2022 ACM-ICPC Teamnote

HeukseokZZANG

October 5, 2022

Contents

1 기본 템플릿

```
#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>;
```

```cpp
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);

}
```

2 주요 알고리즘

2.1 유니온 파인드

```cpp
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

```cpp
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

```
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

```
# 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

```
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

```
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

```
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

```
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

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

3.4 확장 유클리드

```
# 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 페르마 소정리

3.6 중국인의 나머지 정리

3.7 모듈러 곱셈 역원

```
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

```
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

```
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

```
# 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
```

```python
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
    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 펜윅 트리

```
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
```

5.3 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
```

6 테크닉

6.1 비트마스킹

6.2 이분탐색

```
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
```