

# 2022 ACM-ICPC Teamnote

HeukseokZZANG

October 5, 2022

## Contents

### 1 기본 템플릿

### 2 주요 알고리즘

2.1	유니온 파인드	7
2.2	다익스트라	8
2.3	DFS	8
2.4	BFS	8
2.5	선분 교차 판정	8
2.6	소수 리스트 생성	8
2.7	소수 판정 알고리즘	8
2.8	밀러-라빈 소수 판정	8
2.9	폴라드-로 소인수분해	8

### 3 수학

3.1	NTT	7
3.2	스프라그-그런디	7
3.3	유클리드 호제법	8
3.4	확장 유클리드	8
3.5	페르마 소정리	8
3.6	중국인의 나머지 정리	8
3.7	모듈러 곱셈 역원	8
3.8	좌표 압축	8

### 4 그래프

4.1	최대 유량	7
4.2	이분 매칭	8

### 5 트리

5.1	세그먼트 트리	7
5.2	펜윅 트리	8
5.3	2차원 펜윅 트리	8

### 6 테크닉

6.1	비트마스킹	8
6.2	이분탐색	8

## 1 기본 템플릿

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4
5  using vInt = vector<int>;
6  using matInt = vector<vInt>;
7  using pii = pair<int, int>;
8  using vPii = vector<pii>;
9  using matPii = vector<vPii>;
10 using LL = long long;
11 using vLL = vector<LL>;
12 using matLL = vector<vLL>;
13 using pLL = pair<LL, LL>;
14 using vPLL = vector<pLL>;
15 using vBool = vector<bool>;
16 using matBool = vector<vBool>;
17 using vStr = vector<string>;
18
19 int main(){
20     ios::sync_with_stdio(0);
21     cin.tie(0);
22
23 }
```

## 2 주요 알고리즘

### 2.1 유니온 파인드

```
1 int rank[MAX_SIZE];
2
```

```

3  for (int i=0; i<MAX_SIZE; i++)
4      rank[i] = 1;
5
6  int find(int x){
7      if (x==parent[x]){
8          return x;
9      }
10     else{
11         int y = find(parent[x]);
12         parent[x] = y;
13         return y;
14     }
15 }
16
17 void union(int x, int y){
18     x = find(x);
19     y = find(y);
20
21     if (x == y)
22         return;
23
24     if (rank[x] > rank[y]){
25         parent[y] = x;
26         rank[x] += rank[y];
27     }
28     else {
29         parent[x] = y;
30         rank[y] += rank[x];
31     }
32 }

```

## 2.2 다익스트라

```

1  int v,e,st; //정점의 개수, 간선의 개수, 시작 위치
2
3  // {비용, 정점 번호}
4  vector<pair<int,int>> adj[MAX_SIZE]; //adj[i].push_back({w,x}) 면 i->x
   ↳ 이고 거리는 w
5  const int INF = 0x3f3f3f3f;
6  int d[MAX_SIZE]; // 최단 거리 테이블
7  fill(d,d+v+1,INF);
8  while(e--){

```

```

9      int u,x,w;
10     adj[u].push_back({w,x});
11 }
12
13 priority_queue<pair<int,int>, vector<pair<int,int>>,
↳ greater<pair<int,int>> > pq;
14 d[st] = 0;
15 // 우선순위 큐에 (0, 시작점) 추가
16 pq.push({d[st],st});
17 while(!pq.empty()){
18     auto cur = pq.top(); pq.pop(); // {비용, 정점 번호}
19     // 거리가 d에 있는 값과 다를 경우 넘어감
20     if(d[cur.second] != cur.first) continue;
21     for(auto nxt : adj[cur.second]){ //이웃하는 모든 노드들 = nxt에 대하여
↳ 반복
22         if(d[nxt.second] <= d[cur.second]+nxt.X) continue;
23         // cur를 거쳐가는 것이 더 작은 값을 가질 경우
24         // d[nxt.Y]을 갱신하고 우선순위 큐에 (거리, nxt.Y)를 추가
25         d[nxt.second] = d[cur.second]+nxt.first;
26         pq.push({d[nxt.second],nxt.second});
27     }
28 }

```

## 2.3 DFS

