    }
    void add(int a){
        int w=++pv;
        nd.push_back(w);
        vec[w].len=vec[now].len+1;
        int p=now;
        now=w;
        while(p!=-1&&!vec[p].nxt[a]){
            vec[p].nxt[a]=w;
            p=vec[p].slink;
        }if(p!=-1){
            int pre=vec[p].nxt[a];
            int upd=vec[p].nxt[a];
            if(vec[pre].len>vec[p].len+1){
                upd=++pv;
                vec[upd]=vec[pre];
                vec[upd].len=vec[p].len+1;

                vec[pre].slink=upd;
                while(p!=-1&&vec[p].nxt[a]==pre){
                    vec[p].nxt[a]=upd;
                    p=vec[p].slink;
                }
            }vec[now].slink=upd;}}}am;
```

1.4 Sparse Table

```
auto get_min = [&](int s, int e) {
   int lg = log[e-s+1];
   return min(sparse_table[lg][s],sparse_table[lg][e-(1<<lg)+1]);
}
```

1.5 Persistent Segment Tree

```
struct node{
    int l,r,v;
    node(){ l = r = v = 0; }
};
node pst[32'400'000];//
int sz=3;
void update(int l,int r,int s,int e,int idx){
    if(!pst[r].v)pst[r].v=pst[l].v;
    pst[r].v++;
    if(s==e)return;
    int m=s+e>>1;
    if (idx <= m) {
        if (!pst[r].l|| pst[r].l == pst[l].l)
            pst[r].l = ++sz;
        if (!pst[r].r) pst[r].r = pst[l].r;
        update(pst[l].l, pst[r].l, s, m,idx);
    } else {
        if (!pst[r].r|| pst[r].r == pst[l].r)
            pst[r].r = ++sz;
        if (!pst[r].l) pst[r].l = pst[l].l;
        update(pst[l].r, pst[r].r, m + 1, e, idx);
    }}
int query(int sk,int ek, int s,int e,int l,int r){
    if(s>r||e<l)return 0;
    if(s>=l&&e<=r) return pst[ek].v-pst[sk].v;
    int m=s+e>>1;
    return query(pst[sk].l,pst[ek].l,s,m,l,r)+query(pst[sk].r,pst[ek].r,m+1,e,l,r);
}
```

1.6 Link Cut Tree

```
struct node{
    node *l,*r,*p;
    int key=0;
    node* mx;
    bool rev;
    int a,b;
    node(int k,int aa=0,int bb=0){
        l=r=p= nullptr;key=k;rev=false;mx=this;a=aa;b=bb;
    }
};int inf=-1e9;
bool isRoot(node * x) {
    return (!x->p || (x->p->l != x && x->p->r != x));
}
node* max(node* a,node* b){
    if(a->key>b->key)return a;
    return b;
}
void update(node* x){
    x->mx=x;
    if(x->l)x->mx=max(x->mx,x->l->mx);
    if(x->r)x->mx=max(x->mx,x->r->mx);
}
void lazy_up(node* x){
    if(x->rev){
        if(x->l)x->l->rev^=1,swap(x->l->l,x->l->r);
        if(x->r)x->r->rev^=1,swap(x->r->l,x->r->r);
        x->rev=false;
    }update(x);
}
void rotate(node* x){
    node* p=x->p;
    lazy_up(p);lazy_up(x);
    update(x);update(p);
    if(x==p->l){
        p->l=x->r;
        if(p->l)p->l->p=p;
        x->r=p;
    }else{
        p->r=x->l;
        if(p->r)p->r->p=p;
        x->l=p;
    }
    x->p=p->p;
    p->p=x;
    lazy_up(p);lazy_up(x);
    update(x);update(p);
    if(x->p){
        if(p==x->p->l)x->p->l=x;
        else if(p==x->p->r)x->p->r=x;
    }
    lazy_up(p);lazy_up(x);
    update(x);update(p);
}
void splay(node* x){///thinking
    while(!isRoot(x)){
        node* p=x->p;
        if (!isRoot(p)) lazy_up(p->p);
        lazy_up(p);
        lazy_up(x);
        update(x);update(p);
        if(!isRoot(p)){
            if((x==p->l)^(p==p->p->l))rotate(x);
            else rotate(p);
        }
        rotate(x);
    }
    lazy_up(x);update(x);
}
node* access(node* x){//////////thinking
    lazy_up(x);update(x);
    splay(x);
    lazy_up(x);update(x);
    x->r= nullptr;
    node* res=x;
    while(x->p){
        node* p=x->p;
        res=p;
        splay(p);
        p->r=x;
        splay(x);
    }
    lazy_up(x);update(x);
    return res;
}
void makeRoot(node * x) {
    access(x);
    swap(x->l,x->r);
    x->rev = true;
    lazy_up(x);
}
void link(node* x,node* y,int c,int a,int b){
    makeRoot(x);access(y);
    y->r=new node(c,a,b);
    y->r->p=y;
    y->r->r=x;
    x->p=y->r;
    update(y->r);
    update(y);
    access(y);
}
node* lca(node* x,node* y){
    access(x);
    return access(y);
}
```

1.7 Splay Tree

```
#include <iostream>
using namespace std;
struct node{
    node* l;
    node* r;
    node* p;
    long long key;//l<r
    long long cnt;
    bool flip;
    node(long long k,node* pp){
        p=pp;
        flip=false;
        l=r= nullptr;
        cnt=1;
    }
}* tree;
void lazy_up(node* x){
    if(!x->flip)return;
    swap(x->l,x->r);
    if(x->l)x->l->flip^=1;
    if(x->r)x->r->flip^=1;
    x->flip=false;
}
void update(node *x){
    lazy_up(x);
    x->cnt=1;
    if(x->l)x->cnt+=x->l->cnt;
    if(x->r)x->cnt+=x->r->cnt;
}
void rotate(node* x){//x to be parent
    node* p=x->p;
    node* b=nullptr;
    lazy_up(p);lazy_up(x);
    if(!p)return;
    if(x==p->l) p->l=b=x->r,x->r=p;
    else p->r=b=x->l,x->l=p;
    x->p=p->p;p->p=x;
    lazy_up(p);lazy_up(x);
    if(b) b->p=p;
    if(x->p){
        if(x->p->l==p)
            x->p->l=x;
        else x->p->r=x;
    }else tree=x;
    lazy_up(p);lazy_up(x);
    update(p);update(x);
}
void splay(node *x,node *g=nullptr){//x to be g's child
    while(x->p!=g){
        node* p=x->p;
        if(p->p==g){rotate(x);break;}
        node* pp = p->p;
        if((p->l==x)^(pp->l==p)) {rotate(x);rotate(x);}
        else {rotate(p);rotate(x);}
    }if(!g)tree=x;
}
void kth(long long k){
    node* x=tree;
    lazy_up(x);
    while(1){
        while(x->l&&x->l->cnt>k)x=x->l,lazy_up(x);
        if(x->l)k-=x->l->cnt;
        if(!k--)break;
        x=x->r;
        lazy_up(x);
    }
    splay(x);
}
node* gather(int s,int e){
    kth(e+1);
    node* temp=tree;
    kth(s-1);
    splay(temp,tree);
    return tree->r->l;
}
void flip(int s,int e){
    node* x=gather(s,e);
    x->flip^=1;
    update(x);
}
void insert(int key){//insert
    node* p=tree;
    node** pp;
    if(!p){
        tree=new node(key, nullptr);
        return;
    }
    while(1){
        if(key==p->key)return;
        if(key < p->key){
            if(!(p->l)){pp=&p->l;break;}
            p=p->l;
        }else{
            if(!(p->r)){pp=&p->r;break;}
            p=p->r;
        }
    }
    node* x=new node(key,p);
    *pp=x;
    splay(x);
}
bool find(int key){//find
    node* p=tree;
    if(!p)return false;
    while(p){
        if(key==p->key)break;
        if(key< p->key){
            if(!(p->l))break;
            p=p->l;
        }else{
            if(!(p->r))break;
            p=p->r;
        }
    }
    splay(p);
    return key==p->key;
}
void remove(int key){//remove
    if(!find(key))return;
    node* p=tree;
    if(p->l&&p->r){
        tree=p->l;tree->p= nullptr;
        node* x=tree;
        while(x->r)x=x->r;
        x->r=p->r;p->r->p=x;
        delete p;return;
    }if(p->l){
        tree=p->l;tree->p= nullptr;
        delete p;return;
    }if(p->r){
        tree=p->r;tree->p=nullptr;
        delete p;return;
    }
    delete p;tree= nullptr;
}
void insertKth(long long key,int k){//insert
    kth(k);
    node* p=tree;
    if(!p->l) {
        tree->l=new node(key,p);
        splay(tree->l);
        return;
    }
    tree->l->p=new node(key,nullptr);
    tree->l->p->l=tree->l;
    tree->l->p->r=tree;
    tree->p=tree->l->p;
    tree->l= nullptr;
    tree=tree->p;
}
void removeKth(int k){//remove
    kth(k);
    node* p=tree;
    if(p->l&&p->r){
        tree=p->l;tree->p= nullptr;
        node* x=tree;
        while(x->r){
//            lazy_up(x);
            update(x);
            x=x->r;
        }//lazy_up(x);
        update(x);
        x->r=p->r;p->r->p=x;
        update(x);
        splay(x->r);
        return;
    }if(p->l){
        tree=p->l;tree->p= nullptr;
        return;
    }if(p->r){
        tree=p->r;tree->p=nullptr;
        return;
    }tree= nullptr;
}
void shift(int s, int e, int y){
    node *x=gather(s,e);
    if(y>=0){
        y%=e-s+1;
        if(!y)return;
        flip(s,e);
        flip(s,s+y-1);
        flip(s+y,e);
    }else{
        y=abs(y);
        y%=(e-s+1);
        if(!y)return;
        flip(s,e);
        flip(s,e-y);
        flip(e-y+1,e);
    }
}
node* ptr[100010];
int arr[100010];
void init(int n){
    if(tree) delete tree;
    node* now=new node(1, nullptr);
    tree=now;
    for(int i=1; i<=n; i++){
        ptr[i] =  now->r = new node(arr[i],now);
        now = now->r;
    }
    now->r=new node(1, now);
    for(int i=n; i>=1; i--) update(ptr[i]);
    splay(ptr[n/2+1]);
}
```

1.8 Centroid

