2025 ICPC team note

Slower than Python3

Nov 2025

Contents

1 Data Structure

1.1 Fenwick Tree

1.2 Segment Tree

```cpp
// Segtree
#include <bits/stdc++.h>
#define INF 1000000007
using namespace std;

int seg[1<<18];

void build_seg(int idx, int l, int r){
    if(l==r){
        // seg[idx]=
        return;
    }
    int mid=(l+r)>>1;
    build_seg(2*idx,l,mid);
    build_seg(2*idx+1,mid+1,r);
    //seg[idx]=min(seg[2*idx],seg[2*idx+1]);
    // Some other operation
}

// 원소를 찾으면 disable도 같이 해주면 될 거 같음
void update_seg(int idx, int l, int r, int t_idx){
    if(l==r){
        seg[idx]=INF;
        return;
    }
    int mid=(l+r)>>1;
    if(t_idx<=mid) update_seg(2*idx,l,mid,t_idx);
    else update_seg(2*idx+1,mid+1,r,t_idx);
    //seg[idx]=min(seg[2*idx],seg[2*idx+1]);
}

int find_seg(int idx, int l, int r, int t_l, int t_r){
    if(t_l<=l && r<=t_r) return seg[idx];
    int mid=(l+r)>>1, ans=INF;
    if(t_l<=mid) //ans=min(ans,find_seg(2*idx,l,mid,t_l,t_r));
    if(t_r>mid) //ans=min(ans,find_seg(2*idx+1,mid+1,r,t_l,t_r));
    return ans;
}

int main()
{
    ios_base::sync_with_stdio(false); cin.tie(NULL);
    return 0;
}
```

1.3 Lazy Segment Tree

1.4 Persistent Segment Tree

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

#define N_node 2000000 // V log|Y_RANGE| scale
#define NX 100001
#define Y_RANGE 100001

class node{
public:
    node(){}
    node operator+(node b){
        node res;
        // Implement. 예시는 단순히 개수 세는 것.
        // 왼쪽에 더해지는 node의 l_idx, r_idx를 보존한다.
```

```
        res.l_idx=l_idx, res.r_idx=r_idx;
        res.cnt=cnt+b.cnt;
        return res;
    }
    node operator-(node b){
        node res;
        // Implement
        res.l_idx=l_idx, res.r_idx=r_idx;
        res.cnt=cnt-b.cnt;
        return res;
    }
    int l_idx=-1, r_idx=-1; // 여기서 idx는 본인의 idx를 말한다.
    long long cnt=0;
};

class PST{
public:
    int seg_root[NX]; // X-wise root idx.
    node seg[N_node]; // All node datas
    int cur_x=0, s_cnt=0; // s_cnt는 현재까지 사용한 node의 개수
    PST(){}
    void push(long long y){
        // Pusing one element generate anothor root.
        // Regardless of the val x.
        seg_root[cur_x]=s_cnt++;
        update_seg(seg_root[cur_x],(cur_x==0?-1:seg_root[cur_x-1]),
                   0,Y_RANGE-1,y,1);
        cur_x++;
    }
    node query(long long x1, long long x2,
               long long y1, long long y2){
        if(x1==0) return find_seg(seg_root[x2],0,Y_RANGE-1,y1,y2);
        return find_seg(seg_root[x2],0,Y_RANGE-1,y1,y2)
                -find_seg(seg_root[x1-1],0,Y_RANGE-1,y1,y2);
    }
    void clear(){
        for(int i=0; i<s_cnt; i++) seg[i]=node();
        cur_x=s_cnt=0;
    }