```

1  bool visited[9];
2  vector<int> graph[9];
3
4  void dfs(int x)
5  {
6      visited[x] = true;
7      cout << x << " ";
8      for (int i = 0; i < graph[x].size(); i++)
9      {
10         int y = graph[x][i];
11         if (!visited[y])
12             dfs(y);
13     }
14 }

```

## 2.4 BFS

```
1  #define X first
2  #define Y second
3  int board[502][502] =
4  {{1,1,1,0,1,0,0,0,0,0},
5   {1,0,0,0,1,0,0,0,0,0},
6   {1,1,1,0,1,0,0,0,0,0},
7   {1,1,0,0,1,0,0,0,0,0},
8   {0,1,0,0,0,0,0,0,0,0},
9   {0,0,0,0,0,0,0,0,0,0},
10  {0,0,0,0,0,0,0,0,0,0}};
11 bool vis[502][502];
12 int n = 7, m = 10;
13 int dx[4] = {1,0,-1,0};
14 int dy[4] = {0,1,0,-1};
15 int main(void){
16     ios::sync_with_stdio(0);
17     cin.tie(0);
18     queue<pair<int,int> > Q;
19     vis[0][0] = 1;
20     Q.push({0,0});
21     while(!Q.empty()){
22         pair<int,int> cur = Q.front(); Q.pop();
23         cout << '(' << cur.X << ", " << cur.Y << ") -> ";
24         for(int dir = 0; dir < 4; dir++){
25             int nx = cur.X + dx[dir];
26             int ny = cur.Y + dy[dir];
27             if(nx < 0 || nx >= n || ny < 0 || ny >= m) continue;
28             if(vis[nx][ny] || board[nx][ny] != 1) continue;
29             vis[nx][ny] = 1;
30             Q.push({nx,ny});
31         }
32     }
33 }
```

## 2.5 선분 교차 판정

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

```
4
5      if (s > 0) return 1;
6      else if (s == 0) return 0;
7      else return -1;
8  }
9
10 #define pii pair<int,int>
11 bool isIntercept(pair<pii,pii> l1, pair<pii,pii> l2) {
12
13     pii p1 = l1.first;
14     pii p2 = l1.second;
15     pii p3 = l2.first;
16     pii p4 = l2.second;
17
18     int p1p2 = ccw(p1, p2, p3) * ccw(p1, p2, p4); // l1 기준
19     int p3p4 = ccw(p3, p4, p1) * ccw(p3, p4, p2); // l2 기준
20
21     // 두 직선이 일직선 상에 존재
22     if (p1p2 == 0 && p3p4 == 0) {
23         // 비교를 일반화하기 위한 점 위치 변경
24         if (p1 > p2) swap(p2, p1);
25         if (p3 > p4) swap(p3, p4);
26
27         return p3 <= p2 && p1 <= p4; // 두 선분이 포개어져 있는지 확인
28     }
29
30     return p1p2 <= 0 && p3p4 <= 0;
31 }
32 }
```

## 2.6 소수 리스트 생성

```
1  import math
2  def prime_list(limit):
3      if limit < 3:
4          return [2] if limit == 2 else []
5      size = (limit - 3) // 2
6      is_prime = [True] * (size + 1)
7      for i in range(math.isqrt(limit - 3) // 2 + 1):
8          if is_prime[i]:
9              p = i + i + 3
```

```

10         s = p * (i + 1) + i
11         is_prime[s:p] = [False] * ((size - s) // p + 1)
12     return [2] + [i + i + 3 for i, v in enumerate(is_prime) if v]

```

## 2.7 소수 판정 알고리즘

```

1  # NO| sqrt(N) 이하의 소인수로 나누어떨어지는지 검사
2  # primes = prime_list(10000000) 으로 소수 리스트 생성 후 실행
3  # 소수 리스트를 백만(10^7)까지 생성한다면 약 (10^14)까지 판별가능
4  def isprime(x):
5      if x == 1:
6          return False
7      for i in primes:
8          if i > x **.5:
9              break
10             if x % i == 0:
11                 return False
12     return True

```

## 2.8 밀러-라빈 소수 판정

