# 2022 ACM-ICPC Teamnote

HeukseokZZANG

November 18, 2022

## Contents

## 1 기본 템플릿

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

using vInt = vector<int>;
using matInt = vector<vInt>;
using pii = pair<int, int>;
using vPii = vector<pii>;
using matPii = vector<vPii>;
using LL = long long;
using vLL = vector<LL>;
using matLL = vector<vLL>;
using pLL = pair<LL, LL>;
using vPLL = vector<pLL>;
using vBool = vector<bool>;
using matBool = vector<vBool>;
using vStr = vector<string>;
```

```
int main(){
        ios::sync_with_stdio(0);
        cin.tie(0);
        freopen("input.txt", "r", stdin);
        freopen("output.txt", "w", stdout);
//          sys.stdin = open("input.txt", "r")
//          sys.stdout = open("output.txt", "w")

}
```

# 2 주요 알고리즘

## 2.1 유니온 파인드

```
int rank[MAX_SIZE];

for (int i=0; i<MAX_SIZE; i++)
        rank[i] = 1;

int find(int x){
  if (x==parent[x]){
    return x;
  }
  else{
    int y = find(parent[x]);
    parent[x] = y;
    return y;
  }
}

void union(int x, int y){
  x = find(x);
  y = find(y);

  if (x == y)
    return;

  if (rank[x] > rank[y]){
    parent[y] = x;
    rank[x] += rank[y];
  }
  else {
    parent[x] = y;
    rank[y] += rank[x];
  }
}
```

## 2.2 다익스트라

```
int v,e,st; //정점의 개수, 간선의 개수, 시작 위치

// {비용, 정점 번호}
vector<pair<int,int>> adj[MAX_SIZE]; //adj[i].push_back({w,x}) 면 i->x 이고 거리는 w
const int INF = 0x3f3f3f3f;
int d[MAX_SIZE]; // 최단 거리 테이블
  fill(d,d+v+1,INF);
  while(e--){
    int u,x,w;
    adj[u].push_back({w,x});
  }

  priority_queue<pair<int,int>, vector<pair<int,int>>, greater<pair<int,int>> > pq;
  d[st] = 0;
  // 우선순위 큐에 (0, 시작점) 추가
  pq.push({d[st],st});
  while(!pq.empty()){
    auto cur = pq.top(); pq.pop(); // {비용, 정점 번호}
    // 거리가 d에 있는 값과 다를 경우 넘어감
    if(d[cur.second] != cur.first) continue;
    for(auto nxt : adj[cur.second]){ //이웃하는 모든 노드들 = nxt에 대하여 반복
      if(d[nxt.second] <= d[cur.second]+nxt.X) continue;
      // cur를 거쳐가는 것이 더 작은 값을 가질 경우
      // d[nxt.Y]을 갱신하고 우선순위 큐에 (거리, nxt.Y)를 추가
      d[nxt.second] = d[cur.second]+nxt.first;
      pq.push({d[nxt.second],nxt.second});
    }

```

## 2.3 DFS

```
bool visited[9];
vector<int> graph[9];

void dfs(int x)
{
        visited[x] = true;
        cout << x << " ";
        for (int i = 0; i < graph[x].size(); i++)
        {
                int y = graph[x][i];
                if (!visited[y])
            dfs(y);
        }
}
```

## 2.4 BFS

```
#define X first
#define Y second
int board[502][502] =
{{1,1,1,0,1,0,0,0,0,0},
 {1,0,0,0,1,0,0,0,0,0},
 {1,1,1,0,1,0,0,0,0,0},
 {1,1,0,0,1,0,0,0,0,0},
 {0,1,0,0,0,0,0,0,0,0},
 {0,0,0,0,0,0,0,0,0,0},
 {0,0,0,0,0,0,0,0,0,0} };
bool vis[502][502];
int n = 7, m = 10;
int dx[4] = {1,0,-1,0};
int dy[4] = {0,1,0,-1};
int main(void){
  ios::sync_with_stdio(0);
  cin.tie(0);
  queue<pair<int,int> > Q;
  vis[0][0] = 1;
  Q.push({0,0});
  while(!Q.empty()){
    pair<int,int> cur = Q.front(); Q.pop();
    cout << '(' << cur.X << ", " << cur.Y << ") -> ";
    for(int dir = 0; dir < 4; dir++){
      int nx = cur.X + dx[dir];
      int ny = cur.Y + dy[dir];
      if(nx < 0 || nx >= n || ny < 0 || ny >= m) continue;
      if(vis[nx][ny] || board[nx][ny] != 1) continue;
      vis[nx][ny] = 1;
      Q.push({nx,ny});
    }
  }
}
```

## 2.5 선분 교차 판정

```
int ccw(pair<int, int>p1, pair<int, int>p2, pair<int, int>p3) {
    int s = p1.first * p2.second + p2.first * p3.second + p3.first * p1.second;
    s -= (p1.second * p2.first + p2.second * p3.first + p3.second * p1.first);

    if (s > 0) return 1;
    else if (s == 0) return 0;
    else return -1;
}

#define pii pair<int, int>
bool isIntercept(pair<pii, pii> l1, pair<pii, pii> l2) {

    pii p1 = l1.first;
    pii p2 = l1.second;
    pii p3 = l2.first;
    pii p4 = l2.second;

    int p1p2 = ccw(p1, p2, p3) * ccw(p1, p2, p4); // l1 기준
    int p3p4 = ccw(p3, p4, p1) * ccw(p3, p4, p2); // l2 기준