private:
    void update_seg(int idx, int prev_idx, int l, int r, int t_idx,
                    long long val){
        // previous segtree의 idx도 같이 관리. (첫 번째 layer의 경우는 -1)
        if(l==r){
            // Implement. 과거의 데이터에 덮어씌우는 것은 이부분이 유일.
            seg[idx].cnt=val;
            if(prev_idx!=-1) seg[idx]=seg[idx]+seg[prev_idx];
            return;
        }
        int mid=(l+r)>>1;
        if(t_idx<=mid){
            seg[idx].l_idx=s_cnt++;
            seg[idx].r_idx=(prev_idx==-1?-1:seg[prev_idx].r_idx);
            update_seg(seg[idx].l_idx,(prev_idx==-1?-1
                :seg[prev_idx].l_idx),l,mid,t_idx,val);
        }
        else{
            seg[idx].r_idx=s_cnt++;
            seg[idx].l_idx=(prev_idx==-1?-1:seg[prev_idx].l_idx);
            update_seg(seg[idx].r_idx,(prev_idx==-1?-1
                :seg[prev_idx].r_idx),mid+1,r,t_idx,val);
        }
        // 아래의 덧셈연산은 l_idx와 r_idx를 보존하기 위한 방법
        if(seg[idx].l_idx!=-1) seg[idx]=seg[idx]+seg[seg[idx].l_idx];
        if(seg[idx].r_idx!=-1) seg[idx]=seg[idx]+seg[seg[idx].r_idx];
    }
    node find_seg(int idx, int l, int r, int t_l, int t_r){
        if(t_l<=l && r<=t_r) return seg[idx];
        int mid=(l+r)>>1;
        node res;
        if(t_l<=mid && seg[idx].l_idx!=-1)
            res=res+find_seg(seg[idx].l_idx,l,mid,t_l,t_r);
        if(t_r>mid && seg[idx].r_idx!=-1)
            res=res+find_seg(seg[idx].r_idx,mid+1,r,t_l,t_r);
        return res;
    }
};

PST p;
```

```cpp
// LCA(sparse table)

#include <bits/stdc++.h>
#define endl '\n'

using namespace std;

#define N 100100
#define MAX 20 // Should satisfy (1<<(MAX-1)) >= N

vector<int> edges[N];
int sparse[N][MAX];
int level[N];

void DFS_init(int idx, int p){
    sparse[idx][0]=p;
    for(int i=0; i<edges[idx].size(); i++){
        int next=edges[idx][i];
        if(next!=p){
            level[next]=level[idx]+1;
            DFS_init(next,idx);
        }
    }
}

void sparse_init(int s){
    level[s]=0;
    DFS_init(s,-1);
    for(int r=1; r<MAX; r++){
        for(int i=0; i<N; i++){
            if(sparse[i][r-1]==-1) sparse[i][r]=-1;
            else sparse[i][r]=sparse[sparse[i][r-1]][r-1];
        }
    }
}