```

1  def power(x, y, p):
2      res = 1
3
4      while y > 0:
5          if y % 2 != 0:
6              res = (res * x) % p
7              y //= 2
8              x = (x * x) % p
9      return res
10  def miller_rabin(n, a):
11      r = 0
12      d = n - 1
13      while d % 2 == 0:
14          r += 1
15          d = d // 2
16
17      x = power(a, d, n)
18      if x == 1 or x == n - 1:
19          return True
20
21      for i in range(r - 1):

```

```

22         x = power(x, 2, n)
23         if x == n - 1:
24             return True
25     return False

```

## 2.9 폴라드-로 소인수분해

```

1  import random
2  def is_prime(n):
3      alist = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41]
4      if n == 1:
5          return False
6      if n == 2 or n == 3:
7          return True
8      if n % 2 == 0:
9          return False
10     for a in alist:
11         if n == a:
12             return True
13         if not miller_rabin(n, a):
14             return False
15     return True
16
17  def pollardRho(n):
18      if is_prime(n):
19          return n
20      if n == 1:
21          return 1
22      if n % 2 == 0:
23          return 2
24      x = random.randrange(2, n)
25      y = x
26      c = random.randrange(1, n)
27      d = 1
28      while d == 1:
29          x = ((x ** 2 % n) + c + n) % n
30          y = ((y ** 2 % n) + c + n) % n
31          y = ((y ** 2 % n) + c + n) % n
32          d = gcd(abs(x - y), n)
33          if d == n:
34              return pollardRho(n)
35

```

```

36     if is_prime(d):
37         return d
38     else:
39         return pollardRho(d)

```

## 3 수학

### 3.1 NTT

```

1  from decimal import Decimal, setcontext, Context, MAX_EMAX, MAX_PREC
2
3  def multiply(a, b, digit = 0):
4      setcontext(Context(prec=MAX_PREC, Emax=MAX_EMAX))
5      if digit == 0:
6          digit = min(20, len(str(min(len(a), len(b)) * max(a) * max(b))))
7      f = f'0{digit}d'
8      a_dec = Decimal(''.join(format(x, f) for x in a))
9      b_dec = Decimal(''.join(format(x, f) for x in b))
10     c_dec = a_dec * b_dec
11     total_digit = digit * (len(a) + len(b) - 1)
12     c = format(c_dec, f'0{total_digit}f')
13     return [int(c[_i:_i + digit]) for _i in range(0, total_digit, digit)]

```

### 3.2 스프라그-그런디

```

1  def mex(s):
2      if not s:
3          return 0
4      for i in range(100):
5          if i not in s:
6              return i
7
8  b = list(multiinput())
9  dp = [0] * 501
10 for i in range(1, 501):
11     s = set()
12     for bb in b:
13         if i - bb >= 0:
14             s.add(dp[i - bb])
15     dp[i] = mex(s)
16
17 for _ in range(5):

```

```

18     x, y = multiinput()
19     if (dp[x] ^ dp[y]) == 0:
20         print('B')
21     else:
22         print('A')

```

### 3.3 유클리드 호제법

```

1  int GCD(int a, int b)
2  {
3      if(b==0) return a;
4      else return GCD(b,a%b);
5  }

```

### 3.4 확장 유클리드

```

1  # a, b의 gcd가 1일 때만 작동
2  # ax + by = 1의 해를 리턴
3  def eea(a, b):
4      s0, s1, t0, t1 = 1, 0, 0, 1
5      r0, r1 = a, b
6      q1 = r0 // r1
7      while 1:
8          s0, s1, t0, t1 = s1, s0 - s1 * q1, t1, t0 - t1 * q1
9          r0, r1 = r1, r0 - r1 * q1
10         if r1:
11             q1 = r0 // r1
12         else:
13             return s0, t0

```

### 3.5 페르마 소정리

### 3.6 중국인의 나머지 정리

### 3.7 모듈러 곱셈 역원

```

1  def moduluiinv(p, q):
2      mod = 1000000007
3      expo = mod - 2
4      while (expo):
5          if (expo & 1):
6              p = (p * q) % mod

```

```

7         q = (q * q) % mod
8         expo >>= 1
9
10    return p

```

### 3.8 좌표 압축

```

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

```

## 4 그래프

### 4.1 최대 유량

