KT AIVLE 코딩 스터디
Heap

최아현

목차

1. 프로그래머스 힙

힙(Heap)

출제 빈도 보통 | 평균 점수 높음

힙은 특정한 규칙을 가지는 트리로, 힙을 이용해서 우선순위 큐를 구현할 수 있습니다.

많은 언어에서 이미 구현된 우선순위 큐 라이브러리를 제공합니다. 이를 활용하면 효율적으로 문제를 풀 수 있습니다. 우선순위 큐를 이용해서 해결하기에 적합한 문제들을 만나보세요.

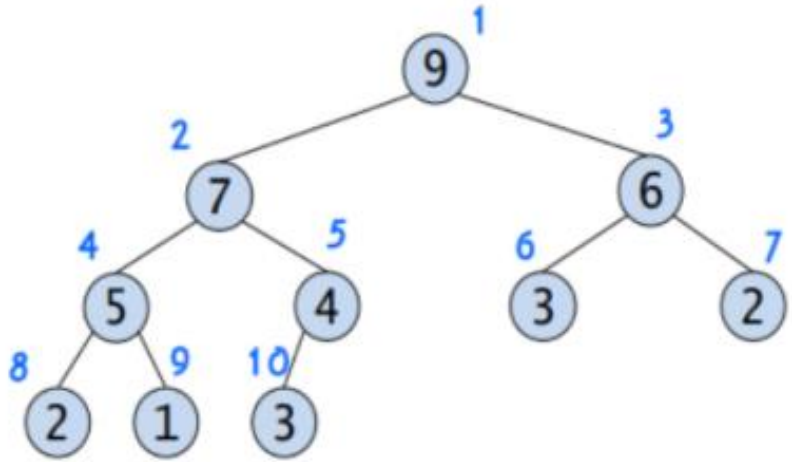
2. 우선순위 큐

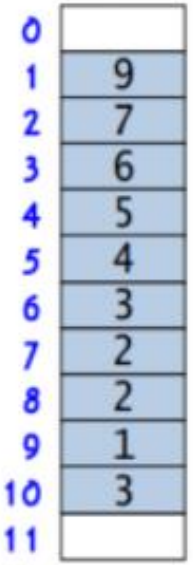
우선순위가 가장 높은 데이터를 가장 먼저 삭제하는 자료구조

우선순위에 따라 작업을 처리하고자 할 때 사용

ex) 물건 데이터를 자료구조에 넣었다가

가치가 높은 물건부터 꺼내서 확인해야하는 경우





- 우선순위 큐를 구현하는 방법

1. 리스트를 기반으로 구현
2. 연결 리스트를 기반으로 구현
3. 힙을 이용하여 구현

우선순위 큐를 구현하는 표현 방법	삽입	삭제
순서 없는 배열	O(1)	O(n)
순서 없는 연결 리스트	O(1)	O(n)
정렬된 배열	O(n)	O(1)
정렬된 연결 리스트	O(n)	O(1)
힙(heap)	O(logn)	O(logn)

3. 힙(Heap)

