

# Team Notebook

SUST\_LateForSubmission

May 9, 2025

## Contents

<b>1 DP</b>	<b>2</b>	<b>3 Extra</b>	<b>6</b>	6.16 sieve all . . . . .	17
1.1 CHT . . . . .	2	3.1 Ordered Set Template . . . . .	6	6.17 totient . . . . .	17
1.2 Grundy . . . . .	2	3.2 Random Number Template . . . . .	6		
1.3 Knurth Optimization . . . . .	2	3.3 build . . . . .	6	<b>7 String</b>	<b>17</b>
1.4 LIS . . . . .	2	3.4 equation . . . . .	6	7.1 Aho . . . . .	17
1.5 digit dp optimize(1 memset) . . . . .	2	3.5 pragma . . . . .	6	7.2 String matching using bitset . . . . .	17
1.6 digit dp . . . . .	2			7.3 Trie . . . . .	17
1.7 divide and conquer . . . . .	2	<b>4 Geo</b>	<b>6</b>	7.4 kmp . . . . .	18
1.8 sos dp . . . . .	3	4.1 <i>geo<sub>t</sub>emplate<sub>2</sub></i> . . . . .	6	7.5 manacher . . . . .	18
				7.6 palindrome <sub>h</sub> ashing . . . . .	18
<b>2 DataStructure</b>	<b>3</b>	<b>5 Graph</b>	<b>13</b>	7.7 pallindromic tree . . . . .	18
2.1 BIT range update and query . . . . .	3	5.1 Flow . . . . .	13	7.8 suffix array occurence of substr in own string	19
2.2 BIT2D . . . . .	3			7.9 z algo . . . . .	19
2.3 Centroid decomposition . . . . .	3	<b>6 NumberTheory</b>	<b>13</b>		
2.4 DSU on tree . . . . .	3	6.1 CRT . . . . .	13	<b>8 Tree</b>	<b>19</b>
2.5 DSU . . . . .	4	6.2 NTT shortened . . . . .	13	8.1 Articulation bridge . . . . .	19
2.6 GP hash table . . . . .	4	6.3 NTT with any prime mod short . . . . .	14	8.2 Articulation point . . . . .	19
2.7 Mo on tree (number of distinct in a path) . .	4	6.4 bigmod . . . . .	14	8.3 Dijkstra . . . . .	19
2.8 Mo's . . . . .	4	6.5 extended euclid . . . . .	14	8.4 Euler tour . . . . .	19
2.9 Persistent Segment Tree . . . . .	5	6.6 fft . . . . .	14	8.5 Floyd Warshall . . . . .	19
2.10 Sparse table . . . . .	5	6.7 linear diphantine equation . . . . .	15	8.6 HLD(update on edge) . . . . .	19
2.11 <i>next<sub>s</sub>mallerprevious<sub>s</sub>maller</i> . . . . .	5	6.8 linear sieve . . . . .	15	8.7 HLD(update on node) . . . . .	20
2.12 ordered set . . . . .	5	6.9 mat expo . . . . .	15	8.8 Inverse Graph . . . . .	20
2.13 trie xor operation . . . . .	5	6.10 mobius . . . . .	15	8.9 LCA . . . . .	21
2.14 wavelet tree . . . . .	5	6.11 ncr for mod . . . . .	16	8.10 dfs <sub>t</sub> ree . . . . .	21
2.15 xor basis . . . . .	6	6.12 ntt . . . . .	16	8.11 strongly connected components . . . . .	21
		6.13 phi . . . . .	16		
		6.14 pollard rho . . . . .	16	<b>9 u blank</b>	<b>21</b>
		6.15 power tower . . . . .	16	9.1 blank . . . . .	21

# 1 DP

## 1.1 CHT

```
lli inf = LLONG_MIN;
//add lines with -m and -b and return -ans to
//make this code work for minimums.(not -x)
struct line {
lli m, b; mutable function<const line*() > succ; bool
operator < (const line& rhs) const {
if (rhs.b != inf) return m < rhs.m; const line* s = succ()
;
if (!s) return 0; lli x = rhs.m; return b - s->b < (s->m -
m) * x;};};

struct CHT : public multiset<line> {
bool bad(iterator y) { auto z = next(y);
if (y == begin()) { if (z == end()) return 0; return y ->
m == z -> m && y -> b <= z -> b; }
auto x = prev(y);
if (z == end()) return y -> m == x -> m && y -> b <= x ->
b;
return 1.0 * (x -> b - y -> b) * (z -> m - y -> m) >= 1.0
* (y -> b - z -> b) * (y -> m - x -> m);}
void add(lli m, lli b) {
auto y = insert({m, b}); y->succ = [=] { return next(y)
} == end() ? 0 : &*next(y);};
if (bad(y)) { erase(y); return; }
while (next(y) != end() && bad(next(y))) erase(next(y));
while (y != begin() && bad(prev(y))) erase(prev(y));}
lli query(lli x) { assert(!empty());
auto l = *lower_bound((line) { x, inf });
return l.m * x + l.b;};
};
```

## 1.2 Grundy

```
int Grundy(int n){ if (n <= 2) return 0;
if (dp[n] != -1)return dp[n];vector<int>vis(1005, 0);
for (int i = 1 ; i < n ; i++){ if (n - i != i)
{vis[(Grundy(n - i)^Grundy(i))]= 1;}}int p = 0;
while(vis[p]) p++;return dp[n] = p;}//pile divided into two
unequal pile
```

## 1.3 Knurth Optimization

```
for (int pos = 1; pos <= N; pos++) Opt[0][pos] = 0;
for (int groupNo = 1; groupNo <= K; groupNo++){ for (int pos
N + 1] = N;
for (int groupNo = 1; groupNo <= K; groupNo++){ for (int pos
= N; pos >= 1; pos--){ for (int endOfLast = Opt[
groupNo - 1][pos];endOfLast <= Opt[groupNo][pos + 1];
endOfLast++){
int ret = dp[groupNo - 1][endOfLast] + Cost(
endOfLast + 1, pos); if (dp[groupNo][pos] <=
ret) continue; dp[groupNo][pos] = ret;Opt[
groupNo][pos] = endOfLast;
}
}
}
```

## 1.4 LIS

```
vector<int>dp, lis;int track[n];dp.push_back(a[0]);track[0]
= 1;
for (int i = 1; i < n; i++){if (a[i] > dp.back()) dp.
push_back(a[i]), track[i] = dp.size();
else{int ind = lower_bound(dp.begin(), dp.end(), a[i]) - dp.
begin();
dp[ind] = a[i]; track[i] = ind + 1;}}int len = dp.size();
for (int i = n - 1 ; i >= 0 ; i--){if (track[i] == len) lis.
push_back(a[i]), len--;
}reverse(lis.begin(), lis.end());
```

## 1.5 digit dp optimize(1 memset)

```
ll dp[20][1030][2][2];ll casio[20][1030][2][2];int cur;int v
[20];
string s;int n;ll foo(int pos, int mask, int ok, bool other)
{
if (pos == -1){int tb = 0;int mxdig = -1;for (int i = 0 ; i
<= 9 ; i++){
if (mask & (1 << i))tb++, mxdig = max(mxdig, i);}return tb
== mxdig;};
ll &R = dp[pos][mask][ok][other];if (casio[pos][mask][ok][
other]==cur)
return R;
int dgt = 9;if (!ok){dgt = v[pos];}R = 0;for (int i = 0 ; i
<= dgt; i++){
int temp = mask;if (other)temp |= (1 << i);elseif (i){other
= 1;temp |= (1 << i);}
```

```
}if (i < dgt || ok)R += foo(pos - 1, temp , true, other);
else
R += foo(pos - 1, temp , false, other);}return R;}void pro()
{
memset(ll x;cin >> x;for(int i=0 ; i<20 ; i++){v[i] = x%10;
x/=10;};
cur++;ll ans = foo(18, 0, 0, 0);cout << ans << endl;}
```

## 1.6 digit dp

```
const int N=20;int a[N];
ll dp[N][11][2][2]; // digit dp te amar number generate hoy
emne : 0, 01, 02, 03, 04, 05.....066,0667
ll getsum(int pos, int dig, int n, bool ok, bool other){if(
pos>n){return 1;};
ll &R = dp[pos][dig][ok][other];if(R!=-1)return R;int
maxdigit = 9;if(!ok)
maxdigit = a[pos];ll res=0;for(int i=0 ; i<=maxdigit ; i++){
if(dig == i && other)
continue;if(i>0)other=1;if(i<maxdigit || ok)res+=getsum(pos
+1, i, n, true, other);else
res+=getsum(pos+1, i, n, false, other);}return R = res;}void
Solve(){string l, r;
cin>>l>>r;int n = r.size();r='*'+r;for(int i=0 ; i<=n ; i++)
{a[i]=r[i]-'0';}
memset(dp, -1, sizeof(dp));ll sumr = getsum(1, -1 , n, 0, 0)
;}
```

## 1.7 divide and conquer

```
mt19937 mt_rand(chrono::high_resolution_clock::now().
time_since_epoch().count());
ll Left = 1, Right = 0;ll cost(ll l, ll r) {while (Right < r
)Add(++Right);
while (Left > l)Add(--Left);while (Left < l)Remove(Left++);
while (Right > r)
Remove(Right--);return Totsum;};ll dp[2][N];
void compute(int group, int l, int r, int optl, int opttr) {
if (l > r)return ;
int mid = (l + r) / 2;dp[group & 1][mid] = LLONG_MAX;int
optnow = optl;
for (int k = optl ; k <= min(mid, opttr) ; k++) {ll ret = dp
[!(group & 1)][k] + cost(k + 1, mid);
if (ret < dp[group & 1][mid]) {dp[group & 1][mid] = ret;
optnow = k;}}
compute(group, l, mid - 1, optl, optnow);compute(group, mid
+ 1, r, optnow, opttr);}
```

```
void Solve() {cin >> n >> k;for (int i = 1 ; i <= n ; i++) {
    cin >> a[i];}
for (int i = 1 ; i <= n ; i++) {dp[1 & 1][i] = cost(1, i);}
    for (int i = 2 ; i <= k ; i++) {
compute(i, 1, n, 1, n);}cout << dp[k & 1][n] << endl;}
```

## 1.8 sos dp

```
void SOS_DP(){for(int i = 0; i<(1<<N); ++i)F[i] = A[i];
for(int i = 0;i < N; ++i)for(int mask = 0; mask < (1<<N); ++
mask){
if(mask & (1<<i))F[mask] += F[mask^(1<<i)];}}
//Istiak
const ll MLOG = 20; const ll MAXN = (1 << MLOG); ll dp[sz +
10], fre[sz + 10], mp[sz + 10];
// forward1: Propagates values from subsets to their
supersets
void forward1() { for (int bit = 0; bit < MLOG; ++bit) { for
(int i = 0; i < MAXN; ++i) { if (i & (1 << bit)) { dp[i]
+= dp[i ^ (1 << bit)]; } } } }
// backward1: Reverses the effect of forward1 by removing
contributions from supersets.This is used when dp[i]
contains info about all subsets of i, and we want to
isolate the info for only i.
void backward1() { for (int bit = 0; bit < MLOG; ++bit) {
for (int i = MAXN - 1; i >= 0; --i) { if (i & (1 << bit
)) { dp[i] -= dp[i ^ (1 << bit)]; } } } }
// forward2: Propagates values from supersets to their
subsets
void forward2() { for (int bit = 0; bit < MLOG; ++bit) { for
(int i = MAXN - 1; i >= 0; --i) { if (i & (1 << bit))
{ dp[i ^ (1 << bit)] += dp[i]; } } } }
// backward2: Reverses the effect of forward2 by removing
contributions from subsets.This is used when dp[i]
contains info about all supersets of i, and we want to
isolate the info for only i.
void backward2() { for (int bit = 0; bit < MLOG; ++bit) {
for (int i = 0; i < MAXN; ++i) { if (i & (1 << bit)) {
dp[i ^ (1 << bit)] -= dp[i]; } } } }
memset(dp, 0, sizeof(dp)); memset(fre, 0, sizeof(fre));
memset(mp, 0, sizeof(mp));}
```

## 2 DataStructure

### 2.1 BIT range update and query

```
const int N = 3e5 + 9;
struct BIT {
    long long M[N], A[N];
    BIT() {memset(M, 0, sizeof M);memset(A, 0, sizeof A);}
    void update(int i, long long mul, long long add) {
        while (i < N) {M[i] += mul;A[i] += add;i |= (i + 1);}
    }
    void upd(int l, int r, long long x) {
        update(l, x, -x * (1 - 1));update(r, -x, x * r);}
    long long query(int i) {
        long long mul = 0, add = 0;int st = i;
        while (i >= 0) {mul += M[i];add += A[i];i = (i & (i + 1))
- 1;}
        return (mul * st + add);}
    long long query(int l, int r) {
        return query(r) - query(l - 1);} } t;
```

### 2.2 BIT2D

```
#include<bits/stdc++.h>using namespace std;const int N =
1010;
struct BIT2D { long long M[N][N][2], A[N][N][2];BIT2D() {
memset(M, 0, sizeof M); memset(A, 0, sizeof A);}
void upd2(long long t[N][N][2], int x, int y, long long mul,
long long add) {
for(int i = x; i < N; i += i & -i) { for(int j = y; j < N; j
+= j & -j) {
t[i][j][0] += mul;t[i][j][1] += add;}}
void upd1(int x, int y1, int y2, long long mul, long long
add) {
upd2(M, x, y1, mul, -mul * (y1 - 1));upd2(M, x, y2, -mul,
mul * y2);
upd2(A, x, y1, add, -add * (y1 - 1));upd2(A, x, y2, -add,
add * y2);}
void upd(int x1, int y1, int x2, int y2, long long val) {
upd1(x1, y1, y2, val, -val * (x1 - 1));upd1(x2, y1, y2, -val
, val * x2);}
long long query2(long long t[N][N][2], int x, int y) { long
long mul = 0, add = 0;
for(int i = y; i > 0; i -= i & -i) { mul += t[x][i][0];add
+= t[x][i][1];}
return mul * y + add;}long long query1(int x, int y) {long
long mul = 0, add = 0;
for(int i = x; i > 0; i -= i & -i) { mul += query2(M, i, y);
add += query2(A, i, y);}
return mul * x + add; } long long query(int x1, int y1, int
x2, int y2) {
return query1(x2, y2) - query1(x1 - 1, y2) - query1(x2, y1 -
1) + query1(x1 - 1, y1 - 1);}
```

```
} t; int main() { int n, m; cin >> n >> m; for(int i = 1; i
<= n; i++) { for(int j = 1; j <= m; j++) {
int k; cin >> k; t.upd(i, j, i, j, k);} } int q; cin >> q;
while(q--) {
int ty, x1, y1, x2, y2; cin >> ty; if(ty == 1) { long long
val;
cin >> x1 >> y1 >> x2 >> y2 >> val;
t.upd(x1, y1, x2, y2, val); // add val from top-left(x1, y1)
to bottom-right (x2, y2);
} else { cin >> x1 >> y1 >> x2 >> y2;
cout << t.query(x1, y1, x2, y2) << '\n'; // output sum from
top-left(x1, y1) to bottom-right (x2, y2);
}}return 0;}
```

### 2.3 Centroid decomposition

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e5 + 9;vector<int> g[N];
int sz[N];int tot, done[N], cenpar[N];
void calc_sz(int u, int p) {tot++;
sz[u] = 1;for (auto v : g[u]) {if(v == p || done[v])
continue;
calc_sz(v, u);sz[u] += sz[v];}
}int find_cen(int u, int p) {for (auto v : g[u]) {
if(v == p || done[v]) continue;else if(sz[v] > tot / 2)
return find_cen(v, u);
}return u;}void decompose(int u, int pre) {
tot = 0;calc_sz(u, pre);int cen = find_cen(u, pre);cenpar[
cen] = pre;
done[cen] = 1;for(auto v : g[cen]) {if(v == pre || done[v])
continue;
decompose(v, cen);}int dep[N];void dfs(int u, int p = 0) {
for(auto v : g[u]) {if(v == p) continue;dep[v] = dep[u] + 1;
dfs(v, u);}int main() {ios_base::sync_with_stdio(0);
cin.tie(0);int n;cin >> n;for(int i = 1; i < n; i++) {
int u, v;cin >> u >> v;g[u].push_back(v);g[v].push_back(u);}
decompose(1, 0);for(int i = 1; i <= n; i++) g[i].clear();int
root;
for(int i = 1; i <= n; i++) {g[cenpar[i]].push_back(i);
g[i].push_back(cenpar[i]);if (cenpar[i] == 0) root = i;}
dfs(root);for(int i = 1; i <= n; i++) cout << char(dep[i] +
'A') << ' ';return 0;}
```

### 2.4 DSU on tree

```
const int N = 1e5 + 9;
```

```
vector<int> g[N];
int ans[N], col[N], sz[N], cnt[N];
bool big[N];
void dfs(int u, int p) {sz[u] = 1;
    for (auto v : g[u]) {if (v == p) continue;dfs(v, u);sz[u]
        += sz[v];}}
void add(int u, int p, int x) {cnt[col[u]] += x;
    for (auto v : g[u]) {if (v == p || big[v] == 1) continue;
        add(v, u, x);}}
void dsu(int u, int p, bool keep) {
    int bigchild = -1, mx = -1;
    for (auto v : g[u]) {if (v == p) continue;if (sz[v] > mx) mx
        = sz[v], bigchild = v;}
    for (auto v : g[u]) {if (v == p || v == bigchild) continue;
        dsu(v, u, 0);}
    if (bigchild != -1) dsu(bigchild, u, 1), big[bigchild] = 1;
    add(u, p, 1);ans[u] = cnt[u];
    if (bigchild != -1) big[bigchild] = 0;if (keep == 0) add(u,
        p, -1);}
```

## 2.5 DSU

```
int n,m;
int parent[MAX],Rank[MAX];
void Init(int n){for(int i=0 ;i<n ;i++)Rank[i]=1,parent[i]=
    i;}
int Find_parent(int v) {if (v == parent[v]) {return v;}
    return parent[v] = Find_parent(parent[v]);}
void Union(int a, int b) {
    a = Find_parent(a);b = Find_parent(b);if (a != b) {if (Rank[
        a] > Rank[b]) {swap (a, b);}parent[a] = b;Rank[b] +=
        Rank[a];}}
```

## 2.6 GP hash table

```
#include <bits/stdc++.h>
using namespace std;
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
struct custom_hash{
    static uint64_t splitmix64(uint64_t x){x += 0
        x9e3779b97f4a7c15;x = (x ^ (x >> 30)) * 0
        xbf58476d1ce4e5b9;x = (x ^ (x >> 27)) * 0
        x94d049bb133111eb;return x ^ (x >> 31);}
    size_t operator()(uint64_t x) const{static const uint64_t
        FIXED_RANDOM = chrono::steady_clock::now().
```