int get_LCA(int a, int b){
    if(level[a]>level[b]) swap(a,b);
    for(int r=MAX-1; r>=0; r--){
        if(sparse[b][r]!=-1 && level[a]<=level[sparse[b][r]])
            b=sparse[b][r];
    }
    if(a==b) return a;
    for(int r=MAX-1; r>=0; r--){
        if(sparse[a][r]!=-1 && sparse[b][r]!=-1
            && sparse[b][r]!=sparse[a][r]){
            a=sparse[a][r], b=sparse[b][r];
        }
    }
    return sparse[a][0];
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL);
    int n,m,a,b;
    cin>>n;
    for(int i=0; i<n-1; i++){
        cin>>a>>b;
        a--, b--;
        edges[a].push_back(b);
        edges[b].push_back(a);
    }
    sparse_init(0);
    cin>>m;
    for(int i=0; i<m; i++){
        cin>>a>>b;
        cout<<get_LCA(a-1,b-1)+1<<endl;
    }
}
```

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

#define N 1000
#define INF 1000000007

using namespace std;

vector<pair<int,int>> edges[N]; // next, cost.
// Must initialize after each testcase
int capa[N][N];
int flow[N][N];
bool inQ[N];
// (inQ[i]) means dist[i] changed ans need to be visited.
int pre[N];
int dist[N];

void add_edge(int a, int b, int w, int f){
    // a->b cost:w capa:f (Reverse edge has negative weight)
    edges[a].push_back({b,w}); capa[a][b]=f; flow[a][b]=0;
    edges[b].push_back({a,-w}); capa[b][a]=0; flow[b][a]=0;
}

void floyd_warshall(int s){
    // Only update pre[], dist[]
    fill(inQ, inQ+N, false);
    fill(dist, dist+N, INF);
    int ff=INF;
    queue<int> q;
    q.push(s);
    inQ[s]=true; // To maintain light queue
    dist[s]=0;
    while(!q.empty()){
        int idx=q.front(); inQ[idx]=false;
        q.pop();
        for(int i=0; i<edges[idx].size(); i++){
            int next=edges[idx][i].first, w=edges[idx][i].second;
            if(capa[idx][next]-flow[idx][next]>0
                && dist[idx]+w<dist[next]){
                dist[next]=dist[idx]+w;
                pre[next]=idx;
                if(!inQ[next]) q.push(next);
                inQ[next]=true;
            }
        }

    }
}

int MCMF(int s, int t){
    // Return minimum cost for maximum flow
    int res=0;
    while(true){
        floyd_warshall(s);
        if(dist[t]==INF) break;
        // Get flow
        int cur=t, ff=INF;
        while(cur!=s){
            ff=min(ff,capa[pre[cur]][cur]-flow[pre[cur]][cur]);
            cur=pre[cur];
        }
        // Flow update
        cur=t;
        while(cur!=s){
            flow[pre[cur]][cur]+=ff;
            flow[cur][pre[cur]]-=ff;
            cur=pre[cur];
        }
        res+=dist[t]*ff;
    }
    return res;
}
```



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

#define N 100000
```

```
vector<int> edges[N];
vector<int> rev_edges[N]; // To find maximum independent set.
int pre[N];
bool visited[N];

bool DFS(int idx){
    // Bipartite matching
    if(visited[idx]) return false;
    visited[idx]=true;
    for(int i=0; i<edges[idx].size(); i++){
        int next=edges[idx][i];
        if(pre[next]==-1
            || (!visited[pre[next]] && DFS(pre[next]))){
            pre[next]=idx;
            return true;
        }
    }
    return false;
}
// fill(visited, visited+n, 0); 을 매번 해줘야 한다.
// N^2이 나오는 게 정상
// if(DFS(i)) result++; 이렇게 유량을 구함.
```

4 Strings

4.1 Aho Corasic

```
// Aho-corasic
// Upper-case code임에 주의.

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

#define NUM_ALPHA 26
#define N 10000

class node{
public:
    node* next[NUM_ALPHA]={NULL,};
    node* pphi; // parent of phi ; i.e. 최대 접미사의 가장 마지막 노드.
    // 다음 원소는 직접 찾아야 함.
    int val; // 부모를 통해 찾아줘도 된다. // 0-based idx of character
    bool end_of_snippet=false;
    // End node of snippet. Propagated in BFS phase.
    int length_of_snippet;
    // Only valid of end of snippet is true. Propagated in BFS phase.
    int depth=0; // Actual length of the sequence
    bool end=true; // All next pointers are NULL
    node(){}
    node(int v){
        val=v;
    }
};

class aho_trie{
public:
    node* root= new node();
    aho_trie(){}
    bool is_s(string &s){
        return is_re(root, 0, s);
    }
    void insert_s(string &s){
        insert_re(root, 0, s);
    }
    void clear(){ // clear_all
        clear_re(root);
        for(int i=0; i<NUM_ALPHA; i++) root->next[i]=NULL;
    }
    void get_phi(){
        queue<pair<node*,node*>> phi_q; // cur_pos, pphi
        phi_q.push({root,root});
        while(!phi_q.empty()){
            pair<node*,node*> temp=phi_q.front();
            phi_q.pop();
            node* cur=temp.first;
            node* ptar=temp.second;
            if(cur->depth>1){ // To prevent phi[self]=self
                while(ptar!=root && (ptar->next[cur->val]==NULL))
                    ptar=ptar->pphi;
                if(ptar->next[cur->val]!=NULL)
                    ptar=ptar->next[cur->val];
            }
            cur->pphi=ptar;
            if(!cur->end_of_snippet && ptar->end_of_snippet){
                cur->end_of_snippet=true;
                cur->length_of_snippet=ptar->length_of_snippet;
            }

