Team notebook

1*SQUARE*

December 20, 2018

Contents

1 DS

1.1 ST LAZY flipcoin

```
const int N = 1e5 + 5;

int n, q;
int T[4*N], L[4*N];

void flip(int c, int tl, int tr) {
    int k = tr - tl + 1;
    T[c] = k - T[c];
}

void pull(int c) {
    T[c] = T[2*c] + T[2*c + 1];
}

void push(int c, int tl, int tr) {
    if (tl != tr) {
        int tm = (tl + tr) / 2;
        L[2*c] ^= 1;
        L[2*c + 1] ^= 1;
    }
    L[c] = 0;
}

void update(int c, int tl, int tr, int l, int r) {
    if (L[c]) {
        flip(c, tl, tr);
        push(c, tl, tr);
    }
    if (r < tl || l > tr) return;
    if (l <= tl && tr <= r) {
        flip(c, tl, tr);
        push(c, tl, tr);
        return;
    }
    int tm = (tl + tr) / 2;
    update(2*c, tl, tm, l, r);
    update(2*c + 1, tm + 1, tr, l, r);
    pull(c);
}

int query(int c, int tl, int tr, int l, int r) {
    if (L[c]) {
        flip(c, tl, tr);
        push(c, tl, tr);
    }
    if (r < tl || l > tr) return 0;
    if (l <= tl && tr <= r) return T[c];
    int tm = (tl + tr) / 2;
    return query(2*c, tl, tm, l, r) + query(2*c + 1, tm + 1, tr,
        l, r);
}

int main() {
```

```
    ios_base::sync_with_stdio(false);
    cin.tie(0);
//  freopen("in.txt", "r", stdin);
    cin >> n >> q;
    while (q--) {
        int x, a, b;
        cin >> x >> a >> b;
        if (x == 0)
            update(1, 0, n-1, a, b);
        else
            cout << query(1, 0, n-1, a, b) << '\n';
    }
    return 0;
}
```

1.2 ST bottom up

```
const int N = 1e6 + 5;

int n, q;
int tree[N<<1];

void build() {
    for (int i = n - 1; i >= 1; i--)
        tree[i] = tree[2*i] + tree[2*i + 1];
}

int query(int l, int r) {
    int s = 0;
    for (l += n, r += n; l <= r; l /= 2, r /= 2) {
        if (l%2 == 1) s += tree[l++];
        if (r%2 == 0) s += tree[r--];
    }
    return s;
}

void update(int k) {
    k += n;
    tree[k] = !tree[k];
    for (k /= 2; k > 0 ; k /= 2)
        tree[k] = tree[2*k] + tree[2*k + 1];
}
```

1.3 ST top down

```
const int N = 1e5 + 5;
const int inf = 0x3f3f3f3f;

int n;
int a[N];
int T[4*N];

void build(int c = 1, int tl = 0, int tr = n-1) {
    if (tl == tr) {
        T[c] = a[tl];
        return;
    }

    int tm = (tl + tr) / 2;

    build(2*c, tl, tm);
    build(2*c + 1, tm+1, tr);

    T[c] = T[2*c] + T[2*c + 1];
}

void update(int c = 1, int tl = 0, int tr = n-1, int idx, int d)
    {
```

```
    if (tl == tr) {
        T[c] = d;
        return;
    }

    int tm = (tl + tr) / 2;

    if (idx <= tm) {
        update(2*c, tl, tm, idx, d);
    } else {
        update(2*c + 1, tm+1, tr, idx, d);
    }

    T[c] = T[2*c] + T[2*c + 1];
}

int query(int c = 1, int tl = 0, int tr = n-1, int ql, int qr) {
    if (qr < tl || tr < ql)
        return -inf;
    if (ql <= tl && tr <= qr)
        return T[c];

    int tm = (tl + tr) / 2;

    int lq = query(2*c, tl, tm, ql, qr);
    int rq = query(2*c + 1, tm + 1, tr, ql, qr);

    return lq + rq;
}

int main() {
    build(1, 0, n - 1);
    return 0;
}
```

1.4 bit

```
void build() {
    for (int i = 1; i <= n; i++) {
        j = i + (i & -i);
        if (j <= n)
            bit[j] += bit[i];
    }
}

void add(int k, int x) {
    while (k <= x) {
        bit[k] += x;
        k += k & -k;
    }
}

void sum(int k) {
    int s = 0;
    while (k >= 1) {
        s += tree[k];
        k -= k & -k;
    }
    return s;
}
```