```

1  INF = 10**9
2  # V = 10
3  # capacity = [[1] * V for _ in range(V)]
4  # flow = [[0] * V for _ in range(V)]
5
6
7  V = 4
8  capacity = [[0, 1, 3, 0], [0, 0, 1, 2], [0, 0, 0, 1], [0, 0, 0, 0]]
9  flow = [[0, 0, 0, 0] for _ in range(4)]
10
11
12 def networkFlow(source, sink):
13     totalFlow = 0
14     while 1:
15         parent = [-1] * V
16         q = deque()
17         parent[source] = source
18         q.append(source)
19         while q and parent[sink] == -1:
20             here = q.popleft()
21             for there in range(0, V):
22                 if capacity[here][there] - flow[here][there] > 0 and
23                     ↪ parent[there] == -1:
24                     q.append(there)
25                     parent[there] = here
26             if parent[sink] == -1:
27                 break

```

```

27     amount = INF
28     p = sink
29     while p != source:
30         amount = min(capacity[parent[p]][p] - flow[parent[p]][p],
31                     ↪ amount)
32         p = parent[p]
33     p = sink
34     while p != source:
35         flow[parent[p]][p] += amount
36         flow[p][parent[p]] -= amount
37         p = parent[p]
38     totalFlow += amount
39     return totalFlow

```

### 4.2 이분 매칭

```

1  # N명의 직원이 M개의 일을 나누어서 할 때,
2  # i번째 직원이 할 수 있는 일이 정해져 있음
3  # 할 수 있는 최대 일의 개수 구하기
4  from collections import deque
5  adj = []
6  n, m = map(int, input().split())
7  for i in range(n):
8      s = list(map(int, input().split()))[1:]
9      ss = [0] * m
10     for j in s:
11         ss[j - 1] = 1
12     adj.append(ss)
13
14  aMatch = [-1] * n
15  bMatch = [-1] * m
16
17  def dfs(a, visited):
18      if visited[a]:
19          return 0
20      visited[a] = 1
21      for b in range(0, m):
22          if adj[a][b]:
23              if bMatch[b] == -1 or dfs(bMatch[b], visited):
24                  aMatch[a] = b
25                  bMatch[b] = a

```

```

26         return 1
27     return 0
28 def bipartiteMatch():
29     size = 0
30     for start in range(0, n):
31         visited = [0] * n
32         if dfs(start, visited):
33             size += 1
34     return size

```

## 5 트리

### 5.1 세그먼트 트리

```

1  #include <iostream>
2  #include <cmath>
3  #include <vector>
4  using namespace std;
5  void init(vector<long long> &a, vector<long long> &tree, int node, int
↳ start, int end) {
6      if (start == end) {
7          tree[node] = a[start];
8      } else {
9          init(a, tree, node*2, start, (start+end)/2);
10         init(a, tree, node*2+1, (start+end)/2+1, end);
11         tree[node] = tree[node*2] + tree[node*2+1];
12     }
13 }
14 void update(vector<long long> &a, vector<long long> &tree, int node, int
↳ start, int end, int index, long long val) {
15     if (index < start || index > end) {
16         return;
17     }
18     if (start == end) {
19         a[index] = val;
20         tree[node] = val;
21         return;
22     }
23     update(a, tree, node*2, start, (start+end)/2, index, val);
24     update(a, tree, node*2+1, (start+end)/2+1, end, index, val);
25     tree[node] = tree[node*2] + tree[node*2+1];
26 }

```

```

27 long long query(vector<long long> &tree, int node, int start, int end,
↳ int left, int right) {
28     if (left > end || right < start) {
29         return 0;
30     }
31     if (left <= start && end <= right) {
32         return tree[node];
33     }
34     long long lsum = query(tree, node*2, start, (start+end)/2, left,
↳ right);
35     long long rsum = query(tree, node*2+1, (start+end)/2+1, end, left,
↳ right);
36     return lsum + rsum;
37 }
38 int main() {
39     ios_base::sync_with_stdio(false);
40     cin.tie(nullptr);
41     int n, m, k;
42     cin >> n >> m >> k;
43     vector<long long> a(n);
44     int h = (int)ceil(log2(n));
45     int tree_size = (1 << (h+1));
46     vector<long long> tree(tree_size);
47     m += k;
48     for (int i=0; i<n; i++) {
49         cin >> a[i];
50     }
51     init(a, tree, 1, 0, n-1);
52     while (m-->0) {
53         int what;
54         cin >> what;
55         if (what == 1) {
56             int index;
57             long long val;
58             cin >> index >> val;
59             update(a, tree, 1, 0, n-1, index-1, val);
60         } else if (what == 2) {
61             int left, right;
62             cin >> left >> right;
63             cout << query(tree, 1, 0, n-1, left-1, right-1) << '\n';
64         }

```