    // 두 직선이 일직선 상에 존재
    if (p1p2 == 0 && p3p4 == 0) {
            // 비교를 일반화하기 위한 점 위치 변경
        if (p1 > p2) swap(p2, p1);
        if (p3 > p4) swap(p3, p4);

        return p3 <= p2 && p1 <= p4; // 두 선분이 포개어져 있는지 확인
    }

    return p1p2 <= 0 && p3p4 <= 0;

}
```

## 2.6 소수 리스트 생성

```python
import math
def prime_list(limit):
    if limit < 3:
        return [2] if limit == 2 else []
    size = (limit - 3) // 2
    is_prime = [True] * (size + 1)
    for i in range(math.isqrt(limit - 3) // 2 + 1):
        if is_prime[i]:
            p = i + i + 3
            s = p * (i + 1) + i
            is_prime[s::p] = [False] * ((size - s) // p + 1)
    return [2] + [i + i + 3 for i, v in enumerate(is_prime) if v]
```

## 2.7 소수 판정 알고리즘

```python
# N이 sqrt(N) 이하의 소인수로 나누어떨어지는지 검사
# primes = prime_list(10000000) 으로 소수 리스트 생성 후 실행
# 소수 리스트를 백만(10^7)까지 생성한다면 약 (10^14)까지 판별가능
def isprime(x):
    if x == 1:
        return False
    for i in primes:
        if i > x ** .5:
            break
        if x % i == 0:
            return False
    return True
```

## 2.8 밀러-라빈 소수 판정

```python
def power(x, y, p):
    res = 1

    while y > 0:
        if y % 2 != 0:
            res = (res * x) % p
        y //= 2
        x = (x * x) % p
    return res
def miller_rabin(n, a):
    r = 0
    d = n - 1
    while d % 2 == 0:
        r += 1
        d = d // 2

    x = power(a, d, n)
    if x == 1 or x == n - 1:
        return True

    for i in range(r - 1):
        x = power(x, 2, n)
        if x == n - 1:
            return True
    return False
```

## 2.9 폴라드-로 소인수분해

```python
import random
def is_prime(n):
    alist = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41]
    if n == 1:
        return False
    if n == 2 or n == 3:
        return True
    if n % 2 == 0:
        return False
    for a in alist:
        if n == a:
            return True
        if not miller_rabin(n, a):
            return False
    return True


def pollardRho(n):
    if is_prime(n):
        return n
    if n == 1:
        return 1
    if n % 2 == 0:
        return 2
    x = random.randrange(2, n)
    y = x
    c = random.randrange(1, n)
    d = 1
    while d == 1:
        x = ((x ** 2 % n) + c + n) % n
        y = ((y ** 2 % n) + c + n) % n
        y = ((y ** 2 % n) + c + n) % n
        d = gcd(abs(x - y), n)
        if d == n:
            return pollardRho(n)
    if is_prime(d):
        return d
    else:
        return pollardRho(d)
```

# 3 수학

## 3.1 NTT

```python
from decimal import Decimal, setcontext, Context, MAX_EMAX, MAX_PREC

def multiply(a, b, digit = 0):
    setcontext(Context(prec=MAX_PREC, Emax=MAX_EMAX))
    if digit == 0:
        digit = min(20, len(str(min(len(a), len(b)) * max(a) * max(b))))
    f = f'0{digit}d'
    a_dec = Decimal(''.join(format(x, f) for x in a))
    b_dec = Decimal(''.join(format(x, f) for x in b))
    c_dec = a_dec * b_dec
    total_digit = digit * (len(a) + len(b) - 1)
    c = format(c_dec, f'0{total_digit}f')
    return [int(c[_i:_i + digit]) for _i in range(0, total_digit, digit)]
```

## 3.2 스프라그-그런디

```python
def mex(s):
    if not s:
        return 0
    for i in range(100):
        if i not in s:
            return i

b = list(multiinput())
dp = [0] * 501
for i in range(1, 501):
    s = set()
    for bb in b:
        if i - bb >= 0:
            s.add(dp[i - bb])
    dp[i] = mex(s)

for _ in range(5):
    x, y = multiinput()
    if (dp[x] ^ dp[y]) == 0:
        print('B')
    else:
        print('A')
```

## 3.3 유클리드 호제법

```cpp
int GCD(int a, int b)
{
    if(b==0) return a;
    else return GCD(b,a%b);
}
```

## 3.4 확장 유클리드

```python
# a, b의 gcd가 1일 때만 작동
# ax + by = 1의 해를 리턴
def eea(a, b):
    s0, s1, t0, t1 = 1, 0, 0, 1
    r0, r1 = a, b
    q1 = r0 // r1
    while 1:
        s0, s1, t0, t1 = s1, s0 - s1 * q1, t1, t0 - t1 * q1
        r0, r1 = r1, r0 - r1 * q1
        if r1:
            q1 = r0 // r1
        else:
            return s0, t0
```

## 3.5 가우스 소거법