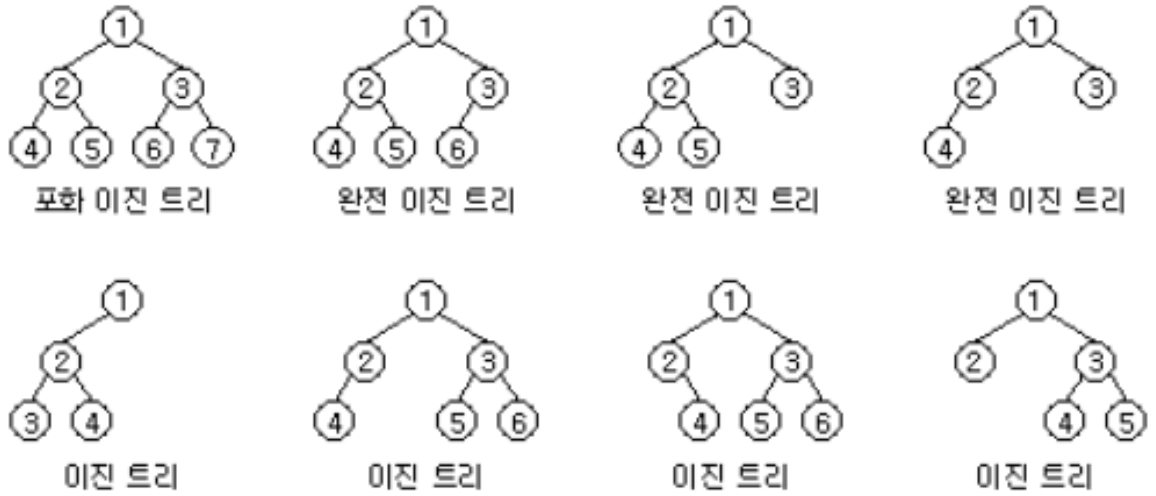
힙은 완전 이진 트리 자료구조의 일종이다. 우선순위 큐를 위해 고안되었다.

힙의 종류

- 최대 힙
- 최소 힙

- 완전이진트리

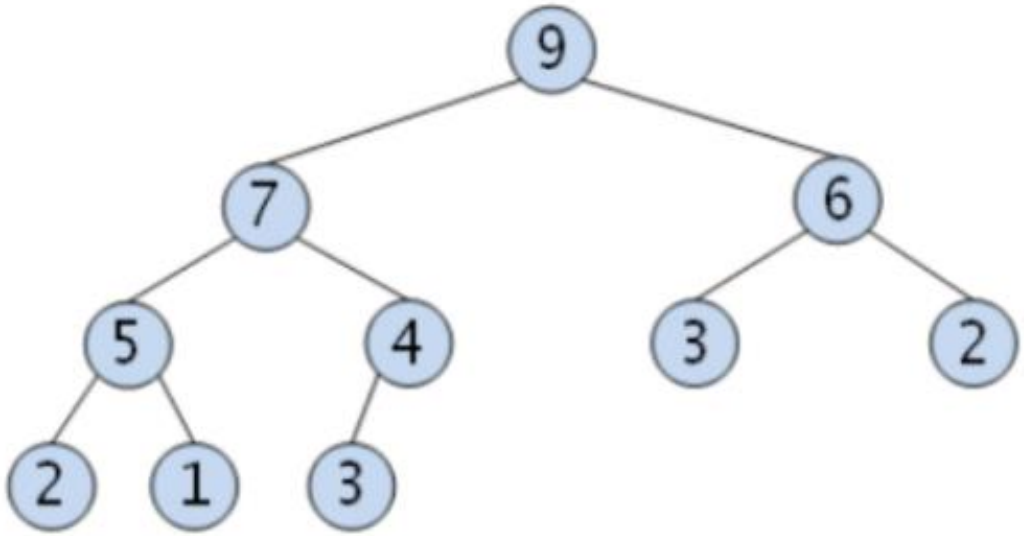
: 이진트리에서 자식 노드들이 왼쪽부터 차곡차곡 채워져 있는 트리를 말하며 완전 이진트리는 마지막 레벨을 제외하고 노드가 모두 채워져 있어야 한다. 마지막 레벨도 왼쪽부터 채워져 있어야 한다.