            for(int i=0; i<NUM_ALPHA; i++){
                if((cur->next[i])!=NULL){
                    phi_q.push({cur->next[i],ptar});
                }
            }
        }
```

```
    }
    void find_matching(string &s_ref, vector<int> &ans){
        int n=s_ref.size();
        ans.clear();
        node* ptar=root;
        for(int i=0; i<n; i++){
            int cur=s_ref[i]-'A'; // transition function
            while(ptar!=root && ptar->next[cur]==NULL)
                ptar=ptar->pphi;
            if(ptar->next[cur]!=NULL) ptar=ptar->next[cur];
            //cout<<(char)(ptar->val+'A')<<' '<<ptar->cnt<<'
            //'<<ptar->depth<<endl;
            if(ptar->end_of_snippet)
                ans.push_back(i+1-(ptar->length_of_snippet));
            if(ptar->end) ptar=ptar->pphi;
        }
    }

private:
    bool is_re(node *cur, int idx, string &s){
        if(idx==s.size()) return (cur->end_of_snippet);
        if((cur->next[s[idx]-'A'])==NULL) return false;
        return is_re(cur->next[s[idx]-'A'],idx+1,s);
    }
    void insert_re(node* cur, int idx, string &s){
        if(idx==s.size()){
            cur->end_of_snippet=true;
            cur->length_of_snippet=s.size();
            return;
        }
        cur->end=false;
        if((cur->next[s[idx]-'A'])==NULL){
            cur->next[s[idx]-'A']=new node(s[idx]-'A');
            cur->next[s[idx]-'A']->depth=idx+1;
        }
        insert_re(cur->next[s[idx]-'A'],idx+1,s);
    }
    void clear_re(node* cur){
        for(int i=0; i<NUM_ALPHA; i++){
            if((cur->next[i])!=NULL){
                clear_re(cur->next[i]);
                delete cur->next[i];
            }
        }
    }
};
```

```
aho_trie aho;
```

4.2 KMP

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

#define N 2020202
string s_ref;
string s_sni;
int phi[N];
vector<int> match_res;

void get_phi(){
    int m=s_sni.size();
    phi[0]=0;
    for(int i=1, j=0; i<m; i++){
        while(j>0 && s_sni[i]!=s_sni[j]) j=phi[j-1];
        if(s_sni[i]==s_sni[j]) j++;
        phi[i]=j;
    }
}

void KMP(){
    int n=s_ref.size(), m=s_sni.size();
    match_res.clear();
    for(int i=0, j=0; i<n; i++){
        while(j!=0 && s_ref[i]!=s_sni[j]) j=phi[j-1];
        if(s_ref[i]==s_sni[j]) j++;
        if(j==m) match_res.push_back(i+1-m), j=phi[j-1];
    }
}
```

4.3 Trie

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

using namespace std;

#define NUM_ALPHA 26

class node{
public:
    node* next[NUM_ALPHA]={NULL,};
    char val; // 대부분 필요없다.
    int cnt=0; // 여기서 끝나는 단어들의 개수
    node(){}
    node(char c){
```

