

자료구조

스택, 큐, 덱, 리스트, set, map

목차

차례

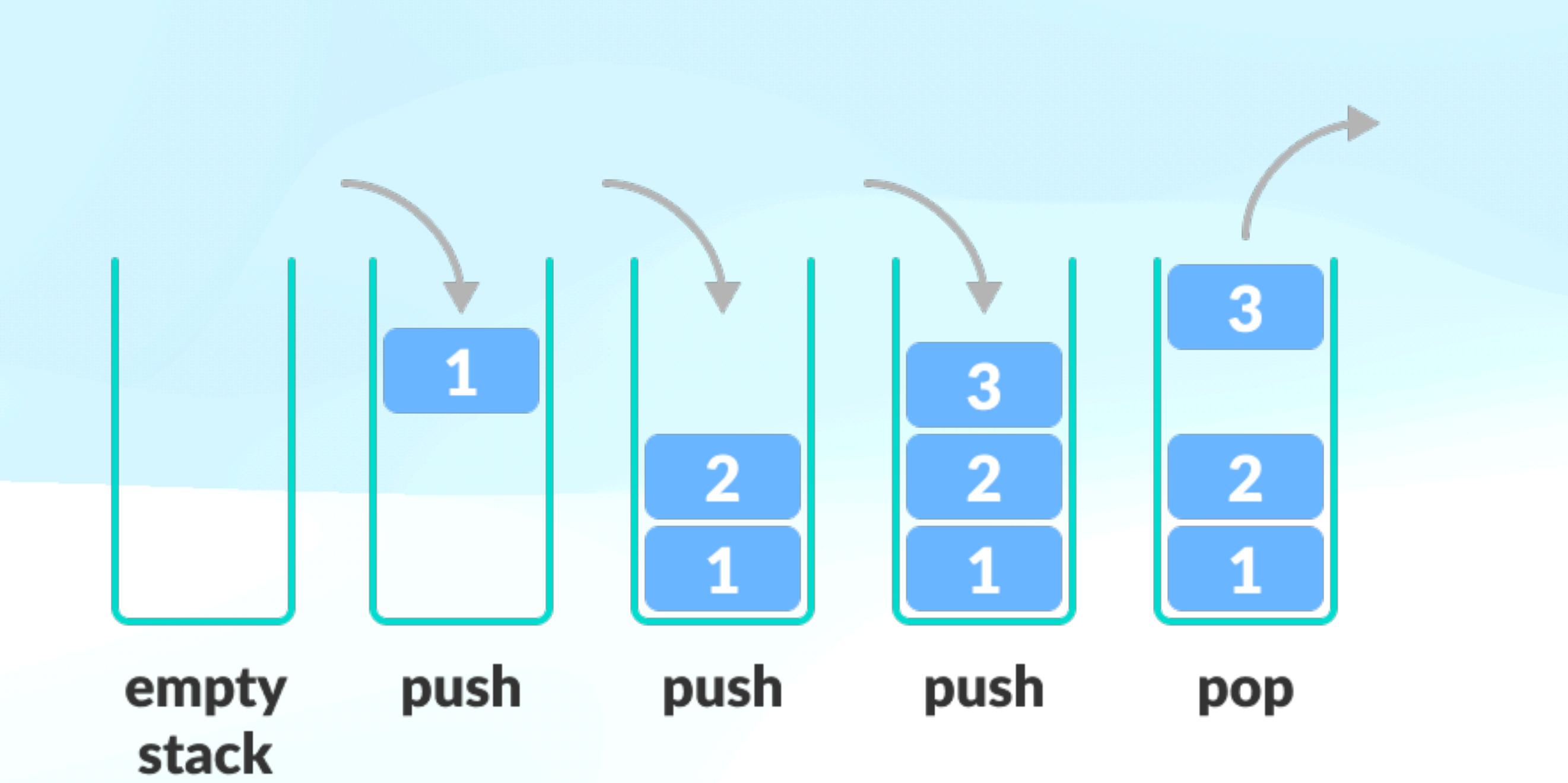
- 0. 2회차 review
- 1. 스택(Stack)
- 2. 큐(queue)
- 3. 덱(deque)
- 4. 리스트(list)
- 5. set, multiset, map
- 6. Priority queue

2회차 리뷰

- 시간 복잡도.. $O(x)$ 의 식을 만드는 것보다 1억이 넘는가 안넘는가를 세는 것이 중요
 - 몇 번 연산하는지를 잘 셀 수 있으면 빅-O표현식은 그냥 알 수 있을 것입니다.
- 브루트포스는 모든 경우를 다 세는 것
 - 27172 수 나누기 게임 : 모든 배수를 다 세는데 꽤 빠르다
- 백트래킹
 - 한번 잘 익혀두면 나올 때마다 비슷한 형식을 조금씩 고쳐서 문제를 풀 수 있음
 - 재귀함수가 어떻게 돌아가는지 공부하기 좋음

스택

생김새



- Monotone stack (단조로움), LIFO(Last In First Out)
- 어떤 상황이 될 때까지 스택을 쌓다가 방출!! 이런 느낌

스택

코드

```
int main(){
    fast_io;
    stack<int> stk;
    stk.push(1);
    stk.pop();
    stk.top();
    stk.empty();
    stk.size();
    stack<pii> stk2;
    // stk2.push(1, 2);
    stk2.emplace(1, 2);
    cout << stk2.top().xx << ' ' << stk2.top().yy;
}
```

스택

문제 풀이

오큰수

성공



4 골드 IV

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	512 MB	84364	30191	21231	34.104%

문제

크기가 N 인 수열 $A = A_1, A_2, \dots, A_N$ 이 있다. 수열의 각 원소 A_i 에 대해서 오큰수 $NGE(i)$ 를 구하려고 한다. A_i 의 오큰수는 오른쪽에 있으면서 A_i 보다 큰 수 중에서 가장 왼쪽에 있는 수를 의미한다. 그러한 수가 없는 경우에 오큰수는 -1이다.

예를 들어, $A = [3, 5, 2, 7]$ 인 경우 $NGE(1) = 5, NGE(2) = 7, NGE(3) = 7, NGE(4) = -1$ 이다. $A = [9, 5, 4, 8]$ 인 경우에는 $NGE(1) = -1, NGE(2) = 8, NGE(3) = 8, NGE(4) = -1$ 이다.

입력

첫째 줄에 수열 A 의 크기 N ($1 \leq N \leq 1,000,000$)이 주어진다. 둘째 줄에 수열 A 의 원소 A_1, A_2, \dots, A_N ($1 \leq A_i \leq 1,000,000$)이 주어진다.