1.5 dsu

```
int par[N], sze[N];

int get(int x) {
    if (par[x] != x)
        par[x] = get(par[x]);
```

```
    return par[x];
}

void init() {
    for (int i = 0; i < N; i++) {
        par[i] = i;
        sze[i] = 1;
    }
}

void unite(int a, int b) {
    a = get(a), b = get(b);
    if (a == b) return;
    if (sze[a] < sze[b]) swap(a, b);
    sze[a] += sze[b], sze[b] = -1;
    par[b] = a;
}
```

1.6 mo

```
const int N;
const int MAXN;

struct query {
        int l, r, ix;
        bool operator<(const query &rhs) {
                if(l/bz != rhs.l/bz) {
                        return l/bz < rhs.l/bz;
                }
                return r < rhs.r;
        }
} Q[N];

void add(int x) {
    //add element at x to the current solution
}

void rem(int x) {
    //remove element at x from the current solution
}

int main() {
    sort(Q, Q + t);
    int l = 1, r = 0;
    for (int i = 0; i < t; i++) {
        while (r < Q[i].r) add(ar[++r]);
        while (r > Q[i].r) rem(ar[r--]);
        while (l < Q[i].l) rem(ar[l++]);
        while (l > Q[i].l) add(ar[--l]);
        ans[Q[i].idx] = cur;
    }
}
```

2 geometry

2.1 circle line intersection

```
double r, a, b, c; // given as input
double x0 = -a*c/(a*a+b*b), y0 = -b*c/(a*a+b*b);
if (c*c > r*r*(a*a+b*b)+EPS)
    puts ("no points");
else if (abs (c*c - r*r*(a*a+b*b)) < EPS) {
    puts ("1 point");
    cout << x0 << ' ' << y0 << '\n';
}
else {
    double d = r*r - c*c/(a*a+b*b);
```

```
    double mult = sqrt (d / (a*a+b*b));
    double ax, ay, bx, by;
    ax = x0 + b * mult;
    bx = x0 - b * mult;
    ay = y0 - a * mult;
    by = y0 + a * mult;
    puts ("2 points");
    cout << ax << ' ' << ay << '\n' << bx << ' ' << by << '\n';
}
```

2.2 lines intersection

```
struct pt { double x, y; };
struct line { double a, b, c; };
const double EPS = 1e-9;
double det(double a,double b,double c,double d) {return a*d -
    b*c;}

bool intersect(line m, line n, pt & res) {
    double zn = det(m.a, m.b, n.a, n.b);
    if (abs(zn) < EPS)
        return false;
    res.x = -det(m.c, m.b, n.c, n.b) / zn;
    res.y = -det(m.a, m.c, n.a, n.c) / zn;
    return true;
}

bool parallel(line m, line n) {
    return abs(det(m.a, m.b, n.a, n.b)) < EPS;
}

bool equivalent(line m, line n) {
    return abs(det(m.a, m.b, n.a, n.b)) < EPS
        && abs(det(m.a, m.c, n.a, n.c)) < EPS
        && abs(det(m.b, m.c, n.b, n.c)) < EPS;
}
```

3 graphs

3.1 bellman ford

```
const int N = 1005;
const int inf = 0x3f3f3f3f;

vector< tuple<int, int, int> > G;
int dis[N], par[N];

void bf(int src) {
    memset(dis, inf, sizeof(dis));
    dis[src] = 0;
    for (int i = 1, f = 1; i < n; i++) {
        if (f == 0) break;
        f = 0;

        for (auto e : G) {
            int fr, to, w;
            tie(fr, to, w) = e;

            if (dis[to] < dis[fr] + w) {
                dis[to] = dis[fr] + w;
                par[to] = par[fr];
                f = 1;
            }
        }
    }
}
```

3.2 bridges

```
const int N = 1005;

vector<int> G[N];
bool vis[N];
int tin[N], fn[N];
int timer;

void dfs(int v, int p = -1) {
    vis[v] = 1;
    tin[v] = fn[v] = timer++;//current v
    for (int to : G[v]) {
        if (to == p) continue;
        if (vis[to]) {//ancestors
            fn[v] = min(fn[v], tin[to]);
        } else {//descendants
            dfs(to, v);
            fn[v] = min(fn[v], fn[to]);
            if (fn[to] > tin[v]) {
                printf("%d %d\n", v, to);
            }
        }
    }
}
```