```
time_since_epoch().count();return splitmix64(x +
    FIXED_RANDOM);}};
gp_hash_table<int, int, custom_hash> mp;
```

## 2.7 Mo on tree (number of distinct in a path)

```
const int mod = 1e9 + 7, LG = 18;const int N = 2e5 + 6;const
    int BLOCK_SIZE = 450;int a[N];
vector<int>node[N];int starttime[N], endtime[N];int ft[N];
    int par[N][LG + 1], dep[N], sz[N];int timer = 1;
void dfs(int u, int p = 0){
    ft[timer] = u;starttime[u] = timer++;par[u][0] = p;dep[u] =
        dep[p] + 1;sz[u] = 1;
    for (int i = 1; i <= LG; i++){par[u][i] = par[par[u][i -
        1]][i - 1];}
    for (auto v : node[u]){if (v == p) continue;dfs(v, u);sz[u]
        += sz[v];ft[timer] = u;endtime[u] = timer++;}
    int lca(int u, int v){ // ache already}
    int freq[N];int colour[N];int res;
    void operation(int id){
        int curnode = ft[id];int c = a[curnode];
        if (freq[curnode] == 0){colour[c]++;
            if (colour[c] == 1)res++;}
        else{colour[c]--;if (colour[c] == 0)res--;}freq[curnode] ^=
            1;}
    void Solve(){
        int n, q;
        while (cin >> n >> q){
            set<ll>st;map<ll , ll>m;
            for (int i = 1 ; i <= n ; i++){cin >> a[i];}int tot = 0;
            for (int i = 1 ; i <= n ; i++){if (m[a[i]])a[i] = m[a[i]];
                else{m[a[i]] = ++tot;a[i] = m[a[i]]};}
            for (int i = 1 ; i < n ; i++){int u, v;cin >> u >> v;node[u
                ].push_back(v);node[v].push_back(u);}
            dfs(1);ll ans[q + 1];Query queries[q];
            for (int i = 0 ; i < q ; i++){int u, v, c;cin >> u >> v;int
                lc = lca(u, v);
                if (dep[u] > dep[v])swap(u, v);
                if (lc == u || lc == v)queries[i] = {starttime[u], starttime
                    [v], i + 1, 1, lc, -1};
                else queries[i] = {endtime[u], starttime[v], i + 1, 1, lc,
                    1};}
            sort(queries, queries + q);
            int Left = 1, Right = 0;
            for (auto i : queries){
                int l = i.l;int r = i.r;int id = i.idx;int c = i.c;int type
                    = i.type;int lc = i.lc;
```

```
while (Right < r)operation(++Right);
while (Left > l)operation(--Left);
while (Left < l)operation(Left++);
while (Right > r)operation(Right--);
if (type == 1){operation(starttime[lc]);}
ans[id] = res;
if (type == 1)operation(starttime[lc]);}
for (int i = 1 ; i <= q ; i++){cout << ans[i] << endl;}}
```

## 2.8 Mo's

```
const int mod = 1e9 + 7;const int N = 5e5 + 6;const int
    BLOCK_SIZE = 500;
struct Query {
    int l, r, idx, lc, type;
    bool operator<(const Query &y) const {
        // Current query x is being compared with other query y
        int x_block = l / BLOCK_SIZE;int y_block = y.l / BLOCK_SIZE;
        // If x and y both lie in the same block, sort in non
            decreasing order of endpoint
        if (x_block == y_block)return r < y.r;
        // x and y lie in different blocks
        return x_block < y_block;}};
ll nc3(ll x){if (x < 3)return 0;return (x * (x - 1) * (x -
    2)) / 6;}
int a[N];ll last[N];ll freq[N];ll res;
void Add(int i){int x = a[i];res -= last[x];freq[x]++;last[x]
    = nc3(freq[x]);res += last[x];}
void Remove(int i){int x = a[i];res -= last[x];freq[x]--;
    last[x] = nc3(freq[x]);res += last[x];}
void Solve(){
    int n, q;cin >> n >> q;
    for (int i = 1 ; i <= n ; i++){cin >> a[i];}
    vector<Query>queries;ll ans[q + 1];
    for (int i = 1 ; i <= q ; i++){int l, r;cin >> l >> r;
        queries.push_back({l, r, i});}
    sort(queries.begin(), queries.end());
    int Left = 1, Right = 0;
    for (auto i : queries){int l = i.l;int r = i.r;int id = i.
        idx;
        while (Right < r)Add(++Right);
        while (Left > l)Add(--Left);
        while (Left < l)Remove(Left++);
        while (Right > r)Remove(Right--);
        ans[i.idx] = res;}
    for (int i = 1 ; i <= q ; i++)cout << ans[i] << endl;}
    //number of triple(l, r) a[i] = a[j] = a[k]
```

## 2.9 Persistent Segment Tree

```
#include<bits/stdc++.h>using namespace std;
struct nd{long long sum;nd *left;nd *right;nd(long long data)
{sum=data;}}
nd(nd l,nd r){sum=l.sum+r.sum;left=&l;right=&r;}};
int n;vector<nd>states;
nd build(int start,int end){
if(start==end)return nd(0);int mid=(start+end)/2;return nd(
    build(start,mid),build(mid+1,end));}
nd update(nd root,int start,int end,int pos,int val){if(
    start==end)return nd(val);
int mid=(start+end)/2;return pos<=mid?nd(update(*root.left,
    start,mid,pos,val),*root.right):nd(*root.left,update(*
    root.right,mid+1,end,pos,val));}
void solve(){cin>>n;states.push_back(build(0,n-1));states.
    push_back(update(states.back(),0,n-1,4,3));return;}
int main(){ios_base::sync_with_stdio(0);cin.tie(0);solve();
    return 0;}
```

## 2.10 Sparse table

```
int Table[N][22], a[N];
void Build(int n){
for (int i = 1 ; i <= n ; i++)Table[i][0] = a[i];
for (int k = 1 ; k < 22 ; k++){
for (int i = 1 ; i + (1 << k) - 1 <= n ; i++)Table[i][k] =
    min(Table[i][k - 1], Table[i + (1 << (k - 1))][k - 1])
    ;}}
int Query(int l, int r){int k = log2(r - l + 1);return min(
    Table[l][k], Table[r - (1 << k) + 1][k]);}
```

## 2.11 next<sub>s</sub>mallerprevious<sub>s</sub>smaller

```
ll Next_smaller[N + 2];ll Prev_smaller[N + 2];
void NEXTSMALLER(){stack<int>st;
for (int i = 1; i <= n; i++){if (st.empty()){st.push(i);}
else{while (!st.empty() && a[st.top()] > a[i]){Next_smaller[
    st.top()] = i;st.pop();}st.push(i);}}
while (!st.empty()){Next_smaller[st.top()] = n + 1;st.pop()
    ;}}
void PREVSMAILLER(){stack<int>st;
for (int i = n; i >= 1; i--){if (st.empty()){st.push(i);}
else{while (!st.empty() && a[st.top()] > a[i]){Prev_smaller[
    st.top()] = i;st.pop();}st.push(i);}}
while (!st.empty()){Prev_smaller[st.top()] = 0;st.pop();}}
//priority_queue<int,vector<int>, greater<int>> >pq;
```

## 2.12 ordered set

```
#include<ext/pb_ds/assoc_container.hpp>#include<ext/pb_ds/
    tree_policy.hpp>using namespace std;using namespace
    __gnu_pbds;
template<class T> using ordered_set =
    tree<T, null_type, less<T>, rb_tree_tag,
    tree_order_statistics_node_update>; // find_by_order,
    order_of_key
//1 2 2 3 3 7
//greater->>descending order => 7 3 2 1
//less ->>ascending order => 1 2 3 7
//less_equal -> ascending but in duplicate value => 1 2 2 3
    3 7 so this will work as multiset
// finding kth element - 4th query
cout << "0th element: " << *A.find_by_order(0) << endl;
// finding number of elements smaller than X - 3rd query
cout << "No. of elems smaller than 6: " << A.order_of_key(6)
    << endl; //
```

## 2.13 trie xor operation

```
int Trie[35 * N][2];int root = 1;int cnt[35 * N];int cur =
    1;
void Update(ll x, ll value){int start = root;cnt[start] +=
    value;
for (int i = 32 ; i >= 0 ; i--){bool bit = x & (1LL << i);
if (Trie[start][bit] == 0){Trie[start][bit] = ++cur;}
start = Trie[start][bit];cnt[start] += value;}}
ll MaxQuery(ll x){int start = root;ll ans = 0;
for (int i = 32 ; i >= 0 ; i--){bool bit = x & (1LL << i);
if (Trie[start][1 ^ bit] == 0 || cnt[Trie[start][1 ^ bit]]
    == 0){ans = ans;}
else {ans += (1LL << i);bit ^= 1;start = Trie[start][bit];}
return ans;}
ll MinQuery(ll x){int start = root;ll ans = 0;
for (int i = 32 ; i >= 0 ; i--){bool bit = x & (1LL << i);
if (Trie[start][bit]) {ans = ans;}
else {ans += (1LL << i);bit ^= 1;start = Trie[start][bit];}
return ans;}}
```

## 2.14 wavelet tree

```
const int MAXN = (int)3e5 + 9;const int MAXV = (int)1e9 + 9;
//maximum value of any element in array
//array values can be negative too, use appropriate minimum
    and maximum value
```

```
struct wavelet_tree {int lo, hi;wavelet_tree *l, *r;int *b,
    *c, bsz, csz; // c holds the prefix sum of elements
wavelet_tree() {lo = 1;hi = 0;bsz = 0;csz = 0, l = NULL;r =
    NULL;}}
void init(int *from, int *to, int x, int y) {lo = x, hi = y;
    if (from >= to) return;
int mid = (lo + hi) >> 1;auto f = [mid](int x) {return x <=
    mid;};
b = (int*)malloc((to - from + 2) * sizeof(int));bsz = 0;b[
    bsz++] = 0;
c = (int*)malloc((to - from + 2) * sizeof(int));csz = 0;c[
    csz++] = 0;
for (auto it = from; it != to; it++) {b[bsz] = (b[bsz - 1] +
    f(*it));c[csz] = (c[csz - 1] + (*it));bsz++;csz++;}
if (hi == lo) return;
auto pivot = stable_partition(from, to, f);
l = new wavelet_tree();
l->init(from, pivot, lo, mid);
r = new wavelet_tree();
r->init(pivot, to, mid + 1, hi);}
//kth smallest element in [l, r]
//for array [1,2,1,3,5] 2nd smallest is 1 and 3rd smallest
    is 2
int kth(int l, int r, int k) {if (l > r) return 0;if (lo ==
    hi) return lo;int inLeft = b[r] - b[l - 1], lb = b[l -
    1], rb = b[r];
if (k <= inLeft) return this->l->kth(lb + 1, rb, k);return
    this->r->kth(l - lb, r - rb, k - inLeft);}
//count of numbers in [l, r] Less than or equal to k
int LTE(int l, int r, int k) {if (l > r || k < lo) return 0;
    if (hi <= k) return r - l + 1;int lb = b[l - 1], rb = b
    [r];return this->l->LTE(lb + 1, rb, k) + this->r->LTE(l
    - lb, r - rb, k);}
//count of numbers in [l, r] equal to k
int count(int l, int r, int k) {if (l > r || k < lo || k >
    hi) return 0;if (lo == hi) return r - l + 1;int lb = b[
    l - 1], rb = b[r];int mid = (lo + hi) >> 1;if (k <= mid
    ) return this->l->count(lb + 1, rb, k);return this->r->
    count(l - lb, r - rb, k);}
//sum of numbers in [l, r] less than or equal to k
int sum(int l, int r, int k) {if (l > r || k < lo) return 0;
    if (hi <= k) return c[r] - c[l - 1];int lb = b[l - 1],
    rb = b[r];return this->l->sum(lb + 1, rb, k) + this->r
    ->sum(l - lb, r - rb, k);}
~wavelet_tree() {delete l;delete r;};
wavelet_tree t;int a[MAXN];
int main() {
int i, j, k, n, m, q, l, r;cin >> n;for (i = 1; i <= n; i++)
    cin >> a[i];t.init(a + 1, a + n + 1, -MAXV, MAXV);}
```

```
//beware! after the init() operation array a[] will not be
samecin >> q;
while (q--) {int x;cin >> x;cin >> l >> r >> k;
if (x == 0) {kth smallestcout << t.kth(l, r, k) << endl;}
else if (x == 1) {less than or equal to Kcout << t.
LTE(l, r, k) << endl;} else if (x == 2) {count
occurence of K in [l, r]cout << t.count(l, r, k) <<
endl;}
if (x == 3) {sum of elements less than or equal to K in [l
, r]cout << t.sum(l, r, k) << endl;}}return 0;}
```

## 2.15 xor basis

```
struct XorBasis { vector<ll> basis;
ll N = 0, tmp = 0; void add(ll x) {
N++; tmp |= x;for (auto &i : basis) x = min(x, x ^ i);if (!x
) return;
for (auto &i : basis) if ((i ^ x) < i) i ^= x;basis.
push_back(x);
sort(basis.begin(), basis.end());}
ll size() {return (ll)basis.size();}
}void clear() {N = 0; tmp = 0;basis.clear();}
}bool possible(ll x) {for (auto &i : basis) x = min(x, x ^ i
);
return !x;}ll maxxor(ll x = 0) {for (auto &i : basis) x =
max(x, x ^ i);
return x;}ll minxor(ll x = 0) {
for (auto &i : basis) x = min(x, x ^ i);
return x;}ll cntxor(ll x) {if (!possible(x)) return 0LL;//
return (1LL<<(N-size()));}
ll ans = 1LL;for (int i = 0; i < N - size(); i++)ans = (ans
* 2) % MOD;
return ans;}ll sumOfAll() {ll ans = tmp * (1LL << (N - 1));
return ans;}ll kth(ll k) {ll sz = size();if (k > (1LL << sz)
) return -1;
k--; ll ans = 0;for (ll i = 0; i < sz; i++) if (k >> i & 1)
ans ^= basis[i];
return ans;}} xb;
```

## 3 Extra

### 3.1 Ordered Set Template

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;
```

```
#define ordered_set tree<lli, null_type,less<lli>,
rb_tree_tag,tree_order_statistics_node_update>
```

### 3.2 Random Number Template

```
#include <random>
#include <chrono>
using namespace __gnu_pbds; using namespace std;
mt19937 rng(chrono::steady_clock::now().time_since_epoch().
count());
```

### 3.3 build

```
//windows
{
"shell_cmd": "g++.exe -std=c++17 \"$file\" -o \"
$file_base_name\".exe && \"$file_base_name.exe\"<\"D://
c programes/in.txt\">\"D://c programes/out.txt\"\"",
"shell":true,
"working_dir": "$file_path",
"selector": "source.c, source.cpp, source.c++"
}

//linux
{
"cmd" : ["g++ -std=c++17 \"$file_name\" && timeout 10s ./a.
out <~/Code/input.txt | head -n 2000000 | head -c
50000000 >~/Code/output.txt"],
"selector" : "source.c,source.cpp,source.c++",
"shell": true,
"working_dir" : "$file_path"
}
```

### 3.4 equation

Some properties of bitwise operations:  
 $a|b = ab + a\&b$ ,  $a(a\&b) = (a|b)b$ ,  
 $b(a\&b) = (a|b)a$ ,  $(a\&b)(a|b) = ab$   
Addition:  $a+b = a|b + a\&b$ ,  $a+b = ab + 2(a\&b)$   
Subtraction:  
 $a-b = (a(a\&b)) - ((a|b)a)$ ,  $a-b = ((a|b)b) - ((a|b)a)$   
 $,a-b = (a(a\&b)) - (b(a\&b))$ ,  $a-b = ((a|b)b) - (b(a\&b))$

### 3.5 pragma

```
// #pragma GCC optimize("O3,unroll-loops,Ofast")
// #pragma GCC target("avx2")
```

## 4 Geo

### 4.1 *geo<sub>t</sub>emplate<sub>2</sub>*

```
#include <bits/stdc++.h>using namespace std;
// https://victorlecomte.com/cp-geo.pdf
const int N = 3e5 + 9;const double inf = 1e100;const double
eps = 1e-9;const double PI = acos((double)-1.0);
int sign(double x) { return (x > eps) - (x < -eps); }
struct PT{double x, y;PT() { x = 0, y = 0; }PT(double x,
double y) : x(x), y(y) {}PT(const PT &p) : x(p.x), y(p.
y) {}PT operator+(const PT &a) const { return PT(x + a.
x, y + a.y); }
PT operator-(const PT &a) const { return PT(x - a.x, y - a.y
); }PT operator*(const double a) const { return PT(x *
a, y * a); }
friend PT operator*(const double &a, const PT &b) { return
PT(a * b.x, a * b.y); }PT operator/(const double a)
const { return PT(x / a, y / a); }bool operator==(PT a)
const { return sign(a.x - x) == 0 && sign(a.y - y) ==
0; }
bool operator!=(PT a) const { return !(*this == a); }bool
operator<(PT a) const { return sign(a.x - x) == 0 ? y <
a.y : x < a.x; }
bool operator>(PT a) const { return sign(a.x - x) == 0 ? y >
a.y : x > a.x; }double norm() { return sqrt(x * x + y
* y); }
double norm2() { return x * x + y * y; }PT perp() { return
PT(-y, x); }double arg() { return atan2(y, x); }
PT truncate(double r){ // returns a vector with norm r and
having same directiondouble k = norm();if (!sign(k))
return *this;r /= k;return PT(x * r, y * r);};
istream &operator>>(istream &in, PT &p) { return in >> p.x
>> p.y; }
ostream &operator<<(ostream &out, PT &p) { return out << "("
<< p.x << "," << p.y << ")"; }
inline double dot(PT a, PT b) { return a.x * b.x + a.y * b.y
; }
inline double dist2(PT a, PT b) { return dot(a - b, a - b);
}
inline double dist(PT a, PT b) { return sqrt(dot(a - b, a -
b)); }
```



```

inline double cross(PT a, PT b) { return a.x * b.y - a.y * b.x; }
inline double cross2(PT a, PT b, PT c) { return cross(b - a, c - a); }
inline int orientation(PT a, PT b, PT c) { return sign(cross(b - a, c - a)); }
PT perp(PT a) { return PT(-a.y, a.x); }
PT rotateccw90(PT a) { return PT(-a.y, a.x); }
PT rotatecw90(PT a) { return PT(a.y, -a.x); }
PT rotateccw(PT a, double t) { return PT(a.x * cos(t) - a.y * sin(t), a.x * sin(t) + a.y * cos(t)); }
PT rotatecw(PT a, double t) { return PT(a.x * cos(t) + a.y * sin(t), -a.x * sin(t) + a.y * cos(t)); }
double SQ(double x) { return x * x; }
double rad_to_deg(double r) { return (r * 180.0 / PI); }
double deg_to_rad(double d) { return (d * PI / 180.0); }
double get_angle(PT a, PT b){double costheta = dot(a, b) / a.norm() / b.norm();return acos(max((double)-1.0, min((double)1.0, costheta)));}
bool is_point_in_angle(PT b, PT a, PT c, PT p){ // does point p lie in angle <bac
assert(orientation(a, b, c) != 0);if (orientation(a, c, b) < 0)swap(b, c);
return orientation(a, c, p) >= 0 && orientation(a, b, p) <= 0;}
bool half(PT p){return p.y > 0.0 || (p.y == 0.0 && p.x < 0.0);}
void polar_sort(vector<PT> &v){ // sort points in counterclockwise
sort(v.begin(), v.end(), [](PT a, PT b){ return make_tuple(half(a), 0.0, a.norm2()) < make_tuple(half(b), cross(a, b), b.norm2()); });}
void polar_sort(vector<PT> &v, PT o){ // sort points in counterclockwise with respect to point o
sort(v.begin(), v.end(), [&](PT a, PT b){ return make_tuple(half(a - o), 0.0, (a - o).norm2()) < make_tuple(half(b - o), cross(a - o, b - o), (b - o).norm2()); });}
struct line{PT a, b; // goes through points a and bPT v; double c; // line form: direction vec [cross] (x, y) = c
line() {}// direction vector v and offset cline(PT v, double c) : v(v), c(c){auto p = get_points();a = p.first;b = p.second;}
// equation ax + by + c = 0
line(double _a, double _b, double _c) : v({_b, -_a}), c(-_c){auto p = get_points();a = p.first;b = p.second;}
// goes through points p and q
line(PT p, PT q) : v(q - p), c(cross(v, p)), a(p), b(q) {}
pair<PT, PT> get_points()

