

Heap, Priority Queue

Priority Queue

Hospital Emergency Queue



Priority Queue

👉 정의

- 우선순위 큐 : 선입선출(FIFO) 구조를 가지고 있는 일반적인 큐와 달리 우선순위가 가장 큰 Element가 먼저 나오는 큐.

👉 예시

- 입력 : 1, 5, 9, 10
- 일반 큐 : (front)[1, 5, 9, 10](rear)
- 우선순위 큐 : (front)[10, 9, 5, 1](rear)

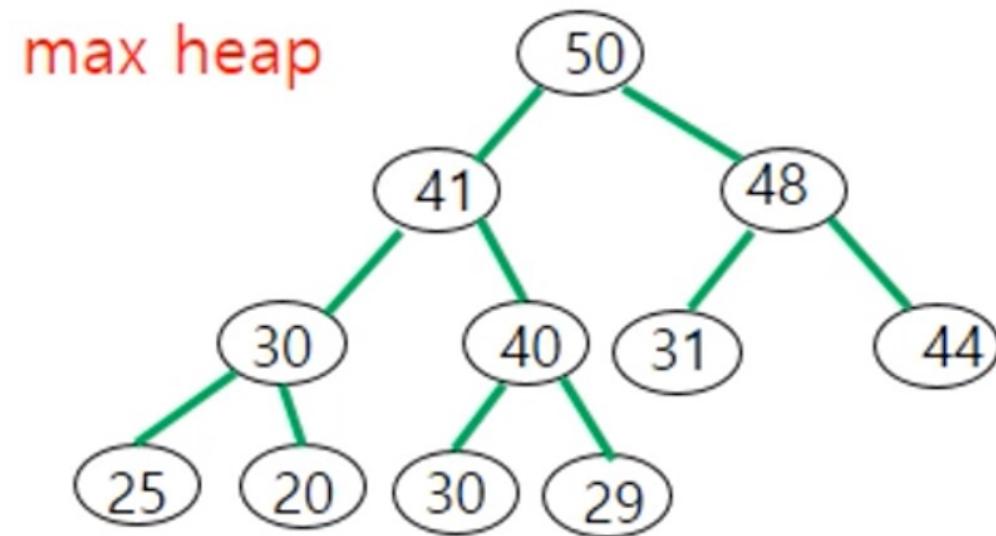
Max(Min) Heap

👉 정의

- **최대(최소) 힙** : 완전 이진 트리이며 모든 루트의 Key가 자식의 Key보다 큰 최대(최소) 트리
- **최대(최소) 힙을 통해 우선순위 큐를 구현**

👉 시간 복잡도

- 삽입 : $O(\log n)$
- 삭제 : $O(\log n)$
- heaptify(or adjust) time 비례



Max(Min) Heap - C++

👉 In C++

- **std::make_heap**
- **<algorithm>**에 정의
- **Vector**를 인자로 받아 힙 구조로 변환 시켜 줌.
- 우선 순위 미지정 시 최대(Max) 힙 Default.
- `push_heap()`, `pop_heap()` 으로 삽입, 삭제 수행
- https://cplusplus.com/reference/algorithm/make_heap/

Max(Min) Heap - C++

👉 make_heap(), push_heap(), pop_heap()

- ⭐ **void make_heap(RandomIt first, RandomIt last, Compare comp)**
- ⭐ **void push_heap(RandomIt first, RandomIt last, Compare comp)**
- ⭐ **void pop_heap(RandomIt first, RandomIt last, Compare comp)**

- **first** : 힙으로 만들 Vector의 시작 지점 iterator
- **last** : 힙으로 만들 Vector의 종료 지점 iterator
- **comp** : Key의 우선순위를 재정의하기 위한 비교 연산자
 - false : 우선순위 ↑, Default : ‘<’

Max(Min) Heap - C++

👉 make_heap(), push_heap(), pop_heap()

```
#include <algorithm>

void print(vector<int>& vs);

void main(){
    vector<int> v =
{1,6,5,2,3,8,4,9,7};
    print(v);
    make_heap(v.begin(), v.end());
    print(v);

    v.push_back(10);
    push_heap(v.begin(),v.end());
    print(v);

    pop_heap(v.begin(),v.end());
    print(v);
    v.pop_back();
}
```