```
const double EPS = 1e-9;
const int INF = 2; // it doesn't actually have to be infinity or a big number

int gauss (vector < vector<double> > a, vector<double> & ans) {
    int n = (int) a.size();
    int m = (int) a[0].size() - 1;

    vector<int> where (m, -1);
    for (int col=0, row=0; col<m && row<n; ++col) {
        int sel = row;
        for (int i=row; i<n; ++i)
            if (abs (a[i][col]) > abs (a[sel][col]))
                sel = i;
        if (abs (a[sel][col]) < EPS)
            continue;
        for (int i=col; i<=m; ++i)
            swap (a[sel][i], a[row][i]);
        where[col] = row;

        for (int i=0; i<n; ++i)
            if (i != row) {
                double c = a[i][col] / a[row][col];
                for (int j=col; j<=m; ++j)
                    a[i][j] -= a[row][j] * c;
            }
        ++row;
    }

    ans.assign (m, 0);
    for (int i=0; i<m; ++i)
        if (where[i] != -1)
            ans[i] = a[where[i]][m] / a[where[i]][i];
    for (int i=0; i<n; ++i) {
        double sum = 0;
        for (int j=0; j<m; ++j)
            sum += ans[j] * a[i][j];
        if (abs (sum - a[i][m]) > EPS)
            return 0;
    }

    for (int i=0; i<m; ++i)
        if (where[i] == -1)
            return INF;
    return 1;
}
```

```
int gauss (vector < bitset<N> > a, int n, int m, bitset<N> & ans) {
    vector<int> where (m, -1);
    for (int col=0, row=0; col<m && row<n; ++col) {
        for (int i=row; i<n; ++i)
            if (a[i][col]) {
                swap (a[i], a[row]);
                break;
            }
        if (! a[row][col])
            continue;
        where[col] = row;

        for (int i=0; i<n; ++i)
            if (i != row && a[i][col])
                a[i] ^= a[row];
        ++row;
    }
        // The rest of implementation is the same as above
}
```

## 3.6 중국인의 나머지 정리

```
int CRT (int a1 , int m1 , int a2 , int m2) {
        return (a1 - a2 % m1 + m1) * (ll) rev(m2, m1) % m1 * m2 + a2 ;
}

int rev (int x, int m) {
        if (x == 1) return 1;
        return (1 - rev(m % x, x) * (ll) m) / x + m;
}
```

```
// Chinese remainder theorem (special case): find z such that
// z % x = a, z % y = b. Here, z is unique modulo M = lcm(x,y).
// Return (z,M). On failure, M = -1.
PII chinese_remainder_theorem(int x, int a, int y, int b) {
        int s, t;
        int d = extended_euclid(x, y, s, t);
        if (a % d != b % d) return make_pair(0, -1);
        return make_pair(mod(s * b * x + t * a * y, x * y) / d, x * y / d);
}

// Chinese remainder theorem: find z such that
// z % x[i] = a[i] for all i. Note that the solution is
// unique modulo M = lcm_i (x[i]). Return (z,M). On
// failure, M = -1. Note that we do not require the a[i]'s
// to be relatively prime.

PII chinese_remainder_theorem(const VI &x, const VI &a) {
        PII ret = make_pair(a[0], x[0]);
        for (int i = 1; i < x.size(); i++) {
                ret = chinese_remainder_theorem(ret.second, ret.first, x[i], a[i]);
                if (ret.second == -1) break;
        }
        return ret;
}

// computes x and y such that ax + by = c; on failure, x = y =-1
void linear_diophantine(int a, int b, int c, int &x, int &y) {
        int d = gcd(a, b);
        if (c % d) {
                x = y = -1;
        } else {
                x = c / d * mod_inverse(a / d, b / d);
                y = (c - a * x) / b;
        }
}
```

### 3.7 모듈러 곱셈 역원

```python
def moduloinv(p, q):
    mod = 1000000007
    expo = mod - 2
    while (expo):
        if (expo & 1):
            p = (p * q) % mod
        q = (q * q) % mod
        expo >>= 1

    return p
```

### 3.8 좌표 압축

```python
def comp(arr):
    dic = {x: i for i, x in enumerate(sorted(set(arr)))}
    return [dic[x] for x in arr]
```

## 4 그래프

### 4.1 최대 유량

```python
INF = 10**9
# V = 10
# capacity = [[1] * V for _ in range(V)]
# flow = [[0] * V for _ in range(V)]


V = 4
capacity = [[0, 1, 3, 0], [0, 0, 1, 2], [0, 0, 0, 1], [0, 0, 0, 0]]
flow = [[0, 0, 0, 0] for _ in range(4)]


def networkFlow(source, sink):
    totalFlow = 0
    while 1:
        parent = [-1] * V
        q = deque()
        parent[source] = source
        q.append(source)
        while q and parent[sink] == -1:
            here = q.popleft()
            for there in range(0, V):
                if capacity[here][there] - flow[here][there] > 0 and parent[there] == -1:
                    q.append(there)
                    parent[there] = here
        if parent[sink] == -1:
            break
        amount = INF
        p = sink
        while p != source:
            amount = min(capacity[parent[p]][p] - flow[parent[p]][p], amount)
            p = parent[p]
        p = sink
        while p != source:
            flow[parent[p]][p] += amount
            flow[p][parent[p]] -= amount
            p = parent[p]
        totalFlow += amount
    return totalFlow
```

## 4.2 이분 매칭

```python
# N명의 직원이 M개의 일을 나누어서 할 때,
# i번째 직원이 할 수 있는 일이 정해져 있음
# 할 수 있는 최대 일의 개수 구하기
from collections import deque
adj = []
n, m = map(int, input().split())
for i in range(n):
    s = list(map(int, input().split()))[1:]
    ss = [0] * m
    for j in s:
        ss[j - 1] = 1
    adj.append(ss)

aMatch = [-1] * n
bMatch = [-1] * m

def dfs(a, visited):
    if visited[a]:
        return 0
    visited[a] = 1
    for b in range(0, m):
        if adj[a][b]:
            if bMatch[b] == -1 or dfs(bMatch[b], visited):
                aMatch[a] = b
                bMatch[b] = a
                return 1
    return 0
def bipartiteMatch():
    size = 0
    for start in range(0, n):
        visited = [0] * n
        if dfs(start, visited):
            size += 1
    return size
```

# 5 트리

## 5.1 세그먼트 트리