```

```

{ // extract any two points from this linePT p, q;double a = -v.y, b = v.x; // ax + by = cif (sign(a) == 0){p = PT(0, c / b);q = PT(1, c / b);}
else if (sign(b) == 0){p = PT(c / a, 0);q = PT(c / a, 1);}
else{p = PT(0, c / b);q = PT(1, (c - a) / b);}return {p, q};}
// ax + by + c = 0array<double, 3> get_abc(){double a = -v.y, b = v.x;return {a, b, -c};}
// 1 if on the left, -1 if on the right, 0 if on the line
int side(PT p) { return sign(cross(v, p) - c); }
// line that is perpendicular to this and goes through point p
line perpendicular_through(PT p) { return {p, p + perp(v)}; }
// translate the line by vector t i.e. shifting it by vector t
line translate(PT t) { return {v, c + cross(v, t)}; }
// compare two points by their orthogonal projection on this line
// a projection point comes before another if it comes first according to vector v
bool cmp_by_projection(PT p, PT q) { return dot(v, p) < dot(v, q); }
line shift_left(double d){PT z = v.perp().truncate(d);return line(a + z, b + z);}
// find a point from a through b with distance d
PT point_along_line(PT a, PT b, double d){assert(a != b);return a + (((b - a) / (b - a).norm()) * d);}
// projection point c onto line through a and b assuming a != b
PT project_from_point_to_line(PT a, PT b, PT c){return a + (b - a) * dot(c - a, b - a) / (b - a).norm2();}
// reflection point c onto line through a and b assuming a != b
PT reflection_from_point_to_line(PT a, PT b, PT c){PT p = project_from_point_to_line(a, b, c);return p + p - c;}
// minimum distance from point c to line through a and b
double dist_from_point_to_line(PT a, PT b, PT c){return fabs(cross(b - a, c - a) / (b - a).norm());}
// returns true if point p is on line segment ab
bool is_point_on_seg(PT a, PT b, PT p){if (fabs(cross(p - b, a - b)) < eps){if (p.x < min(a.x, b.x) - eps || p.x > max(a.x, b.x) + eps) return false;
if (p.y < min(a.y, b.y) - eps || p.y > max(a.y, b.y) + eps) return false;return true;}return false;}
// minimum distance point from point c to segment ab that lies on segment ab
PT project_from_point_to_seg(PT a, PT b, PT c){double r = dist2(a, b);if (sign(r) == 0)return a;

```

```

r = dot(c - a, b - a) / r;if (r < 0)return a;if (r > 1) return b;return a + (b - a) * r;}
// minimum distance from point c to segment ab
double dist_from_point_to_seg(PT a, PT b, PT c){return dist(c, project_from_point_to_seg(a, b, c));}
// 0 if not parallel, 1 if parallel, 2 if collinear
int is_parallel(PT a, PT b, PT c, PT d){double k = fabs(cross(b - a, d - c));
if (k < eps){if (fabs(cross(a - b, a - c)) < eps && fabs(cross(c - d, c - a)) < eps)return 2;
else return 1;}else return 0;}
// check if two lines are same
bool are_lines_same(PT a, PT b, PT c, PT d){if (fabs(cross(a - c, c - d)) < eps && fabs(cross(b - c, c - d)) < eps)return true;return false;}
// bisector vector of <abc
PT angle_bisector(PT &a, PT &b, PT &c){PT p = a - b, q = c - b;return p + q * sqrt(dot(p, p) / dot(q, q));}
// 1 if point is ccw to the line, 2 if point is cw to the line, 3 if point is on the line
int point_line_relation(PT a, PT b, PT p){int c = sign(cross(p - a, b - a));
if (c < 0)return 1;if (c > 0)return 2;return 3;}
// intersection point between ab and cd assuming unique intersection exists
bool line_line_intersection(PT a, PT b, PT c, PT d, PT &ans){
double a1 = a.y - b.y, b1 = b.x - a.x, c1 = cross(a, b);
double a2 = c.y - d.y, b2 = d.x - c.x, c2 = cross(c, d);
double det = a1 * b2 - a2 * b1;if (det == 0)return 0;
ans = PT((b1 * c2 - b2 * c1) / det, (c1 * a2 - a1 * c2) / det);return 1;}
// intersection point between segment ab and segment cd assuming unique intersection exists
bool seg_seg_intersection(PT a, PT b, PT c, PT d, PT &ans){double oa = cross2(c, d, a), ob = cross2(c, d, b);double oc = cross2(a, b, c), od = cross2(a, b, d);if (oa * ob < 0 && oc * od < 0){ans = (a * ob - b * oa) / (ob - oa);return 1;}elsereturn 0;}
// intersection point between segment ab and segment cd assuming unique intersection may not exists
// se.size()==0 means no intersection
// se.size()==1 means one intersection
// se.size()==2 means range intersection
set<PT> seg_seg_intersection_inside(PT a, PT b, PT c, PT d){PT ans;if (seg_seg_intersection(a, b, c, d, ans))return {ans};set<PT> se;

```

```

if (is_point_on_seg(c, d, a))se.insert(a);if (
    is_point_on_seg(c, d, b))se.insert(b);
if (is_point_on_seg(a, b, c))se.insert(c);
if (is_point_on_seg(a, b, d))se.insert(d);return se;}
// intersection between segment ab and line cd
// 0 if do not intersect, 1 if proper intersect, 2 if
// segment intersect
int seg_line_relation(PT a, PT b, PT c, PT d){
    double p = cross2(c, d, a);double q = cross2(c, d, b);
    if (sign(p) == 0 && sign(q) == 0)return 2;
    else if (p * q < 0)return 1;else return 0;}
// intersection between segment ab and line cd assuming
// unique intersection exists
bool seg_line_intersection(PT a, PT b, PT c, PT d, PT &ans){
    bool k = seg_line_relation(a, b, c, d);assert(k != 2);if (k)
        line_line_intersection(a, b, c, d, ans);return k;}
// minimum distance from segment ab to segment cd
double dist_from_seg_to_seg(PT a, PT b, PT c, PT d){PT dummy
    ;if (seg_seg_intersection(a, b, c, d, dummy))return
    0.0;
    else return min({dist_from_point_to_seg(a, b, c),
        dist_from_point_to_seg(a, b, d),dist_from_point_to_seg(
        c, d, a), dist_from_point_to_seg(c, d, b)});}
// minimum distance from point c to ray (starting point a
// and direction vector b)
double dist_from_point_to_ray(PT a, PT b, PT c){b = a + b;
    double r = dot(c - a, b - a);if (r < 0.0)return dist(c,
    a);return dist_from_point_to_line(a, b, c);}
// starting point as and direction vector ad
bool ray_ray_intersection(PT as, PT ad, PT bs, PT bd){
    double dx = bs.x - as.x, dy = bs.y - as.y;double det = bd.x
    * ad.y - bd.y * ad.x;if (fabs(det) < eps)return 0;
    double u = (dy * bd.x - dx * bd.y) / det;
    double v = (dy * ad.x - dx * ad.y) / det;
    if (sign(u) >= 0 && sign(v) >= 0)return 1;elsereturn 0;}
double ray_ray_distance(PT as, PT ad, PT bs, PT bd){if (
    ray_ray_intersection(as, ad, bs, bd))return 0.0;
    double ans = dist_from_point_to_ray(as, ad, bs);ans = min(
    ans, dist_from_point_to_ray(bs, bd, as));return ans;}
struct circle{PT p;double r;
    circle() {circle(PT _p, double _r) : p(_p), r(_r) {}};
    // center (x, y) and radius r
    circle(double x, double y, double _r) : p(PT(x, y)), r(_r)
    {};
    // circumcircle of a triangle
    // the three points must be unique
    circle(PT a, PT b, PT c){b = (a + b) * 0.5;c = (a + c) *
    0.5;line_line_intersection(b, b + rotatecw90(a - b), c,
    c + rotatecw90(a - c), p);r = dist(a, p);}
    // inscribed circle of a triangle

```

```

// pass a bool just to differentiate from circumcircle
circle(PT a, PT b, PT c, bool t){line u, v;double m = atan2(
    b.y - a.y, b.x - a.x), n = atan2(c.y - a.y, c.x - a.x);
    u.a = a;u.b = u.a + (PT(cos((n + m) / 2.0), sin((n + m) /
    2.0)));v.a = b;
    m = atan2(a.y - b.y, a.x - b.x), n = atan2(c.y - b.y, c.x -
    b.x);v.b = v.a + (PT(cos((n + m) / 2.0), sin((n + m) /
    2.0)));
    line_line_intersection(u.a, u.b, v.a, v.b, p);r =
    dist_from_point_to_seg(a, b, p);}
bool operator==(circle v) { return p == v.p && sign(r - v.r)
    == 0; }double area() { return PI * r * r; }double
    circumference() { return 2.0 * PI * r; };
// 0 if outside, 1 if on circumference, 2 if inside circle
int circle_point_relation(PT p, double r, PT b){double d =
    dist(p, b);if (sign(d - r) < 0)return 2;if (sign(d - r)
    == 0)return 1;return 0;}
// 0 if outside, 1 if on circumference, 2 if inside circle
int circle_line_relation(PT p, double r, PT a, PT b){double
    d = dist_from_point_to_line(a, b, p);if (sign(d - r) <
    0)return 2;
    if (sign(d - r) == 0)return 1;return 0;}
// compute intersection of line through points a and b with
// circle centered at c with radius r > 0
vector<PT> circle_line_intersection(PT c, double r, PT a, PT
    b){vector<PT> ret;
    b = b - a;a = a - c;double A = dot(b, b), B = dot(a, b);
    double C = dot(a, a) - r * r, D = B * B - A * C;
    if (D < -eps)return ret;
    ret.push_back(c + a + b * (-B + sqrt(D + eps)) / A);
    if (D > eps)ret.push_back(c + a + b * (-B - sqrt(D)) / A);
    return ret;}
// 5 - outside and do not intersect
// 4 - intersect outside in one point
// 3 - intersect in 2 points
// 2 - intersect inside in one point
// 1 - inside and do not intersect
int circle_circle_relation(PT a, double r, PT b, double R){
    double d = dist(a, b);
    if (sign(d - r - R) > 0)return 5;
    if (sign(d - r - R) == 0)return 4;
    double l = fabs(r - R);
    if (sign(d - r - R) < 0 && sign(d - l) > 0)return 3;
    if (sign(d - l) == 0)return 2;
    if (sign(d - l) < 0)return 1;assert(0);return -1;}
vector<PT> circle_circle_intersection(PT a, double r, PT b,
    double R){
    if (a == b && sign(r - R) == 0)return {PT(1e18, 1e18)};
    vector<PT> ret;double d = sqrt(dist2(a, b));

```

```

if (d > r + R || d + min(r, R) < max(r, R))return ret;double
    x = (d * d - R * R + r * r) / (2 * d);
    double y = sqrt(r * r - x * x);PT v = (b - a) / d;
    ret.push_back(a + v * x + rotateccw90(v) * y);if (y > 0)ret.
    push_back(a + v * x - rotateccw90(v) * y);return ret;}
// returns two circle c1, c2 through points a, b and of
// radius r
// 0 if there is no such circle, 1 if one circle, 2 if two
// circle
int get_circle(PT a, PT b, double r, circle &c1, circle &c2){
    vector<PT> v = circle_circle_intersection(a, r, b, r);
    int t = v.size();if (!t)return 0;
    c1.p = v[0], c1.r = r;if (t == 2)c2.p = v[1], c2.r = r;
    return t;}
// returns two circle c1, c2 which is tangent to line u,
// goes through
// point q and has radius r1; 0 for no circle, 1 if c1 = c2
// , 2 if c1 != c2
int get_circle(line u, PT q, double r1, circle &c1, circle &
    c2){
    double d = dist_from_point_to_line(u.a, u.b, q);if (sign(d -
    r1 * 2.0) > 0)return 0;
    if (sign(d) == 0){cout << u.v.x << ' ' << u.v.y << '\n';
    c1.p = q + rotateccw90(u.v).truncate(r1);c2.p = q +
    rotatecw90(u.v).truncate(r1);c1.r = c2.r = r1;return
    2;}
    line u1 = line(u.a + rotateccw90(u.v).truncate(r1), u.b +
    rotateccw90(u.v).truncate(r1));line u2 = line(u.a +
    rotatecw90(u.v).truncate(r1), u.b + rotatecw90(u.v).
    truncate(r1));
    circle cc = circle(q, r1);PT p1, p2;vector<PT> v;v =
    circle_line_intersection(q, r1, u1.a, u1.b);
    if (!v.size())v = circle_line_intersection(q, r1, u2.a, u2.b
    );v.push_back(v[0]);p1 = v[0], p2 = v[1];
    c1 = circle(p1, r1);if (p1 == p2){c2 = c1;return 1;}c2 =
    circle(p2, r1);return 2;}
// returns the circle such that for all points w on the
// circumference of the circle
// dist(w, a) : dist(w, b) = rp : rq
// rp != rq
// https://en.wikipedia.org/wiki/Circles_of_Apollonius
circle get_apollonius_circle(PT p, PT q, double rp, double
    rq){rq == rq;rp == rp;double a = rq - rp;
    assert(sign(a));double g = rq * p.x - rp * q.x;g /= a;double
    h = rq * p.y - rp * q.y;h /= a;double c = rq * p.x * p
    .x - rp * q.x * q.x + rq * p.y * p.y - rp * q.y * q.y;
    c /= a;PT o(g, h);double r = g * g + h * h - c;r = sqrt(r);
    return circle(o, r);}
// returns area of intersection between two circles

```



```

double circle_circle_area(PT a, double r1, PT b, double r2){
    double d = (a - b).norm();if (r1 + r2 < d + eps)return
    0;
    if (r1 + d < r2 + eps)return PI * r1 * r1;
    if (r2 + d < r1 + eps)return PI * r2 * r2;
    double theta_1 = acos((r1 * r1 + d * d - r2 * r2) / (2 * r1
        * d)),theta_2 = acos((r2 * r2 + d * d - r1 * r1) / (2 *
        r2 * d));
    return r1 * r1 * (theta_1 - sin(2 * theta_1) / 2.) + r2 * r2
        * (theta_2 - sin(2 * theta_2) / 2.);}
// tangent lines from point q to the circle
int tangent_lines_from_point(PT p, double r, PT q, line &u,
    line &v){
    int x = sign(dist2(p, q) - r * r);if (x < 0)return 0; //
        point in cricleif (x == 0)
    { // point on circle
        u = line(q, q + rotateccw90(q - p));v = u;return 1;}
    double d = dist(p, q);double l = r * r / d;double h = sqrt(r
        * r - l * l);u = line(q, p + ((q - p).truncate(1) + (
        rotateccw90(q - p).truncate(h)))));
    v = line(q, p + ((q - p).truncate(1) + (rotatecw90(q - p).
        truncate(h)))));return 2;}
// returns outer tangents line of two circles
// if inner == 1 it returns inner tangent lines
int tangents_lines_from_circle(PT c1, double r1, PT c2,
    double r2, bool inner, line &u, line &v){
    if (inner)r2 = -r2;PT d = c2 - c1;double dr = r1 - r2, d2 =
        d.norm2(), h2 = d2 - dr * dr;
    if (d2 == 0 || h2 < 0){assert(h2 != 0);return 0;}vector<pair
        <PT, PT>> out;for (int tmp : {-1, 1}){PT v = (d * dr +
        rotateccw90(d) * sqrt(h2) * tmp) / d2;out.push_back({c1
        + v * r1, c2 + v * r2});}
    u = line(out[0].first, out[0].second);
    if (out.size() == 2)v = line(out[1].first, out[1].second);
    return 1 + (h2 > 0);}
// O(n^2 log n)
// https://vjudge.net/problem/UVA-12056
struct CircleUnion
{int n;double x[2020], y[2020], r[2020];int covered[2020];
    vector<pair<double, double>> seg, cover;double arc, pol
    ;inline int sign(double x) { return x < -eps ? -1 : x >
        eps; }inline int sign(double x, double y) { return
        sign(x - y); }
    inline double SQ(const double x) { return x * x; }inline
        double dist(double x1, double y1, double x2, double y2)
        { return sqrt(SQ(x1 - x2) + SQ(y1 - y2)); }
    inline double angle(double A, double B, double C)
    {double val = (SQ(A) + SQ(B) - SQ(C)) / (2 * A * B);if (val
        < -1)val = -1;
    if (val > +1)val = +1;return acos(val);}
}