```
        val=c;
    }
};

class trie{
public:
    node* root= new node();
    trie(){}
    bool is_s(string &s){
        return is_re(root, 0, s);
    }
    void insert_s(string &s){
        insert_re(root, 0, s);
    }
    void clear(){ // clear_all
        clear_re(root);
        for(int i=0; i<NUM_ALPHA; i++) root->next[i]=NULL;
    }

private:
    bool is_re(node *cur, int idx, string &s){
        if(idx==s.size()) return ((cur->cnt)>=1);
        // 여기까지 왔으면 그냥 true해도 되지 않나?
        // 처음에 들어올 때 때문인가
        if((cur->next[s[idx]-'A'])==NULL) return false;
        return is_re(cur->next[s[idx]-'A'],idx+1,s);
    }
    void insert_re(node* cur, int idx, string &s){
        if(idx==s.size()){
            cur->cnt++;
            return;
        }
        if((cur->next[s[idx]-'A'])==NULL)
            cur->next[s[idx]-'A']=new node(s[idx]);
        insert_re(cur->next[s[idx]-'A'],idx+1,s);
    }
    void clear_re(node* cur){
        for(int i=0; i<NUM_ALPHA; i++){
            if((cur->next[i])!=NULL){
                clear_re(cur->next[i]);
                delete cur->next[i];
            }
        }
    }
};

trie tr;
```

4.4 Suffix LCP

4.5 Manacher

5 Mathematics

5.1 Arithmetic Inverse

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

#define MOD 998244353
#define MAX_DIGIT 62

// (ak)*(bk)^(-1) === a*b^(-1) === (ak%P)*(bk%P)^(-1) (mod P)
long long ari_inv(long long num){
    // Calculate num^(MOD-2)
    long long res=1, mult=num;
    for(int i=0; i<MAX_DIGIT; i++){
        if((MOD-2)&(1LL<<i)){
            res=(res*mult)%MOD;
        }
        mult=(mult*mult)%MOD;
    }
    return res;
}
```

5.2 Sum Over Subsets

5.3 Floor Sum

5.4 FFT

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

typedef complex<double> cpx;
const int SZ = 1048576;

void FFT(cpx g[], bool inv = false){
    int n = SZ;
    for(int i = 1, j = 0; i < n; ++i){
        int b = n/2;
        while(!((j ^= b) & b)) b /= 2;
        if(i < j) swap(g[i], g[j]);
    }
```

```
    for(int k = 1; k < n; k *= 2){
        double a = (inv ? M_PI/k : -M_PI/k);
        cpx w(cos(a), sin(a));
        for(int i = 0; i < n; i += k*2){
            cpx wp(1, 0);
            for(int j = 0; j < k; ++j){
                cpx x = g[i+j], y = g[i+j+k] * wp;
                g[i+j] = x + y;
                g[i+j+k] = x - y;
                wp *= w;
            }
        }
    }
    if(inv){
        for(int i = 0; i < n; ++i)
            g[i] /= n;
    }
}
```

5.5 NTT

5.6 NTT + Chinese Remainder Thm

```
// NTT + CRT
#include <bits/stdc++.h>
#define endl '\n'
using namespace std;

typedef long long ll;

/* MOD = a 2^b + 1 일 때, [a, b, Primitive root] 순서쌍
N으로 사용하는 값이 2^b보다 크면 사용할 수 없다.
998244353 [119, 23, 3]
2281701377 [17, 27, 3]
2483027969 [37, 26, 3]
2113929217 [63, 25, 5]
104857601 [25, 22, 3]
1092616193 [521, 21, 3]
*/

ll SZ=1<<18;
#define MOD1 998244353
#define AA1 119
#define BB1 23
#define PR1 3
#define MOD2 2483027969
#define AA2 37
#define BB2 26
#define PR2 3
#define MAX_DIGIT 62

long long ari_inv(long long num, bool ver){
    long long res=1, mult=num;
    ll MOD = (ver?MOD1:MOD2);
    for(ll i=0; i<MAX_DIGIT; i++){
        if((MOD-2)&(1LL<<i)){
            res=(res*mult)%MOD;
        }
        mult=(mult*mult)%MOD;
    }
    return res;
}


void NTT(ll g[], bool ver, bool inv=false){
    ll MOD=(ver?MOD1:MOD2), AA=(ver?AA1:AA2), BB=(ver?BB1:BB2),
        PR=(ver?PR1:PR2);
    ll n = SZ;
    ll w_pow[n+1];
    ll bit_cnt=0;
    for(; bit_cnt<=BB; bit_cnt++){
        if((n>>bit_cnt) == 1) break;
    }
    assert ((n>>bit_cnt)==1);