```
#include <iostream>
#include <vector>
#include <map>
using namespace std;
const int maxv=100'101;
vector<int> arr[maxv];
int k,ans=1e9,discon[maxv]{},sz[maxv];
int cc[maxv];
vector<int> v;
int pre(int node,int par){
    sz[node]=1;
    for(auto i:arr[node]){
        if(i==par||discon[i])continue;
        sz[node]+=pre(i,node);
    }
    return sz[node];
}
int getCentroid(int node,int ns,int par){
    for(auto i:arr[node]){
        if(i==par||discon[i]) continue;
        if(sz[i]>ns)return getCentroid(i,ns,node);
    }
    return node;
}
int num[100'010];
void check(int node,int par,int dep){
    ans=min(ans,cc[num[node]]+dep);
    for(auto i:arr[node]){
        if(discon[i]||i==par)continue;
        check(i,node,dep+1);
    }
}
void update(int node,int par,int dep){
    cc[num[node]]=min(cc[num[node]],dep);
    v.push_back(num[node]);
    for(auto i:arr[node]){
        if(discon[i]||i==par)continue;
        update(i,node,dep+1);
    }
}
void solve(int node){
    int ns=pre(node,-1)/2;
    int centroid=getCentroid(node,ns,-1);
    discon[centroid]=1;
    for(auto i:v){
        cc[i]=1e8;
    }
    v.clear();
    v.push_back(num[centroid]);
    cc[num[centroid]]=0;
    for(auto i:arr[centroid]){
        if(discon[i]) continue;
        check(i,centroid,1);
        update(i,centroid,1);
    }
    for(auto i:arr[centroid]){
        if(discon[i]) continue;
        solve(i);
    }
}
int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int n;
    cin>>n;
    int a,b,c;
    for(int i=1;i<=n;i++)cin>>num[i],cc[i]=1e8;
    for(int i=1;i<n;i++){
        cin>>a>>b;
        arr[a].emplace_back(b);
        arr[b].emplace_back(a);
    }
    solve(1);
    cout<<ans;
    return 0;
}
```

1.9 Van Emde Boas Tree

```
#include<bits/stdc++.h>
using namespace std;
inline int read(){
    int x=0,f=0;char ch=getchar();
    while(!isdigit(ch))f|=ch=='-',ch=getchar();
    while(isdigit(ch))x=10*x+ch-'0',ch=getchar();
    return f?-x:x;
}
template<typename T>void print(T x){
    if(x<0)putchar('-'),x=-x;
    if(x>=10)print(x/10);
    putchar(x%10+'0');
}
template<typename T>void print(T x,char ch){
    print(x),putchar(ch);
}
const int N = 500005;
namespace vEB_tree_impl{// Author: wlzhouzhuan
    using u64 = uint64_t;
    static constexpr unsigned int lgW = 6;
    static constexpr unsigned int W = 1u << lgW;
    static constexpr int inf = 1 << 30;
    inline int ctz(u64 n) { return n ? __builtin_ctzll(n) : -1; }
    inline int clz(u64 n) { return n ? 63 - __builtin_clzll(n) : -1; }
    template <int LOG, class D = void>
    struct vEB_tree_node{
        using Chd = vEB_tree_node<(LOG >> 1)>;
        Chd map;
        int mn, mx;
        static constexpr int shift = (LOG >> 1) * lgW;
        array<Chd, 1 << shift> chd;
        inline int mask(u64 key) const { return key & ((1 << shift) - 1); }
        constexpr vEB_tree_node() : mn(inf), mx(-1) {}
        void insert(int key){
            mn = std::min(mn, key), mx = std::max(mx, key);
            int pos = key >> shift;
            if (chd[pos].empty())map.insert(pos);
            chd[pos].insert(mask(key));
        }
        void erase(int key){
            int pos = key >> shift;
            if (chd[pos].empty())return;
            chd[pos].erase(mask(key));
            if (chd[pos].empty())map.erase(pos);
            if (mn == key){
                if (mx == key)mn = inf, mx = -1;
                else{
                    int p = map.min();
                    mn = (p << shift) + chd[p].min();
                }
            }
            else if (mx == key){
                int p = map.max();
                mx = (p << shift) + chd[p].max();
            }
        }
        bool contain(int key) const{
            int pos = key >> shift;
            return chd[pos].contain(mask(key));
        }
        inline bool empty() const { return mx == -1; }
        inline int min() const { return mn == inf ? -1 : mn; }
        inline int max() const { return mx; }
        int find_next(int key) const{
            if (key <= min())return min();
            if (max() < key)return -1;
            int pos = key >> shift;
            if (map.contain(pos) && mask(key) <= chd[pos].max())
                return (pos << shift) + chd[pos].find_next(mask(key));
            int nxt = map.find_next(pos + 1);
            if (nxt == -1)return -1;
            return (nxt << shift) + chd[nxt].min();
        }
        int find_prev(int key) const{
            if (max() < key)return max();
            if (key <= min())return -1;
            int pos = key >> shift;
            if (map.contain(pos) && chd[pos].min() < mask(key))
                return (pos << shift) + chd[pos].find_prev(mask(key));
            int nxt = map.find_prev(pos);
            if (nxt == -1)return -1;
            return (nxt << shift) + chd[nxt].max();
        }int suc(int key) const { return find_next(key + 1); }        // > key
        int suc_or_equ(int key) const { return find_next(key); }     // >= key
        int pre(int key) const { return find_prev(key); }            // < key
        int pre_or_equ(int key) const { return find_prev(key + 1); } // <= key
    };template <int LOG>
    struct vEB_tree_node<LOG, typename std::enable_if<LOG == 1>::type>{
        u64 map;
        vEB_tree_node() : map(0) {}
        inline void insert(int key) { map |= 1ULL << key; }
        inline void erase(int key) { map &= ~(1ULL << key); }
        inline bool contain(int key) const { return (map >> key) & 1; }
        inline bool empty() const { return map == 0; }
        inline int min() const { return ctz(map); }
        inline int max() const { return clz(map); }
        int find_next(int key) const { return ctz(map & ~((1ULL << key) - 1)); }
        int find_prev(int key) const { return clz(map & ((1ULL << key) - 1)); }
    };
} // namespace vEB_tree_impl
using van_Emde_Boas_tree = vEB_tree_impl::vEB_tree_node<4>;
van_Emde_Boas_tree vEB;
////vEB.insert,erase,pre,suc...
```

1.10 non recursive segment tree

```
long long t[MAX_N*2];/////0-base
void init(){
    for(int i=n-1;i>0;i--){
        t[i]=t[i<<1]+t[i<<1|1];
    }
}
ll query(int l,int r){///[l,r)
    ll ans=0;
    for(l+=n,r+=n;l<r;l>>=1,r>>=1){
        if(l&1)ans+=t[l++];
        if(r&1)ans+=t[--r];
    }
    return ans;
}
void update(int pos, ll val) {
    t[pos+n]=val;
    for (pos+=n;pos>1;pos>>=1) {
        t[pos>>1]=t[pos]+t[pos^1];
    }
}
```

1.11 EXT

```
#include <ext/rope>
using namespace __gnu_cxx;
crope rp;
rp[ver]=crope(e.c_str());
rope<int> rp;
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
typedef tree<int, null_type, less_equal<int>, rb_tree_tag,
        tree_order_statistics_node_update> pds;
pds arr;
arr.order_of_key(a);////lower_bound
```

2 Graph

2.1 SCC

```
vector<bool> vis(n);
stack<int> st;
function<void(int, int)> dfs = [&](int v, int p) {
  vis[v] = 1;
  for(int nxt : G[v]) {
    if(nxt == p || vis[nxt])
      continue;
    dfs(nxt, v);
  }
  st.push(v);
};
for(int i=0 ; i<n ; i++) {
  if(vis[i])
    continue;
  dfs(i,-1);
}
vis = vector<bool>(n);
vector<vector<int>> scc;
int cnt = 0;
function<void(int, int)> dfs2 = [&](int v, int p) {
  vis[v] = 1;
  scc[cnt].push_back(v);
  for(int nxt : rG[v]) {
    if(nxt == p || vis[nxt])
```

```
      continue;
    dfs2(nxt, v);
  }
};
for(int i=0 ; i<n ; i++) {
  int cur = st.top(); st.pop();
  if(vis[cur])
    continue;
  scc.push_back(vector<int>{});
  dfs2(cur, -1);
  cnt ++;
}
```

2.2 2-SAT

```
auto negate = [](int x) {
  if(x & 1) return x - 1;
  else return x + 1;
};
vector<vector<int>> G(2*n), rG(2*n);
auto add_edge = [&](int u, int v, bool not1, bool not2) {
  u *= 2; v *= 2;
  if(not1) u = negate(u); if(not2) v = negate(v);
  G[negate(u)].push_back(v);
  G[negate(v)].push_back(u);
  rG[v].push_back(negate(u));
  rG[u].push_back(negate(v));
};
// ADD EDGES HERE
vector<bool> vis(2 * n);
stack<int> st;
function<void(int)> dfs = [&](int v) {
  vis[v] = 1;
  for(auto nxt : G[v]) {
    if(vis[nxt]) continue;
    dfs(nxt);
  }
  st.push(v);
};
for(int i=0 ; i<2*n ; i++) {
  if(vis[i]) continue;
  dfs(i);
}
vis = vector<bool>(2 * n);
vector<vector<int>> scc;
vector<int> num(2 * n);
int cnt = 0;
function<void(int)> dfs2 = [&](int v) {
  vis[v] = 1;
  num[v] = cnt;
  scc[cnt].push_back(v);
  for(auto nxt : rG[v]) {
    if(vis[nxt]) continue;
    dfs2(nxt);
  }
};
assert(sz(st) == 2 * n);
while(!st.empty()) {
  int x = st.top(); st.pop(); if(vis[x]) continue;
  scc.push_back(vector<int>{});
  dfs2(x); cnt += 1;
}
for(int i=0 ; i<n ; i++) {
  if(num[i*2] == num[i*2 + 1]) {
    cout << 0 << "\n";
    return 0;
  }
}
cout << 1 << "\n";
vector<int> res(2*n, -1);
for(int i=0 ; i<cnt ; i++) {
  for(auto x : scc[i]) {
    if(res[x] != -1) continue;
    res[x] = 0;
    res[negate(x)] = 1;
  }
}
for(int i=0 ; i<2*n ; i+=2)
  cout << res[i] << " ";
```

2.3 BCC

```
const int MAXN = 100;
vector<pair<int, int>> graph[MAXN];  // { next vertex id, edge id }
int up[MAXN], visit[MAXN], vtime;
vector<int> stk;
int is_cut[MAXN];             // v is cut vertex if  is_cut[v] > 0
vector<int> bridge;           // list of edge ids
vector<int> bcc_edges[MAXN];  // list of edge ids in a bcc
int bcc_cnt;
void dfs(int nod, int par_edge) {
    up[nod] = visit[nod] = ++vtime;
    int child = 0;
    for (const auto& e : graph[nod]) {
        int next = e.first, eid = e.second;
        if (eid == par_edge) continue;
        if (visit[next] == 0) {
            stk.push_back(eid);
            ++child;
            dfs(next, eid);
            if (up[next] == visit[next]) bridge.push_back(eid);
            if (up[next] >= visit[nod]) {
                ++bcc_cnt;
                do {
                    auto lasteid = stk.back();
                    stk.pop_back();
                    bcc_edges[bcc_cnt].push_back(lasteid);
                    if (lasteid == eid) break;
                } while (!stk.empty());
                is_cut[nod]++;
            }
            up[nod] = min(up[nod], up[next]);
        }
        else if (visit[next] < visit[nod]) {
            stk.push_back(eid);
            up[nod] = min(up[nod], visit[next]);
        }
    }
    if (par_edge == -1 && is_cut[nod] == 1)
        is_cut[nod] = 0;
}
// find BCCs & cut vertexs & bridges in undirected graph
// O(V+E)
void get_bcc() {
    vtime = 0;
    memset(visit, 0, sizeof(visit));
    memset(is_cut, 0, sizeof(is_cut));
    bridge.clear();
    for (int i = 0; i < n; ++i) bcc_edges[i].clear();
    bcc_cnt = 0;
    for (int i = 0; i < n; ++i) {
        if (visit[i] == 0)
            dfs(i, -1);
    }
}
```

2.4 오일러 경로