- 최대 힙

부모 노드의 키 값이 자식 노드의 키 값보다 크거나 같은 완전 이진 트리로 값이 큰 데이터가 우선적으로 제거

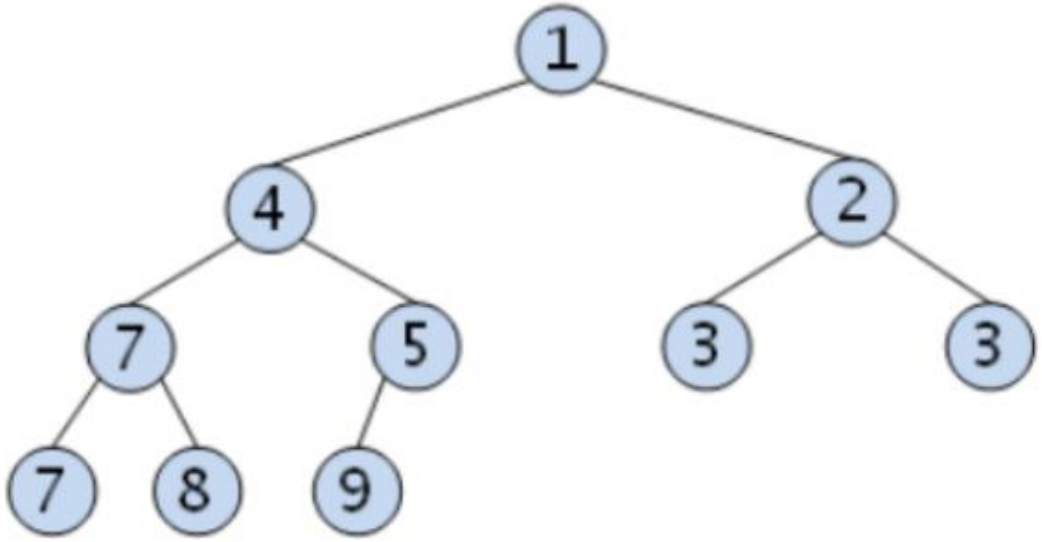
Key(부모 노드) >= key(자식 노드)



- 최소 힙

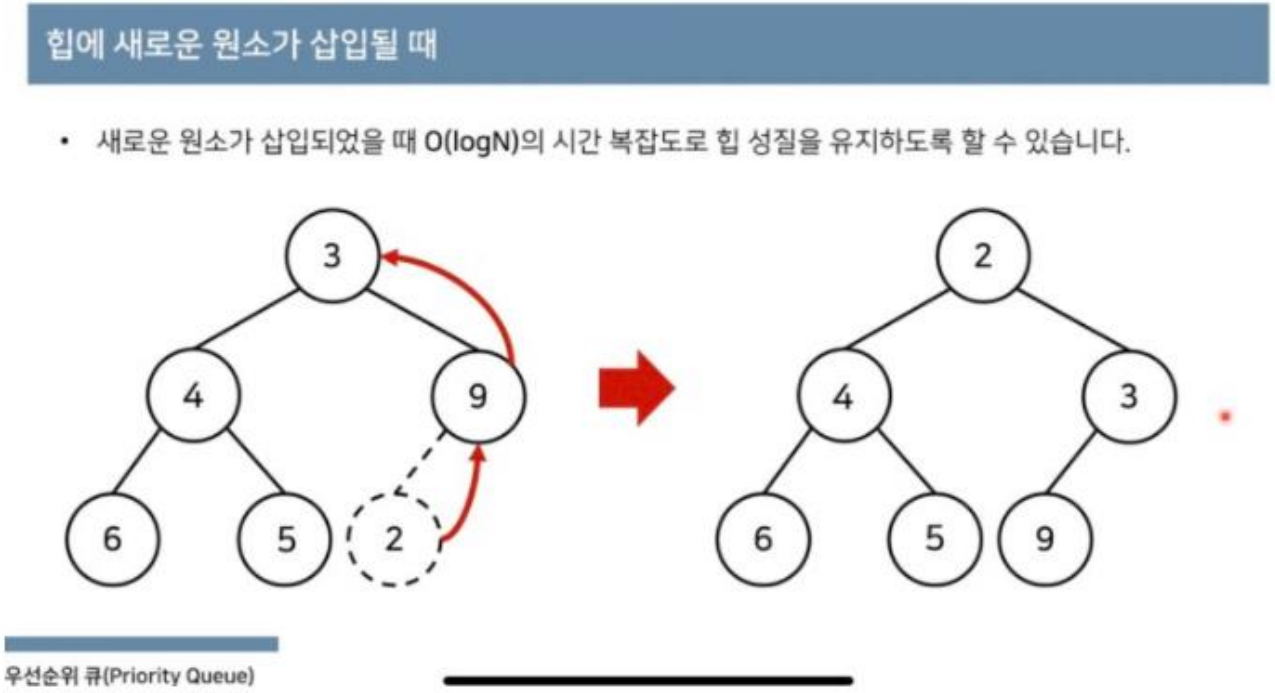
부모 노드의 키 값이 자식 노드의 키 값보다 작거나 같은 완전 이진 트리로 값이 가장 작은 데이터가 우선적으로 제거