스택

문제 풀이

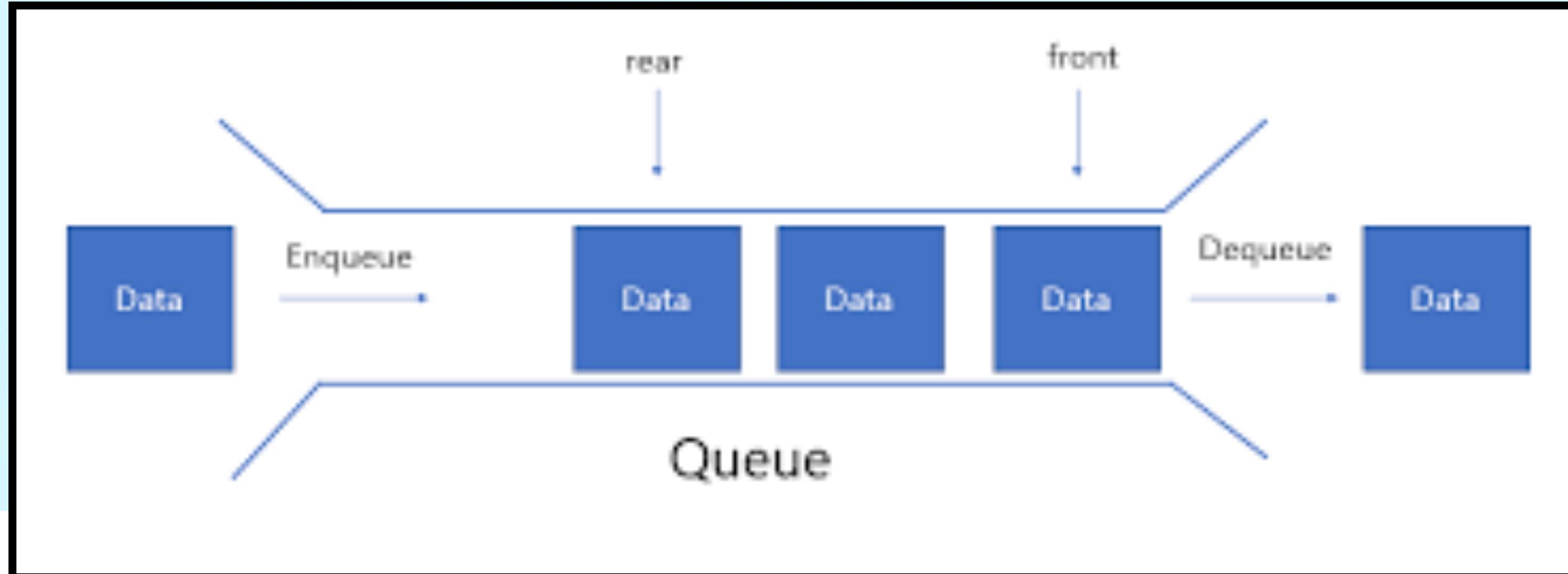
- 클래스로 데이터를 만들어서 넣을 수 있습니다.
- 스택안에는 내림차순으로 저장되어있을 것입니다.
 - Monotone
- 스택이 비어있는데 pop하지 않도록 주의해야함
- 배열을 다 돌고 스택에 남아있는 것을 처리해야함
- 시간복잡도는 $O(n)$: 스택에 n개의 데이터가 들어감

```
class Data{
public:
    int idx, num, NGE = -1;
};

int n;
vector<int> res(1000001);
stack<Data> s;

int main(){
    fast_io;
    cin >> n;
    for (int i = 0; i < n; i++){
        int x; cin >> x;
        while (!s.empty() && s.top().num < x){
            s.top().NGE = x;
            res[s.top().idx] = s.top().NGE;
            s.pop();
        }
        s.push({i, x, -1});
    }
    while (!s.empty()){
        res[s.top().idx] = s.top().NGE;
        s.pop();
    }
    for (int i = 0; i < n; i++)
        cout << res[i] << ' ';
}
```

큐 생김새



큐도 monotone일때

터널에 차를 막아놨다가 어떤 조건 이후에

어디 까지 개방하는 식



1차선 터널과 비슷

큐 코드

- q.back()이 존재하긴하는데 쓰는건 별로..
 - 직관적이지 못함
 - 원형의 정보를 다룰때 효과적임



```
int main(){  
    fast_io  
    queue<int> q;  
    q.push(1);  
    q.pop();  
    q.front();  
    q.size();  
    q.empty();  
    queue<pii> q2;  
    q2.emplace(1, 2);  
}
```

큐 문제 풀이

요세푸스 문제

성공



4 실버 IV

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	256 MB	110369	55472	39052	49.058%

문제

요세푸스 문제는 다음과 같다.

1번부터 N 번까지 N 명의 사람이 원을 이루면서 앉아있고, 양의 정수 K ($\leq N$)가 주어진다. 이제 순서대로 K 번째 사람을 제거한다. 한 사람이 제거되면 남은 사람들로 이루어진 원을 따라 이 과정을 계속해 나간다. 이 과정은 N 명의 사람이 모두 제거될 때까지 계속된다. 원에서 사람들이 제거되는 순서를 (N, K) -요세푸스 순열이라고 한다. 예를 들어 $(7, 3)$ -요세푸스 순열은 $<3, 6, 2, 7, 5, 1, 4>$ 이다.

N 과 K 가 주어지면 (N, K) -요세푸스 순열을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 N 과 K 가 빈 칸을 사이에 두고 순서대로 주어진다. ($1 \leq K \leq N \leq 5,000$)