```cpp
#include <iostream>
#include <cmath>
#include <vector>
using namespace std;
void init(vector<long long> &a, vector<long long> &tree, int node, int start, int end) {
    if (start == end) {
        tree[node] = a[start];
    } else {
        init(a, tree, node*2, start, (start+end)/2);
        init(a, tree, node*2+1, (start+end)/2+1, end);
        tree[node] = tree[node*2] + tree[node*2+1];
    }
}
void update(vector<long long> &a, vector<long long> &tree, int node, int start, int end, int index, long long val) {
    if (index < start || index > end) {
        return;
    }
    if (start == end) {
        a[index] = val;
        tree[node] = val;
        return;
    }
    update(a, tree,node*2, start, (start+end)/2, index, val);
    update(a, tree,node*2+1, (start+end)/2+1, end, index, val);
    tree[node] = tree[node*2] + tree[node*2+1];
}
long long query(vector<long long> &tree, int node, int start, int end, int left, int right) {
    if (left > end || right < start) {
        return 0;
    }
    if (left <= start && end <= right) {
        return tree[node];
    }
    long long lsum = query(tree, node*2, start, (start+end)/2, left, right);
    long long rsum = query(tree, node*2+1, (start+end)/2+1, end, left, right);
    return lsum + rsum;
}
int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    int n, m, k;
    cin >> n >> m >> k;
    vector<long long> a(n);
    int h = (int)ceil(log2(n));
    int tree_size = (1 << (h+1));
    vector<long long> tree(tree_size);
    m += k;
    for (int i=0; i<n; i++) {
        cin >> a[i];
    }
    init(a, tree, 1, 0, n-1);
    while (m--) {
        int what;
        cin >> what;
        if (what == 1) {
            int index;
            long long val;
            cin >> index >> val;
            update(a, tree, 1, 0, n-1, index-1, val);
        } else if (what == 2) {
            int left, right;
            cin >> left >> right;
            cout << query(tree, 1, 0, n-1, left-1, right-1) << '\n';
        }
    }
    return 0;
}
```

## 5.2 레이지 세그먼트 트리

```cpp
#include <iostream>
#include <cmath>
#include <vector>
using namespace std;
void init(vector<long long> &a, vector<long long> &tree, int node, int start, int end) {
    if (start == end) {
        tree[node] = a[start];
    } else {
        init(a, tree, node*2, start, (start+end)/2);
        init(a, tree, node*2+1, (start+end)/2+1, end);
        tree[node] = tree[node*2] + tree[node*2+1];
    }
}
void update_lazy(vector<long long> &tree, vector<long long> &lazy, int node, int start, int end) {
    if (lazy[node] != 0) {
        tree[node] += (end-start+1)*lazy[node];
        if (start != end) {
            lazy[node*2] += lazy[node];
            lazy[node*2+1] += lazy[node];
        }
        lazy[node] = 0;
    }
}
void update_range(vector<long long> &tree, vector<long long> &lazy, int node, int start, int end, int left, int right, long long diff) {
    update_lazy(tree, lazy, node, start, end);
    if (left > end || right < start) {
        return;
    }
    if (left <= start && end <= right) {
        tree[node] += (end-start+1)*diff;
        if (start != end) {
            lazy[node*2] += diff;
            lazy[node*2+1] += diff;
        }
        return;
    }
    update_range(tree, lazy, node*2, start, (start+end)/2, left, right, diff);
    update_range(tree, lazy, node*2+1, (start+end)/2+1, end, left, right, diff);
    tree[node] = tree[node*2] + tree[node*2+1];
}
long long query(vector<long long> &tree, vector<long long> &lazy, int node, int start, int end, int left, int right) {
    update_lazy(tree, lazy, node, start, end);
    if (left > end || right < start) {
        return 0;
    }
    if (left <= start && end <= right) {
        return tree[node];
    }
    long long lsum = query(tree, lazy, node*2, start, (start+end)/2, left, right);
    long long rsum = query(tree, lazy, node*2+1, (start+end)/2+1, end, left, right);
    return lsum + rsum;
}
int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(nullptr);
    int n, m, k;
    cin >> n >> m >> k;
    vector<long long> a(n);
    int h = (int)ceil(log2(n));
    int tree_size = (1 << (h+1));
    vector<long long> tree(tree_size);
    vector<long long> lazy(tree_size);
    m += k;
    for (int i=0; i<n; i++) {
        cin >> a[i];
    }
    init(a, tree, 1, 0, n-1);
    while (m--) {
        int what;
        cin >> what;
        if (what == 1) {
            int left, right;
            long long diff;
            cin >> left >> right >> diff;
            update_range(tree, lazy, 1, 0, n-1, left-1, right-1, diff);
        } else if (what == 2) {
            int left, right;
            cin >> left >> right;
            cout << query(tree, lazy, 1, 0, n-1, left-1, right-1) << '\n';
        }
    }
    return 0;
}
```

## 5.3 펜윅 트리

```python
mod = 998244353
class FenwickTree:
    def __init__(self, size):
        self.data = [0] * (size + 1)
        self.size = size

    # i is exclusive
    def prefix_sum(self, i):
        s = 0
        while i > 0:
            s = (s + self.data[i]) % mod
            i -= i & -i
        return s

    def add(self, i, x):
        i += 1
        while i <= self.size:
            self.data[i] = (self.data[i] + x) % mod
            i += i & -i
```

## 5.4 2차원 펜윅 트리