```
int n, m;
vector<vector<int>> g;
struct edge {
  int u, v;
  bool visit;
  int id;
};
vector<edge> a;
vector<int> ans;
void dfs(int v) {
  while((g[v].size())) {
    int nxt = g[v].back();
    if(a[nxt].visit == 0) {
      a[nxt].visit = 1;
      dfs(a[nxt].u+a[nxt].v-v);
    } else {
      g[v].pop_back();
    }
  }
}
void solve() {
  for(int i=0 ; i<m ; i++) {
    int u, v; cin >> u >> v; u--, v--;
    a[i] = {u,v,0,i};
    g[u].push_back(i);
    g[v].push_back(i);
  }
  dfs(0);
}
```

2.5 Dominator

```
vector<pi> g[200'010];
vector<int> inv[200'010];
vector<int> outv[200'010];
ll dp[200'010];
void bfs(){
    priority_queue<pl> pq;
    pq.push({-1,1});
    dp[1]=1;
    while(!pq.empty()){
        int s=pq.top().second;
        ll w=-pq.top().first;
        pq.pop();
        if(dp[s]!=w)continue;
        for(auto i:g[s]){
            if(!dp[i.first]||dp[i.first]>w+i.second){
                dp[i.first]=w+i.second;
                pq.push({-dp[i.first],i.first});
                inv[i.first]={s};
            }else if(dp[i.first]==w+i.second){
                inv[i.first].push_back(s);
            }
        }
    }
}
int ind[200'010];
pi ret[200'010];
int par[200'010];
int spr[200'010][20];
int dep[200'010];
int lca(int a,int b){
    if(dep[a]<dep[b])swap(a,b);
    for(int i=19;i>=0;i--){
        if(dep[spr[a][i]]>=dep[b])
            a=spr[a][i];
    }if(a==b)return a;
    for(int i=19;i>=0;i--){
        if(spr[a][i]!=spr[b][i]){
            a=spr[a][i];
            b=spr[b][i];
        }
    }return spr[a][0];
}vector<int> v[200'010];
int sz[200'010];
int dfs(int n){
    sz[n]=1;
    for(auto i:v[n]){
        sz[n]+=dfs(i);
    }return sz[n];
}
void makeDomi(int n,int m){
    queue<int> q;
    for(int i=1;i<=n;i++){
        for(auto j:inv[i]){
            outv[j].push_back(i);
        }ind[i]=inv[i].size();
    }q.push(1);dep[1]=1;
    while(!q.empty()){
        int s=q.front();
        q.pop();
        for(auto i:outv[s]){
            ind[i]--;
            if(!par[i])par[i]=s;
            else par[i]=lca(par[i],s);
            if(!ind[i]){
                dep[i]=dep[par[i]]+1;
                v[par[i]].push_back(i);
                spr[i][0]=par[i];
                for(int j=1;j<20;j++)
                    spr[i][j]=spr[spr[i][j-1]][j-1];
                q.push(i);
            }
        }
    }dfs(1);
    for(int i=0;i<m;i++){
        if(dep[ret[i].first]<dep[ret[i].second])
            swap(ret[i].first,ret[i].second);
        if(inv[ret[i].first].size()==1&&
           inv[ret[i].first][0]==ret[i].second){
            cout<<sz[ret[i].first]<<'\n';
        }else cout<<"0\n";
    }
}
signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int n,m,a,b,c;
    cin>>n>>m;
    for(int i=0;i<m;i++){
        cin>>a>>b>>c;
        ret[i]={a,b};
        g[a].emplace_back(b,c);
        g[b].emplace_back(a,c);
    }bfs();
    makeDomi(n,m);
}
```

2.6 Dominator(koo)

```
vector<int> E[MAXN], RE[MAXN], rdom[MAXN];
int S[MAXN], RS[MAXN], cs;
int par[MAXN], val[MAXN], sdom[MAXN], rp[MAXN], dom[MAXN];
void clear(int n) {
  cs = 0;
  for(int i=0;i<=n;i++) {
    par[i] = val[i] = sdom[i] = rp[i] = dom[i] = S[i] = RS[i] = 0;
    E[i].clear(); RE[i].clear(); rdom[i].clear();
  }
}
void add_edge(int x, int y) { E[x].push_back(y); }
void Union(int x, int y) { par[x] = y; }
int Find(int x, int c = 0) {
  if(par[x] == x) return c ? -1 : x;
  int p = Find(par[x], 1);
  if(p == -1) return c ? par[x] : val[x];
  if(sdom[val[x]] > sdom[val[par[x]]]) val[x] = val[par[x]];
  par[x] = p;
  return c ? p : val[x];
}
void dfs(int x) {
  RS[ S[x] = ++cs ] = x;
  par[cs] = sdom[cs] = val[cs] = cs;
  for(int e : E[x]) {
    if(S[e] == 0) dfs(e), rp[S[e]] = S[x];
    RE[S[e]].push_back(S[x]);
  }
}
int solve(int s, int *up) { // Calculate idoms
  dfs(s);
  for(int i=cs;i;i--) {
    for(int e : RE[i]) sdom[i] = min(sdom[i], sdom[Find(e)]);
    if(i > 1) rdom[sdom[i]].push_back(i);
    for(int e : rdom[i]) {
      int p = Find(e);
      if(sdom[p] == i) dom[e] = i;
      else dom[e] = p;
    }
    if(i > 1) Union(i, rp[i]);
  }
  for(int i=2;i<=cs;i++) if(sdom[i] != dom[i]) dom[i] = dom[dom[i]];
  for(int i=2;i<=cs;i++) up[RS[i]] = RS[dom[i]];
  return cs;
}
```

3 Flow

3.1 Dinic

```
struct FlowEdge {
    int v, u;
    ll cap, flow = 0;
    FlowEdge(int v, int u, ll cap) : v(v), u(u), cap(cap) {}
};
struct Dinic {
    const ll flow_inf = 1e18;
    vector<FlowEdge> edges;
    vector<vector<int>> adj;
    int n, m = 0;
    int s, t;
    vector<int> level, ptr;
    queue<int> q;
    Dinic(int n, int s, int t) : n(n), s(s), t(t) {
        adj.resize(n);
        level.resize(n);
        ptr.resize(n);
    }
    void add_edge(int v, int u, ll cap) {
        edges.emplace_back(v, u, cap);
        edges.emplace_back(u, v, 0);
        adj[v].push_back(m);
        adj[u].push_back(m + 1);
        m += 2;
    }
    bool bfs() {
        while (!q.empty()) {
            int v = q.front();
            q.pop();
            for (int id : adj[v]) {
                if (edges[id].cap - edges[id].flow < 1)
                    continue;
                if (level[edges[id].u] != -1)
                    continue;
                level[edges[id].u] = level[v] + 1;
                q.push(edges[id].u);
            }
        }
        return level[t] != -1;
    }
    ll dfs(int v, ll pushed) {
        if (pushed == 0)
            return 0;
        if (v == t)
            return pushed;
        for (int& cid = ptr[v]; cid < (int)adj[v].size(); cid++) {
            int id = adj[v][cid];
            int u = edges[id].u;
            if (level[v] + 1 != level[u] || edges[id].cap - edges[id].flow < 1)
                continue;
            ll tr = dfs(u, min(pushed, edges[id].cap - edges[id].flow));
            if (tr == 0)
                continue;
            edges[id].flow += tr;
            edges[id ^ 1].flow -= tr;
            return tr;
        }
        return 0;
    }
    ll flow() {
        ll f = 0;
        while (true) {
            fill(level.begin(), level.end(), -1);
            level[s] = 0;
            q.push(s);
            if (!bfs())
                break;
            fill(ptr.begin(), ptr.end(), 0);
            while (ll pushed = dfs(s, flow_inf)) {
                f += pushed;
            }
        }
        return f;
    }
};
```

3.2 Dinic Style MCMF

```
int sv[40410]{};
vector<tu> v[40410];
//////node cap flow val
bool spfa(int s,int e){
    bool isIn[40410]{};
    queue<int> q;
    q.push(s);
    memset(sv,-1,sizeof sv);
    sv[s]=0;
    while(!q.empty()){
        int p=q.front();
        q.pop();
        isIn[p]=false;
        for(auto i:v[p]){
            if(i[1]-i[2]<=0)continue;
            if(sv[i[0]]==-1||sv[i[0]]<sv[p]+i[3]){
                sv[i[0]]=sv[p]+i[3];
                if(!isIn[i[0]]){
                    q.push(i[0]);
                    isIn[i[0]]=true;
                }
            }
        }
    }
    return sv[e] > 0;
}
int nv[40410];////start_i
bool vis[40410];
int dfs(int s,int e,int f){
    vis[s]=true;
    if(s==e)return f;
    for(int &i=nv[s];i<v[s].size();i++){
        tu &j=v[s][i];
        if(vis[j[0]]||sv[s]+j[3]!=sv[j[0]]||j[1]-j[2]<=0)continue;
        int w=dfs(j[0],e,min(f,j[1]-j[2]));
        if(w<=0)continue;
        j[2]+=w;
        v[j[0]][j[4]][2]-=w;
//        flow[s][j]+=w;
//        flow[j][s]-=w;
        return w;
    }return 0;
}bool upd(int s,int n){
    int mn=1e9;
    for(int i=s;i<=n;i++){
        if(!vis[i])continue;
        for(auto j:v[i]){
            if(j[1]-j[2]>0&&!vis[j[0]])mn=min(mn,-sv[i]-j[3]+sv[j[0]]);
        }
    }if(mn==1e9)return 0;
    for(int i=s;i<=n;i++)if(!vis[i])sv[i]-=mn;
    return true;
}
signed main(){
    spfa(s,e);
    do{
        if(sv[e]<0)break;
        int now=0;//flow
        memset(vis,0,sizeof vis);
        memset(nv,0,sizeof nv);
        while(now = dfs(s,e,1e9)){
            ans+=sv[e]*now;
//            cout<<ans<<'\n';
            memset(vis,0,sizeof vis);
        }
    }while(upd(m+1,n*m+m+2));
}
```

3.3 Johnson