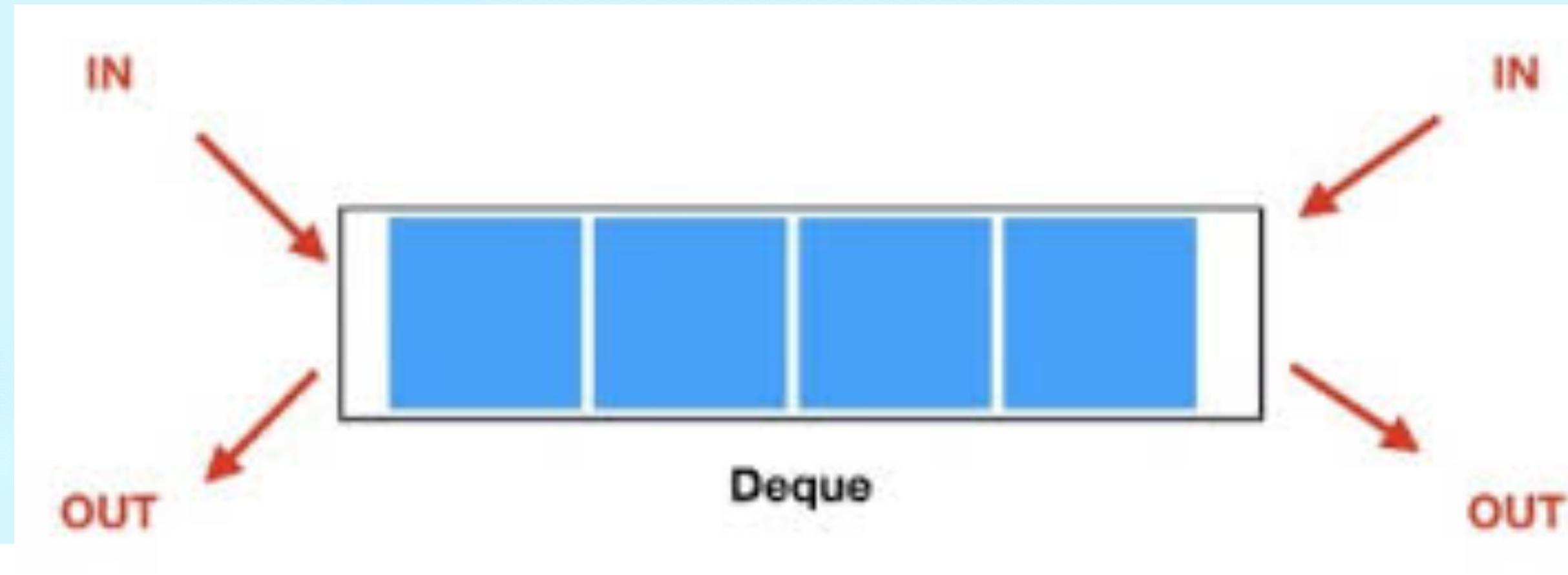
큐

문제 풀이

- 시간 복잡도는 $O(N \times K)$
 - 중간에 $k-1$ 번 회전시키고
 - k 번째꺼를 없앤다.
-
- 마지막엔 3항 연산자로 예쁘게

```
int main(){
    fast_io
    int n, k; cin >> n >> k;
    queue<int> circ;
    for(int i=1;i<=n;i++) circ.push(i);
    vector<int> josephus;
    while(!circ.empty()){
        for(int i=1;i<k;i++){
            circ.push(circ.front());
            circ.pop();
        }
        josephus.push_back(circ.front());
        circ.pop();
    }
    cout << '<';
    for(int i = 0; i< n;i++)
        cout << josephus[i] << ((i == n - 1) ? ">" : ", ");
}
```

덱 생김새



카드 덱은 밑장빼기가 가능하죠.

규칙에 따라 어찌보면 스택일지도?

유용한 자료구조.. 큐랑 덱을 합친 자료구조



덱 코드

- 앞 뒤에 넣을 수 있다는 것이 좋다.
- 그렇다고 무작정 쓰면
 - 문제가 오히려 헷갈릴지도?

```
dq.push_back(1);
dq.push_back(2);
dq.push_back(3);
dq.push_back(4);
cout << dq[1] << ' ' << dq[2] << '\n';
queue<int> q;
q.push(1);
q.push(2);
q.push(3);
cout << q[1] << ' ' << q[2] << '\n';
stack<int> stk;
stk.push(1);
stk.push(2);
stk.push(3);
cout << stk[0] << ' ' << stk[1] << '\n';
```

```
int main(){
    fast_io
    deque<int> dq;
    dq.push_back(1);
    dq.push_front(2);
    dq.front();
    dq.back();
    dq.pop_back();
    dq.pop_front();
    dq.size();
    dq.empty();
}
```

덱 문제 풀이

타노스는 요세푸스가 밉다

성공



2 실버 II

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
0.1 초	512 MB	817	464	401	61.692%

문제

N 마리의 청설모가 1번부터 N 번까지 순서대로 시계 방향으로 원을 이루면서 앉아있다. 타노스는 손을 퉁겨서 순서대로 두 번째 청설모를 제거해 왔는데, 옆 나라의 수학자 요세푸스도 이미 그 방식을 사용해 왔다는 것을 알자 기분이 상했다. 그래서 타노스는 새롭게 청설모를 제거하는 방식을 고안했다.

시작은 1번 청설모를 첫 번째 청설모로 한다. 타노스가 손을 퉁기면 첫 번째 청설모부터 시계 방향으로 K 마리의 청설모가 선택된다. 이후 첫 번째 청설모를 제외한 $2, \dots, K$ 번째 청설모가 번호가 증가하는 순서대로 제거되고 첫 번째 청설모만 살아남는다. 단, 남아 있는 청설모가 K 마리보다 적으면 첫 번째 청설모를 제외한 모든 청설모가 제거된다. 제거된 후 남아있는 청설모가 2마리 이상일 경우 첫 번째 청설모의 오른쪽 청설모가 첫 번째 청설모가 되고, 제거하는 과정을 다시 진행한다. 이 과정은 청설모가 1마리 남을 때까지 계속된다.

N, K 가 주어질 때 마지막으로 남는 청설모의 번호를 구하여라.

덱

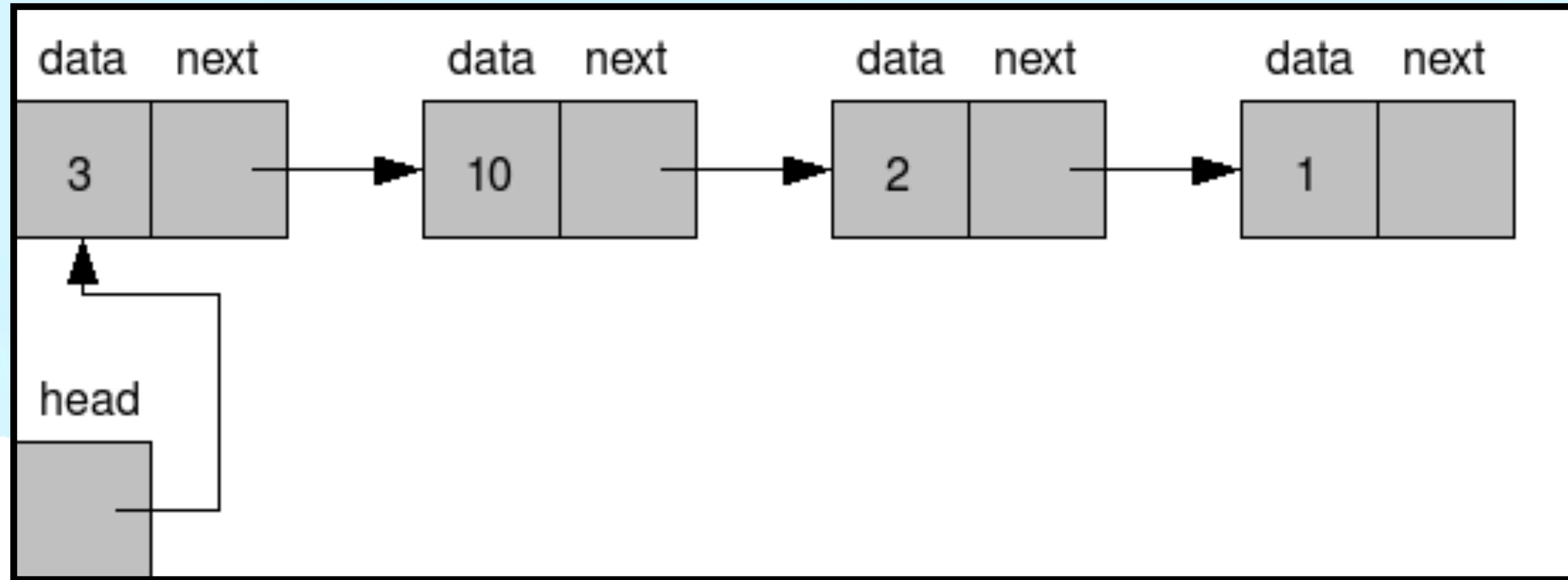
문제 풀이

- 처음 친구는 뒤로가서 살아남고
 - 앞에 k명이 죽는다...(첫번째 친구 포함)
 - 이 짓을 k보다 작아질 때까지 반복
 - 맨 앞을 출력
-
- 왜 덱?
 - 큐,스택으로 풀 수 있으면 덱으로 가능
 - 자기한테 직관적인걸로 푸세요
 - 사실 이 문제는 그냥 큐로 풀리긴함

```
int main(){
    fast_io int n, k;
    cin >> n >> k;
    deque<int> dq;
    for (int i = 1; i <= n; i++)
        dq.push_back(i);
    while (dq.size() >= k){
        dq.push_back(dq.front());
        for (int i = 0; i < k; i++)
            dq.pop_front();
    }
    cout << dq.front();
```

연결리스트(linked list)

생김새



이건 단방향 리스트

연결리스트(linked list)

문제풀이 : 에디터

문제

한 줄로 된 간단한 에디터를 구현하려고 한다. 이 편집기는 영어 소문자만을 기록할 수 있는 편집기로, 최대 600,000글자까지 입력할 수 있다.