3.3 cutpoints

```
const int N = 1005;

vector<int> G[N];
int vis[N], fn[N], tin[N];
int timer, ans;

void dfs(int v, int p = -1) {
    vis[v] = 1;
    fn[v] = tin[v] = timer++;
    int des = 0;

    for (int to : G[v]) {
        if (p == to) continue;
        if (vis[to]) {
            fn[v] = min(fn[v], tin[to]);
        } else {
            dfs(to, v);
            fn[v] = min(fn[v], fn[to]);
            if (fn[to] >= tin[v] && p != -1)
                ans++;
            ++des;
        }
    }

    if (p == -1 && des > 1)
        ans++;
}
```

3.4 dijkstra

```
ll dji(int sou, int des) {
    bool vis[n + 1] = {0};
    ll dis[n + 1];
    priority_queue< pair<ll, int> > Q;

    memset(dis, INF, sizeof(dis));
    vis[so] = 1, dis[so] = 0;
    Q.push({0, so});
```

```
    while (!Q.empty()) {
        int f = Q.top().second; Q.pop();
        if (vis[f]) continue;
        vis[f] = true;
        //if (f == des) return dis[f];
        for (auto x : G[f]) {
            int t = x.to, w = x.we;
            if (dis[t] > dis[f] + w) {
                dis[t] = dis[f] + w;
                Q.push({-dis[t], t});
            }
        }
    }
}
```

3.5 euler circuit-path

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

const int N = 1005;

int n, m;
multiset<int> G[N];
vector<int> ans;

void findPath(int u) {
    while (G[u].size() != 0) {
        int nxt = *G[u].begin();
        G[u].erase(G[u].begin());
        G[nxt].erase(G[nxt].find(u));
        findPath(nxt);
    }
    ans.push_back(u);
}

int main() {
    scanf("%d %d", &n, &m);
    for (int i = 0; i < m; i++) {
        int a, b;
        scanf("%d %d", &a, &b);
        G[a].insert(b);
        G[b].insert(a);
    }
    findPath(5);
    for (int i : ans) {
        printf("%d ", i);
    }
    puts("");
    return 0;
}
```

3.6 floyd warshall

```
const int inf = 0x3f3f3f3f;

int G[N][N], dis[N][N];


void chkneg() {
    for (int i = 1; i <= n; i++) {
        for (int j = 1;j <= n; j++) {
            for (int c = 1; c <= n; c++) {
                if (d[i][c] < inf && d[t][t] < 0 && d[c][j] < inf)
                    d[i][j] = -inf;
            }
```

```
        }
    }
}

void fw() {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            if (i == j) dis[i][j] = 0;
            else if (G[i][j]) dis[i][j] = G[i][j];
            else dis[i][j] = inf;
        }
    }
    for (int k = 1; k <= n; k++) {
        for (int i = 1; i <= n; i++) {
            for (int j = 1; j <= n; j++) {
                if (dis[i][k] < inf && dis[k][j] < inf)
                dis[i][j] = min(dis[i][j], dis[i][k] + dis[k][j]);
            }
        }
    }
    chkneg();
}
```

3.7 kahn

```
vector<int> G[N];
int indeg[N];
priority_queue<int> Q;

void kahn() {
    while (!Q.empty()) {
        int f = abs(Q.top()); Q.pop();
        ans.push_back(f);
        for (int cur : G[f]) {
            --indeg[cur];
            if (!indeg[cur])
                Q.push(-cur);
        }
    }
}
```

3.8 prims

```
vector< pair<int, int> > G[N];//vertex, weight
int used[N];

ll prim(int v) {
        priority_queue< pair<int, int> > pq;//weight, vertex
        used[v] = 1;
        for (int i = 0; i < G[v].size(); i++)
                pq.push({-G[v][i].S, G[v][i].F});
        int c = 0;
        ll ans = 0;
        while (!pq.empty()) {
                if (used[pq.top().S]) {
                        pq.pop();
                        continue;
                }
                int w = -pq.top().F, to = pq.top().S;
                used[to] = 1;
                ans += w;
                for (int i = 0; i < G[to].size(); i++) {
                        if (!used[G[to][i].F]) {
                                pq.push({-G[to][i].S, G[to][i].F});
                        }
                }
        }
        return ans;
```