```
typedef pair<int, int> p;
const int SZ = 888;
struct MCMF{
    int s, t; //source, sink
    struct Edge{ int v, c, d, dual; };
    vector<Edge> g[SZ];
    void addEdge(int s, int e, int c, int d){
        g[s].push_back({e, c, d, (int)g[e].size()});
        g[e].push_back({s, 0, -d, (int)g[s].size()-1});
    }
    int h[SZ], inq[SZ]; //johnson's algorithm, spfa
    int dst[SZ]; //dijkstra
    void init(int _s, int _t){
        s = _s, t = _t;
        memset(h, 0x3f, sizeof h);
        memset(dst, 0x3f, sizeof dst);
        //johnson's algorithm with spfa
        queue<int> q; q.push(s); inq[s] = 1;
        while(q.size()){
            int now = q.front(); q.pop(); inq[now] = 0;
            for(auto i : g[now]){
                if(i.c && h[i.v] > h[now] + i.d){
                    h[i.v] = h[now] + i.d;
                    if(!inq[i.v]) inq[i.v] = 1, q.push(i.v);
                }
            }
        }
        for(int i=0; i<SZ; i++){
            for(auto &j : g[i]) if(j.c) j.d += h[i] - h[j.v];
        }
        //get shortest path DAG with dijkstra
        priority_queue<p> pq; pq.emplace(0, s); dst[s] = 0;
        while(pq.size()){
            int now = pq.top().y;
            int cst = -pq.top().x;
            pq.pop();
            if(dst[now] - cst) continue;
            for(auto i : g[now]){
                if(i.c && dst[i.v] > dst[now] + i.d){
                    dst[i.v] = dst[now] + i.d;
                    pq.emplace(-dst[i.v], i.v);
                }
            }
        }
        for(int i=0; i<SZ; i++) dst[i] += h[t] - h[s];
    }
    int chk[SZ], work[SZ];
    bool update(){ //update shortest path DAG in O(V+E)
        int mn = 1e9;
        for(int i=0; i<SZ; i++){
            if(!chk[i]) continue;
            for(auto j : g[i]){
                if(j.c && !chk[j.v]) mn = min(mn, dst[i] + j.d - dst[j.v]);
            }
        }
        if(mn >= 1e9) return 0;
        for(int i=0; i<SZ; i++){
            if(!chk[i]) dst[i] += mn;
        }
        return 1;
    }
    int dfs(int now, int fl){
        chk[now] = 1;
        if(now == t) return fl;
        for(; work[now] < g[now].size(); work[now]++){
            auto &i = g[now][work[now]];
            if(!chk[i.v] && dst[i.v] == dst[now] + i.d && i.c){
                int ret = dfs(i.v, min(fl, i.c));
                if(ret){
                    i.c -= ret; g[i.v][i.dual].c += ret;
                    return ret;
                }
            }
        }
        return 0;
    }
    p run(int _s, int _t){ //{cost, flow}
        init(_s, _t);
        int cst = 0, fl = 0;
        do{
            memset(chk, 0, sizeof chk);
            memset(work, 0, sizeof work);
            int now = 0;
            while(now = dfs(s, 1e9)){
                cst += dst[t] * now;
                fl += now;
                memset(chk, 0, sizeof chk);
            }
        }while(update());
        return p(cst, fl);
    }
} mcmf;
int main(){
    ios_base::sync_with_stdio(0); cin.tie(0);
    int n, m; cin >> n >> m;
    for(int i=1; i<=n; i++){
        int cnt; cin >> cnt;
        while(cnt--){
            int a, b; cin >> a >> b;
            mcmf.addEdge(i, a+400, 1, b);
        }
    }const int s = 881, t = 882;
    for(int i=1; i<=n; i++) mcmf.addEdge(s, i, 1, 0);
    for(int j=1; j<=m; j++) mcmf.addEdge(j+400, t, 1, 0);
    auto now = mcmf.run(s, t);
    cout << now.y << "\n" << now.x;
}
```

3.4 Hungarian

```
const int MAX = 505;
int w[MAX][MAX], match_x[MAX], match_y[MAX];
int l_x[MAX], l_y[MAX];
bool s[MAX], t[MAX];
int slack[MAX], slack_x[MAX];
int tree_x[MAX], tree_y[MAX];
int hungarian(int n) {
  memset(match_x, -1, sizeof(match_x));
  memset(match_y, -1, sizeof(match_y));
  int ret = 0;
  for (int i = 0; i < n; ++i) {
    for (int j = 0; j < n; ++j) {
      l_x[i] = max(l_x[i], w[i][j]);
    }
  }
  memset(l_y, 0, sizeof(l_y));
  int m = 0;
  while (m != n) { // repeat at most V times
    memset(tree_x, -1, sizeof(tree_x));
    memset(tree_y, -1, sizeof(tree_y));
    memset(s, 0, sizeof(s));
    memset(t, 0, sizeof(t));
    int s_start;
    for (int i = 0; i < n; ++i) { // O(V)
      if (match_x[i] == -1) {
        s[i] = 1;
        s_start = i;
        break;
      }
    }
    for (int i = 0; i < n; ++i) { // init slack
      slack[i] = l_x[s_start] + l_y[i] - w[s_start][i];
      slack_x[i] = s_start;
    }
    here:
    int y = -1;
    for (int i = 0; i < n; ++i) { // compare: O(V)
      if (slack[i] == 0 && !t[i]) y = i;
    }
    if (y == -1) { // n_l = t
      // update label
      int alpha = INF;
      for (int i = 0; i < n; ++i) { // O(V)
        if (!t[i]) {
          alpha = min(alpha, slack[i]);
        }
      }
      for (int i = 0; i < n; ++i) { // O(V)
        if (s[i]) l_x[i] -= alpha;
        if (t[i]) l_y[i] += alpha;
      }
      for (int i = 0; i < n; ++i) { // O(V)
        if (!t[i]) {
          slack[i] -= alpha;
          if (slack[i] == 0) {
            y = i;
          }
        }
      }
    }
    // n_l != t is guaranteed
    if (match_y[y] == -1) { // free
      tree_y[y] = slack_x[y];
      while (y != -1) {
        int x = tree_y[y];
        match_y[y] = x;
        int next_y = match_x[x];
        match_x[x] = y;
        y = next_y;
      }
      m++;
    }
    else { // matched
      int z = match_y[y];
      tree_x[z] = y;
      tree_y[y] = slack_x[y];
      s[z] = 1;
      t[y] = 1;
      // z가 추가되었으므로 slack과 n_l이 update
      for (int i = 0; i < n; ++i) { // O(V)
        if (l_x[z] + l_y[i] - w[z][i] < slack[i]) {
          slack[i] = l_x[z] + l_y[i] - w[z][i];
          slack_x[i] = z;
        }
      }
      goto here;
    }
  }
  for (int i = 0; i < n; ++i) {
    ret += l_x[i];
    ret += l_y[i];
  }
  return ret;
}
```

3.5 Blossom

```
#include <iostream>
#include <deque>
#include <algorithm>
#include <vector>
#include <queue>
#include <cstring>
#include <numeric>
using namespace std;
using ll=long long;
queue<int> q;
vector<int> v[502];
int mat[502];
int col[502],par[502],grp[502];
bool vs[502];
void flip(int b){
    while(b){
        int a=par[b];
        int t=mat[a];
        mat[a]=b;mat[b]=a;
        mat[t]=0;
        b=t;
    }
    {

    }
}int lca(int r,int a,int b){
    memset(vs,0,sizeof vs);
    while(a!=r){
        vs[a]=true;
        a=grp[par[mat[a]]];
    }while(b!=r){
        if(vs[b])return b;
        b=grp[par[mat[b]]];
    }return r;
}void grping(int p,int a,int b){
    while(grp[a]!=p){
        int c=mat[a],d=par[c];
        if(col[c]==1){
            col[c]=0;
            q.push(c);
        }par[a]=b;
        grp[a]=grp[c]=p;
        a=d;
        b=c;
    }
}
bool bfs(int n){
    memset(col,-1,sizeof col);
    memset(par,0,sizeof par);
    iota(grp,grp+502,0);
    while(!q.empty())q.pop();
    q.push(n);col[n]=0;
    while(!q.empty()){
        int p=q.front();
        q.pop();
        for(auto i:v[p]){
            if(col[i]==-1){
                par[i]=p;col[i]=1;
                if(!mat[i]){
                    flip(i);
                    return 1;
                }col[mat[i]]=0;
                q.push(mat[i]);
            }else if(col[i]==0&&grp[i]!=grp[p]){
                int w=lca(grp[n],grp[p],grp[i]);
                grping(w,p,i);
                grping(w,i,p);
            }
        }
    }return 0;
}
signed main(){
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int n,m,a,b,ans=0;
    cin>>n>>m;
    for(int i=0;i<m;i++){
        cin>>a>>b;
        v[a].push_back(b);
        v[b].push_back(a);
    }for(int i=1;i<=n;i++){
        if(!mat[i]){
            for(auto j:v[i]){
                if(!mat[j]){
                    mat[i]=j;mat[j]=i;ans++;
                    break;
                }
            }
        }
    }for(int i=1;i<=n;i++){
        if(!mat[i]&&bfs(i)){
            ans++;
        }
    }cout<<ans;
}
```

4 Strings

4.1 KMP

```
string p, s;
vector<int> fail, ans;
void find_fail() {
  int n = sz(p), j = 0;
  for(int i=1 ; i<n ; i++) {
    while(j>0 && p[i]!=p[j])
      j = fail[j-1];
    if(p[i]==p[j])
      fail[i] = ++j;
  }
}
void kmp() {
  int n = sz(s), m = sz(p), j=0;
  for(int i=0 ; i<n ; i++) {
    while(j>0 && s[i]!=p[j])
      j = fail[j-1];
    if(s[i]==p[j]) {
      if(j==m-1) {
        ans.push_back(i-m+1);
        j = fail[j];
      } else
        j++;
    }
  }
}
```

4.2 Aho-Corasick Algorithm

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <queue>
using ll=long long;
using namespace std;
struct Trie{
    Trie *al[26];
    Trie *fail;
    bool out;
    Trie(){
        for(int i=0;i<26;i++)al[i]=nullptr;
        fail=nullptr;
        out=false;
    }
    void insert(string s,int idx){
        if(idx==s.length()){
            out=true;
            return;
        }
        int g=s[idx]-'a';
        if(al[g]==nullptr)al[g]=new Trie;
        al[g]->insert(s,idx+1);
    }
};
queue<Trie*> q;
int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int n;
    string s;
    cin>>n;
    Trie *r=new Trie;
    while(n--){
        cin>>s;
        r->insert(s,0);
    }
    q.push(r);
    while(!q.empty()){
        Trie*tmp=q.front();
        q.pop();
        for(int i=0;i<26;i++){
            Trie* now=tmp->al[i];
            if(!now)continue;
            if(tmp==r){
                now->fail=r;
            }else{
                Trie*f=tmp->fail;
                while(f!=r&&!f->al[i]){
                    f=f->fail;
                }
                if(f->al[i])f=f->al[i];
                now->fail=f;
            }
            if(now->fail->out)now->out=true;
            q.push(now);
        }
    }
    int qq;
    cin>>qq;
    while(qq--){
        cin>>s;
        Trie* now=r;
        bool flag=false;
        for(int i=0;i<s.length();i++){
            int g=s[i]-'a';
            while(now!=r&&!now->al[g]){
                now=now->fail;
            }
            if(now->al[g])now=now->al[g];
            if(now->out){
                flag=true;
                break;
            }
        }
        cout<<(flag?"YES":"NO")<<'\n';
    }
}
```