```
class Fenwick2D:
    def __init__(self, w, h):
        self.data = [[0] * h for _ in range(w)]
        self.w = w
        self.h = h
    def prefix_sum(self, r, c):
        cnt = 0
        while r > 0:
            cc = c
            while cc > 0:
                cnt += self.data[r][cc]
                cc -= cc & -cc
            r -= r & -r
        return cnt
    def add(self, r, c, diff):
        while r <= self.w:
            cc = c
            while cc <= self.h:
                self.data[r][cc] += diff
                cc += cc & -cc
            r += r & -r
```

## 5.5 레이지 펜윅 트리

```
void update(int bitType, int idx, int diff) {
    int* bit = bitType==1 ? bit1 : bit2;
    while (idx <= n) {
        bit[idx] += diff;
        idx += idx&-idx;
    }
}

void rangeUpdate(int a, int b, int diff) {
    update(1, a, diff);
    update(1, b+1, -diff);
    update(2, a, diff * (a-1));
    update(2, b+1, -diff * b);
}

int getBitValue(int bitType, int idx) {
    int* bit = bitType==1 ? bit1 : bit2;
    int answer = 0;
    while (idx > 0) {
        answer += bit[idx];
        idx -= idx&-idx;
    }
    return answer;
}

int prefixSum(int idx) {
    return getBitValue(1, idx) * idx - getBitValue(2, idx);
}

int query(int a, int b) {
    return prefixSum(b) - prefixSum(a-1);
}
```

```
import sys

# sys.setrecursionlimit(10**6)
# import decimal

# import math
# from collections import deque
# import itertools
# from collections import Counter
# from queue import PriorityQueue
# import heapq
# import decimal
# import random
# from bisect import bisect_left, bisect_right
# import fractions

# import re
# import datetime

input = sys.stdin.readline

```

```python

def multiinput():
    return map(int, input().split())

class LazyFenwick:
    def __init__(self, size):
        self.size = size
        self.bit = [[0] * (size + 1) for _ in range(2)]

    def update(self, bitType, idx, diff):
        while idx <= self.size:
            self.bit[bitType][idx] += diff
            idx += idx & -idx

    def rangeUpdate(self, a, b, diff):
        self.update(0, a, diff)
        self.update(0, b + 1, -diff)
        self.update(1, a, diff * (a - 1))
        self.update(1, b + 1, -diff * b)

    def getBitValue(self, bitType, idx):
        ans = 0
        while idx > 0:
            ans += self.bit[bitType][idx]
            idx -= idx & -idx
        return ans

    def prefixSum(self, idx):
        return self.getBitValue(0, idx) * idx - self.getBitValue(1, idx)

    def query(self, a, b):
        return self.prefixSum(b) - self.prefixSum(a - 1)



# decimal.getcontext().prec = 1111

def main(tc):
    n, m, k = multiinput()
    s = LazyFenwick(n)
    for _ in range(1, n + 1):
        i = int(input())
        s.rangeUpdate(_, _, i)
    for _ in range(m + k):
        a, *q = multiinput()
        if a == 1:
            b, c, d = q
            s.rangeUpdate(b, c, d)
        else:
            b, c = q
            print(s.query(b, c))




# for tc in range(int(input())):
for tc in range(1):
    main(tc)
```

# 6 테크닉

## 6.1 비트마스킹

```python
a = 1234
p = 2
# - p번 비트 켜기
a |= (1 << p)
# - p번 비트 확인하기
a & (1 << p)
# - p번 비트 끄기
a &= ~(1 << p)
# - 최하위 비트 구하기
a & -a
# - 최하위 비트 끄기
a &= (a - 1)
# - p번 비트 토글
a ^= (1 << p)
```

## 6.2 이분탐색

```python
def bisect_left(a, x, lo=0, hi=None, *, key=None):
    """Return the index where to insert item x in list a, assuming a is sorted.
    The return value i is such that all e in a[:i] have e < x, and all e in
    a[i:] have e >= x.  So if x already appears in the list, a.insert(i, x) will
    insert just before the leftmost x already there.
    Optional args lo (default 0) and hi (default len(a)) bound the
    slice of a to be searched.
    """

    if lo < 0:
        raise ValueError('lo must be non-negative')
    if hi is None:
        hi = len(a)
    # Note, the comparison uses "<" to match the
    # __lt__() logic in list.sort() and in heapq.
    if key is None:
        while lo < hi:
            mid = (lo + hi) // 2
            if a[mid] < x:
                lo = mid + 1
            else:
                hi = mid
    else:
        while lo < hi:
            mid = (lo + hi) // 2
            if key(a[mid]) < x:
                lo = mid + 1
            else:
                hi = mid
    return lo
def bisect_right(a, x, lo=0, hi=None, *, key=None):
    """Return the index where to insert item x in list a, assuming a is sorted.
    The return value i is such that all e in a[:i] have e <= x, and all e in
    a[i:] have e > x.  So if x already appears in the list, a.insert(i, x) will
    insert just after the rightmost x already there.
    Optional args lo (default 0) and hi (default len(a)) bound the
    slice of a to be searched.
    """

    if lo < 0:
        raise ValueError('lo must be non-negative')
    if hi is None:
        hi = len(a)
    # Note, the comparison uses "<" to match the
    # __lt__() logic in list.sort() and in heapq.
    if key is None:
        while lo < hi:
            mid = (lo + hi) // 2
            if x < a[mid]:
                hi = mid
            else:
                lo = mid + 1
    else:
        while lo < hi:
            mid = (lo + hi) // 2
            if x < key(a[mid]):
                hi = mid
            else:
                lo = mid + 1
    return
```

# 7 Ext

## 7.1 LCA