key(부모 노드) <= key(자식 노드)



- 힙의 삽입

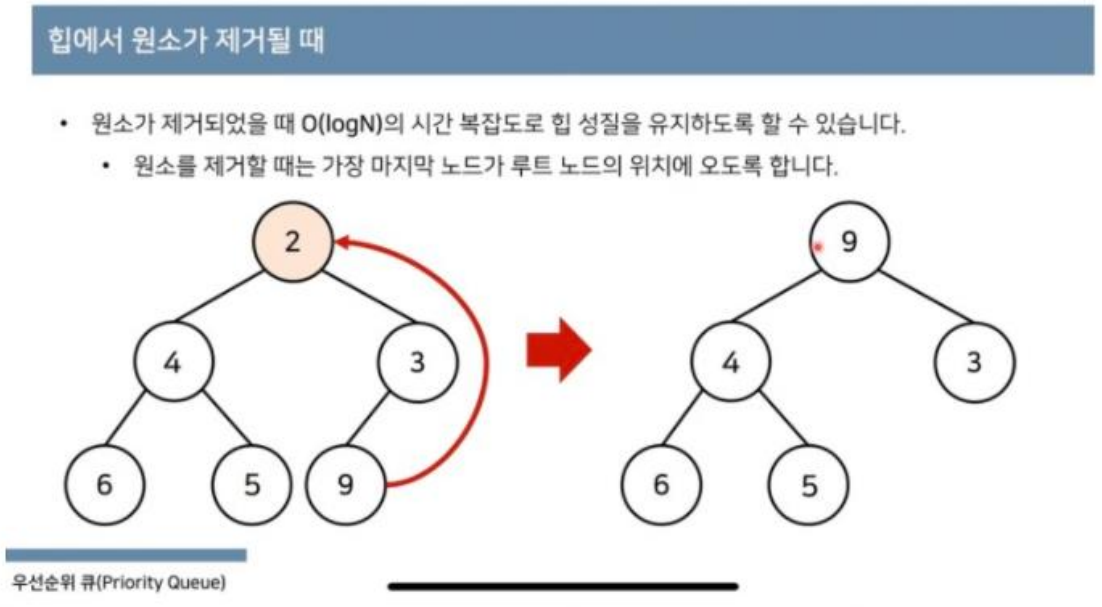
(상향식) 부모 노드로 거슬러 올라가며, 부모보다 자신의 값이 더 작은 경우에 위치를 교체한다.