4.3 Suffix Array(LCP)

```
int sa[500'010],group[500'010],nGroup[500'010],rsa[500'010],lcp[500'010],tmp;
bool cmp(int x, int y) {
    if (group[x] == group[y]) {
        return group[x + tmp] < group[y + tmp];
    }
    return group[x] < group[y];
}
void getSA(const string& str) {
    tmp = 1;
    int n = str.length();
    for(int i=0;i<n;i++){
        sa[i]=i;
        group[i]=str[i]-'a';
    }
    while(tmp<n){
        group[n]=-1;
        sort(sa,sa+n,cmp);
        nGroup[sa[0]]=0;
        if(tmp*2>=n)break;
        for(int i=1;i<n;i++)
            nGroup[sa[i]]=nGroup[sa[i-1]]+cmp(sa[i-1],sa[i]);
        for(int i=0;i<n;i++)
            group[i]=nGroup[i];
        tmp<<=1;
    }
    for(int i=0;i<n;i++)
        rsa[sa[i]]=i;
    tmp=0;
    for(int i=0;i<n;i++){
        int k=rsa[i];
        if(!k)continue;
        int t=sa[k-1];
        while(str[i+tmp]==str[t+tmp])
            tmp++;
        lcp[k]=tmp;
        if(tmp)tmp--;
    }
}
int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    string str;
    cin>>str;
    getSA(str);
    for(int i=0;i<str.length();i++)
        cout<<sa[i]+1<<' ';
    cout<<"\nx ";
    for(int i=1;i<str.length();i++)
        cout<<lcp[i]<<' ';
    return 0;
}
```

4.4 Suffix Array(koo)

```
const int MAXN = 500005;
int ord[MAXN], nord[MAXN], cnt[MAXN], aux[MAXN];
```

```
void solve(int n, char *str, int *sfx, int *rev, int *lcp){
    int p = 1;
    memset(ord, 0, sizeof(ord));
    for(int i=0; i<n; i++){
        sfx[i] = i;
        ord[i] = str[i];
    }
    int pnt = 1;
    while(1){
        memset(cnt, 0, sizeof(cnt));
        for(int i=0; i<n; i++) cnt[ord[min(i+p, n)]]++;
        for(int i=1; i<=n || i<=255; i++) cnt[i] += cnt[i-1];
        for(int i=n-1; i>=0; i--)
            aux[--cnt[ord[min(i+p, n)]]] = i;
        memset(cnt, 0, sizeof(cnt));
        for(int i=0; i<n; i++) cnt[ord[i]]++;
        for(int i=1; i<=n || i<=255; i++) cnt[i] += cnt[i-1];
        for(int i=n-1; i>=0; i--)
            sfx[--cnt[ord[aux[i]]]] = aux[i];
        if(pnt == n) break;
        pnt = 1;
        nord[sfx[0]] = 1;
        for(int i=1; i<n; i++){
            if(ord[sfx[i-1]] != ord[sfx[i]] || ord[sfx[i-1] + p] != ord[sfx[i] + p]){
                pnt++;
            }
            nord[sfx[i]] = pnt;
        }
        memcpy(ord, nord, sizeof(int) * n);
        p *= 2;
    }
    for(int i=0; i<n; i++) rev[sfx[i]] = i;
    int h = 0;
    for(int i=0; i<n; i++){
        if(rev[i]){
            int prv = sfx[rev[i] - 1];
            while(str[prv + h] == str[i + h]) h++;
            lcp[rev[i]] = h;
        }
        h = max(h-1, 0);
    }
}
```

4.5 manacher

```
int p[200'010];
void mana(string str){
    int r=0,c=0;
    //r = maximum last index of palindrome
    //c = center of maximum r
    for(int i=0;i<str.length();i++){
        if(r<i)p[i]=0;
        else p[i]=min(p[c*2-i],r-i);
        while(i-p[i]-1>=0&&i+p[i]+1<str.length()&&str[i-p[i]-1]==str[i+p[i]+1])
            p[i]++;
        if(r<i+p[i]){
            r=i+p[i];c=i;
        }
    }
}int main(){
    string t,str="+";
    cin>>t;for(char i:t)str+=i,str+='+';
    mana(str);int ans=0;
    for(int i=0;i<str.length();i++)ans=max(ans,p[i]);
    cout<<ans;
}
```

4.6 Rolling Hash

```
struct hash_string{
    int v[1'000'010];
    int pw[1'000'010];
    int p1=1564117;
    void init(string s){
        pw[0]=1;
        for(int i=1;i<=s.length();i++){
            pw[i]=pw[i-1]*p1;
            v[i]=v[i-1]*p1+s[i-1];
        }
    }int get(int l,int r){
        return v[r]-v[l-1]*pw[r-l+1];
    }
}tree;
```

5 Geometry

5.1 회전하는 캘리퍼스

```
struct Point2D {
    double x;
    double y;
};
auto dist = [](Point2D p1, Point2D p2) -> double {
    double dx = p1.x - p2.x;
    double dy = p1.y - p2.y;
    return dx*dx + dy*dy;
};
auto check = [](Point2D s1, Point2D e1, Point2D s2, Point2D e2) {
    Point2D p1 = {e1.x - s1.x, e1.y - s1.y};
    Point2D p2 = {e2.x - s2.x, e2.y - s2.y};
    return ccw({0,0},p1,p2) >= 0;
};
int t = 0;
double ret = 0;
for(int i=0 ; i<sz(hull) ; i++) {
    while(t+1 < sz(hull) && check(hull[i], hull[i+1], hull[t], hull[t+1])) {
        ret = max(ret, dist(hull[i], hull[t]));
        t += 1;
    }
    ret = max(ret, dist(hull[i], hull[t]));
}
```

5.2 선분 교차 판정

```
int ccw(pair<ll, ll> p1, pair<ll, ll> p2, pair<ll, ll> p3) {
  // p1p2 -> p2p3
  // returns 1 if CCW, 0 if straight, -1 if CW
  ll CCW = p1.ff * p2.ss + p2.ff * p3.ss + p3.ff * p1.ss - p1.ss * p2.ff - p2.ss * p3.ff -
  p3.ss * p1.ff;
  if(CCW > 0)
    return 1;
  else if(CCW == 0)
```

```
    return 0;
  else if(CCW < 0)
    return -1;
}
int isIntersect(pair<pair<ll, ll>, pair<ll, ll>> x, pair<pair<ll, ll>, pair<ll, ll>> y) {
  pair<ll, ll> a = x.ff;
  pair<ll, ll> b = x.ss;
  pair<ll, ll> c = y.ff;
  pair<ll, ll> d = y.ss;
  int ab = ccw(a,b,c)*ccw(a,b,d);
  int cd = ccw(c,d,a)*ccw(c,d,b);
  if(ab==0 && cd == 0) {
    if(a>b) swap(a, b);
    if(c>d) swap(c,d);
    return c<=b&&a<=d;
  } else {
    return ab<=0&&cd<=0;
  }
}
```

5.3 다각형 점 판정

```
ll gcd(ll a, ll b) { for (; b; a %= b, swap(a, b)); return a; }

pair<ld,ld> p2v(pair<ld,ld> a, pair<ld,ld> b) // 두 점 A,B가 주어지면 벡터 AB를 반환
{
    return { b.first - a.first, b.second - a.second };
}

ll ccw(pair<ld,ld> v1, pair<ld,ld> v2) // 벡터 v1, v2의 CCW
{
    ld res = v1.first * v2.second - v1.second * v2.first;

    if (res > 0) return 1;
    else if (res < 0) return -1;
    else return 0;
}

int n;
vector <pair<ld,ld>> CH;

bool isInside_nonconvex(vector <pair<ld,ld>>& CH, pair<ld,ld> point)
{
    int cnt = 0;
    for (int i = 0; i < CH.size(); i++)
    {
        // x축에 평행하고 point에서 시작하는 반직선과, 선분 p1 p2가 교차하는지 여부
        pair<ld,ld> p1 = CH[i], p2 = CH[(i + 1) % CH.size()];
        if (p1.second < p2.second) swap(p1, p2);

        pair<ld,ld> v1 = p2v(p1, point);
        pair<ld,ld> v2 = p2v(point, p2);

        if (ccw(v1, v2) == 0)
        {
            // 일단 점이 선분위에 있는지 확인
            if (min(p1.first, p2.first) <= point.first && point.first <= max(p1.first,
            p2.first)
                && min(p1.second, p2.second) <= point.second && point.second <=
                max(p1.second, p2.second))
                return true;
        }

        if (max(p1.first, p2.first) < point.first) continue;
        if (p1.second <= point.second) continue; // 1
        if (p2.second > point.second) continue;  // 2 둘 중 하나에만 등호가 들어가야 한다.
        if (min(p1.first, p2.first) > point.first) cnt++;
        else if (ccw(v1, v2) > 0) cnt++;
    }

    return cnt % 2;
    // 홀수번 교차하면 다각형 내부이다.
}
```

5.4 반평면 교집합

```
const double eps = 1e-8;
typedef pair<long double, long double> pi;
bool z(long double x){ return fabs(x) < eps; }
struct line{
   long double a, b, c;
   bool operator<(const line &l)const{
      bool flag1 = pi(a, b) > pi(0, 0);
      bool flag2 = pi(l.a, l.b) > pi(0, 0);
      if(flag1 != flag2) return flag1 > flag2;
      long double t = ccw(pi(0, 0), pi(a, b), pi(l.a, l.b));
      return z(t) ? c * hypot(l.a, l.b) < l.c * hypot(a, b) : t > 0;
   }
   pi slope(){ return pi(a, b); }
};
pi cross(line a, line b){
   long double det = a.a * b.b - b.a * a.b;
   return pi((a.c * b.b - a.b * b.c) / det, (a.a * b.c - a.c * b.a) / det);
}
bool bad(line a, line b, line c){
   if(ccw(pi(0, 0), a.slope(), b.slope()) <= 0) return false;
   pi crs = cross(a, b);
   return crs.first * c.a + crs.second * c.b >= c.c;
}
bool solve(vector<line> v, vector<pi> &solution){ // ax + by <= c;
   sort(v.begin(), v.end());
   deque<line> dq;
   for(auto &i : v){
      if(!dq.empty() && z(ccw(pi(0, 0), dq.back().slope(), i.slope()))) continue;
      while(dq.size() >= 2 && bad(dq[dq.size()-2], dq.back(), i)) dq.pop_back();
      while(dq.size() >= 2 && bad(i, dq[0], dq[1])) dq.pop_front();
      dq.push_back(i);
   }
   while(dq.size() > 2 && bad(dq[dq.size()-2], dq.back(), dq[0])) dq.pop_back();
   while(dq.size() > 2 && bad(dq.back(), dq[0], dq[1])) dq.pop_front();
   vector<pi> tmp;
   for(int i=0; i<dq.size(); i++){
      line cur = dq[i], nxt = dq[(i+1)%dq.size()];
      if(ccw(pi(0, 0), cur.slope(), nxt.slope()) <= eps) return false;
      tmp.push_back(cross(cur, nxt));
   }
   solution = tmp;
```

```
    return true;
}
```