```

```

CircleUnion(){n = 0;seg.clear(), cover.clear();arc = pol =
    0;}
void init(){n = 0;seg.clear(), cover.clear();arc = pol = 0;}
void add(double xx, double yy, double rr)
{x[n] = xx, y[n] = yy, r[n] = rr, covered[n] = 0, n++;}
void getarea(int i, double lef, double rig){
    arc += 0.5 * r[i] * r[i] * (rig - lef - sin(rig - lef));
    double x1 = x[i] + r[i] * cos(lef), y1 = y[i] + r[i] * sin(
        lef);
    double x2 = x[i] + r[i] * cos(rig), y2 = y[i] + r[i] * sin(
        rig);pol += x1 * y2 - x2 * y1;}
double solve(){for (int i = 0; i < n; i++)
    {for (int j = 0; j < i; j++){if (!sign(x[i] - x[j]) && !sign
        (y[i] - y[j]) && !sign(r[i] - r[j]))
        {r[i] = 0.0;break;}}}
    for (int i = 0; i < n; i++){for (int j = 0; j < n; j++){if (
        i != j && sign(r[j] - r[i]) >= 0 && sign(dist(x[i], y[i]
        ], x[j], y[j]) - (r[j] - r[i])) <= 0){
        covered[i] = 1;break;}}}for (int i = 0; i < n; i++){
        if (sign(r[i]) && !covered[i]){seg.clear();
        for (int j = 0; j < n; j++){if (i != j){
            double d = dist(x[i], y[i], x[j], y[j]);
            if (sign(d - (r[j] + r[i])) >= 0 || sign(d - abs(r[j] - r[i]
                )) <= 0)
            {continue;}double alpha = atan2(y[j] - y[i], x[j] - x[i]);
            double beta = angle(r[i], d, r[j]);pair<double, double> tmp(
                alpha - beta, alpha + beta);
            if (sign(tmp.first) <= 0 && sign(tmp.second) <= 0){
                seg.push_back(pair<double, double>(2 * PI + tmp.first, 2 *
                PI + tmp.second));}
            else if (sign(tmp.first) < 0){seg.push_back(pair<double,
                double>(2 * PI + tmp.first, 2 * PI));seg.push_back(pair
                <double, double>(0, tmp.second));}
            else{seg.push_back(tmp);}}}
        sort(seg.begin(), seg.end());double rig = 0;
        for (vector<pair<double, double>>::iterator iter = seg.begin
            (); iter != seg.end(); iter++){
            if (sign(rig - iter->first) >= 0){rig = max(rig, iter->
                second);}
            else{getarea(i, rig, iter->first);rig = iter->second;}}
        if (!sign(rig)){arc += r[i] * r[i] * PI;}
        else{getarea(i, rig, 2 * PI);}}return pol / 2.0 + arc;}} CU
    ;
    double area_of_triangle(PT a, PT b, PT c){return fabs(cross(
        b - a, c - a) * 0.5);}
// -1 if strictly inside, 0 if on the polygon, 1 if strictly
    outside
int is_point_in_triangle(PT a, PT b, PT c, PT p){
    if (sign(cross(b - a, c - a)) < 0)swap(b, c);
}

```

```

int c1 = sign(cross(b - a, p - a));int c2 = sign(cross(c - b
    , p - b));int c3 = sign(cross(a - c, p - c));
    if (c1 < 0 || c2 < 0 || c3 < 0)return 1;if (c1 + c2 + c3 !=
        3)return 0;return -1;}
double perimeter(vector<PT> &p){
    double ans = 0;int n = p.size();
    for (int i = 0; i < n; i++)ans += dist(p[i], p[(i + 1) % n])
        ;return ans;}
double area(vector<PT> &p){double ans = 0;int n = p.size();
    or (int i = 0; i < n; i++)ans += cross(p[i], p[(i + 1)
        % n]);return fabs(ans) * 0.5;}
// centroid of a (possibly non-convex) polygon,
// assuming that the coordinates are listed in a clockwise
    or
// counterclockwise fashion. Note that the centroid is often
    known as
// the "center of gravity" or "center of mass".
PT centroid(vector<PT> &p){int n = p.size();PT c(0, 0);
    double sum = 0;
    for (int i = 0; i < n; i++)sum += cross(p[i], p[(i + 1) % n
        ]);double scale = 3.0 * sum;
    for (int i = 0; i < n; i++){int j = (i + 1) % n;c = c + (p[i]
        + p[j]) * cross(p[i], p[j]);}return c / scale;}
// 0 if cw, 1 if ccw
bool get_direction(vector<PT> &p){double ans = 0;int n = p.
    size();
    for (int i = 0; i < n; i++)ans += cross(p[i], p[(i + 1) % n
        ]);if (sign(ans) > 0)return 1;return 0;}
// it returns a point such that the sum of distances
// from that point to all points in p is minimum
// O(n log^2 MX)
PT geometric_median(vector<PT> p){auto tot_dist = [&](PT z){
    double res = 0;
    for (int i = 0; i < p.size(); i++)res += dist(p[i], z);
    return res;};
    auto findY = [&](double x){double y1 = -1e5, yr = 1e5;for (
        int i = 0; i < 60; i++){
        double ym1 = y1 + (yr - y1) / 3;double ym2 = yr - (yr - y1)
            / 3;double d1 = tot_dist(PT(x, ym1));
        double d2 = tot_dist(PT(x, ym2));if (d1 < d2)yr = ym2;
        else y1 = ym1;}return pair<double, double>(y1, tot_dist(PT(x
            , y1)));};double x1 = -1e5, xr = 1e5;
    for (int i = 0; i < 60; i++){double xm1 = x1 + (xr - x1) /
        3;
        double xm2 = xr - (xr - x1) / 3;double y1, d1, y2, d2;auto z
            = findY(xm1);y1 = z.first;d1 = z.second;z = findY(xm2)
            ;
        y2 = z.first;d2 = z.second;
        if (d1 < d2)xr = xm2;elsex1 = xm1;}return {x1, findY(x1).
            first};}
}

```

```

vector<PT> convex_hull(vector<PT> &p){
if (p.size() <= 1)return p;
vector<PT> v = p;sort(v.begin(), v.end());vector<PT> up, dn;
for (auto &p : v){
while (up.size() > 1 && orientation(up[up.size() - 2], up.
back(), p) >= 0){up.pop_back();}
while (dn.size() > 1 && orientation(dn[dn.size() - 2], dn.
back(), p) <= 0){dn.pop_back();}
up.push_back(p);dn.push_back(p);}
v = dn;if (v.size() > 1)v.pop_back();reverse(up.begin(), up.
end());up.pop_back();
for (auto &p : up){v.push_back(p);}
if (v.size() == 2 && v[0] == v[1])v.pop_back();return v;}
// checks if convex or not
bool is_convex(vector<PT> &p){bool s[3];s[0] = s[1] = s[2] =
0;int n = p.size();for (int i = 0; i < n; i++){int j =
(i + 1) % n;int k = (j + 1) % n;
s[sign(cross(p[j] - p[i], p[k] - p[i])) + 1] = 1;if (s[0] &&
s[2])return 0;}return 1;}
// -1 if strictly inside, 0 if on the polygon, 1 if strictly
outside
// it must be strictly convex, otherwise make it strictly
convex first
int is_point_in_convex(vector<PT> &p, const PT &x){ // 0(log
n)int n = p.size();assert(n >= 3);
int a = orientation(p[0], p[1], x), b = orientation(p[0], p[
n - 1], x);if (a < 0 || b > 0)return 1;int l = 1, r = n
- 1;while (l + 1 < r){
int mid = l + r >> 1;if (orientation(p[0], p[mid], x) >= 0)l
= mid;else = mid;}int k = orientation(p[l], p[r], x);
if (k <= 0)return -k;
if (l == 1 && a == 0)return 0;if (r == n - 1 && b == 0)
return 0;return -1;}
bool is_point_on_polygon(vector<PT> &p, const PT &z){int n =
p.size();
for (int i = 0; i < n; i++){if (is_point_on_seg(p[i], p[(i +
1) % n], z))return 1;}return 0;}
// returns 1e9 if the point is on the polygon
int winding_number(vector<PT> &p, const PT &z){ // 0(n)
if (is_point_on_polygon(p, z))return 1e9;
int n = p.size(), ans = 0;
for (int i = 0; i < n; ++i){int j = (i + 1) % n;bool below =
p[i].y < z.y;
if (below != (p[j].y < z.y)){
auto orient = orientation(z, p[j], p[i]);
if (orient == 0)return 0;
if (below == (orient > 0))ans += below ? 1 : -1;}}return ans
;}
// -1 if strictly inside, 0 if on the polygon, 1 if strictly
outside

```

```

int is_point_in_polygon(vector<PT> &p, const PT &z){ // 0(n)
int k = winding_number(p, z);return k == 1e9 ? 0 : k == 0 ?
1 : -1;}
// id of the vertex having maximum dot product with z
// polygon must need to be convex
// top - upper right vertex
// for minimum dot product negate z and return -dot(z, p[id
])
int extreme_vertex(vector<PT> &p, const PT &z, const int top
){ // 0(log n)
int n = p.size();if (n == 1)return 0;
double ans = dot(p[0], z);int id = 0;
if (dot(p[top], z) > ans)ans = dot(p[top], z), id = top;
int l = 1, r = top - 1;
while (l < r){int mid = l + r >> 1;
if (dot(p[mid + 1], z) >= dot(p[mid], z))l = mid + 1;
else r = mid;}
if (dot(p[l], z) > ans)ans = dot(p[l], z), id = l;l = top +
1, r = n - 1;while (l < r){
int mid = l + r >> 1;if (dot(p[(mid + 1) % n], z) >= dot(p[
mid], z))l = mid + 1;elser = mid;}l %= n;
if (dot(p[l], z) > ans)ans = dot(p[l], z), id = l;return id
;}
// maximum distance from any point on the perimeter to
another point on the perimeter
double diameter(vector<PT> &p){int n = (int)p.size();if (n
== 1)return 0;if (n == 2)return dist(p[0], p[1]);double
ans = 0;int i = 0, j = 1;
while (i < n){
while (cross(p[(i + 1) % n] - p[i], p[(j + 1) % n] - p[j])
>= 0){ans = max(ans, dist2(p[i], p[j]));j = (j + 1) % n
;}}
ans = max(ans, dist2(p[i], p[j]));i++;}return sqrt(ans);}
// minimum distance between two parallel lines (non
necessarily axis parallel)
// such that the polygon can be put between the lines
double width(vector<PT> &p){int n = (int)p.size();if (n <=
2)return 0;double ans = inf;int i = 0, j = 1;while (i <
n){
while (cross(p[(i + 1) % n] - p[i], p[(j + 1) % n] - p[j])
>= 0)j = (j + 1) % n;ans = min(ans,
dist_from_point_to_line(p[i], p[(i + 1) % n], p[j]));i
++;}return ans;}
// minimum perimeter
double minimum_enclosing_rectangle(vector<PT> &p){int n = p.
size();
if (n <= 2)return perimeter(p);int mndot = 0;double tmp =
dot(p[1] - p[0], p[0]);
for (int i = 1; i < n; i++){if (dot(p[1] - p[0], p[i]) <=
tmp){tmp = dot(p[1] - p[0], p[i]);mndot = i;}}

```

```

double ans = inf;int i = 0, j = 1, mxdot = 1;
while (i < n){PT cur = p[(i + 1) % n] - p[i];while (cross(
cur, p[(j + 1) % n] - p[j]) >= 0)j = (j + 1) % n;while
(dot(p[(mxdot + 1) % n], cur) >= dot(p[mxdot], cur))
mxdot = (mxdot + 1) % n;
while (dot(p[(mndot + 1) % n], cur) <= dot(p[mndot], cur))
mndot = (mndot + 1) % n;ans = min(ans, 2.0 * ((dot(p[
mxdot], cur) / cur.norm() - dot(p[mndot], cur) / cur.
norm()) + dist_from_point_to_line(p[i], p[(i + 1) % n],
p[j])));i++;}return ans;}
// given n points, find the minimum enclosing circle of the
points
// call convex_hull() before this for faster solution
// expected O(n)
circle minimum_enclosing_circle(vector<PT> &p){
random_shuffle(p.begin(), p.end());int n = p.size();
circle c(p[0], 0);
for (int i = 1; i < n; i++){if (sign(dist(c.p, p[i]) - c.r)
> 0){c = circle(p[i], 0);
for (int j = 0; j < i; j++){
if (sign(dist(c.p, p[j]) - c.r) > 0){
c = circle((p[i] + p[j]) / 2, dist(p[i], p[j]) / 2);
for (int k = 0; k < j; k++){if (sign(dist(c.p, p[k]) - c.r)
> 0){c = circle(p[i], p[j], p[k]);}}}return c;}
// returns a vector with the vertices of a polygon with
everything
// to the left of the line going from a to b cut away.
vector<PT> cut(vector<PT> &p, PT a, PT b){vector<PT> ans;int
n = (int)p.size();
for (int i = 0; i < n; i++){double c1 = cross(b - a, p[i] -
a);double c2 = cross(b - a, p[(i + 1) % n] - a);
if (sign(c1) >= 0)ans.push_back(p[i]);if (sign(c1 * c2) < 0)
{if (!is_parallel(p[i], p[(i + 1) % n], a, b)){PT tmp;
line_line_intersection(p[i], p[(i + 1) % n], a, b, tmp)
;ans.push_back(tmp);}}return ans;}
// not necessarily convex, boundary is included in the
intersection
// returns total intersected length
// it returns the sum of the lengths of the portions of the
line that are inside the polygon
double polygon_line_intersection(vector<PT> p, PT a, PT b){
int n = p.size();p.push_back(p[0]);line l = line(a, b);
double ans = 0.0;vector<pair<double, int>> vec;for (int
i = 0; i < n; i++){int s1 = orientation(a, b, p[i]);
int s2 = orientation(a, b, p[i + 1]);
if (s1 == s2)continue;line t = line(p[i], p[i + 1]);PT inter
= (t.v * l.c - l.v * t.c) / cross(l.v, t.v);double tmp
= dot(inter, l.v);int f;
if (s1 > s2)f = s1 && s2 ? 2 : 1;

```

```

else f = s1 && s2 ? -2 : -1; vec.push_back(make_pair((f > 0 ?
    tmp - eps : tmp + eps), f)); // keep eps very small
    like 1e-12}
sort(vec.begin(), vec.end()); for (int i = 0, j = 0; i + 1 <
    (int)vec.size(); i++) {j += vec[i].second;
    if (j) ans += vec[i + 1].first - vec[i].first; // if this
    portion is inside the polygon // else ans = 0; // if we
    want the maximum intersected length which is totally
    inside the polygon, uncomment this and take the maximum
    of ans}
ans = ans / sqrt(dot(l.v, l.v)); p.pop_back(); return ans;}
// given a convex polygon p, and a line ab and the top
// vertex of the polygon
// returns the intersection of the line with the polygon
// it returns the indices of the edges of the polygon that
// are intersected by the line
// so if it returns i, then the line intersects the edge (p[i],
// p[(i + 1) % n])
array<int, 2> convex_line_intersection(vector<PT> &p, PT a,
    PT b, int top) {int end_a = extreme_vertex(p, (a - b).
    perp(), top); int end_b = extreme_vertex(p, (b - a).perp
    (), top);
    auto cmp_l = [&](int i) {return orientation(a, p[i], b); };
    if (cmp_l(end_a) < 0 || cmp_l(end_b) > 0) return {-1,
    -1}; // no intersection
    array<int, 2> res; for (int i =
    0; i < 2; i++) {int lo = end_b, hi = end_a, n = p.size()
    ;
    while ((lo + 1) % n != hi) {int m = ((lo + hi + (lo < hi ? 0
    : n)) / 2) % n; (cmp_l(m) == cmp_l(end_b) ? lo : hi) = m
    ;} res[i] = (lo + !cmp_l(hi)) % n;
    swap(end_a, end_b);}
    if (res[0] == res[1]) return {res[0], -1}; // touches the
    vertex res[0] if (!cmp_l(res[0]) && !cmp_l(res[1]))
    switch ((res[0] - res[1] + (int)p.size() + 1) % p.size()) {
        case 0: return {res[0], res[0]}; // touches the edge (
        res[0], res[0] + 1)
        case 2: return {res[1], res[1]}; //
        touches the edge (res[1], res[1] + 1)}
    return res; // intersects the edges (res[0], res[0] + 1) and
    (res[1], res[1] + 1)}

pair<PT, int> point_poly_tangent(vector<PT> &p, PT Q, int
    dir, int l, int r) {
    while (r - l > 1) {int mid = (l + r) >> 1; bool pvs =
    orientation(Q, p[mid], p[mid - 1]) != -dir;
    bool nxt = orientation(Q, p[mid], p[mid + 1]) != -dir;
    if (pvs && nxt) return {p[mid], mid};
    if (!pvs || !nxt) {auto p1 = point_poly_tangent(p, Q, dir,
    mid + 1, r);
    auto p2 = point_poly_tangent(p, Q, dir, l, mid - 1);
    return orientation(Q, p1.first, p2.first) == dir ? p1 : p2;}
}

```

```

if (!pvs) {if (orientation(Q, p[mid], p[l]) == dir) r = mid -
    1;
    else if (orientation(Q, p[l], p[r]) == dir) r = mid - 1; else
    = mid + 1;}
    if (!nxt) {if (orientation(Q, p[mid], p[l]) == dir) l = mid +
    1;
    else if (orientation(Q, p[l], p[r]) == dir) r = mid - 1;
    else l = mid + 1;}
    pair<PT, int> ret = {p[l], l};
    for (int i = l + 1; i <= r; i++) ret = orientation(Q, ret.
    first, p[i]) != dir ? make_pair(p[i], i) : ret;
    return ret;}
// (ccw, cw) tangents from a point that is outside this
// convex polygon
// returns indexes of the points
// ccw means the tangent from Q to that point is in the same
// direction as the polygon ccw direction
pair<int, int> tangents_from_point_to_polygon(vector<PT> &p,
    PT Q) {int ccw = point_poly_tangent(p, Q, 1, 0, (int)p.
    size() - 1).second; int cw = point_poly_tangent(p, Q,
    -1, 0, (int)p.size() - 1).second; return make_pair(ccw,
    cw);}
// minimum distance from a point to a convex polygon
// it assumes point lie strictly outside the polygon
double dist_from_point_to_polygon(vector<PT> &p, PT z) {
    double ans = inf; int n = p.size(); if (n <= 3) {for (int
    i = 0; i < n; i++) ans = min(ans, dist_from_point_to_seg
    (p[i], p[(i + 1) % n], z)); return ans;}
    auto [r, l] = tangents_from_point_to_polygon(p, z); if (l > r
    ) r += n;
    while (l < r) {int mid = (l + r) >> 1; double left = dist2(p[
    mid % n], z), right = dist2(p[(mid + 1) % n], z); ans =
    min({ans, left, right}); if (left < right) r = mid; else
    = mid + 1;}
    ans = sqrt(ans); ans = min(ans, dist_from_point_to_seg(p[l %
    n], p[(l + 1) % n], z)); ans = min(ans,
    dist_from_point_to_seg(p[l % n], p[(l - 1 + n) % n], z)
    ); return ans;}
// minimum distance from convex polygon p to line ab
// returns 0 is it intersects with the polygon
// top - upper right vertex
double dist_from_polygon_to_line(vector<PT> &p, PT a, PT b,
    int top) { // O(log n)
    PT orth = (b - a).perp();
    if (orientation(a, b, p[0]) > 0) orth = (a - b).perp();
    int id = extreme_vertex(p, orth, top); if (dot(p[id] - a,
    orth) > 0) return 0.0; // if orth and a are in the same
    half of the line, then poly and line intersects
    return dist_from_point_to_line(a, b, p[id]); // does not
    intersect}
}