이 편집기에는 '커서'라는 것이 있는데, 커서는 문장의 맨 앞(첫 번째 문자의 왼쪽), 문장의 맨 뒤(마지막 문자의 오른쪽), 또는 문장 중간 임의의 곳(모든 연속된 두 문자 사이)에 위치할 수 있다. 즉 길이가 L인 문자열이 현재 편집기에 입력되어 있으면, 커서가 위치할 수 있는 곳은 L+1가지 경우가 있다.

이 편집기가 지원하는 명령어는 다음과 같다.

L	커서를 왼쪽으로 한 칸 옮김 (커서가 문장의 맨 앞이면 무시됨)
D	커서를 오른쪽으로 한 칸 옮김 (커서가 문장의 맨 뒤이면 무시됨)
B	커서 왼쪽에 있는 문자를 삭제함 (커서가 문장의 맨 앞이면 무시됨) 삭제로 인해 커서는 한 칸 왼쪽으로 이동한 것처럼 나타나지만, 실제로 커서의 오른쪽에 있던 문자는 그대로임
P \$	\$라는 문자를 커서 왼쪽에 추가함

초기에 편집기에 입력되어 있는 문자열이 주어지고, 그 이후 입력한 명령어가 차례로 주어졌을 때, 모든 명령어를 수행하고 난 후 편집기에 입력되어 있는 문자열을 구하는 프로그램을 작성하시오. 단, 명령어가 수행되기 전에 커서는 문장의 맨 뒤에 위치하고 있다고 한다.

입력

첫째 줄에는 초기에 편집기에 입력되어 있는 문자열이 주어진다. 이 문자열은 길이가 N이고, 영어 소문자로만 이루어져 있으며, 길이는 100,000을 넘지 않는다. 둘째 줄에는 입력할 명령어의 개수를 나타내는 정수 M($1 \leq M \leq 500,000$)이 주어진다. 셋째 줄부터 M개의 줄에 걸쳐 입력할 명령어가 순서대로 주어진다. 명령어는 위의 네 가지 중 하나의 형태로만 주어진다.

연결리스트(linked list)

문제풀이 : 에디터

- 직접 리스트 구현

```
14 class Node{
15 public:
16     char data;
17     Node* nxt;
18     Node* pre;
19
20     Node(char c, Node *next, Node * prev) : data(c), nxt(next), pre(prev) {}
21     ~Node() {delete nxt; delete pre;}
22 };
23
24 class Linked_list{
25 public:
26     Node* create_list(){
27         Node* head = new Node('?', NULL, NULL);
28         return head;
29     }
30
31     Node* add_node(Node *now, char c){
32         Node * new_node = new Node(c, now -> nxt, now);
33         if(now -> nxt) now -> nxt -> pre = new_node;
34         now -> nxt = new_node;
35         return new_node;
36     }
37
38     Node * del_node(Node *now){
39         if(!(now -> pre)) return now;
40         Node * tmp = now -> pre;
41         if(now -> pre) now -> pre -> nxt = now -> nxt;
42         if(now -> nxt) now -> nxt -> pre = tmp;
43         return tmp;
44     }
45
46     void print(Node* now){
47         while (now -> nxt){
48             now = now -> nxt;
49             cout << now -> data;
50         }
51     }
52 }
53 }
```

```
55 int main(){
56     fast_io
57     string s; cin >> s;
58     Linked_list lst;
59     Node * pnt = lst.create_list();
60     Node * start = pnt;
61     for(int i=0;i<s.length();i++){
62         pnt = lst.add_node(pnt, s[i]);
63     }
64     int q; cin >> q;
65     while(q--){
66         char op; cin >> op;
67         if(op == 'L'){
68             if(pnt -> pre){
69                 pnt = pnt -> pre;
70             }
71         }else if(op == 'D'){
72             if(pnt -> nxt){
73                 pnt = pnt -> nxt;
74             }
75         }else if(op == 'B'){
76             pnt = lst.del_node(pnt);
77         }else{
78             char c; cin >> c;
79             pnt = lst.add_node(pnt, c);
80         }
81     }
82     lst.print(start);
83 }
```

연결리스트(linked list)

문제풀이 : 에디터

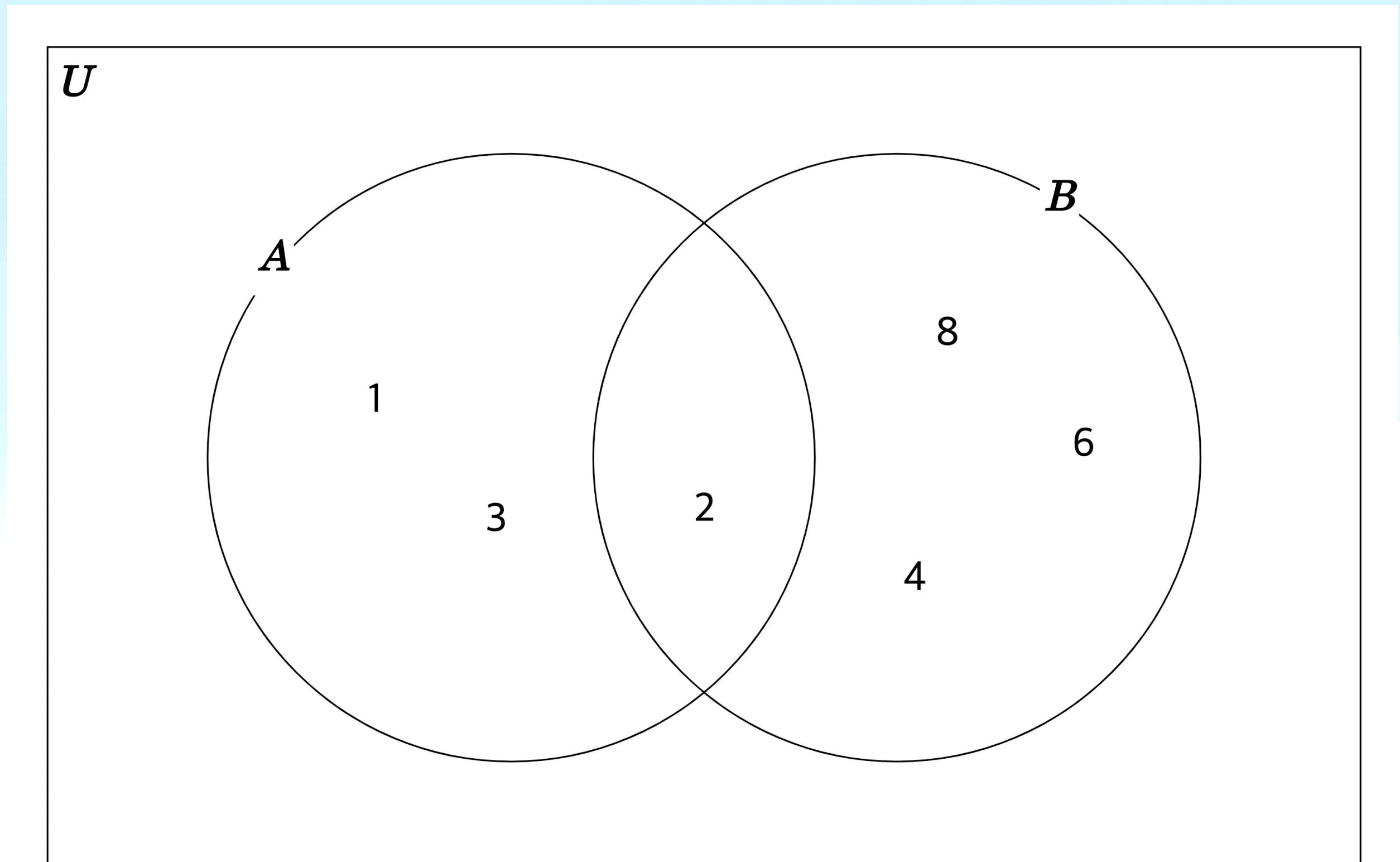
- 이번에는 stl 라이브러리
 - list도 다 구현되어 있습니다.
 - Iterator를 잘 기억하세요
 - 그냥 auto 써도 됨
- 근데 직접 한번 구현해보세요..
 - 저보다 예쁘게 짜보세요

```
13 int main(){
14     fast_io
15     string s; cin >> s;
16     list<char> lst;
17     list<char> :: iterator it = lst.begin();
18     for(char c : s) lst.insert(it, c);
19     int q; cin >> q;
20     while(q--){
21         char op; cin >> op;
22         if(op == 'L' && it != lst.begin()) it--;
23         else if(op == 'D' && it != lst.end()) it++;
24         else if(op == 'B' && it != lst.begin()) it = lst.erase(--it);
25         else if(op == 'P'){
26             char k; cin >> k;
27             lst.insert(it, k);
28         }
29     }
30     for(char x : lst) cout << x;
31 }
```