```cpp
int n, lef[MAX], rig[MAX], dist[MAX], table[2 * MAX][18];
vi graph[MAX], stk;

void dfs(int u, int p, int d)
{
        dist[u] = d;
        lef[u] = rig[u] = stk.size();
        stk.pb(u);
        for (auto v : graph[u])
        {
                if (v == p) continue;
                dfs(v, u, d + 1);
                rig[u] = stk.size();
                stk.pb(u);
        }
}

int lca(int u, int v)
{
        int l = min(lef[u], lef[v]);
        int r = max(rig[u], rig[v]);
        int g = __builtin_clz(r - l + 1) ^ 31;
        return dist[table[l][g]] < dist[table[r - (1 << g) + 1][g]] ?
               table[l][g] : table[r - (1 << g) + 1][g];
}

void build()
{
        dfs(1, -1, 0);

        for (int i = 0; i < stk.size(); i++) table[i][0] = stk[i];
        for (int j = 1; (1 << j) <= stk.size(); j++)
        {
                for (int i = 0; i + (1 << j) <= stk.size(); i++)
                {
                        table[i][j] = (dist[table[i][j - 1]] <
↪  dist[table[i + (1 << (j - 1))][j - 1]] ?
                                       table[i][j - 1] : table[i + (1 <<
↪  (j - 1))][j - 1]);
                }
        }
}
```

```
vi graph[100];
int P[100], L[100], table[100][20];

void dfs(int from, int to, int depth)
{
        P[to]=from;
        L[to]=depth;
        FOR(i,0,(int)graph[to].size())
        {
                int v=graph[to][i];
                if(v==from)
                        continue;
                dfs(to,v,depth+1);
        }
}

int query(int n, int p, int q)
{
        if(L[p]<L[q]) swap(p,q);

        int x=1;

        while(true)
        {
                if((1<<(x+1))>L[p])
                        break;
                x++;
        }

        FORr(i,x,0)
        {
                if(L[p]-(1<<i) >= L[q])
                        p=table[p][i];
        }

        if(p==q) return p;

        FORr(i,x,0)
        {
                if(table[p][i]!=-1 && table[p][i]!=table[q][i])
                {
                        p=table[p][i];
                        q=table[q][i];
                }
        }

        return P[p];
}


void build(int n)
{
        ms(table,-1);

        FOR(i,0,n)
                table[i][0]=P[i];
        for(int j=1; 1<<j < n; j++)
        {
                for(int i=0; i<n; i++)
                {
                        if(table[i][j-1]!=-1)
                                table[i][j]=table[table[i][j-1]][j-1];
                }
        }
}
```

## 7.2 FFT

```
#include <bits/stdc++.h>
using namespace std;
typedef long long lint;
typedef pair<int, int> pi;

static char _buffer[1 << 19];
static int _currentChar = 0;
static int _charsNumber = 0;

static inline int _read() {
        if (_charsNumber < 0) {
                exit(1);
        }
        if (!_charsNumber || _currentChar == _charsNumber) {
                _charsNumber = (int)fread(_buffer, sizeof(_buffer[0]), sizeof(_buffer), stdin);
                _currentChar = 0;
        }
        if (_charsNumber <= 0) {
                return -1;
        }
        return _buffer[_currentChar++];
}

static inline int _readInt() {
        int c, x, s;
        c = _read();
        while (c <= 32) c = _read();
        x = 0;
        s = 1;
        if (c == '-') {
                s = -1;
                c = _read();
        }
        while (c > 32) {
                x *= 10;
                x += c - '0';
                c = _read();
        }
        if (s < 0) x = -x;
        return x;
}

namespace fft{
        typedef complex<double> base;
        void fft(vector<base> &v, bool inv){
                int n = v.size();
                vector<base> w(n/2), aux(n);
                for(int i=0; i<n/2; i++){
                        int k = i&-i;
                        if(i == k){
                                double ang = 2 * M_PI * i / n;
                                if(inv) ang *= -1;
                                w[i] = base(cos(ang), sin(ang));
                        }
                        else w[i] = w[i-k] * w[k];
                }
                for(int i=n/2; i; i>>=1){
                        aux = v;
                        for(int k=0; 2*k<n; k+=i){
                                for(int j=0; j<i; j++){
```