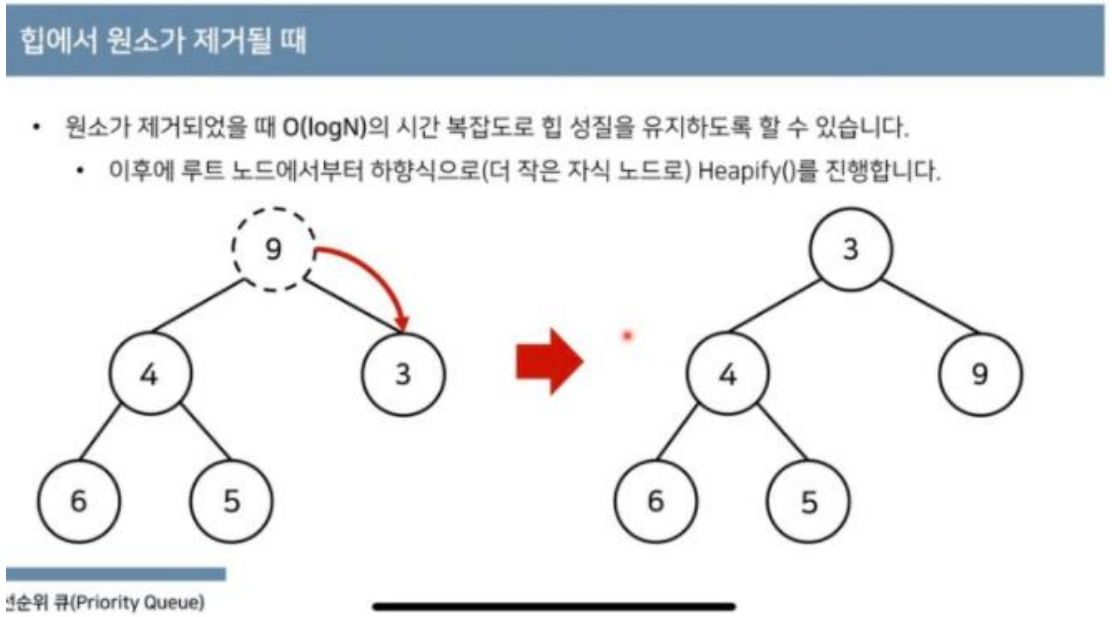


- 힙의 삭제

힙의 삭제의 경우 루트 노드자리에 가장 마지막 노드를 대체 시킨다.

그리고 다음과 같이 힙 성질을 만족하기 위한 연산을 수행한다.





5. 문제 - 11279 최대 힙(실버2)

문제

널리 잘 알려진 자료구조 중 최대 힙이 있다. 최대 힙을 이용하여 다음과 같은 연산을 지원하는 프로그램을 작성하시오.

1. 배열에 자연수 x를 넣는다.
2. 배열에서 가장 큰 값을 출력하고, 그 값을 배열에서 제거한다.

프로그램은 처음에 비어있는 배열에서 시작하게 된다.

입력

첫째 줄에 연산의 개수 N(1 ≤ N ≤ 100,000)이 주어진다. 다음 N개의 줄에는 연산에 대한 정보를 나타내는 정수 x가 주어진다. 만약 x가 자연수라면 배열에 x라는 값을 넣는(추가하는) 연산이고, x가 0이라면 배열에서 가장 큰 값을 출력하고 그 값을 배열에서 제거하는 경우이다. 입력되는 자연수는 2^{31}보다 작다.

출력

입력에서 0이 주어진 회수만큼 답을 출력한다. 만약 배열이 비어 있는 경우인데 가장 큰 값을 출력하라고 한 경우에는 0을 출력하면 된다.

예제 입력 1 복사

```
13
0
1
2
0
0
3
2
1
0
0
0
0
0
```

예제 출력 1 복사

```
0
2
1
3
2
1
0
0
```

```cpp
#include <cstdio>
#include <iostream>
#include <queue>

using namespace std;

int main() {
    //ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);
    ios_base::sync_with_stdio(false); cin.tie(NULL);

    int n, x;
    priority_queue<int> pq;

    cin >> n;

    for (int i=0; i<n; i++) {
        cin >> x;
        if (x == 0) {
            if (pq.empty()) {
                cout << 0 << "₩n";
            }
            else {
                cout << pq.top() << "₩n";
                pq.pop();
            }
        }
        else {
            pq.push(x);
        }
    }
    return 0;
}
```

5. 문제 - 14235번 크리스마스 선물(실버3)

문제

크리스마스에는 산타가 착한 아이들에게 선물을 나눠준다. 올해도 산타는 선물을 나눠주기 위해 많은 노력을 하고 있는데, 전세계를 돌아댕기며 착한 아이들에게 선물을 나눠줄 것이다. 하지만 산타의 썰매는 그렇게 크지 않기 때문에, 세계 곳곳에 거점들을 세워 그 곳을 방문하며 선물을 충전해 나갈 것이다. 또한, 착한 아이들을 만날 때마다 자신이 들고있는 가장 가치가 큰 선물 하나를 선물해 줄 것이다.