```

```

// minimum distance from a convex polygon to another convex
// polygon
// the polygon doesnot overlap or touch
// tested in https://toph.co/p/the-wall
double dist_from_polygon_to_polygon(vector<PT> &p1, vector<
    PT> &p2) { // O(n log n)
    double ans = inf;
    for (int i = 0; i < p1.size(); i++) {ans = min(ans,
    dist_from_point_to_polygon(p2, p1[i]));}
    for (int i = 0; i < p2.size(); i++) {ans = min(ans,
    dist_from_point_to_polygon(p1, p2[i]));}
    return ans;}
// maximum distance from a convex polygon to another convex
// polygon
double maximum_dist_from_polygon_to_polygon(vector<PT> &u,
    vector<PT> &v) { // O(n)
    int n = (int)u.size(), m = (int)v.size(); double ans = 0;
    if (n < 3 || m < 3) {
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < m; j++) ans = max(ans, dist2(u[i], v[j]));
            return sqrt(ans);}
        if (u[0].x > v[0].x) swap(n, m), swap(u, v); int i = 0, j = 0,
        step = n + m + 10;
        while (j + 1 < m && v[j].x < v[j + 1].x) j++;
        while (step-- > 0) {if (cross(u[(i + 1) % n] - u[i], v[(j + 1) %
        m] - v[j]) >= 0) j = (j + 1) % m; else i = (i + 1) % n; ans
        = max(ans, dist2(u[i], v[j]));}
        return sqrt(ans);}
    // calculates the area of the union of n polygons (not
    // necessarily convex).
    // the points within each polygon must be given in CCW order
    .
    // complexity: O(N^2), where N is the total number of points
    double rat(PT a, PT b, PT p) {return !sign(a.x - b.x) ? (p.y
    - a.y) / (b.y - a.y) : (p.x - a.x) / (b.x - a.x);}
    double polygon_union(vector<vector<PT>> &p) {
        int n = p.size(); double ans = 0;
        for (int i = 0; i < n; i++) {
            for (int v = 0; v < (int)p[i].size(); ++v) {
                PT a = p[i][v], b = p[i][(v + 1) % p[i].size()]; vector<pair<
                double, int>> segs; segs.emplace_back(0, 0), segs.
                emplace_back(1, 0);
                for (int j = 0; j < n; ++j) {
                    if (i != j) {for (size_t u = 0; u < p[j].size(); ++u) {
                        PT c = p[j][u], d = p[j][(u + 1) % p[j].size()]; int sc =
                        sign(cross(b - a, c - a)), sd = sign(cross(b - a, d - a
                        )); if (!sc && !sd) {
                            if (sign(dot(b - a, d - c)) > 0 && i > j) {segs.emplace_back(
                            rat(a, b, c), 1), segs.emplace_back(rat(a, b, d), -1)
                        }
                    }
                }
            }
        }
    }
}

```

```

    };}else{
double sa = cross(d - c, a - c), sb = cross(d - c, b - c);
if (sc >= 0 && sd < 0)segs.emplace_back(sa / (sa - sb), 1);
else if (sc < 0 && sd >= 0)segs.emplace_back(sa / (sa - sb), -1);}}
sort(segs.begin(), segs.end());double pre = min(max(segs[0].first, 0.0), 1.0), now, sum = 0;int cnt = segs[0].second;
for (int j = 1; j < segs.size(); ++j){now = min(max(segs[j].first, 0.0), 1.0);if (!cnt)sum += now - pre;cnt += segs[j].second;pre = now;}ans += cross(a, b) * sum;}return ans * 0.5;}
// contains all points p such that: cross(b - a, p - a) >= 0
struct HP{PT a, b;HP() {}HP(PT a, PT b) : a(a), b(b) {}HP(const HP &rhs) : a(rhs.a), b(rhs.b) {}
int operator<(const HP &rhs) const{PT p = b - a;PT q = rhs.b - rhs.a;
int fp = (p.y < 0 || (p.y == 0 && p.x < 0));int fq = (q.y < 0 || (q.y == 0 && q.x < 0));
if (fp != fq)return fp == 0;if (cross(p, q))return cross(p, q) > 0;return cross(p, rhs.b - a) < 0;PT
line_line_intersection(PT a, PT b, PT c, PT d)
{b = b - a;d = c - d;c = c - a;return a + b * cross(c, d) / cross(b, d);}
PT intersection(const HP &v){return line_line_intersection(a, b, v.a, v.b);}};
int check(HP a, HP b, HP c){return cross(a.b - a.a, b.intersection(c) - a.a) > -eps; //-eps to include polygons of zero area (straight lines, points)}
// consider half-plane of counter-clockwise side of each line
// if lines are not bounded add infinity rectangle
// returns a convex polygon, a point can occur multiple times though
// complexity: O(n log(n))
vector<PT> half_plane_intersection(vector<HP> h){sort(h.begin(), h.end());vector<HP> tmp;
for (int i = 0; i < h.size(); i++){if (!i || cross(h[i].b - h[i].a, h[i - 1].b - h[i - 1].a)){tmp.push_back(h[i]);}}
h = tmp;vector<HP> q(h.size() + 10);int qh = 0, qe = 0;
for (int i = 0; i < h.size(); i++){
while (qe - qh > 1 && !check(h[i], q[qh], q[qe - 2], q[qe - 1]))qe--;
while (qe - qh > 1 && !check(h[i], q[qh], q[qh + 1]))qh++;q[qe++] = h[i];}
while (qe - qh > 2 && !check(q[qh], q[qe - 2], q[qe - 1]))qe--;
while (qe - qh > 2 && !check(q[qe - 1], q[qh], q[qh + 1]))qh++;
}

```

```

vector<HP> res;
for (int i = qh; i < qe; i++)res.push_back(q[i]);vector<PT> hull;
if (res.size() > 2){for (int i = 0; i < res.size(); i++){hull.push_back(res[i].intersection(res[(i + 1) % ((int)res.size())]));}
return hull;}
// rotate the polygon such that the (bottom, left)-most point is at the first position
void reorder_polygon(vector<PT> &p){int pos = 0;for (int i = 1; i < p.size(); i++){if (p[i].y < p[pos].y || (sign(p[i].y - p[pos].y) == 0 && p[i].x < p[pos].x))pos = i;}
rotate(p.begin(), p.begin() + pos, p.end());}
// a and b are convex polygons
// returns a convex hull of their minkowski sum
// min(a.size(), b.size()) >= 2
// https://cp-algorithms.com/geometry/minkowski.html
vector<PT> minkowski_sum(vector<PT> a, vector<PT> b){reorder_polygon(a);reorder_polygon(b);
int n = a.size(), m = b.size();int i = 0, j = 0;a.push_back(a[0]);a.push_back(a[1]);b.push_back(b[0]);b.push_back(b[1]);vector<PT> c;
while (i < n || j < m){c.push_back(a[i] + b[j]);double p = cross(a[i + 1] - a[i], b[j + 1] - b[j]);if (sign(p) >= 0)++i;if (sign(p) <= 0)++j;}
return c;}
// returns the area of the intersection of the circle with center c and radius r
// and the triangle formed by the points c, a, b
double _triangle_circle_intersection(PT c, double r, PT a, PT b){
double sd1 = dist2(c, a), sd2 = dist2(c, b);
if (sd1 > sd2)swap(a, b), swap(sd1, sd2);double sd = dist2(a, b);double d1 = sqrtl(sd1), d2 = sqrtl(sd2), d = sqrt(sd);double x = abs(sd2 - sd - sd1) / (2 * d);double h = sqrtl(sd1 - x * x);
if (r >= d2)return h * d / 2;double area = 0;
if (sd + sd1 < sd2){if (r < d1)area = r * r * (acos(h / d2) - acos(h / d1)) / 2;else{area = r * r * (acos(h / d2) - acos(h / r)) / 2;double y = sqrtl(r * r - h * h);area += h * (y - x) / 2;}}
else{if (r < h)area = r * r * (acos(h / d2) + acos(h / d1)) / 2;else{area += r * r * (acos(h / d2) - acos(h / r)) / 2;double y = sqrtl(r * r - h * h);area += h * y / 2;if (r < d1){area += r * r * (acos(h / d1) - acos(h / r)) / 2;area += h * y / 2;elsearea += h * x / 2;}}
return area;}
// intersection between a simple polygon and a circle
double polygon_circle_intersection(vector<PT> &v, PT p, double r){int n = v.size();double ans = 0.00;PT org =

```

```

{0, 0};
for (int i = 0; i < n; i++){int x = orientation(p, v[i], v[(i + 1) % n]);if (x == 0)continue;double area = _triangle_circle_intersection(org, r, v[i] - p, v[(i + 1) % n] - p);if (x < 0)ans -= area;elseans += area;}
return abs(ans);}
// find a circle of radius r that contains as many points as possible
// O(n^2 log n);
double maximum_circle_cover(vector<PT> p, double r, circle &c){int n = p.size();int ans = 0;int id = 0;double th = 0;
for (int i = 0; i < n; ++i){
// maximum circle cover when the circle goes through this point
vector<pair<double, int>> events = {{-PI, +1}, {PI, -1}};
for (int j = 0; j < n; ++j){if (j == i)continue;double d = dist(p[i], p[j]);if (d > r * 2)continue;
double dir = (p[j] - p[i]).arg();double ang = acos(d / 2 / r);double st = dir - ang, ed = dir + ang;
if (st > PI)st -= PI * 2;if (st <= -PI)st += PI * 2;if (ed > PI)ed -= PI * 2;if (ed <= -PI)ed += PI * 2;
events.push_back({st - eps, +1}); // take care of precisions !
events.push_back({ed, -1});if (st > ed){events.push_back({-PI, +1});events.push_back({+PI, -1});}sort(events.begin(), events.end());int cnt = 0;for (auto &e : events){cnt += e.second;if (cnt > ans){ans = cnt;id = i;th = e.first;}}PT w = PT(p[id].x + r * cos(th), p[id].y + r * sin(th));c = circle(w, r); // best_circle
return ans;}
// radius of the maximum inscribed circle in a convex polygon
double maximum_inscribed_circle(vector<PT> p){int n = p.size();if (n <= 2)return 0;double l = 0, r = 20000;
while (r - l > eps){double mid = (l + r) * 0.5;vector<HP> h;const int L = 1e9;h.push_back(HP(PT(-L, -L), PT(L, -L)));h.push_back(HP(PT(L, -L), PT(L, L)));h.push_back(HP(PT(L, L), PT(-L, L)));h.push_back(HP(PT(-L, L), PT(-L, -L)));for (int i = 0; i < n; i++){PT z = (p[(i + 1) % n] - p[i]).perp();z = z.truncate(mid);PT y = p[i] + z, q = p[(i + 1) % n] + z;h.push_back(HP(p[i] + z, p[(i + 1) % n] + z));}vector<PT> nw = half_plane_intersection(h);if (!nw.empty())l = mid;elser = mid;}
return l;}
// ear decomposition, O(n^3) but faster
vector<vector<PT>> triangulate(vector<PT> p){vector<vector<PT>> v;
while (p.size() >= 3){

```



```
for (int i = 0, n = p.size(); i < n; i++){int pre = i == 0 ?
    n - 1 : i - 1;;int nxt = i == n - 1 ? 0 : i + 1;int
    ori = orientation(p[i], p[pre], p[nxt]);if (ori < 0){
    int ok = 1;for (int j = 0; j < n; j++){if (j == i || j
    == pre || j == nxt)continue;if (is_point_in_triangle(p[
    i], p[pre], p[nxt], p[j]) < 1){ok = 0;break;}}if (ok){v
    .push_back({p[pre], p[i], p[nxt]});p.erase(p.begin() +
    i);break;}}}}
return v;}
```

```
struct star{
int n;    // number of sides of the star
double r; // radius of the circumcircle
star(int _n, double _r){n = _n;r = _r;}
double area(){double theta = PI / n;double s = 2 * r * sin(
    theta);double R = 0.5 * s / tan(theta);double a = 0.5 *
    n * s * R;double a2 = 0.25 * s * s / tan(1.5 * theta);
    return a - n * a2;};}
```

```
// given a list of lengths of the sides of a polygon in
    counterclockwise order
// returns the maximum area of a non-degenerate polygon that
    can be formed using those lengths
```

```
double get_maximum_polygon_area_for_given_lengths(vector<
    double> v){
if (v.size() < 3){return 0;}
int m = 0;double sum = 0;
for (int i = 0; i < v.size(); i++){if (v[i] > v[m]){m = i;}
    sum += v[i];}
if (sign(v[m] - (sum - v[m])) >= 0){return 0; // no non-
    degenerate polygon is possible}
```

```
// the polygon should be a circular polygon
// that is all points are on the circumference of a circle
double l = v[m] / 2, r = 1e6; // fix it correctlyint it =
    60;
```

```
auto ang = [](double x, double r) { // x = length of the
    chord, r = radius of the circle
return 2 * asin((x / 2) / r);};
auto calc = [](double r){double sum = 0;for (auto x : v){
    sum += ang(x, r);}return sum; };
```

```
// compute the radius of the circle
while (it--){double mid = (l + r) / 2;if (calc(mid) <= 2 *
    PI){r = mid;}else{l = mid;}}
if (calc(r) <= 2 * PI - eps){ // the center of the circle is
    outside the polygon
```

```
auto calc2 = [&](double r){double sum = 0;for (int i = 0; i
    < v.size(); i++){double x = v[i];double th = ang(x, r);
    if (i != m){sum += th;}else{sum += 2 * PI - th;}}return
    sum;};l = v[m] / 2;r = 1e6;it = 60;while (it--){
```

```
double mid = (l + r) / 2;if (calc2(mid) > 2 * PI){r = mid;}
    else{l = mid;}}
auto get_area = [](double r){double ans = 0;
for (int i = 0; i < v.size(); i++){double x = v[i];double
    area = r * r * sin(ang(x, r)) / 2;if (i != m){ans +=
    area;}else{ans -= area;}}return ans;};return get_area(r
    );};else{ // the center of the circle is inside the
    polygon
auto get_area = [](double r){double ans = 0;for (auto x : v
    ){ans += r * r * sin(ang(x, r)) / 2;}return ans;};
    return get_area(r);};}
```

## 5 Graph

### 5.1 Flow

```
lli n, m; vector<set<lli>> roads; vector<vector<lli>> adjm;
    vector<lli> parents;
lli bfs(lli s, lli t){
fill(parents.begin(), parents.end(), -1); parents[s] = -2;
queue<pair<lli, lli>> q; q.push({s, LLONG_MAX});
while (!q.empty()){ lli ind = q.front().first; lli flow = q.
    front().second; q.pop();
for (lli cind: roads[ind]){ if (parents[cind] == -1 && adjm[
    ind][cind] > 0){
    parents[cind] = ind; if (cind == t) return min(flow, adjm[
    ind][cind]);
    q.push({cind, min(adjm[ind][cind], flow)});}} return 0; }
lli getFlow(){
lli totflow = 0, nflow;
while (nflow = bfs(0, n - 1)){ totflow += nflow; lli ind = n
    - 1;
while (ind != 0){ adjm[ind][parents[ind]] += nflow;
    adjm[parents[ind]][ind] -= nflow; ind = parents[ind];
    }}
return totflow;
}
```

## 6 NumberTheory

### 6.1 CRT

```
using T = __int128;
// ax + by = __gcd(a, b)
// returns __gcd(a, b)
```

```
T extended_euclid(T a, T b, T &x, T &y) {
T xx = y = 0;T yy = x = 1;
while (b) {T q = a / b;T t = b; b = a % b; a = t;
t = xx; xx = x - q * xx; x = t;t = yy; yy = y - q * yy; y =
    t;}
return a;}// finds x such that x % m1 = a1, x % m2 = a2. m1
    and m2 may not be coprime
// here, x is unique modulo m = lcm(m1, m2). returns (x, m).
    on failure, m = -1.
pair<T, T> CRT(T a1, T m1, T a2, T m2) {
T p, q; T g = extended_euclid(m1, m2, p, q);
if (a1 % g != a2 % g) return make_pair(0, -1); T m = m1 / g
    * m2;
p = (p % m + m) % m;q = (q % m + m) % m;
return make_pair((p * a2 % m * (m1 / g) % m + q * a1 % m * (
    m2 / g) % m) % m, m);}
```

### 6.2 NTT shortened

```
const int N = 1 << 20;
const int mod = 998244353;
const int root = 3;
int lim, rev[N], w[N], wn[N], inv_lim;
void reduce(int &x) { x = (x + mod) % mod; }
int POW(int x, int y, int ans = 1) {
for (; y; y >>= 1, x = (long long) x * x % mod) if (y & 1)
    ans = (long long) ans * x % mod;
return ans;}
void precompute(int len) {
lim = wn[0] = 1; int s = -1; while (lim < len) lim <= 1,
    ++s;
for (int i = 0; i < lim; ++i) rev[i] = rev[i >> 1] >> 1 |
    (i & 1) << s;
const int g = POW(root, (mod - 1) / lim); inv_lim = POW(
    lim, mod - 2);
for (int i = 1; i < lim; ++i) wn[i] = (long long) wn[i -
    1] * g % mod;
}
void ntt(vector<int> &a, int typ) {
for (int i = 0; i < lim; ++i) if (i < rev[i]) swap(a[i], a
    [rev[i]]);
for (int i = 1; i < lim; i <= 1) { for (int j = 0, t =
    lim / i / 2; j < i; ++j) w[j] = wn[j * t];
for (int j = 0; j < lim; j += i << 1) { for (int k = 0; k
    < i; ++k) {
const int x = a[k + j], y = (long long) a[k + j + i]
    * w[k] % mod;
reduce(a[k + j] += y - mod), reduce(a[k + j + i] = x
    - y); } } }
```



```

    if (!typ) { reverse(a.begin() + 1, a.begin() + lim);
        for (int i = 0; i < lim; ++i) a[i] = (long long) a[i] *
            inv_lim % mod; }
}
vector<int> multiply(vector<int> &f, vector<int> &g) {
    if (f.empty() or g.empty()) return {};
    int n = (int)f.size() + (int)g.size() - 1;
    if (n == 1) return {(int)((long long) f[0] * g[0] % mod)};
    precompute(n); vector<int> a = f, b = g; a.resize(lim); b.
        resize(lim);
    ntt(a, 1), ntt(b, 1); for (int i = 0; i < lim; ++i) a[i] =
        (long long) a[i] * b[i] % mod;
    ntt(a, 0); a.resize(n + 1); return a;
}