```
                    base a = aux[2*k + j], b =
↪ aux[2*k + j + i] * w[k];
                    v[k + j] = a + b;
                    v[k + j + n/2] = a - b;
                }
            }
        }
        if(inv){
            for(int i=0; i<n; i++){
                v[i] /= n;
            }
        }
    }
    vector<lint> multiply(vector<lint> &v, vector<lint> &w){
        vector<base> fv(v.begin(), v.end()), fw(w.begin(),
↪ w.end());
        int n = 1;
        while(n < max(v.size(), w.size())) n <<= 1;
        n <<= 1;
        fv.resize(n);
        fw.resize(n);
        fft(fv, 0);
        fft(fw, 0);
        for(int i=0; i<n; i++) fv[i] *= fw[i];
        fft(fv, 1);
        vector<lint> ret(n);
        for(int i=0; i<n; i++) ret[i] = round(fv[i].real());
        return ret;
    }
    vector<lint> multiply(vector<lint> &v, vector<lint> &w, int b){
    int n = 2; while(n < v.size() + w.size()) n <<= 1;
    vector<base> v1(n), v2(n), r1(n), r2(n);
    for(int i=0; i<v.size(); i++){
        v1[i] = base(v[i] >> 15, v[i] & 32767);
    }
    for(int i=0; i<w.size(); i++){
        v2[i] = base(w[i] >> 15, w[i] & 32767);
    }
    fft(v1, 0);
    fft(v2, 0);
    for(int i=0; i<n; i++){
        int j = (i ? (n - i) : i);
        base ans1 = (v1[i] + conj(v1[j])) * base(0.5, 0);
        base ans2 = (v1[i] - conj(v1[j])) * base(0, -0.5);
        base ans3 = (v2[i] + conj(v2[j])) * base(0.5, 0);
        base ans4 = (v2[i] - conj(v2[j])) * base(0, -0.5);
        r1[i] = (ans1 * ans3) + (ans1 * ans4) * base(0, 1);
        r2[i] = (ans2 * ans3) + (ans2 * ans4) * base(0, 1);
    }
    fft(r1, 1);
    fft(r2, 1);
    vector<lint> ret(n);
    for(int i=0; i<n; i++){
        lint av = (lint)round(r1[i].real());
        lint bv = (lint)round(r1[i].imag()) +
↪ (lint)round(r2[i].real());
        lint cv = (lint)round(r2[i].imag());
        ret[i] = (av << 30) + (bv << 15) + cv;
    }
    return ret;
    }
}

int n, m;
vector<lint> v, w;

int main(){
    n = _readInt();
    m = _readInt();
    for(int i=0; i<=n; i++){
        v.push_back(_readInt());
    }
    for(int i=0; i<=m; i++){
        w.push_back(_readInt());
    }
    auto poly = fft::multiply(v, w, 32768);
    lint ret = 0;
    for(int i=0; i<=n+m; i++){
        ret ^= poly[i];
    }
    cout << ret;
}
```

## 7.3 HLD

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

struct Seg{
    int tree[1 << 18];
    int sz = 1 << 17;

    void update(int x, int v){
        x |= sz; tree[x] += v;
        while(x >>= 1){
            tree[x] = tree[x << 1] + tree[x << 1 | 1];
        }
    }

    int query(int l, int r){
        l |= sz, r |= sz;
        int ret = 0;
        while(l <= r){
            if(l & 1) ret += tree[l++];
            if(~r & 1) ret += tree[r--];
            l >>= 1, r >>= 1;
        }
        return ret;
    }
}seg;

int sz[101010], dep[101010], par[101010], top[101010], in[101010], out[101010];
vector<int> g[101010];
vector<int> inp[101010]; //입력 / 양방향 그래프

int chk[101010];
void dfs(int v = 1){
        chk[v] = 1;
        for(auto i : inp[v]){
                if(chk[i]) continue;
                chk[i] = 1;
                g[v].push_back(i);
                dfs(i);
        }
}

void dfs1(int v = 1){
        sz[v] = 1;
        for(auto &i : g[v]){
                dep[i] = dep[v] + 1; par[i] = v;
                dfs1(i); sz[v] += sz[i];
                if(sz[i] > sz[g[v][0]]) swap(i, g[v][0]);
        }
}

int pv;
void dfs2(int v = 1){
        in[v] = ++pv;
        for(auto i : g[v]){
                top[i] = i == g[v][0] ? top[v] : i;
                dfs2(i);
        }
        out[v] = pv;
}

void update(int v, int w){
    seg.update(in[v], w);
}

int query(int a, int b){
    int ret = 0;
    while(top[a] ^ top[b]){
        if(dep[top[a]] < dep[top[b]]) swap(a, b);
        int st = top[a];
        ret += seg.query(in[st], in[a]);
        a = par[st];
    }
    if(dep[a] > dep[b]) swap(a, b);
    ret += seg.query(in[a], in[b]);
    return ret;
}

int main(){
    ios_base::sync_with_stdio(0); cin.tie(0);
    int n, q; cin >> n >> q; //정점 개수, 쿼리 개수
    for(int i=1; i<n; i++){
        int s, e; cin >> s >> e;
        inp[s].push_back(e);
        inp[e].push_back(s);
    }
    dfs(); dfs1(); dfs2();
    while(q--){
        //1 v w : update v w
        //2 s e : query s e
        int op, a, b; cin >> op >> a >> b;
        if(op == 1) update(a, b);
        else cout << query(a, b) << "\n";
    }
}
```

## 7.4 KMP

```
const int MAX = 1000;

char text[MAX], patt[MAX];
int pi[MAX], n, m;

void Process()
{
    int now=-1;
    pi[0]=-1;

    for(int i=1; i<m; i++)
    {
        while(now!=-1 && patt[now+1]!=patt[i])
            now=pi[now];
        if(patt[now+1]==patt[i]) pi[i]=++now;
        else pi[i]=now=-1;
    }
}

void Search()
{
    int now=-1;

    for(int i=0; i<n; i++)
    {
        while(now!=-1 && patt[now+1]!=text[i])
            now=pi[now];
        if(patt[now+1]==text[i]) ++now;
        else now=-1;
        if(now==m-1)
        {
            cout<<"match at "<<i-now<<endl;
            now=pi[now]; // match again
        }
    }
}

int main()
{
    // ios_base::sync_with_stdio(0);
    // cin.tie(NULL); cout.tie(NULL);
    // freopen("in.txt","r",stdin);

    cin>>text>>patt;

    n=strlen(text); m=strlen(patt);

    Process();
    Search();
//    FOR(i, 0, m) cout << pi[i] << " "; cout << endl;
    return 0;
}
```