Set

생김새

- 집합
 - 들어가 있는 원소의 순서는 별로 안중요함
 - 파이썬에선 정렬 안되어있음
 - C++은 알아서 정렬해줌
 - 내가 원하는 정보가 집합에 있는지?
 - $O(\log N)$ 만에 찾을 수 있음
 - 내부적으로 red-black tree 같은 트리를 이용해 balanced한 트리를 만들 것입니다. (따라서 log시간)



Set

코드

- 삽입과 삭제가 $O(\log N)$
- 쿼리를 처리할 때 효과적
- 집합의 합연산(merge)도 구현되어있음
 - 차는 메소드로는 딱히 없음..ㅠㅠ

```
int main(){
    fast_io
    set<int> st1, st2;
    st1.insert(1);
    st1.insert(2);
    st1.insert(3);

    st2.insert(3);
    st2.insert(4);
    st2.insert(5);

    st1.merge(st2);
    for(int x : st1) cout << x << ' ';
    // 1 2 3 4 5

    st1.count(1);
    st1.size();
    st1.clear();
    st1.erase(st1.begin()); //첫번째 원소를 삭제
    //set에서는 원소의 삭제도 꽤 중요함

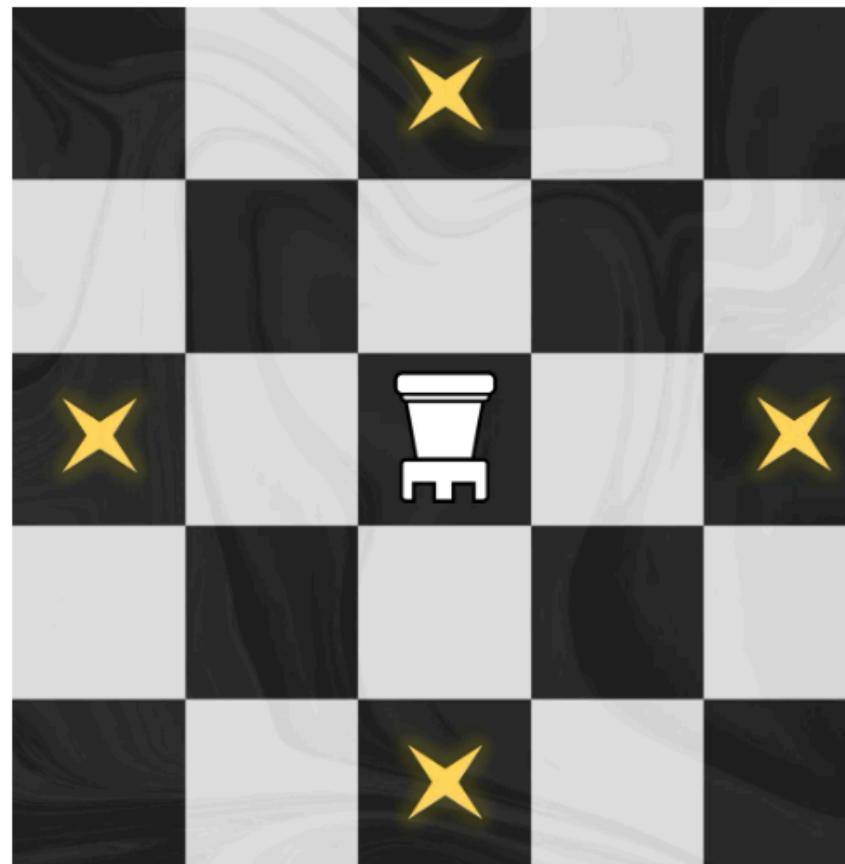
}
```

Set

문제 풀이

문제

은동이는 체스를 두는 것보다 체스 기물을 가지고 노는 것을 좋아한다. 오늘은 변형 체스 기물 중 하나인 다바바(Dabbaba)를 가지고 크기가 $N \times N$ 인 체스판 위에서 놀아보려고 한다.



다바바의 행마

다바바는 상하좌우 중 하나로 2칸을 이동하며 나이트처럼 다른 기물을 뛰어넘을 수 있는 변형 체스 기물이다. 단, 체스판 밖으로 이동하려 하거나 이동하려는 칸에 다른 기물이 있으면 이동하지 못한다.

체스 기물을 가지고 놀던 중 브실이가 실수로 체스판 위에 다바바를 흘뿌려 버렸다. 흘뿌려진 체스판을 본 은동이는 잠시 생각하다가 현재 체스판 위에 있는 다바바가 한 번 이동하여 도착할 수 있는 칸이 총 몇 칸인지 궁금해졌다.

체스판의 크기 N 과 다바바의 개수 K 가 주어졌을 때 다바바가 한 번 이동하여 도착할 수 있는 모든 칸의 개수를 구해보자. 동일한 칸에 가는 방법이 여러 가지라도 칸의 개수는 하나로 센다.