5.5 Rotate SweepLine

```
struct po{
    int x,y;
    bool operator<(const po a)const{
        if(x==a.x)return y<a.y;
        return x<a.x;
    }
}arr[2020];
struct line{
    int i,j,dy,dx;
    bool operator<(line a)const{
        ///dy/dx<ddy/ddx ///dy*ddx<ddy*dx
        return dy*a.dx<a.dy*dx;
    }bool operator==(const line a)const{
        return dy*a.dx==a.dy*dx;
    }
};line init(int i,int j){
    return {i,j,arr[j].x-arr[i].x,arr[j].y-arr[i].y};
}ll shoelace(po a,po b,po c){
    return abs((a.x*b.y+b.x*c.y+c.x*a.y)-(a.y*b.x+b.y*c.x+c.y*a.x));
}vector<line> v;
int loc[2022];
void solve(){
    int n,a,b;
    cin>>n;if(!n)exit(0);
    for(int i=1;i<=n;i++){
        cin>>a>>b;
        arr[i]={a,b};
    }sort(arr+1,arr+n+1);v.clear();
    for(int i=1;i<=n;i++)for(int j=i+1;j<=n;j++){
            v.push_back(init(i,j));
        }stable_sort(all(v));
    for(int i=1;i<=n;i++)loc[i]=i;
    ll mn=1e18,mx=-1e18;
    for(int i=0,j=0;i<v.size();i=j){
        while(j<v.size()&&v[i]==v[j])j++;
        for(int k=i;k<j;k++){
            int f=v[k].i,s=v[k].j;
            int ff=loc[f],ss=loc[s];
            swap(arr[ff],arr[ss]);
            swap(loc[f],loc[s]);
            if(ff>ss)swap(ff,ss);
            if(ff>1){
                mn=min(mn,shoelace(arr[ff],arr[ss],arr[ff-1]));
                mx=max(mx,shoelace(arr[ff],arr[ss],arr[1]));
            }if(ss<n){
                mn=min(mn,shoelace(arr[ff],arr[ss],arr[ss+1]));
                mx=max(mx,shoelace(arr[ff],arr[ss],arr[n]));
            }
        }
    }cout<<mn/2<<'.'<<(mn&1)*5<<' ';cout<<mx/2<<'.'<<(mx&1)*5<<'\n';
}
```

5.6 Convex Hull

```
// 1. Monotone Chain
pii operator-(pii a, pii b){ return {a.x-b.x, a.y-b.y}; }
ll cross(pii a, pii b){ return b.y*1LL*a.x - b.x*1LL*a.y; }
bool ccw(pii a, pii b, pii c){ return cross(b-a, c-a) >= 0; }
// Calculates upper & lower hull. O(NlgN) time & O(N) space.
pair<vector<pii>, vector<pii>> getConvexHull(vector<pii> pt){
    sort(pt.begin(), pt.end());
    vector<pii> uh, dh;
    int un=0, dn=0; // for easy coding
    for(auto& tmp:pt){
        while(un >= 2 && ccw(uh[un-2], uh[un-1], tmp)) uh.pop_back(), --un;
        uh.push_back(tmp); ++un;
    }
    reverse(pt.begin(), pt.end());
    for(auto& tmp:pt){
        while(dn >= 2 && ccw(dh[dn-2], dh[dn-1], tmp)) dh.pop_back(), --dn;
        dh.push_back(tmp); ++dn;
    }
    return {uh, dh};
}
// 2. Graham Scan
using pdd = pair<double, double>;
double size(pdd x){ return hypot(x.first, x.second); }
int sign(ll x){ return x < 0? -1 : x > 0? 1 : 0; }
pii operator-(pii a, pii b){ return {a.x-b.x, a.y-b.y}; }
pii operator+(pii a, pii b){ return {a.x+b.x, a.y+b.y}; }
ll operator^(const pii &l, const pii &r){ return (ll)l.first * r.second - (ll)l.second *
r.first; }
template<typename T>
void convex_hull(vector<T> &L, vector<T> &R){
    int mn = 0;
    for(int i = 1; i < L.size(); i++)
        if( L[mn] > L[i] ) mn = i;
    swap(L[mn], L[0]);
    T t = L[0];
    for(int i = 1; i < L.size(); i++) L[i] = L[i] - L[0];
    L[0] = T(0, 0);
    sort(L.begin()+1, L.end(), [](T &l, T &r){
            if( sign(l^r) != 0 ) return sign(l^r) < 0;
            return size(l) < size(r);
        });
    for(T &c : L){
        while(R.size() >= 2 && sign((R[R.size()-2] - R.back()) ^ (c - R.back())) <= 0 )
        R.pop_back();
        R.push_back(c);
    }
    for(T &c : R) c = c + t;
}
```

6 Math

6.1 Fast Mod

```
typedef __uint128_t L;
struct FastMod {
    ull b, m;
    FastMod(ull b) : b(b), m(ull((L(1) << 64) / b)) {}
    ull reduce(ull a) {
        ull q = (ull)((L(m) * a) >> 64);
        ull r = a - q * b; // can be proven that 0 <= r < 2*b
        return r >= b ? r - b : r;
    }
};FastMod F(2);int N, P;
int main(){
    scanf("%d %d", &N, &P); F = FastMod(P);ll res = 1;
    for (int i = 1; i <= N; i++){
        res *= i;res = F.reduce(res);
    }printf("%lld", res);
}
```

6.2 Floor Sum

```
// @param n `n < 2^32`
// @param m `1 <= m < 2^32`
// @return sum_{i=0}^{n-1} floor((ai + b) / m) (mod 2^64)
unsigned long long floor_sum_unsigned(unsigned long long n,unsigned long long m,unsigned
long long a,unsigned long long b) {
    unsigned long long ans = 0;
    while (true) {
        if (a >= m) {
            ans += n * (n - 1) / 2 * (a / m);
            a %= m;
        }if (b >= m) {
            ans += n * (b / m);
            b %= m;
        }
        unsigned long long y_max = a * n + b;
        if (y_max < m) break;
        // y_max < m * (n + 1)
        // floor(y_max / m) <= n
        n = (unsigned long long)(y_max / m);
        b = (unsigned long long)(y_max % m);
        std::swap(m, a);
    }return ans;
}
```

6.3 FFT / NTT

```
typedef complex<double> base;
void fft(vector<base> &a, bool inv){
  int n = a.size(), j = 0;
  vector<base> roots(n/2);
  for(int i=1; i<n; i++){
    int bit = (n >> 1);
    while(j >= bit){
      j -= bit;
      bit >>= 1;
    }
    j += bit;
    if(i < j) swap(a[i], a[j]);
  }
  double ang = 2 * acos(-1) / n * (inv ? -1 : 1);
  for(int i=0; i<n/2; i++){
    roots[i] = base(cos(ang * i), sin(ang * i));
  }
  /* In NTT, let prr = primitive root. Then,
  int ang = ipow(prr, (mod - 1) / n);
  if(inv) ang = ipow(ang, mod - 2);
  for(int i=0; i<n/2; i++){
    roots[i] = (i ? (1ll * roots[i-1] * ang % mod) : 1);
  }
  XOR Convolution : set roots[*] = 1.
  OR Convolution : set roots[*] = 1, and do following:
    if (!inv) {
        a[j + k] = u + v;
        a[j + k + i/2] = u;
    } else {
        a[j + k] = v;
        a[j + k + i/2] = u - v;
    }
  */
  for(int i=2; i<=n; i<<=1){
    int step = n / i;
    for(int j=0; j<n; j+=i){
      for(int k=0; k<i/2; k++){
        base u = a[j+k], v = a[j+k+i/2] * roots[step * k];
        a[j+k] = u+v;
        a[j+k+i/2] = u-v;
      }
    }
  }
  if(inv) for(int i=0; i<n; i++) a[i] /= n; // skip for OR convolution.
}
vector<lint> multiply(vector<lint> &v, vector<lint> &w){
  vector<base> fv(v.begin(), v.end()), fw(w.begin(), w.end());
  int n = 2; while(n < v.size() + w.size()) n <<= 1;
  fv.resize(n); fw.resize(n);
  fft(fv, 0); fft(fw, 0);
  for(int i=0; i<n; i++) fv[i] *= fw[i];
  fft(fv, 1);
  vector<lint> ret(n);
  for(int i=0; i<n; i++) ret[i] = (lint)round(fv[i].real());
  return ret;
}
vector<lint> multiply(vector<lint> &v, vector<lint> &w, lint mod){
  int n = 2; while(n < v.size() + w.size()) n <<= 1;
  vector<base> v1(n), v2(n), r1(n), r2(n);
  for(int i=0; i<v.size(); i++){
    v1[i] = base(v[i] >> 15, v[i] & 32767);
  }
  for(int i=0; i<w.size(); i++){
    v2[i] = base(w[i] >> 15, w[i] & 32767);
  }
  fft(v1, 0);
  fft(v2, 0);
  for(int i=0; i<n; i++){
    int j = (i ? (n - i) : i);
    base ans1 = (v1[i] + conj(v1[j])) * base(0.5, 0);
    base ans2 = (v1[i] - conj(v1[j])) * base(0, -0.5);
    base ans3 = (v2[i] + conj(v2[j])) * base(0.5, 0);
    base ans4 = (v2[i] - conj(v2[j])) * base(0, -0.5);
    r1[i] = (ans1 * ans3) + (ans1 * ans4) * base(0, 1);
    r2[i] = (ans2 * ans3) + (ans2 * ans4) * base(0, 1);
  }
  fft(r1, 1);
  fft(r2, 1);
  vector<lint> ret(n);
  for(int i=0; i<n; i++){
    lint av = (lint)round(r1[i].real());
    lint bv = (lint)round(r1[i].imag()) + (lint)round(r2[i].real());
    lint cv = (lint)round(r2[i].imag());
    av %= mod, bv %= mod, cv %= mod;
    ret[i] = (av << 30) + (bv << 15) + cv;
    ret[i] %= mod;
    ret[i] += mod;
    ret[i] %= mod;
  }
  return ret;
}
```

6.4 Hell-Joseon FFT

```
#include <smmintrin.h>
#include <immintrin.h>
#pragma GCC target("avx2")
#pragma GCC target("fma")
__m256d mult(__m256d a, __m256d b){
  __m256d c = _mm256_movedup_pd(a);
  __m256d d = _mm256_shuffle_pd(a, a, 15);
  __m256d cb = _mm256_mul_pd(c, b);
  __m256d db = _mm256_mul_pd(d, b);
  __m256d e = _mm256_shuffle_pd(db, db, 5);
  __m256d r = _mm256_addsub_pd(cb, e);
  return r;
}
void fft(int n, __m128d a[], bool invert){
  for(int i=1, j=0; i<n; ++i){
    int bit = n>>1;
    for(;j>=bit;bit>>=1) j -= bit;
    j += bit;
    if(i<j) swap(a[i], a[j]);
  }
  for(int len=2; len<=n; len<<=1){
    double ang = 2*3.14159265358979/len*(invert?-1:1);
    __m256d wlen; wlen[0] = cos(ang), wlen[1] = sin(ang);
    for(int i=0; i<n; i += len){
      __m256d w; w[0] = 1; w[1] = 0;
      for(int j=0; j<len/2; ++j){
        w = _mm256_permute2f128_pd(w, w, 0);
        wlen = _mm256_insertf128_pd(wlen, a[i+j+len/2], 1);
        w = mult(w, wlen);
        __m128d vw = _mm256_extractf128_pd(w, 1);
        __m128d u = a[i+j];
        a[i+j] = _mm_add_pd(u, vw);
        a[i+j+len/2] = _mm_sub_pd(u, vw);
      }
    }
  }
  if(invert){
    __m128d inv; inv[0] = inv[1] = 1.0/n;
    for(int i=0; i<n; ++i) a[i] = _mm_mul_pd(a[i], inv);
  }
}
vector<int64_t> multiply(vector<int64_t>& v, vector<int64_t>& w){
  int n = 2; while(n < v.size()+w.size()) n<<=1;
  __m128d* fv = new __m128d[n];
  for(int i=0; i<n; ++i) fv[i][0] = fv[i][1] = 0;
  for(int i=0; i<v.size(); ++i) fv[i][0] = v[i];
  for(int i=0; i<w.size(); ++i) fv[i][1] = w[i];
  fft(n, fv, 0); // (a+bi) is stored in FFT
  for(int i=0; i<n; i += 2){
    __m256d a;
    a = _mm256_insertf128_pd(a, fv[i], 0);
    a = _mm256_insertf128_pd(a, fv[i+1], 1);
    a = mult(a, a);
    fv[i] = _mm256_extractf128_pd(a, 0);
    fv[i+1] = _mm256_extractf128_pd(a, 1);
  }
  fft(n, fv, 1);
  vector<int64_t> ret(n);
  for(int i=0; i<n; ++i) ret[i] = (int64_t)round(fv[i][1]/2);
  delete[] fv;
  return ret;
}
```