이제 산타가 선물을 나눠줄 것이다. 차례대로 방문한 아이들과 거점지의 정보들이 주어졌을 때, 아이들이 준 선물들의 가치들을 출력하시오. 만약 아이들에게 줄 선물이 없다면 -1을 출력하시오.

입력

첫 번째 줄에서는 아이들과 거점지를 방문한 횟수 n이 주어진다.($1 \le n \le 5,000$)

다음 n줄에는 a가 들어오고, 그 다음 a개의 숫자가 들어온다. 이는 거점지에서 a개의 선물을 충전하는 것이고, 그 숫자들이 선물의 가치이다. 만약 a가 0이라면 거점지가 아닌 아이들을 만난 것이다. 선물의 가치는 100,000보다 작은 양의 정수이다.($1 \le a \le 100$)

출력

a가 0일 때마다, 아이들에게 준 선물의 가치를 출력하시오. 만약 줄 선물이 없다면 -1을 출력하라. 적어도 하나의 출력이 있음을 보장한다.

5. 문제 - 1766번 문제집(골드2)

문제

민오는 1번부터 N번까지 총 N개의 문제로 되어 있는 문제집을 풀려고 한다. 문제는 난이도 순서로 출제되어 있다. 즉 1번 문제가 가장 쉬운 문제이고 N번 문제가 가장 어려운 문제가 된다.

어떤 문제부터 풀까 고민하면서 문제를 훑어보던 민오는, 몇몇 문제들 사이에는 '먼저 푸는 것이 좋은 문제'가 있다는 것을 알게 되었다. 예를 들어 1번 문제를 풀고 나면 4번 문제가 쉽게 풀린다거나 하는 식이다. 민오는 다음의 세 가지 조건에 따라 문제를 풀 순서를 정하기로 하였다.

1. N개의 문제는 모두 풀어야 한다.
2. 먼저 푸는 것이 좋은 문제가 있는 문제는, 먼저 푸는 것이 좋은 문제를 반드시 먼저 풀어야 한다.
3. 가능하면 쉬운 문제부터 풀어야 한다.

예를 들어서 네 개의 문제가 있다고 하자. 4번 문제는 2번 문제보다 먼저 푸는 것이 좋고, 3번 문제는 1번 문제보다 먼저 푸는 것이 좋다고 하자. 만일 4-3-2-1의 순서로 문제를 풀게 되면 조건 1과 조건 2를 만족한다. 하지만 조건 3을 만족하지 않는다. 4보다 3을 충분히 먼저 풀 수 있기 때문이다. 따라서 조건 3을 만족하는 문제를 풀 순서는 3-1-4-2가 된다.

문제의 개수와 먼저 푸는 것이 좋은 문제에 대한 정보가 주어졌을 때, 주어진 조건을 만족하면서 민오가 풀 문제의 순서를 결정해 주는 프로그램을 작성하시오.

입력

첫째 줄에 문제의 수 N(1 ≤ N ≤ 32,000)과 먼저 푸는 것이 좋은 문제에 대한 정보의 개수 M(1 ≤ M ≤ 100,000)이 주어진다. 둘째 줄부터 M개의 줄에 걸쳐 두 정수의 순서쌍 A,B가 빈칸을 사이에 두고 주어진다. 이는 A번 문제는 B번 문제보다 먼저 푸는 것이 좋다는 의미이다.

항상 문제를 모두 풀 수 있는 경우만 입력으로 주어진다.

출력

첫째 줄에 문제 번호를 나타내는 1 이상 N 이하의 정수들을 민오가 풀어야 하는 순서대로 빈칸을 사이에 두고 출력한다.

5. 문제 - 1655번 가운데를 말해요(골드2)

문제

백준이는 동생에게 "가운데를 말해요" 게임을 가르쳐주고 있다. 백준이가 정수를 하나씩 외칠때마다 동생은 지금까지 백준이가 말한 수 중에서 중간값을 말해야 한다. 만약, 그동안 백준이가 외친 수의 개수가 짝수개라면 중간에 있는 두 수 중에서 작은 수를 말해야 한다.