Set

문제 풀이

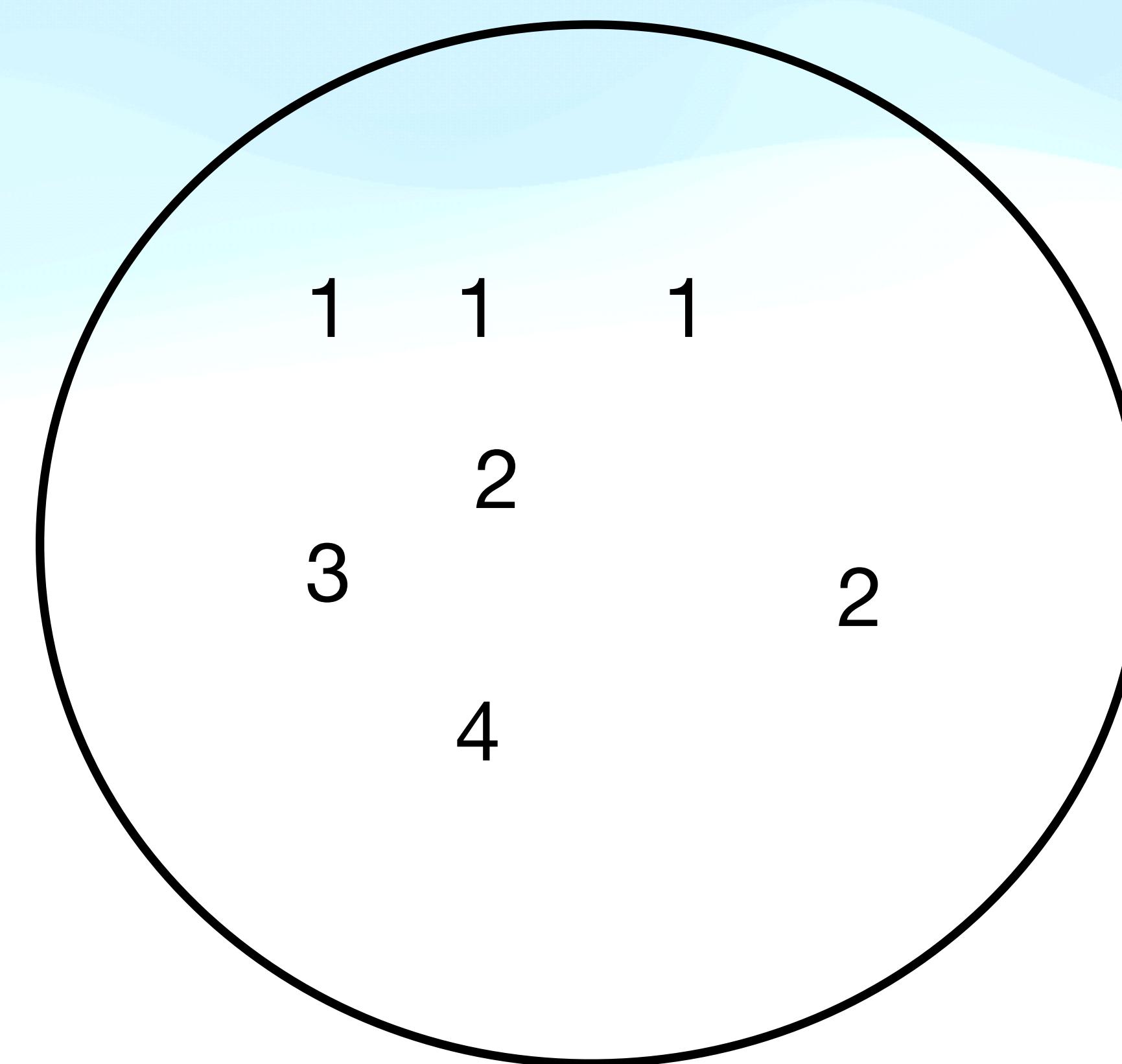
- 방향 처리를 어떻게 하는지 잘보세요!
- 나중에 그래프할때 유용하게 씁니다.
- Const : 못 바꾸게 상수로 만듦
- set에 pii 를 넣는 것을 보여줌
- 시간 복잡도는 $O(K \times \log(\min(N^2, 4K)))$
 - 말이 있을만한 자리를
 - 모두 저장하는 배열을 만들면 메모리초과

```
14     const int dy[] = {0, 0, -2, 2};
15     const int dx[] = {-2, 2, 0, 0};
16
17     int main(){
18         fast_io
19         int N, K; cin >> N >> K;
20         vector<pii> C(K);
21         set<pii> st;
22         for (int i = 0; i < K; i++){
23             int y, x;
24             cin >> y >> x;
25             C[i] = {y, x};
26             st.insert(C[i]);
27         }
28
29         for (int i = 0; i < K; i++){
30             auto [y, x] = C[i];
31             for (int j = 0; j < 4; j++){
32                 int ny = y + dy[j];
33                 int nx = x + dx[j];
34                 if (ny >= 1 && ny <= N && nx >= 1 && nx <= N)
35                     st.insert({ny, nx});
36             }
37         }
38         cout << st.size() - K;
39     }
```

Multiset

생김새

- set과 개념이 비슷하지만 중복된 정보를 따로 본다.
- 내가 찾는 정보가 집합에 몇개 있는지?
 - $O(\log N)$ 만에 찾을 수 있음
- 매우 유용해서 자주 씀
- set과 multiset에서 이분탐색을 하는
 - `lower_bound`나 `upper_bound`가 있는데
 - 다음시간에 시간이 남으면 다뤄볼게요



Multiset

문제 풀이

문제

상훈이는 N 개의 선물 상자를 가지고 있다. 선물 상자에는 현재 담겨있는 선물의 개수가 적혀있다.

선물을 받을 아이들이 M 명 있다. 아이들은 각자 1에서 M 까지의 서로 다른 번호를 하나씩 부여받았다.

1번 아이부터 M 번 아이까지 한 번에 한 명씩, 현재 선물이 가장 많이 담겨있는 상자에서 각자 원하는 만큼 선물을 가져간다. 이 때, 앞서 누군가 선물을 가져갔던 선물 상자에서 또다시 가져가도 상관없다.

하지만 상자에 자신이 원하는 것보다 적은 개수의 선물이 들어있다면, 선물을 가져가지 못해 실망한다.

상훈이는 한 명이라도 실망하지 않고 모두가 선물을 가져갈 수 있는지 궁금하다.

입력

첫째 줄에 선물 상자의 수 N 과 아이들의 수 M 이 공백을 사이에 두고 주어진다. ($1 \leq M \leq N \leq 10^5$)

둘째 줄에 각 선물 상자에 들어있는 선물의 개수 c_1, c_2, \dots, c_N 이 공백을 사이에 두고 주어진다. ($1 \leq c_i \leq 10^5$)

셋째 줄에 아이들의 번호 순으로 각 아이가 원하는 선물의 개수 w_1, w_2, \dots, w_M 이 공백을 사이에 두고 주어진다. ($1 \leq w_i \leq 10^5$)

Multiset

문제 풀이

```
int main(){
    fast_io
    int n, m; cin >> n >> m;
    multiset<int> mt;
    for (int i = 0; i < n; i++){
        int k;
        cin >> k;
        mt.insert(k);
    }
    bool ok = true;
    for (int i = 0; i < m; i++){
        int k;
        cin >> k;
        multiset<int>::iterator it = mt.end(); //end()는 마지막요소의 다음꺼를 가리킴(제일 큰 요소일것이다.)
        it--;
        if (*it >= k)
        {
            int tmp = *it - k;
            mt.erase(it);
            if (tmp)
                mt.insert(tmp);
        }
        else
            ok = false;
    }
    cout << ok;
}
```