6.5 확장 유클리드

```
// k*x+l*y=gcd
ll euclid(ll x, ll y, ll &k, ll &l) {
  if (y == 0) {
    k = 1;
    l = 0;
    return x;
  }
  ll g = euclid(y, x % y, l, k);
  l -= k * (x / y);
  return g;
}
```

6.6 CRT + Modular Inverse + 확장 유클리드

```
// gcd(a,b), s,t where a*s + b*t = gcd(a,b)
pair<ll,pair<ll,ll>> xGCD(ll a, ll b) {
    if(b == 0) return {a,{1,0}};
    pair<ll,pair<ll,ll>> ret = xGCD(b, a%b);
    ll g, x, y;
    g = ret.first;
    tie(x,y) = ret.second;
    return {g,{y,x-(a/b)*y}};
}
int mod_inverse(int a, int mod) {
    auto res = xGCD(a,mod);
    if(res.first > 1) return -1;
    return (res.second.first + mod) % mod;
}
// A = [a_1, a_2, ... , a_N]
// M = [m_1, m_2, ... , m_N]
// each equation is x = a_i (mod m_i)
// it returns {-1,-1} if there's no solution satisfying N linear congruence equations.
pair<ll,ll> CRT(vector<ll> &A, vector<ll> &M) {
    if(A.size() != M.size()) return {-1,-1};
    int N = A.size();
    ll a1 = A[0];
    ll m1 = M[0];
    a1 %= m1;
    for(int i=1;i<N;++i) {
        ll a2 = A[i];
        ll m2 = M[i];
        ll g = __gcd(m1, m2);
        if(a1 % g != a2 % g) return {-1,-1};
        ll p, q;
        auto res = xGCD(m1/g, m2/g);
        tie(p,q) = res.second;
        i128 mod = (i128)m1 / g * m2;
        a1 = ((i128)a1 * (m2/g) % mod) * q % mod + ((i128)a2*(m1/g)%mod)*p % mod;
        a1 = (a1 + mod) % mod;
        m1 = mod;
    }
    return {a1, m1};
}
```

6.7 뤼카의 정리(lucas theorem)

```
// calculate nCm % p  when p is prime
int lucas_theorem(const char *n, const char *m, int p) {
    vector<int> np, mp;
    int i;
    for (i = 0; n[i]; i++) {
        if (n[i] == '0' && np.empty()) continue;
        np.push_back(n[i] - '0');
    }
    for (i = 0; m[i]; i++) {
        if (m[i] == '0' && mp.empty()) continue;
        mp.push_back(m[i] - '0');
    }
    int ret = 1;
    int ni = 0, mi = 0;
    while (ni < np.size() || mi < mp.size()) {
        int nmod = 0, mmod = 0;
        for (i = ni; i < np.size(); i++) {
            if (i + 1 < np.size())
                np[i + 1] += (np[i] % p) * 10;
            else
                nmod = np[i] % p;
            np[i] /= p;
        }
        for (i = mi; i < mp.size(); i++) {
            if (i + 1 < mp.size())
                mp[i + 1] += (mp[i] % p) * 10;
            else
                mmod = mp[i] % p;
            mp[i] /= p;
        }
        while (ni < np.size() && np[ni] == 0) ni++;
        while (mi < mp.size() && mp[mi] == 0) mi++;
        // implement binomial. binomial(m,n) = 0 if m < n
        ret = (ret * binomial(nmod, mmod)) % p;
    }
    return ret;
}ll gets(ll n,ll k){
    if(n<k)return 0;
    if(n==k||k==0)return 1;
    if(n<m&&k<m)return 1ll*pack[n%m]*ipack[k%m]%m*ipack[n%m-k%m]%m;
    return gets(n/m,k/m)*gets(n%m,k%m)%m;
}
```

6.8 Linear-sieve with Multiplicative Function)

```
// n = k개의 소인수 p에 대해 : p_i ** e_i 의 곱
// phi[n] : n 이하의 자연수 중 n과 서로소인 수의 개수
// mu[n] : n의 약수 중 지수가 2 이상인 것이 있다면 0, 그렇지 않다면 (-1)^k를 나타내는 함수
// tau[n] : n의 양의 약수의 개수
// sigma[n] : n의 양의 약수의 합
// sp[n] : n이 소수이면 0, 아니면 최소 소인수
// p : 소수가 담긴 벡터
vector<int> p;
const int sz=101010;
int sp[sz], e[sz], phi[sz], mu[sz], tau[sz], sigma[sz];
int pw(int a, int b){
    int ret = 1;
    while(b){
        if(b&1)ret*a;
        a*=a; b>>=1;
    }
    return ret;
}
// 1. 그냥 linear_sieve(최소 소인수, 소수 목록) 구할 때
void linear_sieve(int n)
{
  for(int i=2;i<=n;i++){
    if(!sp[i]) p.push_back(i);
    for(auto j: p){
      if(i*j>n) break;
      sp[i*j]=j;
      if(i%j==0) break;
    }
  }
}
// 2. multiplicative function까지 구할 때
void linear_sieve_multiplicative_function(){
    int i, j, temp=0;
  phi[1]=mu[1]=tau[1]=sigma[1]=1;
  for(i=2;i<sz;i++)
  {
    if(!sp[i])
    {
      p.push_back(i);
      e[i]=1;
      phi[i]=i-1;
      mu[i]=-1;
      tau[i]=2;
      sigma[i]=i+1;
    }
    for(auto j:p)
    {
      if(i*j>=sz) break;
      sp[i*j]=j;
      if(i%j==0)
      {
        e[i*j]=e[i]+1;
        phi[i*j]=phi[i]*j;
        mu[i*j]=0;
        tau[i*j]=tau[i]/e[i*j]*(e[i*j]+1);
        sigma[i*j]=sigma[i]*(j-1)/(pw(j, e[i*j])-1)*(pw(j, e[i*j]+1)-1)/(j-1);//overflow
        break;
      }
      e[i*j]=1;
      phi[i*j]=phi[i]*phi[j];
      mu[i*j]=mu[i]*mu[j];
      tau[i*j]=tau[i]*tau[j];
      sigma[i*j]=sigma[i]*sigma[j];
    }
  }
}
```

6.9 Pollard rho, Miller-Rabin

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
using namespace std;
using ll=unsigned long long;
ll pow(ll n,ll k,ll mod){
    ll res=1;n=n%mod;
    while(k){
        if(k&1)res=(__int128)res*n%mod;
        n=(__int128)n*n%mod;
        k>>=1;
    }return res;
}
bool isPrime(ll v){
    ll p;
    p=v;
    int cnt=0;
    v--;
    while(v%2==0){
        cnt++;
        v/=2;
    }
    bool flag=false;
    for(auto i:{2,3,5,7,11,13,17,17,19,23,29,31,37}){
        if(i==p){
            flag=true;
            break;
        }
        ll d=v;
        ll now=pow(i,d,p);
        flag=false;
        if(now==1||now==p-1){
            flag=true;
            continue;
        }
        for(int j=1;j<cnt;j++){
            now=(__int128)now*now%p;
            if(now==p-1){
                flag=true;
                break;
            }
        }
        if(!flag)break;
    }if(!flag&&v<=40)return 0;
    return flag;
}vector<ll> ans;
ll func(ll t,ll c,ll n){
    return ((__int128)t*t)%n+c;
}ll abs(ll t){
    if(t>0)return t;
    return -t;
}
void fack(ll n){
    if(n==1)return;
    if(~n&1){
        ans.push_back(2);
        fack(n/2);
        return;
    }
    if(isPrime(n)){
        ans.push_back(n);
        return;
    }ll x,y,c,g=n;
    do{
        if(g==n){
            x=y=rand()%(n-1)+1;
            c=rand()%23+1;
        }x=func(x,c,n);
        y=func(y,c,n);
        y=func(y,c,n);
        g=__gcd(abs(x-y),n);
    }while(g==1);
    fack(g);
    fack(n/g);
}
int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    ll n;
    cin>>n;
    fack(n);
    sort(ans.begin(),ans.end());
    for(auto i:ans)cout<<i<<'\n';
}
```

7 DP

7.1 컨벡스헐 트릭

```
// 점화식이 DP[i] = min_{j<i}(A[i]*B[j]+C[j])+D[i] 꼴인 경우 사용 가능하다.
struct CHT{
  bool isInc;
  CHT(){}
  // A[i]가 증가함수인 경우 isInc를 True로 설정하면 전체 시간복잡도를 O(NlogN)에서 O(N)으로 줄일 수
  있다.
  CHT(bool _isInc){
    isInc = _isInc;
  }
  deque<pll> line;
  double inter(int i, int j){
    return 1.00 * (line[i].second - line[j].second) / (line[j].first - line[i].first);
  }
  ll calc(ll i, ll x){
    return line[i].first * x + line[i].second;
  }
  //f(x)=B[j]*x+C[j]일 때 insert(B[j], C[j])로 직선을 저장한다.
  void insert(ll a, ll b){
    line.push_back({a, b});
    int i = line.size() - 1;
    while(i > 1 && inter(i, i-1) < inter(i-1, i-2)){
      line[i-1] = line.back();
      line.pop_back();
      i--;
    }
  }
  int bin(ll k){
    int l = 0;
    int r = line.size() - 1;
    while(l < r){
      int m = l + r >> 1;
      if (k < inter(m, m+1)) r = m;
      else l = m + 1;
    }
    return r;
  }
  // min_{j<i}(k*B[j]+C[j])를 리턴한다.
  ll get(ll k){
    if(isInc){
      if(line.empty()) return 3e17; //assert
      while(line.size() > 1 && calc(0, k) > calc(1, k)){ //max를 구하는 경우 calc(0, k) <
      calc(1, k)로 수정한다.
        line.pop_front();
      }
      return calc(0, k);
    }
    if(line.empty()) return 3e17; //assert
    if(line.size() == 1) return calc(0, k);
    return calc(bin(k), k);
  }
};
```

7.2 Lichao Tree