    // w^n % MOD == 1 이어야 함.
    // w = PR^((p-1) / n) = PR ^ (a*2^(b-bit_cnt))
    ll w = 1;
    ll pow = AA<<(BB-bit_cnt);
    ll mul = PR;
    for(int i=0; i<MAX_DIGIT; i++){
        if(pow&(1LL<<i)) w=(w*mul)%MOD;
        mul=(mul*mul)%MOD;
    }
    w = (inv ? w : ari_inv(w,ver));

    for(ll i = 1, j = 0; i < n; ++i){
        ll b = n/2;
        while(!((j ^= b) & b)) b /= 2;
        if(i < j) swap(g[i], g[j]);
    }

    ll cnt=0;
    for(ll k = 1; k < n; k *= 2){
        cnt++;
```

```
        ll ww = w; // ww^(2*k) % MOD == 1 이어야 함.
        // ww=w^(n/(2*k))=w^(2^(bit_cnt-cnt))
        for(ll i=0; i<bit_cnt-cnt; i++) ww=(ww*ww)%MOD;
        for(ll i=0; i<k; i++) w_pow[i]=(i==0?1:(w_pow[i-1]*ww)%MOD);
        for(ll i = 0; i < n; i += k*2){
            ll wp = 1;
            for(ll j = 0; j < k; ++j){
                ll x = g[i+j], y = g[i+j+k]*w_pow[j];
                g[i+j] = (x + y);
                // MOD*MOD+MOD가 ll로 표현된다면 중간과정에서 %MOD 필요없음.
                g[i+j+k] = (x - y);
            }
        }
        for(int i=0; i<n; i++) g[i]%=MOD;
        // 위의 for문 동안 한 번만 방문
    }
    if(inv){
        ll ari_n = ari_inv(n,ver);
        for(ll i = 0; i < n; ++i)
            g[i] = (g[i]*ari_n)%MOD;
    }
}

inline ll large_mul(ll a, ll b){
    // 이것 때문에 TLE가 뜨진 않는다.
    ll MOD=(MOD1*MOD2);
    a=(a%MOD+MOD)%MOD;
    b=(b%MOD+MOD)%MOD;
    ll mul=a;
    ll res=0;
    for(int i=0; i<MAX_DIGIT; i++){
        if(b & (1LL<<i)) res=(res+mul)%MOD;
        mul=(mul+mul)%MOD;
    }
    return res;
}

/*
// for MOD1
NTT(datas1_M1, true, false);
NTT(datas2_M1, true, false);
for(ll i=0; i<SZ; i++) datas1_M1[i]=(datas1_M1[i]*datas2_M1[i])%MOD1;
NTT(datas1_M1, true, true);

// for MOD2
NTT(datas1_M2, false, false);
NTT(datas2_M2, false, false);
for(ll i=0; i<SZ; i++) datas1_M2[i]=(datas1_M2[i]*datas2_M2[i])%MOD2;
NTT(datas1_M2, false, true);

// Chinese Theorem
ll n1 = MOD2, n2=MOD1, MOD=(MOD1*MOD2);
ll s1 = ari_inv(MOD2, true), s2=ari_inv(MOD1, false);
for(int i=0; i<2*l; i++){
    datas1_M1[i] = (large_mul(datas1_M1[i]*n1,s1)
    + large_mul(datas1_M2[i]*n2,s2)) % MOD;
}
*/
```

5.7 Ternary Search

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

#define INF 1e18

long long f(long long x){
    long long res;
    // calculate f(x)
    return res;
}

long long ternary(long long s, long long e){
    // Return smallest x if there are multiple minimum values.
    while(3<=e-s){
        long long l=(s+s+e)/3, r=(s+e+e)/3;
        if(f(l)>f(r)) s=l;
        else e=r;
    }
    long long mx=INF, res;
    for(long long i=s; i<=e; i++){
        long long temp=f(i);
        if(mx>temp) mx=temp, res=i;
    }
    return res;
}
```