Map

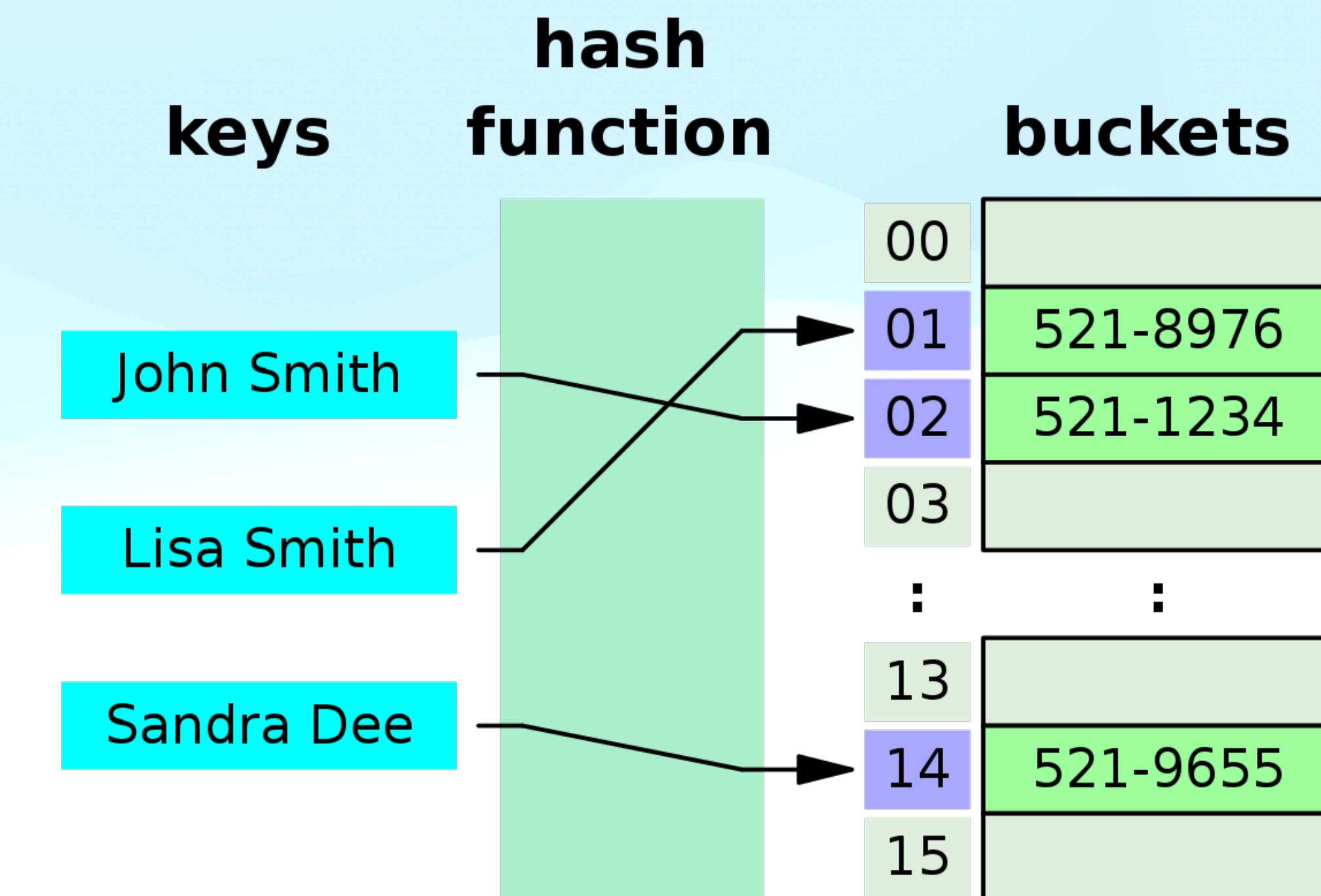
생김새

- 일단 테이블을 떠올리는 게 좋습니다.

- mapping을 시키는 것

- 일단 log 시간복잡도라고 생각하되

- 꽤나 느리다고 생각합시다.



파이썬은 내부적으로 맵을 구현할 때 해시 테이블을 쓸 것입니다.

Map 문제풀이

문제

동현이는 볼링을 사랑하는 훌륭한 프로그래머다. 오늘도 볼링을 치고 싶은 동현이는 자신의 볼링공 컬렉션을 보면서 어떤 볼링공을 가져갈지 고민에 빠졌다. 동현이는 매일의 컨디션에 따라 아주 미세한 무게까지 컨트롤하고 싶기 때문에 다양한 무게의 볼링공이 매우 많다. 볼링공을 관리하는 사물함에는 사물함 안에 들어 있는 볼링공들의 무게가 적혀 있고, 하나의 사물함에 여러 개의 볼링공이 들어갈 수 있다. 동현이는 오로지 볼링에만 집중하고 싶기 때문에 볼링공 관리는 여러분에게 맡기기로 했다.

동현이의 요청에 따라 볼링공을 관리해 보자!

요청은 다음과 같다.

- 1 x w : x 번 사물함에 w 무게의 볼링공을 넣는다.
- 2 w : w 무게를 가진 볼링공이 들어 있는 사물함의 번호를 출력한다.

동현이의 볼링공은 무게가 모두 다르기 때문에 2번 요청의 답은 항상 하나만 존재한다.

2번 요청이 들어올 때, w 무게를 가진 볼링공은 무조건 어딘가에 존재한다.

Map

문제풀이

- 이 문제를 배열로 풀 수 있을까?
 - w범위 때문에 안된다!!
 - 이처럼 배열로 나타낼 수 없는
 - String
 - Pair
 - Char
 - 인덱스로 위와 같은 것을 넣을 수 있음