⭐ 결과

⭐ 1 6 5 2 3 8 4 9 7

⭐ 9 7 8 6 3 5 4 2 1 // make_heap

⭐ 10 9 8 6 7 5 4 2 1 3 // push_heap

⭐ 9 7 8 6 3 5 4 2 1 10 // pop_heap

👉 삽입 : v.push_back => push_heap, $O(\log n)$

👉 삭제 : v.push_back => push_heap, $O(\log n)$

Max(Min) Heap - Java

👉 In Java

- **Class PriorityQueue<E>**
- java.lang.Object - java.util.AbstractCollection<E> - java.util.PriorityQueue<E>에 정의
- **Priority Heap 기반 구현**
- 우선 순위 미지정 시 최소(Min) 힙 Default.
- 삽입 : add(), offer()
- 삭제 : peek(), poll(), remove()
- <https://docs.oracle.com/javase/7/docs/api/java/util/PriorityQueue.html>

Max(Min) Heap - Java

👉 생성자

- **PriorityQueue(Collection<? extends E> c)**

👉 메쏘드

⭐️ 삽입

- bool **add(E e)**, bool **offer(E e)**
: PQ에 e 삽입, 성공 시 true, 실패 시 Exception throw

⭐️ 삭제

- E **peek()**, E **poll()**
: PQ에서 우선순위가 높은 요소 삭제 및 해당 요소 반환, Empty시 null 반환

Max(Min) Heap - Java

👉 PriorityQueue

```
import java.util.PriorityQueue;  
  
public class Main {  
    public static void main(String[] args){  
        PriorityQueue<Integer> priorityQueue = new PriorityQueue<>(Collections.reverseOrder());  
  
        priorityQueue.add(1);  
        priorityQueue.add(2);  
        priorityQueue.offer(3);  
  
        priorityQueue.poll();  
        priorityQueue.poll();  
        priorityQueue.peek();  
  
        Iterator iter = priorityQueue.iterator();  
        while(iter.hasNext()){  
            int value = iterator.next();  
            System.out.println(value);  
        }  
    }  
}
```

⭐ 결과

⭐ 1

⭐ 2 1

⭐ 3 2 1

⭐ 2 1

⭐ 1

⭐ 출력 : 1

👉 삽입 : $O(\log n)$

👉 삭제 : $O(\log n)$

Max(Min) Heap - Python

👉 In Python 3.~

1 heapq 모듈

- import **heapq**
- $\text{heap}[k] \leq \text{heap}[2*k+1]$ and $\text{heap}[k] \leq \text{heap}[2*k+2]$
- 최소(Min) 힙 Default.
- Thread-non-safe
- 삽입 : heappush(*heap, item*)
- 삭제 : heappop(*heap*)
- 기타 : heapify(*x*), heappushpop(*heap, item*), ...
- <https://docs.python.org/ko/3.9/library/hearq.html>

Max(Min) Heap - Python

👉 In Python 3.~

1 heapq 모듈

```
1 import heapq
2
3 heap = [4, 1, 7, 3, 8, 5]
4
5 heapq.heapify(heap)
6
7 heapq.heappush(heap, 9)
8 heapq.heappush(heap, 2)
9 print(heap)
10
11 heapq.heappop(heap)
12 print(heap)
13 heapq.heappop(heap)
14 print(heap)
15
```

⭐ 결과

⭐ [1, 3, 5, 4, 8, 7]

⭐ [1, 3, 2, 4, 8, 5, 7, 9]

⭐ [2, 3, 5, 4, 8, 9, 7]

⭐ [3, 4, 5, 7, 8, 9]

👉 삽입 : $O(\log n)$

👉 삭제 : $O(\log n)$

Max(Min) Heap - Python

👉 In Python 3.~

② PriorityQueue 클래스

- from queue import PriorityQueue
- 최소(Min) 힙 Default.
- Thread-safe
- heapq 모듈 통해 구현 되어 있음.
- 삽입 : put(*item*)
- 삭제 : get()

Max(Min) Heap - Python

👉 In Python 3.~

2 PriorityQueue 클래스

```
1 from queue import PriorityQueue
2
3 que = PriorityQueue()
4
5 que.put(4)
6 que.put(1)
7 que.put(7)
8 que.put(3)
9
10 print(que.get())
11 print(que.get())
12 print(que.get())
13 print(que.get())
```

⭐ 결과

⭐ 1
⭐ 3
⭐ 4
⭐ 7

Max(Min) Heap - Python

👉 In Python 3.~

2 PriorityQueue 클래스

```
1 from queue import PriorityQueue
2
3 nums = [4, 1, 7, 3, 8, 5]
4 que = PriorityQueue()
5
6 for num in nums:
7     que.put((-num, num)) # 우선 순위, 값
8
9 while not que.empty():
10    print(que.get()[1]) # index 1
```

⭐ 결과

- <tuple>을 이용해 Max Heap을 만들어 줄 수 있음.