```
struct line{//y=ax+b
    ll a,b;
    ll y(ll x){
        return a*x+b;
    }
};
struct node{
    int l,r; //child
    line ln;
    node(){
        l=-1;r=-1;ln={0,inf};
    }
};
vector<node> tree(100'010);
void update(int n,int pre,line v,ll s=0,ll e=mv){
    if(pre!=-1)tree[n]=tree[pre];
    ll m=s+e>>1;
    line b=tree[n].ln,t=v;
    if(t.y(s)>b.y(s))swap(t,b);
    if(t.y(e)<=b.y(e)){tree[n].ln=t;return;}
    if(t.y(m)<b.y(m)){
        tree[n].ln=t;
        tree[n].r=tree.size(),tree.push_back(node());
        if(pre==-1||!~tree[pre].r)pre=-1;
        else pre=tree[pre].r;
        update(tree[n].r,pre,b,m+1,e);
    }else{
        tree[n].ln=b;
        tree[n].l=tree.size(),tree.emplace_back();
        if(pre==-1||!~tree[pre].l)pre=-1;
        else pre=tree[pre].l;
        update(tree[n].l,pre,t,s,m);
    }
}
ll query(int n,ll v,ll s=0,ll e=mv){
    if(!~n)return inf;
    ll m=s+e>>1;
    if(v<=m)return min(tree[n].ln.y(v),query(tree[n].l,v,s,m));
    else return min(tree[n].ln.y(v),query(tree[n].r,v,m+1,e));
}
```

7.3 Monotone Queue opt

```
#include <iostream>
#include <vector>
using namespace std;
using ll=long long;
ll pre[50'010],dp[50'010];
int iq[50'010],cq[50'010],use[50'010],N;
ll func(int i,int j){
    return dp[i]+(pre[j]-pre[i])*(j-i);
}int cross(int i,int j){
    int l=j+1,r=N;
    while(l<=r){
        int mid=l+r>>1;
        if(func(i,mid)<=func(j,mid))l=mid+1;
        else r=mid-1;
    }return l-1;
}int opt(ll k){
    int pv=0,pv2=1;
    iq[0]=0,cq[0]=N;
    for(int i=1;i<=N;i++){
        while(cq[pv]<i)pv++;
        dp[i]=func(iq[pv],i)+k;
        use[i]=use[iq[pv]]+1;
        while(pv+1<pv2&&cq[pv2-2]>=cross(iq[pv2-1],i))pv2--;
        cq[pv2-1]=cross(iq[pv2-1],i);
        iq[pv2]=i;cq[pv2++]=N;
    }return use[N];
}
int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int n,k;
    cin>>n>>k;N=n;
    for(int i=1;i<=n;i++){
        cin>>pre[i];pre[i]+=pre[i-1];
    }ll l=0,r=1e14;
    while(l<=r){
        ll mid=l+r>>1;
        if(opt(mid)>=k)l=mid+1;
        else r=mid-1;
    }opt(r+1);
    cout<<dp[n]-(r+1)*k;
}
```

7.4 SOS DP

```
for(int j=m-1;j>=0;j--)for(int i=(1<<m)-1;i>=0;i--){
            if(i&(1<<j)){
                B[i^(1<<j)]+=B[i];
            }
        }
```

7.5 Berlekamp-Massey

```
const int mod = 998244353;
using lint = long long;
lint ipow(lint x, lint p){
  lint ret = 1, piv = x;
  while(p){
    if(p & 1) ret = ret * piv % mod;
    piv = piv * piv % mod;
    p >>= 1;
  }
  return ret;
}
vector<int> berlekamp_massey(vector<int> x){
  vector<int> ls, cur;
  int lf, ld;
  for(int i=0; i<x.size(); i++){
    lint t = 0;
    for(int j=0; j<cur.size(); j++){
      t = (t + 1ll * x[i-j-1] * cur[j]) % mod;
    }
    if((t - x[i]) % mod == 0) continue;
    if(cur.empty()){
      cur.resize(i+1);
      lf = i;
      ld = (t - x[i]) % mod;
      continue;
    }
    lint k = -(x[i] - t) * ipow(ld, mod - 2) % mod;
    vector<int> c(i-lf-1);
    c.push_back(k);
    for(auto &j : ls) c.push_back(-j * k % mod);
    if(c.size() < cur.size()) c.resize(cur.size());
    for(int j=0; j<cur.size(); j++){
      c[j] = (c[j] + cur[j]) % mod;
    }
    if(i-lf+(int)ls.size()>=(int)cur.size()){
      tie(ls, lf, ld) = make_tuple(cur, i, (t - x[i]) % mod);
    }
    cur = c;
  }
  for(auto &i : cur) i = (i % mod + mod) % mod;
  return cur;
}
int get_nth(vector<int> rec, vector<int> dp, lint n){
  int m = rec.size();
  vector<int> s(m), t(m);
  s[0] = 1;
  if(m != 1) t[1] = 1;
  else t[0] = rec[0];
  auto mul = [&rec](vector<int> v, vector<int> w){
    int m = v.size();
    vector<int> t(2 * m);
    for(int j=0; j<m; j++){
      for(int k=0; k<m; k++){
        t[j+k] += 1ll * v[j] * w[k] % mod;
        if(t[j+k] >= mod) t[j+k] -= mod;
      }
    }
    for(int j=2*m-1; j>=m; j--){
      for(int k=1; k<=m; k++){
        t[j-k] += 1ll * t[j] * rec[k-1] % mod;
        if(t[j-k] >= mod) t[j-k] -= mod;
      }
    }
    t.resize(m);
    return t;
  };
  while(n){
    if(n & 1) s = mul(s, t);
    t = mul(t, t);
    n >>= 1;
  }
  lint ret = 0;
  for(int i=0; i<m; i++) ret += 1ll * s[i] * dp[i] % mod;
  return ret % mod;
}
int guess_nth_term(vector<int> x, lint n){
  if(n < x.size()) return x[n];
  vector<int> v = berlekamp_massey(x);
  if(v.empty()) return 0;
  return get_nth(v, x, n);
}
struct elem{int x, y, v;}; // A_(x, y) <- v, 0-based. no duplicate please..
vector<int> get_min_poly(int n, vector<elem> M){
  // smallest poly P such that A^i = sum_{j < i} {A^j \times P_j}
  vector<int> rnd1, rnd2;
  mt19937 rng(0x14004);
  auto randint = [&rng](int lb, int ub){
    return uniform_int_distribution<int>(lb, ub)(rng);
  };
  for(int i=0; i<n; i++){
    rnd1.push_back(randint(1, mod - 1));
    rnd2.push_back(randint(1, mod - 1));
  }
  vector<int> gobs;
  for(int i=0; i<2*n+2; i++){
    int tmp = 0;
    for(int j=0; j<n; j++){
      tmp += 1ll * rnd2[j] * rnd1[j] % mod;
      if(tmp >= mod) tmp -= mod;
    }
    gobs.push_back(tmp);
    vector<int> nxt(n);
    for(auto &i : M){
      nxt[i.x] += 1ll * i.v * rnd1[i.y] % mod;
      if(nxt[i.x] >= mod) nxt[i.x] -= mod;
    }
    rnd1 = nxt;
  }
  auto sol = berlekamp_massey(gobs);
  reverse(sol.begin(), sol.end());
  return sol;
}
lint det(int n, vector<elem> M){
  vector<int> rnd;
  mt19937 rng(0x14004);
  auto randint = [&rng](int lb, int ub){
    return uniform_int_distribution<int>(lb, ub)(rng);
  };
  for(int i=0; i<n; i++) rnd.push_back(randint(1, mod - 1));
  for(auto &i : M){
    i.v = 1ll * i.v * rnd[i.y] % mod;
  }
  auto sol = get_min_poly(n, M)[0];
  if(n % 2 == 0) sol = mod - sol;
  for(auto &i : rnd) sol = 1ll * sol * ipow(i, mod - 2) % mod;
  return sol;
}
```

8 기타

8.1 fastio(jthis)

```
static char buf[1 << 19]; // size : any number geq than 1024
static int idx = 0;
static int bytes = 0;
static inline int _read() {
  if (!bytes || idx == bytes) {
    bytes = (int)fread(buf, sizeof(buf[0]), sizeof(buf), stdin);
    idx = 0;
  }
  return buf[idx++];
}
static inline int _readInt() {
  int x = 0, s = 1;
  int c = _read();
  while (c <= 32) c = _read();
  if (c == '-') s = -1, c = _read();
  while (c > 32) x = 10 * x + (c - '0'), c = _read();
  if (s < 0) x = -x;
  return x;
}
```

8.2 SA

```
#include <chrono>
#include <random>
void tring(int n,int m){
    double k=1.5,T=1.0,delta=0.999999;
    mt19937 rd(0x34832a);
    uniform_int_distribution<int> rnd(1, n);
    uniform_int_distribution<int> rnd2(0, 100);
    int pre=chking(m);
    for(int q=0;q<5e5;q++){
        int idx=rnd(rd);
        arr[idx]^=1;
        int now=chking(m);
        double p=exp((now-pre)/(k*T));
        if(p>(double)rnd2(rd)/100){
            pre=now;
        }else arr[idx]^=1;////not Change
        k*=delta;
    }
}
mt19937 rd((unsigned)chrono::steady_clock::now().time_since_epoch().count());
```

8.3 Bit Hack

```
x&-x;   //x의 가장 마지막 비트를 계산함
unsigned int t = (v | (v - 1)) + 1;     //v랑 크기가 같고, 사전순으로 다음에 오는 집합을 찾음
w = t | ((((t & -t) / (v & -v)) >> 1) - 1);
for(int i=0; i=(i-x)&x; )   //x의 모든 부분집합을 순회함.
for(int i=x; i>0; i=(i-1)&x)    //x의 모든 부분집합을 역순으로 순회함
__builtin_clz(x);   //gcc내장함수 이용, x의 앞에 있는 0의 갯수를 셈 (x의 가장 큰 원소 = log2(x) =
31-clz(x))
__builtin_ctz(x);   //gcc내장함수 이용, x의 뒤에 있는 0의 갯수를 셈 (x의 가장 작은 원소를 가져 옴,
x&-x는 1<<__builtin_ctz(x)과 같음)
```

8.4 Template Wonsei

```
#include <bits/stdc++.h>
using namespace std;
#define all(x) x.begin(), x.end()
#define ff first
#define ss second
#define LLINF 0x3f3f3f3f3f3f3f3f
#define INF 0x3f3f3f3f
#define uniq(x) sort(all(x)); x.resize(unique(all(x))-x.begin());
#define sz(x) (int)x.size()
#define pw(x) (1LL<<x)
using pii = pair<int, int>;
using ll = long long;
const ll MOD = 1e9 + 7;
const long double PI = acos(-1.0);
int main() { ios::sync_with_stdio(0), cin.tie(0); }
100003,100019,200003,200009,407521,1e9 + 7,1e9 + 9,1e9 - 63,998244353,1234567891
# pragma GCC optimize ("O3")
# pragma GCC optimize ("Ofast")
# pragma GCC optimize ("unroll-loops")
# pragma GCC target("sse,sse2,sse3,ssse3,sse4,avx,avx2")
```

8.5 그동안 나온 알고리즘

suffix array   sparse table   Hash   게임 이론   MITM   이분탐색   flow   DSU dp - SOS dp, bit dp..   FFT 세그먼트 트리 + 스위핑   기하학   센트로이드   small to large   DNC KMP PBS 아호코라식   벌레캠프   삼분탐색   그리디   브루트포스   v-e+f   이분매칭   mobius function(약.배수) tree dp min-cut DP optimization 분할 정복을 이용한 거듭제곱 parametric search   HLD 단절점/단절선 convex hull trick   MCMF   PST   offline query   set/map two pointer trie   offline dynamic connectivity   다각형 점 판   manacher(팰린드롬 판정) floyd warshall   2-sat randomization   splay tree