```
}
```

3.9 scc

```
vector<int> ord, comp;

void dfs1(int u) {
    vis[u] = 1;
    for (int v : G[u]) {
        if (!vis[v])
            dfs1(v);
    }
    ord.push_back(u);
}

void dfs2(int u) {
    vis[u] = 1;
    comp.push_back(u);
    for (int v : G[u]) {
        if (!vis[v])
            dfs2(v);
    }
}

void get() {
    for (int i = 0; i < n; i++) {
        if (!vis[i])
            dfs1(i);
    }
    memset(vis, 0, sizeof(vis));
    int c = 1;
    for (int i : ord) {
        if (!vis[i]) {
            dfs2(i);
            printf("\ncomponent #%d : ", c++);
            for (int k : comp) printf("%d ", k);
            comp.clear();
        }
    }
}
```

3.10 toposort

```
int n;
vector<vector<int>> adj;
vector<bool> visited;
vector<int> ans;

void dfs(int v) {
  visited[v] = true;
  for (int u : adj[v]) {
    if (!visited[u]) dfs(u);
  }
  ans.push_back(v);
}

void topological_sort() {
  visited.assign(n, false);
  ans.clear();
  for (int i = 0; i < n; ++i) {
    if (!visited[i]) dfs(i);
  }
  reverse(ans.begin(), ans.end());
}
```

4 maths

4.1 binexp

```
ll binexp(ll x, int n) {
    ll res = 1;
    while (n) {
        if (n&1) res = (res * x);
        x = (x * x);
        n >>= 1;
    }
    return res;
}
```

4.2 extented euclid

```
int egcd(int a, int b, int &x, int &y) {
    if (a == 0) {
        x = 0;
        y = 1;
        return b;
    }
    int x1, x2;
    int d = egcd(b % a, a, x1, x2);
    x = y1 - (b/a) * x1;
    y = x1;
    return d;
}

int modinv(int a, int m) {
    int x, y;
    int g = egcd(a, m, x, y);
    if (g != 1) {
        cerr << "No solution!";
        return -1;
    }
    return (x % m + m) % m;
}
```

4.3 matrix exponentiation

```
/*
int a = 1, b = 2, c = 3;
for (int i = 0; i < n; i++) {
  int a1 = a + b - c;
  int b1 = 2 * a + c;
  int c1 = b + 4 * c;
  a = a1;
  b = 1;
  c = c1;
}
v = { { 1 2 0}
      { 1 0 1}
      {-1 1 4} }
*/
const ll MOD = 1000000007;
typedef vector<vector<ll>> matrix;

matrix mul(matrix A, matrix B) {
  matrix C(A.size(), vector<ll>(B[0].size()));
  for (ll i = 0; i < A.size(); i++)
    for (ll j = 0; j < B[0].size(); j++)
      for (ll w = 0; w < A.size(); w++)
        C[i][j] = (C[i][j] + A[i][w] * B[w][j]) % MOD;
  return C;
}
```

```
matrix pow(matrix A, ll p) {
  if (p == 1)
    return A;
  if (p % 2)
    return mul(A, pow(A, p - 1));
  matrix X = pow(A, p / 2);
  return mul(X, X);
}

int main() {
  ll t;
  cin >> t;
  for (ll cas = 0; cas < t; cas++) {
    ll k;
    cin >> k;
    matrix T(k, vector<ll>(k));
    matrix F1(k, vector<ll>(1));
    for (ll i = 0; i < k; i++)
      cin >> F1[i][0];
    for (ll i = k - 1; i > -1; i--)
      cin >> T[k - 1][i];
    for (ll i = 0; i < k - 1; i++)
      T[i][i + 1] = 1;
    ll n;
    cin >> n;
    if (n <= k) {
      cout << F1[n - 1][0] % MOD << endl;
    } else {
      matrix powered_T = pow(T, n - k);
      matrix Fn = mul(powered_T, F1);
      cout << Fn[k - 1][0] << endl;
    }
  }
  return 0;
}
```

4.4 modular inverse range

```
//compute modular inverses [1...M] in O(M) time
void inverses() {
    inv[1] = 1;
    for (int i = 2; i < M; i++)
        inv[i] = (mod - (mod/i) * inv[mod%i] % mod) % mod;
}
```

4.5 nCk

```
for (int n = 0; n < N; n++) {
    C[n][0] = C[n][n] = 1;
    for (int k = 1; k <= n; k++) {
        C[n][k] = (C[n-1][k] + C[n-1][k-1]) % (mod);
    }
}
```

4.6 sieve