```

### 6.3 NTT with any prime mod short

```

const int N = 3e5 + 9, mod = 998244353;

struct base {
    double x, y;
    base() { x = y = 0; }
    base(double x, double y): x(x), y(y) {}
};
inline base operator + (base a, base b) { return base(a.x +
    b.x, a.y + b.y); }
inline base operator - (base a, base b) { return base(a.x -
    b.x, a.y - b.y); }
inline base operator * (base a, base b) { return base(a.x *
    b.x - a.y * b.y, a.x * b.y + a.y * b.x); }
inline base conj(base a) { return base(a.x, -a.y); }
int lim = 1; vector<base> roots = {{0, 0}, {1, 0}}; vector<
    int> rev = {0, 1};
const double PI = acos(-1.0);
void ensure_base(int p) {
    if(p <= lim) return; rev.resize(1 << p);
    for(int i = 0; i < (1 << p); i++) rev[i] = (rev[i >> 1] >>
        1) + ((i & 1) << (p - 1));
    roots.resize(1 << p); while(lim < p) { double angle = 2 *
        PI / (1 << (lim + 1));
        for(int i = 1 << (lim - 1); i < (1 << lim); i++) {
            roots[i << 1] = roots[i]; double angle_i = angle * (2 *
                i + 1 - (1 << lim));
            roots[(i << 1) + 1] = base(cos(angle_i), sin(angle_i));
        } lim++;
    }
}
void fft(vector<base> &a, int n = -1) {

```

```

    if(n == -1) n = a.size(); assert((n & (n - 1)) == 0); int
        zeros = __builtin_ctz(n);
    ensure_base(zeros); int shift = lim - zeros;
    for(int i = 0; i < n; i++) if(i < (rev[i] >> shift)) swap(a[
        i], a[rev[i] >> shift]);
    for(int k = 1; k < n; k <= 1) { for(int i = 0; i < n; i +=
        2 * k) {
        for(int j = 0; j < k; j++) { base z = a[i + j + k] * roots[j
            + k]; a[i + j + k] = a[i + j] - z;
            a[i + j] = a[i + j] + z; }}}
    }
    //eq = 0: 4 FFTs in total
    //eq = 1: 3 FFTs in total
    vector<int> multiply(vector<int> &a, vector<int> &b, int eq
        = 0) {
        int need = a.size() + b.size() - 1; int p = 0; while((1 << p
            ) < need) p++;
        ensure_base(p); int sz = 1 << p; vector<base> A, B; if(sz >
            (int)A.size()) A.resize(sz);
        for(int i = 0; i < (int)a.size(); i++) {int x = (a[i] % mod
            + mod) % mod;
            A[i] = base(x & ((1 << 15) - 1), x >> 15); }
        fill(A.begin() + a.size(), A.begin() + sz, base{0, 0});
        fft(A, sz); if(sz > (int)B.size()) B.resize(sz);
        if(eq) copy(A.begin(), A.begin() + sz, B.begin());
        else { for(int i = 0; i < (int)b.size(); i++) {
            int x = (b[i] % mod + mod) % mod; B[i] = base(x & ((1 <<
                15) - 1), x >> 15); }
            fill(B.begin() + b.size(), B.begin() + sz, base{0, 0}); fft
                (B, sz); }
        double ratio = 0.25 / sz;
        base r2(0, -1), r3(ratio, 0), r4(0, -ratio), r5(0, 1);
        for(int i = 0; i <= (sz >> 1); i++) { int j = (sz - i) & (sz
            - 1);
            base a1 = (A[i] + conj(A[j])), a2 = (A[i] - conj(A[j])) *
                r2;
            base b1 = (B[i] + conj(B[j])) * r3, b2 = (B[i] - conj(B[j]))
                * r4;
            if(i != j) { base c1 = (A[j] + conj(A[i])), c2 = (A[j] -
                conj(A[i])) * r2;
                base d1 = (B[j] + conj(B[i])) * r3, d2 = (B[j] - conj(B[
                    i])) * r4;
                A[i] = c1 * d1 + c2 * d2 * r5; B[i] = c1 * d2 + c2 * d1; }
                A[j] = a1 * b1 + a2 * b2 * r5; B[j] = a1 * b2 + a2 * b1; }
                fft(A, sz); fft(B, sz); vector<int> res(need);
                for(int i = 0; i < need; i++) { long long aa = A[i].x +
                    0.5;
                    long long bb = B[i].x + 0.5; long long cc = A[i].y + 0.5;
                    res[i] = (aa + ((bb % mod) << 15) + ((cc % mod) << 30))%
                        mod; }
    }
}

```

```

    return res;
}
vector<int> pow(vector<int> &a, int p) {
    vector<int> res; res.emplace_back(1);
    while(p) { if(p & 1) res = multiply(res, a); a = multiply(
        a, a, 1);
        p >>= 1; } return res;
}

```

### 6.4 bigmod

```

ll bigmod(ll a, ll b, ll n) {ll res = 1;
    if (b == 0) return 1; a = a % n; if (a == 0)
    return 0; while (b > 0) {if (b % 2) res = (res * a) % n;
    b = b / 2; a = (a * a) % n; } return res; }

```

### 6.5 extended euclid

```

#include <bits/stdc++.h>
using namespace std; int x, y;
int gcdExtended(int a, int b, int *x, int *y) {
    if (b==0) { *x = 1; *y = 0; return a; }
    int x1, y1; int gcd = gcdExtended(b, a%b, &x1, &y1);
    *x = y1; *y = x1 - y1 * (a/b); return gcd; }
int main() { int a = 50, b = 10;
    cout << "gcd " << gcdExtended(a, b, &x, &y) << endl;
    cout << x << " " << y << endl; return 0; }

```

### 6.6 fft

```

const double PI = acos(-1); struct base { double a, b; base(
    double a = 0, double b = 0) : a(a), b(b) {}
const base operator + (const base &c) const { return base(a +
    c.a, b + c.b); }
const base operator - (const base &c) const { return base(a -
    c.a, b - c.b); }
const base operator * (const base &c) const { return base(a *
    c.a - b * c.b, a * c.b + b * c.a); }
}; void fft(vector<base> &p, bool inv = 0) { int n = p.size(),
    i = 0; for(int j = 1; j < n - 1; ++j) {
    for(int k = n >> 1; k > (i ^= k); k >>= 1); if(j < i) swap(p[
        i], p[j]); }
    for(int l = 1, m; (m = 1 << 1) <= n; l <= 1) { double ang =
        2 * PI / m;

```

```
base wn = base(cos(ang), (inv ? 1. : -1.) * sin(ang)), w;for
    (int i = 0, j, k; i < n; i += m) {
for(w = base(1, 0), j = i, k = i + 1; j < k; ++j, w = w * wn
    ) {
base t = w * p[j + 1];p[j + 1] = p[j] - t;p[j] = p[j] + t
    ;}}
if(inv) for(int i = 0; i < n; ++i) p[i].a /= n, p[i].b /= n
    ;}
vector<long long> multiply(vector<int> &a, vector<int> &b) {
int n = a.size(), m = b.size(), t = n + m - 1, sz = 1;while(
    sz < t) sz <= 1;
vector<base> x(sz), y(sz), z(sz);for(int i = 0; i < sz; ++i
    ) {
x[i] = i < (int)a.size() ? base(a[i], 0) : base(0, 0);
y[i] = i < (int)b.size() ? base(b[i], 0) : base(0, 0);}
fft(x), fft(y);for(int i = 0; i < sz; ++i) z[i] = x[i] * y[i
    ];fft(z, 1);
vector<long long> ret(sz);for(int i = 0; i < sz; ++i) ret[i]
    = (long long) round(z[i].a);
// while((int)ret.size() > 1 && ret.back() == 0) ret.
    pop_back();
return ret;}
```

## 6.7 linear diophantine equation

```
#include<bits/stdc++.h> using namespace std;using ll = long
    long;
ll extended_euclid(ll a, ll b, ll &x, ll &y) {
ll xx = y = 0;ll yy = x = 1;while (b) {
ll q = a / b;ll t = b; b = a % b; a = t;
t = xx; xx = x - q * xx; x = t;
t = yy; yy = y - q * yy; y = t;}return a;}
// a*x+b*y=c. returns valid x and y if possible.
// all solutions are of the form (x0 + k * b / g, y0 - k * b
    / g)
bool find_any_solution (ll a, ll b, ll c, ll &x0, ll &y0, ll
    &g) {
if (a == 0 and b == 0) {if (c) return false;
x0 = y0 = g = 0; return true;}
g = extended_euclid (abs(a), abs(b), x0, y0);if (c % g != 0)
    return false;
x0 *= c / g;y0 *= c / g;if (a < 0) x0 *= -1;if (b < 0) y0 *=
    -1;
return true;}void shift_solution(ll &x, ll &y, ll a, ll b,
    ll cnt) {
x += cnt * b;y -= cnt * a;}
// returns the number of solutions where x is in the range[
    minx, maxx] and y is in the range[miny, maxy]
```

```
ll find_all_solutions(ll a, ll b, ll c, ll minx, ll maxx, ll
    miny, ll maxy) {ll x, y, g;
if (find_any_solution(a, b, c, x, y, g) == 0) return 0;if (a
    == 0 and b == 0) {
assert(c == 0);return 1LL * (maxx - minx + 1) * (maxy - miny
    + 1);}
if (a == 0) {return (maxx - minx + 1) * (miny <= c / b and c
    / b <= maxy);}
if (b == 0) {return (maxy - miny + 1) * (minx <= c / a and c
    / a <= maxx);}
a /= g, b /= g;ll sign_a = a > 0 ? +1 : -1;ll sign_b = b > 0
    ? +1 : -1;
shift_solution(x, y, a, b, (minx - x) / b);if (x < minx)
    shift_solution(x, y, a, b, sign_b);
if (x > maxx) return 0;ll lx1 = x;shift_solution(x, y, a, b,
    (maxx - x) / b);
if (x > maxx) shift_solution (x, y, a, b, -sign_b);ll rx1 =
    x;shift_solution(x, y, a, b, -(miny - y) / a);
if (y < miny) shift_solution (x, y, a, b, -sign_a);if (y >
    maxy) return 0;
ll lx2 = x;shift_solution(x, y, a, b, -(maxy - y) / a);if (y
    > maxy) shift_solution(x, y, a, b, sign_a);
ll rx2 = x;if (lx2 > rx2) swap (lx2, rx2);ll lx = max(lx1,
    lx2);ll rx = min(rx1, rx2);
if (lx > rx) return 0;return (rx - lx) / abs(b) + 1;}
int32_t main() { ios_base::sync_with_stdio(0);cin.tie(0);int
    t, cs = 0; cin >> t;
while (t--) {ll a, b, c, x1, x2, y1, y2; cin >> a >> b >> c
    >> x1 >> x2 >> y1 >> y2;
cout << "Case " << ++cs << ": " << find_all_solutions(a, b,
    -c, x1, x2, y1, y2) << '\n';}
return 0;}// https://lightoj.com/problem/solutions-to-an-
    equation
```

## 6.8 linear sieve

```
const ll N = 1e7 + 7;bool isPrime[N];vector < ll > p;
void lin_sieve () {ll i;for (i = 2; i < N; i++) {
if (!isPrime[i]) p.push_back (i);for (ll j : p) {
if (i * j >= N) break;isPrime[i * j] = 1;if (i % j == 0)
    break;}}}
```

## 6.9 mat expo

```
mt19937 mt_rand(chrono::high_resolution_clock::now().
    time_since_epoch().count());
const int mod = 1e9 + 7;// const int N = 5e5 + 6;
```

```
// No of terms in the Recurrence Relation.
const int N = 4;const long long M = 1000000007;
// Multiplies two matrices A and B and stores the result in
    A.
void multiply (long long A[N][N], long long B[N][N]){long
    long R[N][N];
// Multiply A and B and store result in R.
for (int i = 0; i < N; i++){for (int j = 0; j < N; j++){
R[i][j] = 0;for (int k = 0; k < N; k++){R[i][j] = (R[i][j] +
    A[i][k] * B[k][j]) % M;}}}
// Copy contents of R in A.
for (int i = 0; i < N; i++){for (int j = 0; j < N; j++){A[i
    ][j] = R[i][j];}}}
// Raise matrix A to the power of n in O(log n).
void power_matrix (long long A[N][N], ll n){long long B[N][N
    ];// B = Identity Matrix.
for (int i = 0; i < N; i++){for (int j = 0; j < N; j++){B[i
    ][j] = A[i][j];}}
// A = A * A ^ (n - 1).
n = n - 1;while (n > 0){// If n is odd, A = A * B.
if (n & 1)multiply (A, B);// B = B * B.
multiply (B, B);// n = n / 2.
n = n >> 1;}// A = Coefficient Matrix, B = Base Matrix.
// It returns the nth term of the recurrence relation formed
    from A and B in O(log n).
long long solve_recurrence (long long A[N][N], long long B[N
    ][1], ll n){//Base Cases.
if (n < N)return B[n - 1 - n][0];// A = A ^ (n - N + 1).
power_matrix (A, n - N + 1);long long result = 0;for (int i
    = 0; i < N; i++)
result = (result + A[0][i] * B[i][0]) % M;return result;}
void Solve(){
/*
The recurrence relation used here is: -
R(n) = 2 * R(n-1) + R(n-2) + 3 * R(n-3) + 3.
Base Cases: R(0) = 1, R(1) = 2, R(2) = 3.
*/
// Forming the Coefficient Matrix
long long A[N][N] = {{3, 2, 1, 1}, {1, 0, 0, 0}, {0, 1, 0,
    0}, {0, 0, 0, 1}};
//Forming the Base Matrix
long long B[N][1] = {{3}, {2}, {1}, {3}};ll n;cin >> n;if (n
    <= 2){
cout << n + 1 << endl;return ;}long long R_n =
    solve_recurrence (A, B, n + 1);
cout << R_n << endl;}
```

## 6.10 mobius

---

```
void mobius() {mob[1] = 1;for (int i = 2; i < N; i++){
mob[i]--;for (int j = i + i; j < N; j += i) {mob[j] -= mob[i]
];}}}
```

---

## 6.11 ncr for mod

---

```
ll fact[200008];ll bigmod(ll b, ll p) {if(p == 0) return 1;
ll h = bigmod(b, p/2);h = h * h % mod;if(p&1) h = h * b %
mod;
return h;};ll ncr(ll n, ll r) {if(n<r)return 0;
return fact[n] * bigmod(fact[r] * fact[n-r] % mod, mod - 2)
% mod;};
void Fact(){fact[0] = 1;for(int i=1; i<=200002; i++)
fact[i] = fact[i-1] * i % mod;}
```

---

## 6.12 ntt

---

```
const int N = 1 << 20;const int mod = 998244353;const int
root = 3;
int lim, rev[N], w[N], wn[N], inv_lim;void reduce(int &x) {
x = (x + mod) % mod; }
int POW(int x, int y, int ans = 1) {for (; y >= 1, x = (
long long) x * x % mod)
if (y & 1) ans = (long long) ans * x % mod;return ans;}void
precompute(int len) {
lim = wn[0] = 1; int s = -1;while (lim < len) lim <= 1, ++s
;
for (int i = 0; i < lim; ++i) rev[i] = rev[i >> 1] >> 1 | (i
& 1) << s;
const int g = POW(root, (mod - 1) / lim);inv_lim = POW(lim,
mod - 2);
for (int i = 1; i < lim; ++i) wn[i] = (long long) wn[i - 1]
* g % mod;};
void ntt(vector<int> &a, int typ) {
for (int i = 0; i < lim; ++i) if (i < rev[i]) swap(a[i], a[
rev[i]]);
for (int i = 1; i < lim; i <= 1) {for (int j = 0, t = lim /
i / 2; j < i; ++j)
w[j] = wn[j * t];for (int j = 0; j < lim; j += i << 1) {for
(int k = 0; k < i; ++k) {
const int x = a[k + j], y = (long long) a[k + j + i] * w[k]
% mod;
reduce(a[k + j] += y - mod), reduce(a[k + j + i] = x - y)
;}}}if (!typ) {
reverse(a.begin() + 1, a.begin() + lim);for (int i = 0; i <
lim; ++i)
```

---

```
a[i] = (long long) a[i] * inv_lim % mod;}}
vector<int> multiply(vector<int> &f, vector<int> &g) {
if (f.empty() or g.empty()) return {};int n = (int)f.size()
+ (int)g.size() - 1;
if (n == 1) return {(int)((long long) f[0] * g[0] % mod)};
precompute(n);
vector<int> a = f, b = g;a.resize(lim); b.resize(lim);
ntt(a, 1), ntt(b, 1);for (int i = 0; i < lim; ++i)
a[i] = (long long) a[i] * b[i] % mod;ntt(a, 0);a.resize(n +
1);return a;}
```

---

## 6.13 phi

---

```
// for n<=1e6 by nloglogn
void phi(int n){vector<int>phi(n+1) ;
for(int i = 0; i <= n; i++){phi[i] = i;for(int i = 2; i <=n;
i++){
if(phi[i] == i){for(int j = i; j <= n; j += i){
phi[j] -= phi[j]/i ;}}}}
// for single number by sqrt(n)
int phi(int n){int res = n ;for(int i = 2; i * i <= n ; i++)
{
if(n % i == 0){while(n % i == 0) n /= i ;res -= res / i ;}}
if(n > 1) res -= res / n ;return res ;}
```

---

## 6.14 pollard rho

---

```
using ll = long long;namespace PollardRho {
mt19937 rnd(chrono::steady_clock::now().time_since_epoch().
count());
const int P = 1e6 + 9;ll seq[P];int primes[P], spf[P];
inline ll add_mod(ll x, ll y, ll m) {return (x += y) < m ? x
: x - m;};
inline ll mul_mod(ll x, ll y, ll m) {ll res = __int128(x) *
y % m;
return res;// ll res = x * y - (ll)((long double)x * y / m +
0.5) * m;
// return res < 0 ? res + m : res;
}inline ll pow_mod(ll x, ll n, ll m) {ll res = 1 % m;
for (; n; n >= 1) {if (n & 1) res = mul_mod(res, x, m);
x = mul_mod(x, x, m);return res;}
// 0(it * (logn)^3), it = number of rounds performed
inline bool miller_rabin(ll n) {if (n <= 2 || (n & 1 ^ 1))
return (n == 2);
if (n < P) return spf[n] == n;ll c, d, s = 0, r = n - 1;for
(; !(r & 1); r >= 1, s++) {}
// each iteration is a round
```