```

65     }
66     return 0;
67 }

```

## 5.2 펜윅 트리

```

1 mod = 998244353
2 class FenwickTree:
3     def __init__(self, size):
4         self.data = [0] * (size + 1)
5         self.size = size
6
7     # i is exclusive
8     def prefix_sum(self, i):
9         s = 0
10        while i > 0:
11            s = (s + self.data[i]) % mod
12            i -= i & -i
13        return s
14
15    def add(self, i, x):
16        i += 1
17        while i <= self.size:
18            self.data[i] = (self.data[i] + x) % mod
19            i += i & -i

```

## 5.3 2차원 펜윅 트리

```

1 class Fenwick2D:
2     def __init__(self, w, h):
3         self.data = [[0] * h for _ in range(w)]
4         self.w = w
5         self.h = h
6     def prefix_sum(self, r, c):
7         cnt = 0
8         while r > 0:
9             cc = c
10            while cc > 0:
11                cnt += self.data[r][cc]
12                cc -= cc & -cc
13            r -= r & -r
14        return cnt

```

```

15 def add(self, r, c, diff):
16     while r <= self.w:
17         cc = c
18         while cc <= self.h:
19             self.data[r][cc] += diff
20             cc += cc & -cc
21         r += r & -r

```

## 6 테크닉

### 6.1 비트마스킹

### 6.2 이분탐색

```

1 def bisect_left(a, x, lo=0, hi=None, *, key=None):
2     """Return the index where to insert item x in list a, assuming a is
3     ↪ sorted.
4     The return value i is such that all e in a[:i] have e < x, and all e
5     ↪ in
6     a[i:] have e >= x. So if x already appears in the list, a.insert(i,
7     ↪ x) will
8     insert just before the leftmost x already there.
9     Optional args lo (default 0) and hi (default len(a)) bound the
10    slice of a to be searched.
11    """
12
13    if lo < 0:
14        raise ValueError('lo must be non-negative')
15    if hi is None:
16        hi = len(a)
17    # Note, the comparison uses "<" to match the
18    # __lt__() logic in list.sort() and in heapq.
19    if key is None:
20        while lo < hi:
21            mid = (lo + hi) // 2
22            if a[mid] < x:
23                lo = mid + 1
24            else:
25                hi = mid
26    else:
27        while lo < hi:
28            mid = (lo + hi) // 2

```



<pre> 26         if key(a[mid]) &lt; x: 27             lo = mid + 1 28         else: 29             hi = mid 30     return lo 31 def bisect_right(a, x, lo=0, hi=None, *, key=None): 32     """Return the index where to insert item x in list a, assuming a is 33     ↪ sorted. 34     ↪ The return value i is such that all e in a[:i] have e &lt;= x, and all e 35     ↪ in 36     ↪ a[i:] have e &gt; x. So if x already appears in the list, a.insert(i, 37     ↪ x) will 38     ↪ insert just after the rightmost x already there. 39     ↪ Optional args lo (default 0) and hi (default len(a)) bound the 40     ↪ slice of a to be searched. 41     """ 42     if lo &lt; 0: 43         raise ValueError('lo must be non-negative') </pre>	<pre> 42     if hi is None: 43         hi = len(a) 44     # Note, the comparison uses "&lt;" to match the 45     # __lt__() logic in list.sort() and in heapq. 46     if key is None: 47         while lo &lt; hi: 48             mid = (lo + hi) // 2 49             if x &lt; a[mid]: 50                 hi = mid 51             else: 52                 lo = mid + 1 53     else: 54         while lo &lt; hi: 55             mid = (lo + hi) // 2 56             if x &lt; key(a[mid]): 57                 hi = mid 58             else: 59                 lo = mid + 1 60     return </pre>
--	--