예를 들어 백준이가 동생에게 1, 5, 2, 10, -99, 7, 5를 순서대로 외쳤다고 하면, 동생은 1, 1, 2, 2, 2, 2, 5를 차례대로 말해야 한다. 백준이가 외치는 수가 주어졌을 때, 동생이 말해야 하는 수를 구하는 프로그램을 작성하시오.

입력

첫째 줄에는 백준이가 외치는 정수의 개수 N이 주어진다. N은 1보다 크거나 같고, 100,000보다 작거나 같은 자연수이다. 그 다음 N줄에 걸쳐서 백준이가 외치는 정수가 차례대로 주어진다. 정수는 -10,000보다 크거나 같고, 10,000보다 작거나 같다.

출력

한 줄에 하나씩 N줄에 걸쳐 백준이의 동생이 말해야 하는 수를 순서대로 출력한다.

5. 문제 - 7662번 이중 우선순위 큐(골드5) *도전

문제

이중 우선순위 큐(dual priority queue)는 전형적인 우선순위 큐처럼 데이터를 삽입, 삭제할 수 있는 자료 구조이다. 전형적인 큐와의 차이점은 데이터를 삭제할 때 연산(operation) 명령에 따라 우선순위가 가장 높은 데이터 또는 가장 낮은 데이터 중 하나를 삭제하는 점이다. 이중 우선순위 큐를 위해선 두 가지 연산이 사용되는데, 하나는 데이터를 삽입하는 연산이고 다른 하나는 데이터를 삭제하는 연산이다. 데이터를 삭제하는 연산은 또 두 가지로 구분되는데 하나는 우선순위가 가장 높은 것을 삭제하기 위한 것이고 다른 하나는 우선순위가 가장 낮은 것을 삭제하기 위한 것이다.

정수만 저장하는 이중 우선순위 큐 Q가 있다고 가정하자. Q에 저장된 각 정수의 값 자체를 우선순위라고 간주하자.

Q에 적용될 일련의 연산이 주어질 때 이를 처리한 후 최종적으로 Q에 저장된 데이터 중 최댓값과 최솟값을 출력하는 프로그램을 작성하라.

입력

입력 데이터는 표준입력을 사용한다. 입력은 T개의 테스트 데이터로 구성된다. 입력의 첫 번째 줄에는 입력 데이터의 수를 나타내는 정수 T가 주어진다. 각 테스트 데이터의 첫째 줄에는 Q에 적용할 연산의 개수를 나타내는 정수 k ($k \leq 1{,}000{,}000$)가 주어진다. 이어지는 k 줄 각각엔 연산을 나타내는 문자('D' 또는 'I')와 정수 n이 주어진다. 'I n'은 정수 n을 Q에 삽입하는 연산을 의미한다. 동일한 정수가 삽입될 수 있음을 참고하기 바란다. 'D 1'는 Q에서 최댓값을 삭제하는 연산을 의미하며, 'D -1'는 Q 에서 최솟값을 삭제하는 연산을 의미한다. 최댓값(최솟값)을 삭제하는 연산에서 최댓값(최솟값)이 둘 이상인 경우, 하나만 삭제됨을 유념하기 바란다.

만약 Q가 비어있는데 적용할 연산이 'D'라면 이 연산은 무시해도 좋다. Q에 저장될 모든 정수는 32-비트 정수이다.

출력

출력은 표준출력을 사용한다. 각 테스트 데이터에 대해, 모든 연산을 처리한 후 Q에 남아 있는 값 중 최댓값과 최솟값을 출력하라. 두 값은 한 줄에 출력하되 하나의 공백으로 구분하라. 만약 Q가 비어있다면 'EMPTY'를 출력하라.

참고 :

https://gmlwjd9405.github.io/2018/05/10/data-structure-heap.html