```
bitset<N> np;

void sieve() {
    for (int i = 2; i * i <= N; i++)
        if (!np[i])
            for (int j = i + i; j <= N; j += i)
                np[j] = 1;
}
```

4.7 totient

```
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;
}

const int N = 1e6 + 5;

int phi[N];

int getPHI() {
    phi[1] = 1;
    for (int i = 2; i < N; i++) phi[i] = i - 1;
    for (int i = 2; i < N; i++)
        for (int j = 2*i; j < N; j += i)
            phi[j] -= phi[i];
}
```

5 miscellaneous

5.1 FIO

```
int read() {
    bool neg = false;
    int x = 0; char ch = getchar_unlocked();
    while(ch < '0' || ch > '9') { if(ch == '-') neg = true; ch =
        getchar_unlocked(); }
    while(ch >= '0' && ch <= '9') { x = x*10+ch-48; ch =
        getchar_unlocked(); }
    if(neg) return -x;
    return x;
}

void write(int x) {
    if(x < 0) { x = -x; putchar_unlocked('-'); }
    if(x > 9) write(x/10);
    putchar_unlocked(x%10+48);
}
```

5.2 binary search

```
int lo = , hi = ;
while (lo < hi) {
    int m = (lo + hi) / 2;
    if (p(m)) {
        hi = m;
    } else {
        lo = m + 1;
    }
}

//FFFFFF T TTTTTT
//      *
if (lo == hi && p(lo)) {
    //ans = lo
} else {
    //ans don't exist
}
```

```
//over real numbers
double EPS = 1e-10, lo = -1000.0, hi = 1000.0;

while (hi - lo > EPS) {
    double mid = (lo + hi) / 2.0;
    if (p(mid)) {
        hi = mid;
    } else {
        lo = mid;
    }
}
```

5.3 builtin bit manupulation

```
/*
    __builtin_clz(x): the number of zeros at the beginning of
    the bit represen-
tation
    __builtin_ctz(x): the number of zeros at the end of the bit
    representation
    __builtin_popcount(x): the number of ones in the bit
    representation
    __builtin_parity(x): the parity (even or odd) of the number
    of ones in the
bit representation
*/
```

5.4 gedit scripts

```
#!/bin/sh
g++ -O2 -std=gnu++14 -DLOCAL $GEDIT_CURRENT_DOCUMENT_NAME -o
    ${GEDIT_CURRENT_DOCUMENT_NAME%.*}.out && gnome-terminal -x
    bash -c "./${GEDIT_CURRENT_DOCUMENT_NAME%.*}.out; bash"
```

5.5 generating subsets

```
vector<int> subset;

void search(int k) {
  if (k == n + 1) {
    // process subset
  } else {
    // include k in the subset
    subset.push_back(k);
    search(k + 1);
    subset.pop_back();
    //  dont  include k in the subset
    search(k + 1);
  }
}
```

5.6 pbs

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

typedef tree<int,null_type,less<int>,rb_tree_tag,
tree_order_statistics_node_update> indexed_set;

indexed_set s;
s.insert(2);
s.insert(3);
```

```
s.insert(7);
s.insert(9);

auto x = s.find_by_order(2);
cout << *x << "\n"; // 7

cout << s.order_of_key(7) << "\n"; // 2

cout << s.order_of_key(6) << "\n"; // 2
cout << s.order_of_key(8) << "\n"; // 3
```

5.7 permutations

```
int n;

vector<int> permutation;

bool chosen[n+1];

void search() {
  if (permutation.size() == n) {
    // process permutation
  } else {
    for (int i = 1; i <= n; i++) {
      if (chosen[i]) continue;
      chosen[i] = true;
      permutation.push_back(i);
      search();
      chosen[i] = false;
      permutation.pop_back();
    }
  }
}
```

```
//stl

for (int i = 1; i <= n; i++) {
  permutation.push_back(i);
}
do {
  // process permutation
} while (next_permutation(permutation.begin(),
    permutation.end()));
```