---

```
for (int i = 0; primes[i] < n && primes[i] < 32; i++) {c =
pow_mod(primes[i], r, n);
for (int j = 0; j < s; j++) {d = mul_mod(c, c, n);if (d == 1
&& c != 1 && c != (n - 1)) return false;
c = d;}if (c != 1) return false;}return true;} void init() {
int cnt = 0;
for (int i = 2; i < P; i++) { if (!spf[i]) primes[cnt++] =
spf[i] = i;
for (int j = 0, k; (k = i * primes[j]) < P; j++) { spf[k] =
primes[j];
if (spf[i] == spf[k]) break;}}// returns 0(n^(1/4))
ll pollard_rho(ll n) {while (1) {ll x = rnd() % n, y = x, c
= rnd() % n, u = 1, v, t = 0;
ll *px = seq, *py = seq;while (1) {*py++ = y = add_mod(
mul_mod(y, y, n), c, n);
*py++ = y = add_mod(mul_mod(y, y, n), c, n);if ((x = *px++)
== y) break;v = u;
u = mul_mod(u, abs(y - x), n);if (!u) return __gcd(v, n);
if (++t == 32) {t = 0;if ((u = __gcd(u, n)) > 1 && u < n)
return u;}}
if (t && (u = __gcd(u, n)) > 1 && u < n) return u;}}
vector<ll> factorize(ll n) {if (n == 1) return vector<ll>{};
if (miller_rabin(n)) return vector<ll>{n};
vector<ll> v, w; while (n > 1 && n < P) { v.push_back(spf[n
]);n /= spf[n];}
if (n >= P) { ll x = pollard_rho(n); v = factorize(x); w =
factorize(n / x);
v.insert(v.end(), w.begin(), w.end());}return v;}}
int32_t main() { ios_base::sync_with_stdio(0); cin.tie(0);
PollardRho::init();
int t; cin >> t; while (t--) { ll n; cin >> n; auto f =
PollardRho::factorize(n);
sort(f.begin(), f.end()); cout << f.size() << ' '; for (auto
x: f) cout << x << ' '; cout << '\n';}return 0;}
// https://judge.yosupo.jp/problem/factorize
```

---

## 6.15 power tower

---

```
const int N = 1e5 + 9;using ll = long long;map<ll, ll> mp;
ll phi(ll n) { if (mp.count(n)) return mp[n];ll ans = n, m =
n;
for (ll i = 2; i * i <= m; i++) { if (m % i == 0) {while (m
% i == 0) m /= i;
ans = ans / i * (i - 1);}}if (m > 1) ans = ans / m * (m - 1)
;return mp[n] = ans;}
inline ll MOD(ll x, ll m) { if (x < m) return x;return x % m
+ m;}
ll power(ll n, ll k, ll mod) { ll ans = MOD(1, mod); while (
k) {
```

---

```

if (k & 1) ans = MOD(ans * n, mod); n = MOD(n * n, mod); k
>>= 1; } return ans; }
int a[N]; // if x >= log2(m), then a^x = a^(MOD(x, phi(m)))
% m
ll yo(ll l, ll r, ll m) { if (l == r) return MOD(a[l], m); if
(m == 1) return 1;
return power(a[l], yo(l + 1, r, phi(m)), m); }
int32_t main() { ios_base::sync_with_stdio(0); cin.tie(0);
int n, m; cin >> n >> m; for (int i = 1; i <= n; i++) { cin
>> a[i]; }
int q; cin >> q; while (q--) { int l, r; cin >> l >> r; cout
<< yo(l, r, m) % m << '\n'; }
return 0; } // https://codeforces.com/contest/906/problem/D

```

## 6.16 sieve all

```

vector<bool> prime(N, true); vector<int> vec;
void seive() { prime[0] = false; prime[1] = false;
for (int i = 2; i * i < N; i++) { if (prime[i]) {
for (int j = i * i; j < N; j += i) { prime[j] = false; } } }
for (int i = 2; i < N; i++) if (prime[i]) vec.push_back(i); }
void pro() { int n; cin >> n; int ans = 1;
for (auto it : vec) { if (it * it > n) break;
if (n % it == 0) { int cnt = 1; while (n % it == 0) {
n /= it; cnt++; } ans *= cnt; } if (n > 1) ans *= 2;
cout << ans - 1 << endl; } } // Segmented Sieve
void pro() { int n, m; cin >> n >> m; bool ara[m - n + 1];
memset(ara, true, sizeof(ara)); for (auto it : vec) { if (it
* it > m) break;
int fml = (n + it - 1) / it; fml *= it; int strt = max(
fml, it * it);
for (int j = strt; j <= m; j += it) { ara[j - n] = false; } }
if (n == 1) ara[0] = false; for (int i = n; i <= m; i++) if (
ara[i - n]) cout << i << endl; }

```

## 6.17 totient

```

#include<bits/stdc++.h> using namespace std;
const double pi = 2 * acos(0.0); const int N = 5000006;
const int INF = INT_MAX; const int mod = 1000000007;
vector<int> phi(N, 0); void totient_seive() {
for (int i = 1; i < N; i++) phi[i] = i; for (int i = 2; i < N; i++) {
if (phi[i] == i) { for (int j = i; j < N; j += i) {
phi[j] = (phi[j] - (phi[j] / i)); } } } }
// FOR ANY SINGLE NUMBER --- CALCULATING THE VALUE OF PHI
USING SQRT COMPLEXITY

```

```

const int N = 1000009; const int INF = INT_MAX; const int mod
= 1000000007;
int sqrt_phi(int n) { int ans = n; for (int i = 2; i * i <= n; i++) {
if (n % i == 0) { while (n % i == 0) n /= i; ans -= (ans / i); } }
if (n > 1) ans -= (ans / n); return ans; }
// CALCULATING PHI VALUE USING SUM OF PHI...
// SUM OF TOTAL VALUE FOR ALL DIVISOR OF N IS EQUAL TO N
// PHI(10) + PHI(5) + PHI(2) + PHI(1)
// = 4 + 4 + 1 + 1
// 10
const int N = 10000007; const int INF = INT_MAX; const int mod
= 1000000007;
vector<int> phi(N, 0); void calc_phi() { phi[0] = 0; phi[1] = 1;
for (int i = 2; i < N; i++) phi[i] = i - 1;
// (loop er modde 1 divisor hisabe ani nai..... tai 1 er
contribution bad)
for (int i = 2; i < N; i++) { for (int j = 2 * i; j < N; j += i) { phi[j] -=
phi[i]; } } }

```

## 7 String

### 7.1 Aho

```

#include<bits/stdc++.h>
using namespace std; const int N = 1e5 + 9; // credit: Alpha_Q
struct AC { int N, P; const int A = 26; vector<vector<int>
>> next;
vector<int> link, out_link; vector<vector<int>> out;
AC(): N(0), P(0) { node(); } int node() { next.emplace_back(A,
0);
link.emplace_back(0); out_link.emplace_back(0);
out.emplace_back(0); return N++; } inline int get(char c) {
return c - 'a'; }
} int add_pattern(const string T) { int u = 0; for (auto c :
T) {
if (!next[u][get(c)]) next[u][get(c)] = node(); u = next[u][
get(c)]; }
out[u].push_back(P); return P++; } void compute() { queue<int>
q;
for (q.push(0); !q.empty(); ) { int u = q.front(); q.pop();
for (int c = 0; c < A; ++c) { int v = next[u][c];
if (!v) next[u][c] = next[link[u]][c]; else { link[v] = u ?
next[link[u]][c] : 0;
out_link[v] = out[link[v]].empty() ? out_link[link[v]] :
link[v]; q.push(v); } } }
int advance(int u, char c) { while (u && !next[u][get(c)])
u = link[u];
u = next[u][get(c)]; return u; } }

```

```

int32_t main() { ios_base::sync_with_stdio(0);
cin.tie(0); auto st = clock(); int t, cs = 0; cin >> t;
while (t--) { int n; cin >> n; vector<string> v;
for (int i = 0; i < n; i++) { string s; cin >> s;
v.push_back(s); } sort(v.begin(), v.end());
v.erase(unique(v.begin(), v.end()), v.end()); AC aho;
vector<int> len(n + 3, 0); for (auto s : v) { len[aho.
add_pattern(s)] = s.size(); }
aho.compute(); string s; cin >> s; n = s.size();
vector<int> dp(n, n + 10); int u = 0; for (int i = 0; i < n; i
++) {
char c = s[i]; u = aho.advance(u, c); for (int v = u; v; v =
aho.out_link[v]) {
for (auto p : aho.out[v]) { dp[i] = min(dp[i], (i - len[p]
>= 0 ? dp[i - len[p]] : 0) + 1); } } }
cout << "Case " << ++cs << ": "; if (dp[n - 1] == n + 10) {
cout << "impossible\n"; }
else { cout << dp[n - 1] << '\n'; } } cout << 1.0 * (clock() -
st) / 1000 << '\n'; return 0; }

```

## 7.2 String matching using bitset

```

#include<bits/stdc++.h>
using namespace std; const int N = 1e5 + 9;
vector<int> v; bitset<N> bs[26], oc;
int main() { int i, j, k, n, q, l, r; string s, p;
cin >> s; for (i = 0; i < s.size(); i++) bs[s[i] - 'a'][i] = 1;
cin >> q; while (q--) { cin >> p; oc.set(0);
for (i = 0; i < p.size(); i++) oc |= (bs[p[i] - 'a'] >> i);
cout << oc.count() << endl; // number of occurrences
int ans = N, sz = p.size(); int pos = oc._Find_first();
v.push_back(pos); pos = oc._Find_next(pos); while (pos < N) {
v.push_back(pos); pos = oc._Find_next(pos); }
for (auto x : v) cout << x << ' '; // position of occurrences
cout << endl; v.clear(); cin >> l >> r; // number of
occurrences from l to r, where l and r is 1-indexed
if (sz > r - l + 1) cout << 0 << endl; else cout << (oc >> (l
- 1)).count() - (oc >> (r - 1)).count() << endl;
} return 0; }

```

## 7.3 Trie

```

struct node { bool endmark; node* next[26 + 1]; node() {
endmark = false; for (int i = 0; i < 26; i++) next[i] = NULL; }
} * root; void insert(char* str, int len) { node* curr = root;
for (int i = 0; i < len; i++) { int id = str[i] - 'a';
if (curr->next[id] == NULL) curr->next[id] = new node();

```

```
curr = curr->next[id];}curr->endmark = true;}
bool search(char* str, int len){node* curr = root;for (int i
= 0; i < len; i++) {
int id = str[i] - 'a';if (curr->next[id] == NULL)return
false;curr = curr->next[id];
}return curr->endmark;}void del(node* cur){for (int i = 0; i
< 26; i++)if (cur->next[i])
del(cur->next[i]);delete (cur);}puts("ENTER NUMBER OF WORDS"
);
root = new node();int num_word;cin >> num_word;for (int i =
1; i <= num_word; i++) {
char str[50];scanf("%s", str);insert(str, strlen(str));}
puts("ENTER NUMBER OF QUERY");int query;cin >> query;
for (int i = 1; i <= query; i++) {char str[50];scanf("%s",
str);if (search(str, strlen(str)))
puts("FOUND");elseputs("NOT FOUND");}del(root);return 0;
```

## 7.4 kmp

```
// returns the longest proper prefix array of pattern p
// where lps[i]=longest proper prefix which is also suffix
of p[0...i]
vector<int> build_lps(string p) {int sz = p.size();vector<
int> lps;
lps.assign(sz + 1, 0);int j = 0;lps[0] = 0;for (int i = 1; i
< sz; i++) {
while (j >= 0 && p[i] != p[j]) {if (j >= 1) j = lps[j - 1];
else j = -1;}
j++;lps[i] = j;}return lps;}vector<int> ans;
// returns matches in vector ans in 0-indexed
void kmp(vector<int> lps, string s, string p) {int psz = p.
size(), sz = s.size();
int j = 0;for (int i = 0; i < sz; i++) {while (j >= 0 && p[j
] != s[i])
if (j >= 1) j = lps[j - 1];else j = -1;j++;if (j == psz) {
j = lps[j - 1];// pattern found in string s at position i-
psz+1
ans.push_back(i - psz + 1);}
// after each loop we have j=longest common suffix of s[0..i
] which is also prefix of p}}
```

## 7.5 manacher

```
struct Manacher {vector<int> p[2];
// p[1][i] = (max odd length palindrome centered at i) / 2 [
floor division]
// p[0][i] = same for even, it considers the right center
```

```
// e.g. for s = "abbabba", p[1][3] = 3, p[0][2] = 2
Manacher(string s) {int n = s.size();p[0].resize(n + 1);
p[1].resize(n);for (int z = 0; z < 2; z++) {
for (int i = 0, l = 0, r = 0; i < n; i++) {int t = r - i + !
z;
if (i < r) p[z][i] = min(t, p[z][l + t]);int L = i - p[z][i
], R = i + p[z][i] - !z;
while (L >= 1 && R + 1 < n && s[L - 1] == s[R + 1])p[z][i
]++, L--, R++;
if (R > r) l = L, r = R;}}}bool is_palindrome(int l, int r)
{
int mid = (l + r + 1) / 2, len = r - l + 1;return 2 * p[len
% 2][mid] + len % 2 >= len;}};
```

## 7.6 palindrome<sub>hashing</sub>

```
#include <bits/stdc++.h>
using namespace std;
vector<vector<long long>> HASH, REV_HASH, POW;
vector<int> BASE = {1231, 1567}, MOD = {10000000000 + 7,
10000000000 + 9};
#define lim 1000006 string text, pattern;void init(){
POW = vector<vector<long long>>(2, vector<long long>(lim));
POW[0][0] = POW[1][0] = 1;for (int b = 0; b < 2; b++)
for (int j = 1; j < lim; j++)POW[b][j] = (POW[b][j - 1] *
BASE[b]) % MOD[b];return;
}void initHash(string str){int len = str.size();HASH[0][0] =
HASH[1][0] = 0;
for (int b = 0; b < 2; b++)for (int i = 1; i <= len; i++)
HASH[b][i] = (HASH[b][i - 1] * BASE[b] + (str[i - 1] - 'a' +
1)) % MOD[b];
REV_HASH[0][len + 1] = REV_HASH[1][len + 1] = 0;for (int b =
0; b < 2; b++)
for (int i = len; i; i--)REV_HASH[b][i] = (REV_HASH[b][i +
1] * BASE[b] + (str[i - 1] - 'a' + 1)) % MOD[b];
return;}long long getHash(int left, int right, int hsh){int
len = (right - left + 1);
long long ret = (HASH[hsh][right] - HASH[hsh][left - 1] *
POW[hsh][len]) % MOD[hsh];
if (ret < 0)ret += MOD[hsh];return ret;}
pair<long long, long long> getHash(int left, int right){long
long hsh0 = getHash(left, right, 0);
long long hsh1 = getHash(left, right, 1);return {hsh0, hsh1
};}
long long getRevHash(int left, int right, int hsh){int len =
(right - left + 1);
long long ret = (REV_HASH[hsh][left] - REV_HASH[hsh][right +
1] * POW[hsh][len]) % MOD[hsh];}
```

```
if (ret < 0)ret += MOD[hsh];return ret;}pair<long long, long
long> getRevHash(int left, int right){
long long hsh0 = getRevHash(left, right, 0);long long hsh1 =
getRevHash(left, right, 1);
return {hsh0, hsh1};}bool palindrome(int l, int r){return
getHash(l, r) == getRevHash(l, r);}
void solve(){string s = "aaabbabbaaac";HASH = vector<vector<
long long>>(2, vector<long long>(s.size() + 5));
REV_HASH = vector<vector<long long>>(2, vector<long long>(s.
size() + 5));initHash(s);
cout << (palindrome(1, s.size()) ? "YES\n" : "NO\n");cout <<
(palindrome(1, s.size() - 1) ? "YES\n" : "NO\n");
return;}int32_t main(){ios_base::sync_with_stdio(0);cin.tie
(0);init();solve();return 0;}
```

## 7.7 pallindromic tree

```
// s = "#" + s;
struct PaliTree{ #define sz 26 struct node{ int
lng; int link; int next[sz]; int occ; node(int _
lng){ lng = _lng; link = 0; occ = 0; memset(next
,-1,sizeof(next)); } }; vector<node> tree; strin
g s; int cur; PaliTree(){ tree.push_back(node(-
1)); //img
tree.push_back(node(0)); //root
cur = 1; } void clear(){ tree.clear(); tree.push
_back(node(-1)); //img
tree.push_back(node(0)); //root
cur = 1; } int get_id(char c){return c-'a';} int
get_link(int now,int i){ char c = s[i]; while(1
){ if(now == 0 or (i-1-tree[now].lng > 0 and s[i -1-tree[now
].lng] == c)) break; now = tree[now].
link; } int id = get_id(c); return (tree[now].ne
xt[id] == -1)?now:tree[now].next[id]; } void add
(int i){ char c = s[i]; int id = get_id(c); whil
e(1){ if(cur == 0 or (i-1-tree[cur].lng > 0 and
s[i-1-tree[cur].lng] == c)) break; cur = tree[cu
r].link; } if(cur == 0 and s[i] == s[i-1]) cur =
1; if(tree[cur].next[id] == -1){ node tmp(tree[
cur].lng+2); if(tmp.lng == 1) tmp.link = 1; else
tmp.link = get_link(tree[cur].link,i); tree.pus
h_back(tmp); tree[cur].next[id] = tree.size()-1;
} cur = tree[cur].next[id]; tree[cur].occ++; }
void calc(){ for(int i = tree.size()-1; i > 1; i --) tree[
tree[i].link].occ += tree[i].occ; } };
```