5.8 Gale-Shapley Algorithm

6 Geometry

6.1 Line Intersection

6.2 Basic Operations

6.3 Convex Hull

```cpp
// Convex hull
#include <bits/stdc++.h>
#define fi first
#define se second
using namespace std;
typedef long long ll;
#define N 100000

pair<ll,ll> datas[N]; // Vertices
// Convex hull algorithm never permute initial vertices.
int sorted_idx[N]; // Idx of 'sorted' datas.
int ans[N]; // Elements of convex hull

inline bool CCW(const pair<ll,ll>& a,
    const pair<ll,ll>& b, const pair<ll,ll>& c){
    return (b.fi-a.fi)*(c.se-a.se)-(b.se-a.se)*(c.fi-a.fi)>0;
    // 변 위의 점도 필요하다면 >=
}
inline long long dist(const pair<ll,ll>& a,const pair<ll,ll>& b){
    return (a.fi-b.fi)*(a.fi-b.fi)+(a.se-b.se)*(a.se-b.se);
}

class comp_c{
public:
    pair<ll,ll> mp;
    comp_c(pair<ll,ll> minpos){
        mp=minpos;
    }
    bool operator()(int a_idx, int b_idx){
        pair<ll,ll> a=datas[a_idx], b=datas[b_idx];
        ll com=(a.fi-mp.fi)*(b.se-mp.se)-(a.se-mp.se)*(b.fi-mp.fi);
        if(com==0) return dist(a,mp)<dist(b,mp);
        else return com>0;
    }
};

int convex_hull(int n){ // # number of vertices
    // Phase 1: Sort
    int m_idx=0;
    for(int i=1; i<n; i++) if(datas[m_idx]>datas[i]) m_idx=i;
    for(int i=0; i<n; i++) sorted_idx[i]=i;
    swap(sorted_idx[0], sorted_idx[m_idx]);
    sort(sorted_idx,sorted_idx+n,comp_c(datas[m_idx]));

    // Phase 2: Get Convexhull
    int st[N]; // Elements of st are the idx of datas
    int cur=0, st_cnt=0;
    // cur : # of visited elements of s,
    // st_cnt : # of elements in stack
    while(cur<n){
        while(st_cnt>=2 && !CCW(datas[st[st_cnt-2]],
        datas[st[st_cnt-1]],datas[sorted_idx[cur]])) st_cnt--;
        st[st_cnt++]=sorted_idx[cur++];
    }
    // Counter clockwise, starts with leftmost vertices.
    for(int i=0; i<st_cnt; i++) ans[i]=st[i];
    return st_cnt;
}
```