5.8 polymul

```
using cd = complex<double>;

const double PI = acos(-1);

void fft(vector<cd> &a, bool inv) {
        int n = a.size();
        if (n == 1) return;

        vector<cd> a0(n/2), a1(n/2);
        for (int i = 0; 2*i < n; i++) {
                a0[i] = a[2*i];
                a1[i] = a[2*i + 1];
        }

        fft(a0, inv);
        fft(a1, inv);

        double ang = 2 * PI / n * (inv ? -1 : 1);
        cd w(1), wn(cos(ang), sin(ang));
        for (int i = 0; 2*i < n; i++) {
                a[i] = a0[i] + w * a1[i];
                a[i + (n/2)] = a0[i] - w * a1[i];
```

```
                if (inv) {
                        a[i] /= 2;
                        a[i + (n/2)] /= 2;
                }
                w *= wn;
        }
}

void mul(vector<ll> &res, const vector<ll> &a, const vector<ll>
    &b) {
        vector<cd> fa(a.begin(), a.end()), fb(b.begin(), b.end());
        int n = 1;
        while (n < a.size() + b.size()) {
                n <<= 1;
        }
        fa.resize(n);
        fb.resize(n);

        fft(fa, 0);
        fft(fb, 0);

        for (int i = 0; i < n; i++)
                fa[i] *= fb[i];

        fft(fa, 1);

        res.resize(n);
        for (int i = 0; i < n; i++)
                res[i] = round(fa[i].real());
}

int t, n;

int main() {
        scanf("%d", &t);
        while (t--) {
                scanf("%d", &n);

                vector<ll> a(n + 1), b(n + 1), ans;

                for (int i = 0; i <= n; i++)
                        scanf("%lld", &a[i]);
                for (int i = 0; i <= n; i++)
                        scanf("%lld", &b[i]);

                mul(ans, a, b);

                for (int i = 0; i < 2*n + 1; i++)
                        printf("%lld ", ans[i]);
                puts("");
        }
        return 0;
}
```

6 strings

6.1 hashing

```
//find the positions of all occurences of a given string in
    another string
const int MAX = 1e7 + 5;
const int p = 53;
const int m = 1e9 + 7;

vector<ll> pi(MAX);

void precomp() {
  pi[0] = 1;
```

```
  for (int i = 1; i < MAX; i++) {
    pi[i] = (pi[i - 1] * p) % m;
  }
}

int t, nn;

int main() {
  precomp();
  while (cin >> t) {
    string n, h;
    cin >> n >> h;

    if (sz(n) > sz(h)) {
      cout << "\n";
      continue;
    }

    ll tar = 0;
    for (int i = 0; i < sz(n); i++) {
      tar = (tar + pi[i] * (n[i] - 'a' + 1)) % m;
    }

    vector<ll> pref(sz(h));
    pref[0] = 0;
    for (int i = 0; i < sz(h); i++) {
      pref[i + 1] = (pref[i] + pi[i] * (h[i] - 'a' + 1)) % m;
    }

    vector<int> ans;

    for (int i = 0; i + sz(n) - 1 < sz(h); i += 1) {
      ll cur = (pref[i + sz(n)] + m - pref[i]) % m;
      if (cur == tar * pi[i] % m) {
        ans.push_back(i);
      }
    }
    if (sz(ans)) {
      for (int i = 0; i < sz(ans); i++) {
        cout << ans[i] << '\n';
      }
    } else {
      cout << "\n\n";
    }
  }
  return 0;
}
```

6.2 kmp

```
/*
Returns a vector containing the zero based index of
the start of each match of the string K in S.
Matches may overlap
*/
vector<int> KMP(string S, string K) {
  vector<int> T(K.size() + 1, -1);
  vector<int> matches;

  if (K.size() == 0) {
    matches.push_back(0);
    return matches;
  }

  for (int i = 1; i <= K.size(); i++) {
    int pos = T[i - 1];
    while (pos != -1 && K[pos] != K[i - 1])
      pos = T[pos];
    T[i] = pos + 1;
```

```
  }

  int sp = 0;
  int kp = 0;
  while (sp < S.size()) {
    while (kp != -1 && (kp == K.size() || K[kp] != S[sp]))
      kp = T[kp];
    kp++;
    sp++;
    if (kp == K.size())
      matches.push_back(sp - K.size());
  }

  return matches;
}
```