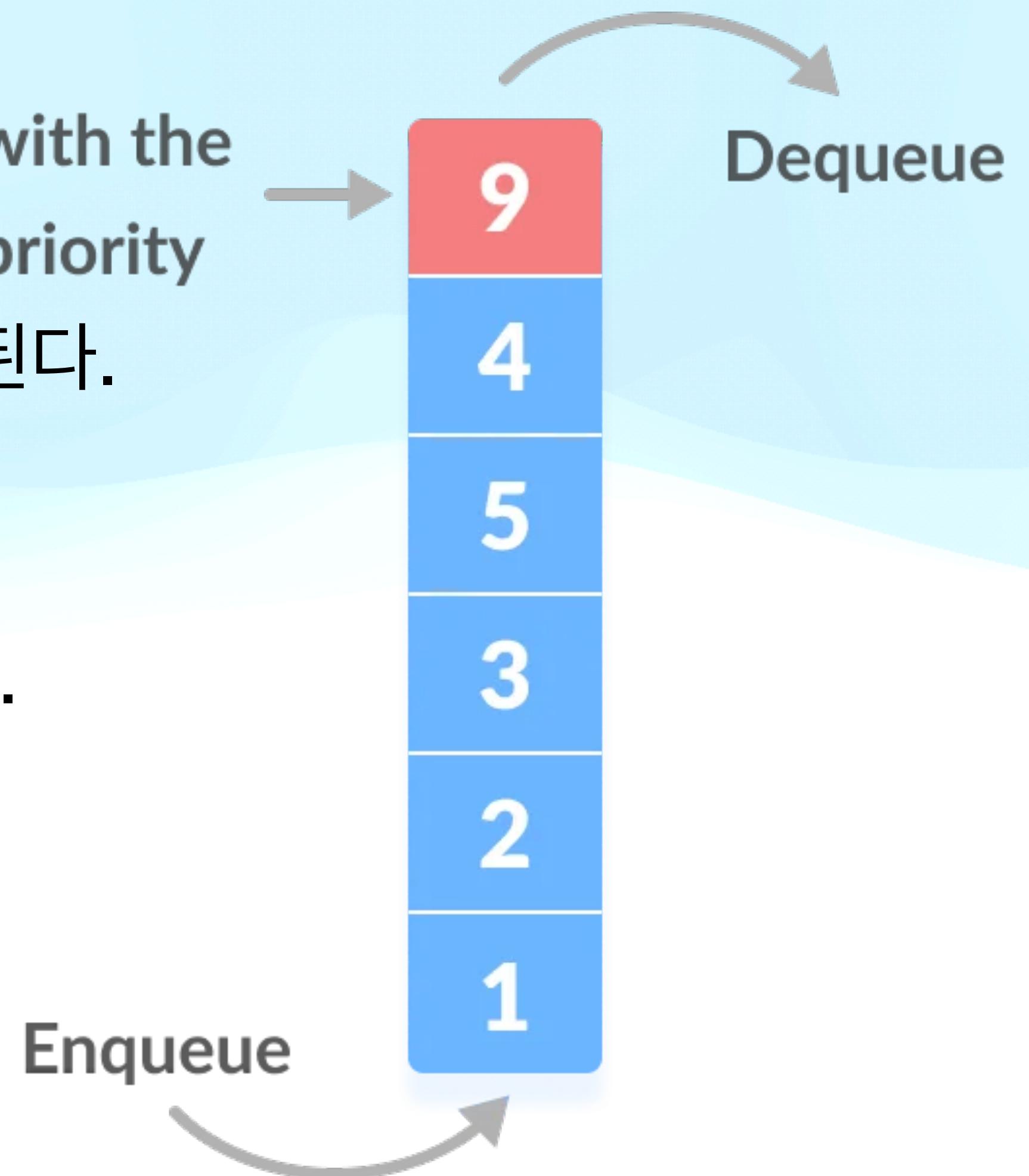
```
13 int main(){
14     fast_io
15     int m ; cin >> m;
16     map<int, int> mp;
17     while(m--){
18         int op; cin >> op;
19         if(op == 1){
20             int x, w; cin >> x >> w;
21             mp[w] = x;
22         }else{
23             int w; cin >> w;
24             cout << mp[w] << '\n';
25         }
26     }
27 }
```

Priority queue

생김새

- 넣으면 우선순위로 자동으로 정렬된다
- pair<int, int>를 넣으면 앞에거 기준으로 먼저 정렬된다.
 - 이건 비교함수가 정의가 되어있어서 그렇다.
- 우리의 노드를 만든다면 연산자 오버로딩을 해야한다.

Element with the
highest priority



Priority queue

문제풀이

문제

넥슨의 **FIFA ONLINE 4**는 축구 경기 시뮬레이션을 생동감 있게 플레이할 수 있는 게임이다. 각 선수별로 포지션이 있으며, 선수별로 능력치와 선수 가치가 존재하여 경기에 영향을 미치게 된다. 또한 선발 선수와 후보 선수를 구분하여 선발 선수의 컨디션이 안 좋은 경우 후보 선수와 교체할 수도 있다.



FIFA ONLINE 4를 즐겨하는 주원이는 다음 과정대로 팀을 구성하면 K 년 12월에 되었을 때 팀의 선발 선수 가치의 합은 얼마가 되어 있을지 궁금해 하였다.

1. 팀은 총 11명으로, 포지션 번호는 1 이상 11 이하인 정수이다.
2. 3월에 같은 포지션 중 선수 가치가 가장 높은 선수가 선발 선수가 된다. 선수 가치가 가장 높은 선수가 여러 명 있을 경우, 그 중 아무나 선발 선수로 선택한다.
3. 포지션에 해당하는 선수가 없을 시, 해당 포지션을 공석으로 팀을 구성한다.
4. 8월에 현재 팀에 있는 선발 선수의 선수 가치는 모두 1이 떨어진다. 선수 가치는 0보다 작아지지 않는다.
5. 11월에 2의 조건대로 선발 선수를 재구성한다.

선수 N 명의 포지션 번호와 선수 가치가 주어졌을 때, $1, 2, \dots, K$ 년이 될 때마다 매년 위 과정대로 새로운 팀을 구성한다. K 년 12월에 되었을 때 만든 팀의 선발 선수 가치의 합을 구하여라.

Priority queue

문제풀이

- 원소가 pq인 배열을 만들었습니다.
- Pq 안에 몇개의 원소가 들어가는지 봅시다.
 - Pq안에 원소는 N개 들어갑니다.
 - 따라서 $O(N \log N)$

```
priority_queue<int> pq[12];
int main(){
    fast_io
    int N, K; cin >> N >> K;
    for(int i=0;i<N;i++){
        int p, w; cin >> p >> w;
        pq[p].push(w);
    }
    int res = 0;
    for(int i=0;i<K;i++){
        for(int j=1;j<=11;j++){
            if(pq[j].empty()) continue;
            int x = pq[j].top();
            pq[j].pop();
            pq[j].push(x-1);
        }
        res = 0;
        for(int j=1;j<=11;j++){
            if(pq[j].empty()) continue;
            res += pq[j].top();
        }
    }
    cout << res;
}
```

마무리

- Stack
- Queue
- Deque
- List
- Set
- Multiset
- Map
- Priority queue
- 엄청나게 많은 도구들을 배웠습니다. 이 것들이 나중에도 중간중간 나올 겁니다.

마무리

추천문제

- 9012 괄호 - 스택
- 17298 오큰수 - index를 같이 저장한 스택
- 6198 옥상 정원 꾸미기 - 스택, 정수형 범위 주의!
- 1158 요세푸스 문제 - 큐, 원형 배열 처리
- 28066 타노스는 요세푸스가 밟다 - 덱, 큐
- 24511 Queuestack - 덱
- 1406 에디터, 5397 키로거 - 연결리스트 기본문제

마무리

추천문제

- 29721 변형 체스놀이 : 다바바 - set에 pair저장
- 23757 아이들과 선물 상자 - multiset, 정렬되는 성질 이용
- 7662 이중 우선순위 큐 - 우선순위 큐 쓸 순있는데 어려움,, multiset으로 하면 편함
- 28446 볼링공 찾아주기 - map
- 31562 전주듣고 노래 맞히기 - map
- 29160 나의 FIFA 팀 가치는? - 우선순위 큐
- 11003 최솟값 찾기 - 도전문제임... 덱으로 푸는 것이 가장 직관적이고 편했음 (우선순위 큐도 가능)
 - 덱 안에 index를 같이 저장해서 범위를 관리한다.
 - 덱 안의 데이터들은 오름차순으로 저장되어 있다면 맨앞은 구간 내 가장 작은 값이 된다.