## 7.8 suffix array occurence of substr in own string

```
int Table[N][20], a[N]; void Build(vector<int> lcp){
int n = lcp.size(); for (int i = 1; i <= n; i++){
Table[i][0] = lcp[i - 1]; for (int k = 1; k < 20; k++){
for (int i = 1; i + (1 << k) - 1 <= n; i++){
Table[i][k] = min(Table[i][k - 1], Table[i + (1 << (k - 1))
][k - 1]);}}
int Query(int l, int r){l++, r++; int k = log2(r - l + 1);
return min(Table[l][k], Table[r - (1 << k) + 1][k]);}
pair<int, int> FindRight(int low, int high, int val) // Find
maximum R such that lcp(low, low+1...) > val and return
lcp(low, R)
{int l = low, r = high, mid; int ans = low - 1; while (l <= r)
{
mid = (l + r) / 2; if (Query(low, mid) > val){ans = mid, l =
mid + 1;}
else r = mid - 1; if (ans == low - 1) return {low, -1};
else return {ans + 1, Query(low, ans)};
}
void Solve(){string s; cin >> s; SuffixArray ehhe(s);
ll n = s.size(); vector<int> p = ehhe.sa; vector<int> lcp;
lcp = ehhe.lcp; Build(lcp); ll ans = 0;
for (int i = 0; i < n; i++){ int high = n - 1; int pans = i
? lcp[i - 1] : 0;
int len = n - p[i]; while (pans < len){
pair<int, int> pt = FindRight(i, high, pans); // pt = {
maximum r such that lcp(i, r) > val, lcp(i, r)}
int right = pt.f; ll templ = right - i + 1;
if (pt.f == i) pt.s = len; ll contr = (pt.s - pans);
ans += (contr * (templ * (templ))); // len of contr occurs
templ times
high = pt.f; pans = pt.s; } cout << ans << endl; }
// Problem link : https://codeforces.com/contest/802/problem
/I
```

## 7.9 z algo

```
// An element Z[i] of Z array stores length of the longest
substring
// starting from str[i] which is also a prefix of str[0..n
-1].
// The first entry of Z array is meaning less as complete
string is always prefix of itself.
// Here Z[0]=0.
vector<int> z_function(string s) {int n = (int) s.length();
vector<int> z(n);
```

```
for (int i = 1, l = 0, r = 0; i < n; ++i) {if (i <= r) z[i] =
min (r - i + 1, z[i - l]);
while (i + z[i] < n && s[z[i]] == s[i + z[i]]) ++z[i];
if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1; } return z; }
```

## 8 Tree

### 8.1 Articulation bridge

```
vector<int> node[10003]; int lowtime[10003], intime[10003], vis
[10003];
vector<pair<int, int>> edge; int timer;
void dfs(int p, int parent) {intime[p] = lowtime[p] = timer; timer
++;
vis[p] = 1; for (int child : node[p]) {if (child == parent) continue;
if (vis[child]) {lowtime[p] = min(lowtime[p], intime[child]); //
node - child is a back edge
} else { // node - child is a forward edge
dfs(child, p); if (lowtime[child] > intime[p]) edge.push_back({p
, child});
lowtime[p] = min(lowtime[p], lowtime[child]); } } }
```

### 8.2 Articulation point

```
vector<int> node[10003]; int lowtime[10003], intime[10003], vis
[10003];
set<int> cut_vertex; int timer;
void IS_CUTPOINT(int x) {cut_vertex.insert(x);}
void dfs(int p, int parent) {intime[p] = lowtime[p] = timer;
timer++; int children = 0; vis[p] = 1; for (int child : node[p]) {
if (child == parent) continue; if (vis[child]) {
lowtime[p] = min(lowtime[p], intime[child]); // node - child
is a back edge
} else { // node - child is a forward edge
dfs(child, p); if (lowtime[child] >= intime[p] && parent != -1)
IS_CUTPOINT(p); lowtime[p] = min(lowtime[p], lowtime[child]);
;
children++; } } if (parent == -1 && children > 1) // for root
IS_CUTPOINT(p); }
```

### 8.3 Dijkstra

```
priority_queue<pair<ll, ll>, vector<pair<ll, ll>>,
greater<pair<ll, ll>>> pq;
```

```
int parrent[100003]; void dijkstra(ll p) {parrent[p] = 1; pq.
push({0, p});
dis[p] = 0; ans.push_back(p); while (!pq.empty()) {
ll curr_node = pq.top().second; ll curr_dis = pq.top().first;
pq.pop(); for (pair<ll, ll> child : node[curr_node]) {
if (child.second + curr_dis < dis[child.first]) {
parrent[child.first] = curr_node; dis[child.first] = child.
second + curr_dis;
pq.push({dis[child.first], child.first}); } } }
```

### 8.4 Euler tour

```
vector<int> node[N]; int Intime[N], Outtime[N], Level[N], a[N
];
int timer = 1; int n, q; void EulerTour(int p, int par, int d)
{
Intime[p] = timer++; Level[p] = d; for (auto i : node[p]) {if
(i == par)
continue; EulerTour(i, p, Level[p] + 1); } Outtime[p] = timer;
// if timer++ then intime[u] to intime[v] can be find path
query
} struct BIT { T[2]; void Solve() { cin >> n >> q;
for (int i = 1; i <= n; i++) cin >> a[i];
for (int i = 1; i < n; i++) { int u, v; cin >> u >> v;
node[u].push_back(v); node[v].push_back(u); EulerTour(1, 0,
0);
for (int i = 1; i <= n; i++) { T[Level[i] % 2].upd(Intime[i
], Intime[i], a[i]);
T[!(Level[i] % 2)].upd(Intime[i], Intime[i], 0); } while (q--)
{ int type;
cin >> type; if (type == 1) { ll x, val; cin >> x >> val;
T[Level[x] % 2].upd(Intime[x], Outtime[x] - 1, val);
T[!(Level[x] % 2)].upd(Intime[x], Outtime[x] - 1, -val); } else
{ ll x;
cin >> x; cout << T[Level[x] % 2].query(Intime[x], Intime[x
]) << endl; } } }
```

### 8.5 Floyd Warshall

```
ll vis[504][504]; void warshall(ll n) {
for (int k = 1; k <= n; k++) { for (int i = 1; i <= n; i++)
{ for (int j = 1; j <= n; j++) {
dis[i][j] = min(dis[i][j], dis[i][k] + dis[k][j]); } } }
```

### 8.6 HLD(update on edge)

```

ll n, q; vector<ll> node[N]; ll a[N];
struct SegmentTree { vector<ll> tree;
vector<ll> lazy; vector<ll> aa; SegmentTree() {
tree.resize(4 * N); lazy.resize(4 * N); aa.resize(4 * N);}
void build(ll node, ll b, ll e) { if (b == e) {
tree[node] = 0; lazy[node] = -1; return;}
ll mid = (b + e) >> 1; build(2 * node, b, mid);
build(2 * node + 1, mid + 1, e); lazy[node] = -1;
tree[node] = tree[2 * node] + tree[2 * node + 1];}
void push(ll node, ll b, ll e) { tree[node] = (e - b + 1) *
lazy[node];
if (b != e) { lazy[2 * node] = lazy[2 * node + 1] = lazy[
node];}
lazy[node] = -1;} void update(ll node, ll b, ll e, ll l, ll
r, ll x) {
if (lazy[node] != -1) push(node, b, e); if (l > e || r < b)
return;
if (l <= b && r >= e) { tree[node] = (e - b + 1) * x;
if (b != e) { lazy[2 * node] = lazy[2 * node + 1] = x; }
lazy[node] = -1; return;} ll mid = (b + e) >> 1;
update(2 * node, b, mid, l, r, x); update(2 * node + 1, mid
+ 1, e, l, r, x);
tree[node] = tree[2 * node] + tree[2 * node + 1];}
ll query(ll node, ll b, ll e, ll l, ll r) { if (lazy[node]
!= -1)
push(node, b, e); if (l > e || r < b) return 0; if (l <= b
&& r >= e) {return tree[node];} ll mid = (b + e) >> 1;
return query(2 * node, b, mid, l, r) + query(2 * node +
1, mid + 1, e, l, r);
}
}; SegmentTree st; ll par[N][LG + 1], dep[N], sz[N]; void dfs(
int u, int p = 0) {
par[u][0] = p; dep[u] = dep[p] + 1; sz[u] = 1; for (int i = 1;
i <= LG; i++) {
par[u][i] = par[par[u][i - 1]][i - 1];} for (auto v : node[u
]){
if (v == p) continue; dfs(v, u); sz[u] += sz[v];}
int lca(int u, int v) {
// ache already}
int intime[N], head[N]; int timer = 1; map<ll, ll> alledge[N];
void decompose(int p, int parent, int Head_node) { intime[p] =
timer++;
head[p] = Head_node; st.update(1, 1, n, intime[p], intime[p],
alledge[parent][p]);
int heavysize = -1, heavychild = -1; for (auto i : node[p]) {
if (i != parent) {
if (sz[i] > heavysize) heavysize = sz[i], heavychild = i;}
if (heavychild == -1)
return; decompose(heavychild, p, Head_node); for (auto i :
node[p]) {

```

```

if (i == heavychild || i == parent) continue; decompose(i, p,
i);}
}
ll sumpath(int u, int v) { ll ans = 0;
//cout << "here " << u << " " << v << endl;
if (u == v) return 0; while (head[u] != head[v]) { if (dep[
head[u]] > dep[head[v]])
swap(u, v); ans += st.query(1, 1, n, intime[head[v]], intime
[v]); v = par[head[v]][0];}
if (dep[u] > dep[v]) swap(u, v); if (u != v) ans += st.query(1,
1, n, intime[u] + 1, intime[v]);
return ans;} void reset(int n) { for (int i = 0; i <= n; i++)
{
intime[i] = head[i] = dep[i] = sz[i] = 0; a[i] = 0; node[i].
clear();
timer = 1; alledge[i].clear();} } void Solve() { cin >> n; //reset
(n);
st.build(1, 1, n); vector<pair<int, int>> edge; for (int i =
1; i < n; i++) {
ll u, v, w; cin >> u >> v >> w; alledge[u][v] = w; alledge[v][u
] = w; node[u].push_back(v);
node[v].push_back(u); edge.push_back({u, v});} dfs(1);
decompose(1, 0, 1); int q;
cin >> q; while (q--) { int type; cin >> type; if (type == 1) {
ll id, x; cin >> id >> x; id--; int p = edge[id].f; int q = edge
[id].s;
if (dep[p] > dep[q]) swap(p, q); st.update(1, 1, n, intime[q
], intime[q], x);
alledge[p][q] = x; alledge[q][p] = x;} else { int u, v; cin >> u
>> v;
int l = lca(u, v); if (u == v) { cout << 0 << endl; continue;}
if (l == u || l == v) { if (dep[u] > dep[v]) swap(u, v); cout
<< sumpath(u, v) << endl;}
else { cout << sumpath(1, u) + sumpath(1, v) << endl;}}} }

```

## 8.7 HLD(update on node)

```

int n, q; vector<int> node[N]; int a[N];
struct SegmentTree {
vector<int> tree; vector<int> lazy; vector<int> aa;
SegmentTree() { tree.resize(4 * N); lazy.resize(4 * N); aa.
resize(4 * N);}
void build(int node, int b, int e) { if (b == e) { tree[node] = 0;
lazy[node] = -1; return;} int mid = (b + e) >> 1; build(2 * node,
b, mid); build(2 * node + 1, mid + 1, e); lazy[node] = -1; tree[
node] = max(tree[2 * node], tree[2 * node + 1]);}
void push(int node, int b, int e) { tree[node] = lazy[node]; if
(b != e) { lazy[2 * node] = lazy[2 * node + 1] = lazy[node]; } lazy[
node] = -1;}

```

```

void update(int node, int b, int e, int l, int r, int x) { if (
lazy[node] != -1) push(node, b, e); if (l > e || r < b) return; if (
l <= b && r >= e) { tree[node] = x; if (b != e) { lazy[2 * node] = lazy
[2 * node + 1] = x; } lazy[node] = -1; return;} int mid = (b + e)
>> 1; update(2 * node, b, mid, l, r, x); update(2 * node + 1, mid
+ 1, e, l, r, x); tree[node] = max(tree[2 * node], tree[2 * node
+ 1]);}
int query(int node, int b, int e, int l, int r) { if (lazy[node]
!= -1) push(node, b, e); if (l > e || r < b) return 0; if (l <= b && r
>= e) { return tree[node]; } int mid = (b + e) >> 1; return max(
query(2 * node, b, mid, l, r), query(2 * node + 1, mid + 1, e, l, r))
;}
};
SegmentTree st; int par[N][LG + 1], dep[N], sz[N];
void dfs(int u, int p = 0) { par[u][0] = p; dep[u] = dep[p] + 1; sz[u] = 1;
for (int i = 1; i <= LG; i++) { par[u][i] = par[par[u][i - 1]][i - 1]; }
for (auto v : node[u]) { if (v == p) continue; dfs(v, u); sz[u]
+= sz[v];}
int lca(int u, int v) { if (dep[u] < dep[v]) swap(u, v); for (int k = LG
; k >= 0; k--) if (dep[par[u][k]] >= dep[par[v][k]]) u = par[u][k]; if (u == v)
return u; for (int k = LG; k >= 0; k--) if (par[u][k] != par[v][k
]) u = par[u][k], v = par[v][k]; return par[u][0];}
int intime[N], head[N]; int timer = 1;
void decompose(int p, int parent, int Head_node) { intime[p] =
timer++; head[p] = Head_node; st.update(1, 1, n, intime[p],
intime[p], a[p]); int heavysize = -1, heavychild = -1; for (auto
i : node[p]) { if (i != parent) { if (sz[i] > heavysize) heavysize =
sz[i], heavychild = i; } if (heavychild == -1) return; decompose
(heavychild, p, Head_node); for (auto i : node[p]) { if (i ==
heavychild || i == parent) continue; decompose(i, p, i);}
int maxnode(int u, int v) { int ans = 0; while (head[u] != head[v]) {
if (dep[head[u]] > dep[head[v]]) swap(u, v); ans = max(ans, st.
query(1, 1, n, intime[head[v]], intime[v])); v = par[head[v]
][0]; if (dep[u] > dep[v]) swap(u, v); ans = max(ans, st.query
(1, 1, n, intime[u], intime[v])); return ans;}
void Solve() { cin >> n >> q; for (int i = 1; i <= n; i++) { cin >> a[i]; } st.
build(1, 1, n); for (int i = 1; i < n; i++) { int u, v; cin >> u >> v;
node[u].push_back(v); node[v].push_back(u); } dfs(1);
decompose(1, 0, 1); while (q--) { int type; cin >> type; if (type
== 1) { int u, x; cin >> u >> x; st.update(1, 1, n, intime[u], intime
[u], x); } else { int u, v; cin >> u >> v; int l = lca(u, v); cout << max
(maxnode(1, u), maxnode(1, v)) << " ";}}}

```

## 8.8 Inverse Graph

```

void bfs(int u) { queue<int> que; que.push(u); wh_cmpnnt[u] =
cmpnnt; while (!que.empty()) { auto u = que.front(); que.
pop(); vector<int> restricted; for (auto v : graph[u]) { if (!
wh_cmpnnt[v]) { restricted.push_back(v); adj[v] = 1; } } for

```

```
(auto v: not_visited){if(v == u || adj[v]) continue ;
que.push(v) ;wh_cmpnnt[v] = cmpnnt ;}for(auto v:
restricted)adj[v] = 0 ;not_visited = restricted ;}}
```

## 8.9 LCA

```
const int N = 3e5 + 9, LG = 18;vector<int> g[N];int par[N][
    LG + 1], dep[N], sz[N];
void dfs(int u, int p = 0) {par[u][0] = p;dep[u] = dep[p] +
    1;sz[u] = 1;
for (int i = 1; i <= LG; i++) par[u][i] = par[par[u][i -
    1]][i - 1];
for (auto v: g[u]) if (v != p) {dfs(v, u);sz[u] += sz[v];}}
int lca(int u, int v) {
if (dep[u] < dep[v]) swap(u, v);
for (int k = LG; k >= 0; k--) if (dep[par[u][k]] >= dep[v])
    u = par[u][k];
if (u == v) return u;
for (int k = LG; k >= 0; k--) if (par[u][k] != par[v][k]) u
    = par[u][k], v = par[v][k];return par[u][0];}
int kth(int u, int k) {assert(k >= 0);
for (int i = 0; i <= LG; i++) if (k & (1 << i)) u = par[u][i
    ];return u;}
int dist(int u, int v) {
int l = lca(u, v);return dep[u] + dep[v] - (dep[l] << 1);}
//kth node from u to v, 0th node is u
int go(int u, int v, int k) {
int l = lca(u, v);int d = dep[u] + dep[v] - (dep[l] << 1);
assert(k <= d);if (dep[l] + k <= dep[u]) return kth(u,
    k);k -= dep[u] - dep[l];return kth(v, dep[v] - dep[l] -
    k);}
int32_t main() {int n; cin >> n;for (int i = 1; i < n; i++)
    {int u, v; cin >> u >> v;g[u].push_back(v);g[v].
    push_back(u);}dfs(1);int q; cin >> q;while (q--) {int u
    , v; cin >> u >> v;cout << dist(u, v) << '\n';}return
    0;}
```

## 8.10 dfs<sub>tree</sub>

```
void dfs(int u, int p){dp[u] = 0 ;for(auto v: graph[u]){if(v
    == p)continue ;if(level[v] == 0){level[v] = level[u] +
    1 ;vp.push_back({u, v}) ;dfs(v, u) ;dp[u] += dp[v] ;}
else if(level[v] < level[u]){vp.push_back({u, v}) ;dp[u]
    ]++ ;}else{dp[u]-- ;}}if(level[u] > 1 && dp[u] == 0)
    bridges++ ;}
```

## 8.11 strongly connected components

```
vector<int>node[10000], transpose_node[10000];
int visit[10000], out_time[10000], in_time[10000];
vector<int>order;//by out_time we can sort by out_time but
    stack/vector reduce complexity
vector<int>SCC;int timer;void dfs(int x) {
visit[x] = 1;in_time[x] = ++timer;for (auto i : node[x])
if (!visit[i]) dfs(i); out_time[x] = ++timer;
order.push_back(x); // all the child of this node (X)
    already visited
}void dfs_for_scc(int x) //This dfs for find scc
{visit[x] = 1;SCC.push_back(x);for (auto child :
    transpose_node[x])
if (!visit[child]) dfs_for_scc(child);}while (m--){
cin >> a >> b;node[a].push_back(b);transpose_node[b].
    push_back(a);
//to find scc we need to run dfs is transpose graph of main
    graph
}for (int i = 0 ; i <= n ; i++)if (visit[i] == 0) dfs(i);
for (int i = 0 ; i <= n ; i++) visit[i] = 0;
cout << "here is node list by order of out time\n";
for (int i = n - 1 ; i >= 0 ; i--)
cout << order[i] << " out time is -> " << out_time[order[i]]
    << endl;
for (int i = n - 1 ; i >= 0 ; i--) {if (visit[order[i]] ==
    0) //order[i] is by largest outtime
{SCC.clear(); //previous SCC cleard
dfs_for_scc(order[i]); // for finding scc we run dfs in
    transpose graph
cout << "Strongly Connected Components are \n";for (auto
    child : SCC)
cout << child << " ";cout << endl;}}
```

## 9 u blank

### 9.1 blank

```
/*
```



---



\*/