6.3 suffix array

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

const int N = 500050;

void getSA(string &s, vector<int> &p) {
        int n = s.size();
        const int alp = 256;

        p.resize(n);
        vector<int> c(n), cnt(N, 0);

        for (int i = 0; i < n; i++)
                cnt[s[i]]++;
        for (int i = 1; i < alp; i++)
                cnt[i] += cnt[i - 1];
        for (int i = 0; i < n; i++)
                p[--cnt[s[i]]] = i;

        c[p[0]] = 0;
        int clas = 1;

        for (int i = 1; i < n; i++) {
                if (s[p[i]] != s[p[i - 1]])
                        clas++;
                c[p[i]] = clas - 1;
        }

        vector<int> pn(n), cn(n);
        for (int h = 1; h < n; h = h << 1) {
                for (int i = 0; i < n; i++) {
                        pn[i] = p[i] - h;
                        if (pn[i] < 0) pn[i] += n;
                }

                fill(cnt.begin(), cnt.end(), 0);
                for (int i = 0; i < n; i++)
                        cnt[c[pn[i]]]++;
                for (int i = 1; i < clas; i++)
                        cnt[i] += cnt[i - 1];
                for (int i = n - 1; i >= 0; i--)
                        p[--cnt[c[pn[i]]]] = pn[i];

                cn[p[0]] = 0;
                clas = 1;

                for (int i = 1; i < n; i++) {
                        pair<int, int> cur = {c[p[i]], c[(p[i] +
                            h) % n]};
                        pair<int, int> prev = {c[p[i - 1]], c[(p[i
                            - 1] + h) % n]};
```

```
                if (cur != prev) clas++;
                cn[p[i]] = clas - 1;
            }
            c.swap(cn);
        }
}

void getLCP(string &s, vector<int> &sa, vector<int> &lcp) {
        int n = s.size(), k = 0;

    lcp.resize(n, 0);
        vector<int> rank(n, 0);

        for (int i = 0; i < n; i++)
            rank[sa[i]] = i;

        for (int i = 0; i < n; i++, k ? k-- : 0) {
            if (rank[i] == n - 1) {
                k = 0;
                continue;
            }
            int j = sa[rank[i] + 1];
            while (i + k < n && j + k < n && s[i + k] == s[j +
                k])
                k++;
            lcp[rank[i]] = k;
        }
}

int sz;

int clr(int idx) {
    if (idx > sz) return 1;
    return 0;
}

int main() {
    string s1, s2;
    vector<int> sa, lcp;
    cin >> s1 >> s2;
    sz = s1.size();
    s1 += "#" + s2 + "$";
    getSA(s1, sa);

    getLCP(s1, sa, lcp);

    int ans = 0;
    for (int i = 0; i < lcp.size() - 1; i++) {
        if (clr(sa[i]) + clr(sa[i + 1]) == 1) {
            ans = max(ans, lcp[i]);
        }
    }
    cout << ans << '\n';

    return 0;
}
```

6.4 trie

```
const int N = 100005;
const int SIG = 27;

int tr[N*10][SIG], eow[N*10];

int sz = 0;

void insert(string s) {
    int v = 0, cur;
```

```
    for (char c : s) {
        cur = c - 'a' + 1;
        if (!tr[v][cur])
            v = tr[v][cur] = ++sz;
        else
            v = tr[v][cur];
    }
    eow[v] = 1;
}

bool search(string s) {
    int v = 0;
    for (int i = 0; i < s.size(); i++) {
        int c = s[i] - 'a' + 1;
        if (!tr[v][c])
            return false;
        v = tr[v][c];
    }
    if (eow[w] == 1)
        return true;
    return false;
}

//lexicographically first non-empty substring with MAX LCP with s
void search(string s, string &res) {
    int v = 0, cur;
    for (char c : s) {
        cur = c - 'a' + 1;
        if (!tr[v][cur]) break;
        res += c;
        v = tr[v][cur];
    }
    while (!eow[v]) {
        for (int i = 1; i <= 26; i++) {
            if (tr[v][i]) {
                res += (char)(i + 'a' - 1);
                v = tr[v][i];
                break;
            }
        }
    }
}
```

7 trees

7.1 LCA binary Lifting depth