6.4 Rotating Calipers

```cpp
// Rotating calipers
#include <bits/stdc++.h>
#define fi first
#define se second

using namespace std;
typedef long long ll;

#define N 100000

pair<ll,ll> datas[N]; // Vertices
// Convex hull algorithm never permute initial vertices.
int sorted_idx[N]; // Idx of 'sorted' datas.
int ans[N]; // Elements of convex hull

inline bool CCW(const pair<ll,ll>& a, const pair<ll,ll>& b,
    const pair<ll,ll>& c){
    return (b.fi-a.fi)*(c.se-a.se)-(b.se-a.se)*(c.fi-a.fi)>0;
    // 변 위의 점도 필요하다면 >=
}
inline bool CCW_4(const pair<ll,ll>& a,
    const pair<ll,ll>& na, const pair<ll,ll>& b, pair<ll,ll> &nb){
    return (na.fi-a.fi)*(nb.se-b.se)-(na.se-a.se)*(nb.fi-b.fi)>=0;
}
inline long long dist(const pair<ll,ll>& a, const pair<ll,ll>& b){
```

```
    return (a.fi-b.fi)*(a.fi-b.fi)+(a.se-b.se)*(a.se-b.se);
}

class comp_c{
public:
    pair<ll,ll> mp;
    comp_c(pair<ll,ll> minpos){
        mp=minpos;
    }
    bool operator()(int a_idx, int b_idx){
        pair<ll,ll> a=datas[a_idx], b=datas[b_idx];
        ll com=(a.fi-mp.fi)*(b.se-mp.se)-(a.se-mp.se)*(b.fi-mp.fi);
        if(com==0) return dist(a,mp)<dist(b,mp);
        else return com>0;
    }
};

int convex_hull(int n){ // # number of vertices
    // Phase 1: Sort
    int m_idx=0;
    for(int i=1; i<n; i++) if(datas[m_idx]>datas[i]) m_idx=i;
    for(int i=0; i<n; i++) sorted_idx[i]=i;
    swap(sorted_idx[0], sorted_idx[m_idx]);
    sort(sorted_idx,sorted_idx+n,comp_c(datas[m_idx]));

    // Phase 2: Get Convexhull
    int st[N]; // Elements of st are the idx of datas
    int cur=0, st_cnt=0;
    // cur : # of visited elements of s,
    // st_cnt : # of elements in stack
    while(cur<n){
        while(st_cnt>=2 && !CCW(datas[st[st_cnt-2]],
            datas[st[st_cnt-1]],datas[sorted_idx[cur]])) st_cnt--;
        st[st_cnt++]=sorted_idx[cur++];
    }
    for(int i=0; i<st_cnt; i++) ans[i]=st[i];
    // Counter clockwise, starts with leftmost vertices.
    return st_cnt;
}

pair<int,int> rotating_calipers(int sz_hull){
    // # of vertices in convex hull.
    // ans[] must be filled before call this function.
    int mx_a, mx_b; // mx_a and mx_b are the idx of two vertices
                    // making the maximum distance.
    long long mx=0;
    for(int a=0, p=0; a<sz_hull; a++){
        while(p+1<sz_hull && CCW_4(datas[ans[a]], datas[ans[a+1]],
                datas[ans[p]], datas[ans[p+1]])){
            ll temp=dist(datas[ans[a]], datas[ans[p]]);
            if(mx<temp) mx=temp, mx_a=ans[a], mx_b=ans[p];
            p++;
        }
        ll temp=dist(datas[ans[a]], datas[ans[p]]);
        if(mx<temp) mx=temp, mx_a=ans[a], mx_b=ans[p];
    }
    return {mx_a, mx_b}; // Idx of initial vertices. i.e. datas[]
}
```

7 Miscellaneous

7.1 Mathematics

Green's Theorem Let C be a positively oriented, piecewise smooth, simple closed curve in a plane. Then

$$\oint_C (Ldx + Mdy) = \iint_D \left(\frac{\partial M}{\partial x} - \frac{\partial L}{\partial y}\right) dxdy,$$

where the path integral is taken counterclockwise.

Shoelace Lemma

```
for (int i=0; i<n; i++){
    area += X[i]*Y[(i+1)%len] - X[i]*Y[(i+len-1)%len];
}
return abs(area/2);
```

Burnside's Lemma

$$|X/G| = \sum_{g \in G} |X^g|$$

Pisano Period Pisano Period refers to the period with which Fibonacci numbers mod some P repeats. In other words, for some modulus P, its Pisano Period is equal to some M such that

$$F_a \equiv F_{(a\%M)} \mod P \quad \forall\, a.$$

for moduli of form 10^k, its $15 * 10^k$.

Euler Characteristics For a planar graph, the following identity holds:

$$x = V - E + F$$

where V = vertices, E = edges, F = faces