## 7.5 kth power

```
LL mod;
LL S[105][105];
// Find 1^k+2^k+...+n^k % mod
void solve() {
        LL n, k;
        scanf("%lld %lld %lld", &n, &k, &mod);
        S[0][0] = 1 % mod;
        for (int i = 1; i <= k; i++) {
                for (int j = 1; j <= i; j++) {
                        if (i == j) S[i][j] = 1 % mod;
                        else S[i][j] = (j * S[i - 1][j] + S[i - 1][j - 1]) % mod;
                }
        }

        LL ans = 0;
        for (int i = 0; i <= k; i++) {
                LL fact = 1, z = i + 1;
                for (LL j = n - i + 1; j <= n + 1; j++) {
                        LL mul = j;
                        if (mul % z == 0) {
                                mul /= z;
                                z /= z;
                        }
                        fact = (fact * mul) % mod;
                }
                ans = (ans + S[k][i] * fact) % mod;
        }
        printf("%lld\n", ans);
}
```

## 7.6 convex

```
struct PT
{
        int x, y;
        PT(){}
        PT(int x, int y) : x(x), y(y) {}
        bool operator < (const PT &P) const
        {
                return x<P.x || (x==P.x && y<P.y);
        }
};



ll cross(const PT p, const PT q, const PT r)
{
        return (ll)(q.x-p.x)*(ll)(r.y-p.y)-(ll)(q.y-p.y)*(ll)(r.x-p.x);
}

vector<PT> Points, Hull;

void findConvexHull()
{
        int n=Points.size(), k=0;

        SORT(Points);

        // Build lower hull

        FOR(i,0,n)
        {
                while(Hull.size()>=2 && cross(Hull[Hull.size()-2],Hull.back(),Points[i])<=0)
                {
                        Hull.pop_back();
                        k--;
                }
                Hull.pb(Points[i]);
                k++;
        }

        // Build upper hull

        for(int i=n-2, t=k+1; i>=0; i--)
        {
                while(Hull.size()>=t && cross(Hull[Hull.size()-2],Hull.back(),Points[i])<=0)
                {
                        Hull.pop_back();
                        k--;
                }
                Hull.pb(Points[i]);
                k++;
        }

        Hull.resize(k);
}
```

## 7.7 LIS

```
vector<int> d;
int ans, n;

int main() {
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        int x;
        scanf("%d", &x);
        vector<int>::iterator it = lower_bound(d.begin(), d.end(), x);
        if (it == d.end()) d.push_back(x);
        else *it = x;
    }
    printf("LIS = %d", d.size());
    return 0;
}
```

## 7.8 LCS

```
string a, b;
int dp[100][100];
string l;
void printLcs(int i, int j)
{
        if (a[i] == '\0' || b[j] == '\0')
        {
                cout << l << endl;
                return;
        }
        if (a[i] == b[j])
        {
                l += a[i];
                printLcs(i + 1, j + 1);
        }
        else
        {
                if (dp[i + 1][j] > dp[i][j + 1])
                        printLcs(i + 1, j);
                else
                        printLcs(i, j + 1);
        }
}
void printAll(int i, int j)
{
        if (a[i] == '\0' || b[j] == '\0')
        {
                prnt(l);
                return;
        }
        if (a[i] == b[j])
        {
                l += a[i];
                printAll(i + 1, j + 1);
                l.erase(l.end() - 1);
        }
        else
        {
                if (dp[i + 1][j] > dp[i][j + 1])
                        printAll(i + 1, j);
                else if (dp[i + 1][j] < dp[i][j + 1])
                        printAll(i, j + 1);
                else
                {
                        printAll(i + 1, j);
                        printAll(i, j + 1);
                }
        }
}
int lcslen (int i, int j)
{
        if (a[i] == '\0' || b[j] == '\0')
                return 0;
        if (dp[i][j] != -1)
                return dp[i][j];
        int ans = 0;
        if (a[i] == b[j])
        {
                ans = 1 + lcslen(i + 1, j + 1);
        }
        else
        {
                int x = lcslen(i, j + 1);
                int y = lcslen(i + 1, j);
                ans = max(x, y);
        }
        return dp[i][j] = ans;
}

int main()
```

```
{
        cin >> a >> b;
        ms(dp, -1);
        cout << lcslen(0, 0) << endl;
        printLcs(0, 0);
        l.clear();
        printAll(0, 0);
        return 0;
}
```

## 7.9 Matrix Exponential

```
struct Matrix
{
        ll mat[MAX][MAX];

        Matrix(){}

        // This initialization is important.
        // Input matrix should be initialized separately

        void init(int sz)
        {
                ms(mat,0);
                for(int i=0; i<sz; i++) mat[i][i]=1;
        }
} aux;

void matMult(Matrix &m, Matrix &m1, Matrix &m2, int sz)
{
        ms(m.mat,0);

        // This only works for square matrix

        FOR(i,0,sz)
        {
                FOR(j,0,sz)
                {
                        FOR(k,0,sz)
                        {
                                m.mat[i][k]=(m.mat[i][k]+m1.mat
                                        [i][j]*m2.mat[j][k])%mod;
                        }
                }
        }
}

Matrix expo(Matrix &M, int n, int sz)
{
        Matrix ret;
        ret.init(sz);

        if(n==0) return ret;
        if(n==1) return M;

        Matrix P=M;

        while(n!=0)
        {
                if(n&1)
                {
                        aux=ret;
                        matMult(ret,aux,P,sz);
                }

                n>>=1;

                aux=P; matMult(P,aux,aux,sz);
        }

        return ret;
}
```