```
//<nlogn, logn>
const int N = 2e5;
const int LOG = 20;

int n;
int up[N + 1][LOG + 1], depth[N + 1];
vector<int> G[N + 1];

void dfs(int v, int p) {
    up[v][0] = p;
    for (int i = 1; i <= LOG; i++) {
        up[v][i] = up[up[v][i-1]][i-1];
    }
    for (int u : G[v]) if (u != p) {
        depth[u] += depth[v] + 1;
        dfs(u, v);
    }
}

int lca(int a, int b) {
    if (depth[a] < depth[b]) swap(a, b);
```

```
    int k = depth[a] - depth[b];

    for (int i = 0; i <= LOG && a != 0; i++) {
        if (k & (1<<i)) {
            a = up[a][i];
        }
    }

    if (a == b) return a;

    for (int i = LOG; i >= 0; i--) {
        if (up[a][i] != up[b][i]) {
            a = up[a][i];
            b = up[b][i];
        }
    }

    return up[a][0];
}

int main() {
    cin >> n;
    for (int i = 1; i < n; i++) {
        int a, b;
        cin >> a >> b;
        G[a].push_back(b);
        G[b].push_back(a);
    }

    dfs(1, 0);

    cout << lca(2, 4) << '\n';

    return 0;
}
```

7.2 LCA

```
const int inf = 0x3f3f3f3f;
const int N = 1e3 + 5;

int n, sz;

vector<int> G[N];
int F[N], H[N];
vector<int> tour, st;

void euler(int u, int p, int d) {
    F[u] = tour.size();
    H[u] = d;
    tour.push_back(u);
    for (int v : G[u]) {
        if (v == p) continue;
        euler(v, u, d + 1);
        tour.push_back(u);
    }
}

void build() {
    sz = tour.size();
    st.resize(3 * sz);
    for (int i = 0; i < sz; i++) st[i + sz] = tour[i];
    for (int i = sz-1; i >= 1; i--) {
        int tl = 2*i, tr = tl + 1;
        st[i] = H[st[tl]] < H[st[tr]] ? st[tl] : st[tr];
    }
}
```

```
int query(int l, int r) {
    l = F[l] + sz, r = F[r] + sz;
    if (r < l) swap(l, r);
    int ans = 0;
    while (l <= r) {
        if (l%2 == 1) {
            ans = H[ans] < H[st[l]] ? ans : st[l];
            l++;
        }
        if (r%2 == 0) {
            ans = H[ans] < H[st[r]] ? ans : st[r];
            r--;
        }
        l /= 2, r /= 2;
    }
    return ans;
}

void lca(int rt) {
    H[0] = inf;
    euler(rt, -1, 1);
    build();
}

int main() {
    ifstream in("emaxx-eng-lca-example.txt");
    in >> n;
    for (int i = 0; i < n-1; i++) {
        int a, b;
        in >> a >> b;
        G[a].push_back(b);
        G[b].push_back(a);
    }
    lca(1);
    for (int i = 0; i < int(tour.size()); i++) {
        cout << tour[i] << " ";
    } cout << endl;
    for (int i = 0; i < int(tour.size()); i++) {
        cout << H[tour[i]] << " ";
    } cout << endl;

    cout << "\n\n\nQUERIES:\n\n\n";
    in.close();
    while (1) {
        int a, b;
        cin >> a >> b;
        cout << query(a, b) << endl;
    }
    return 0;
}
```

7.3 Learning Dishes

```
const int N = 1e6 + 5;
const int LOG = 21;

vector<int> G[N];
int a[N], cnt[N], mx[N];
int up[N][LOG];

void dfs(int v, int p = 0) {
    up[v][0] = p;
    for (int j = 1; j < LOG; j++)
        up[v][j] = up[up[v][j-1]][j-1];
    if (p == 0 || a[v] > mx[p]) {
        cnt[v] = cnt[p] + 1;
    } else {
        cnt[v] = cnt[p];
```

```
    }
    mx[v] = max(a[v], mx[p]);
    for (int u : G[v]) {
        dfs(u, v);
    }
}

int main() {
    int t;
    cin >> t;
    while (t--) {
        int n;
        cin >> n;
        for (int i = 1; i <= n; i++) {
            cin >> a[i];
            G[i].clear();
        }
        cnt[1] = 1;
        for (int i = 2; i <= n; i++) {
            int x;
            cin >> x;
            G[x].push_back(i);
        }
        dfs(1);
        int q;
        cin >> q;
        int ans = 0;
        while (q--) {
            int v, w;
            cin >> v >> w;
            v ^= ans, w ^= ans;
            if (w < mx[v]) {
                int k = v;
                for (int i = LOG-1; i >= 0; i--) {
                    if (mx[up[k][i]] > w)
                        k = up[k][i];
                }
                ans = cnt[v] - cnt[up[k][0]];
            }
            else {
                ans = 0;
            }
            cout << ans << '\n';
        }
        memset(cnt, 0, sizeof(cnt));
    }
    return 0;
}
```