⭐ 8
⭐ 7
⭐ 5
⭐ 4
⭐ 3
⭐ 1

Max(Min) Heap - Python

👉 In Python 3.~

③ Heapq vs PriorityQueue?

```
순차적인 숫자 1 ~ 1000 push 후 pop  
PriorityQueue : 2.880911350250244  
Heapq : 0.293992280960083
```

```
랜덤 숫자 1000번 push 후 1000번 pop 10회 평균  
PriorityQueue : 3.8655668020248415  
Heapq : 0.11064200401306153
```

<https://slowsure.tistory.com/130>

Time 10~30배 차이

✨ Thread-Safe를 요구하는 상황
-> PriorityQueue

✨ Thread-Non-Safe 해도 된다
-> heapq



- 코딩테스트로 문제를 푸는 상황.
- Thread-Safe를 요구하지 않음.
- **heapq 사용!**

Baekjoon - #11279

👉 #11279, ‘최대 힙’

⭐ 문제 조건



시간 제한	메모리 제한	정답 비율
1 초 (추가 시간 없음) (하단 참고)	256 MB	46.272%

⭐ 문제

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

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

프로그램은 처음에 비어있는 배열에서 시작하게 된다.

Baekjoon - #11279

👉 #11279, ‘최대 힘’

⭐ 입력

첫째 줄에 연산의 개수 $N(1 \leq N \leq 100,000)$ 이 주어진다. 다음 N 개의 줄에는 연산에 대한 정보를 나타내는 정수 x 가 주어진다. 만약 x 가 자연수라면 배열에 x 라는 값을 넣는(추가하는) 연산이고, x 가 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.

Baekjoon - #11279

⭐ 입력, 출력 예시

⭐ 입력 ⭐ 출력

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

```
1 #include <iostream>
2 #include <algorithm>
3 #include <vector>
4
5 using namespace std;
6
7 int main(){
8     ios::sync_with_stdio(false);
9     cin.tie(NULL);
10    cout.tie(NULL);
11
12    int N;
13    int input[100000];
14
15    vector<int> heap;
16
17    cin >> N;
18
19    for(int i = 0; i < N; i++){
20        cin >> input[i];
21    }
22
23    for(int i = 0; i < N; i++){
24        if(input[i] == 0){
25            if(heap.empty())
26                cout << 0 << "\n";
27            else {
28                cout << heap.front() << "\n";
29                pop_heap(heap.begin(), heap.end());
30                heap.pop_back();
31            }
32        }
33        else {
34            heap.push_back(input[i]);
35            make_heap(heap.begin(), heap.end());
36        }
37    }
38
39    return 0;
40 }
```

제출 번호	아이디	문제	문제 제목	결과	메모리	시간	언어	코드 길이	제출한 시간
45098841	euije	11279	최대 힙	시간 초과			C++17	827 B	3일 전



Baekjoon - #11279

👉 In C++

- **std::priority_queue**
- **<queue>**에 정의
- **Vector**를 인자로 받아 힙 구조로 변환 시켜 줌.
- 우선 순위 미지정 시 최대(Max) 힙 Default.
- push(), pop() 으로 삽입, 삭제 수행
- <https://docs.microsoft.com/ko-kr/cpp/standard-library/priority-queue-class?view=msvc-170>

Baekjoon - #11279

👉 In C++

```
1 #include <iostream>
2 #include <queue>
3
4 using namespace std;
5
6 int main(){
7     priority_queue<int> pq;
8
9     pq.push(5);
10    pq.push(3);
11    pq.push(6);
12    pq.push(1);
13    pq.push(2);
14
15    cout << pq.size() << endl;
16
17    while(!pq.empty()) {
18        int temp = pq.top();
19        cout << temp;
20        pq.pop();
21    }
22
23    return 0;
24 }
```

⭐ 결과

- **priority_queue**는 내부적으로 **vector**와 **make_heap**을 사용하고 있음.

⭐ 5

⭐ 6

⭐ 5

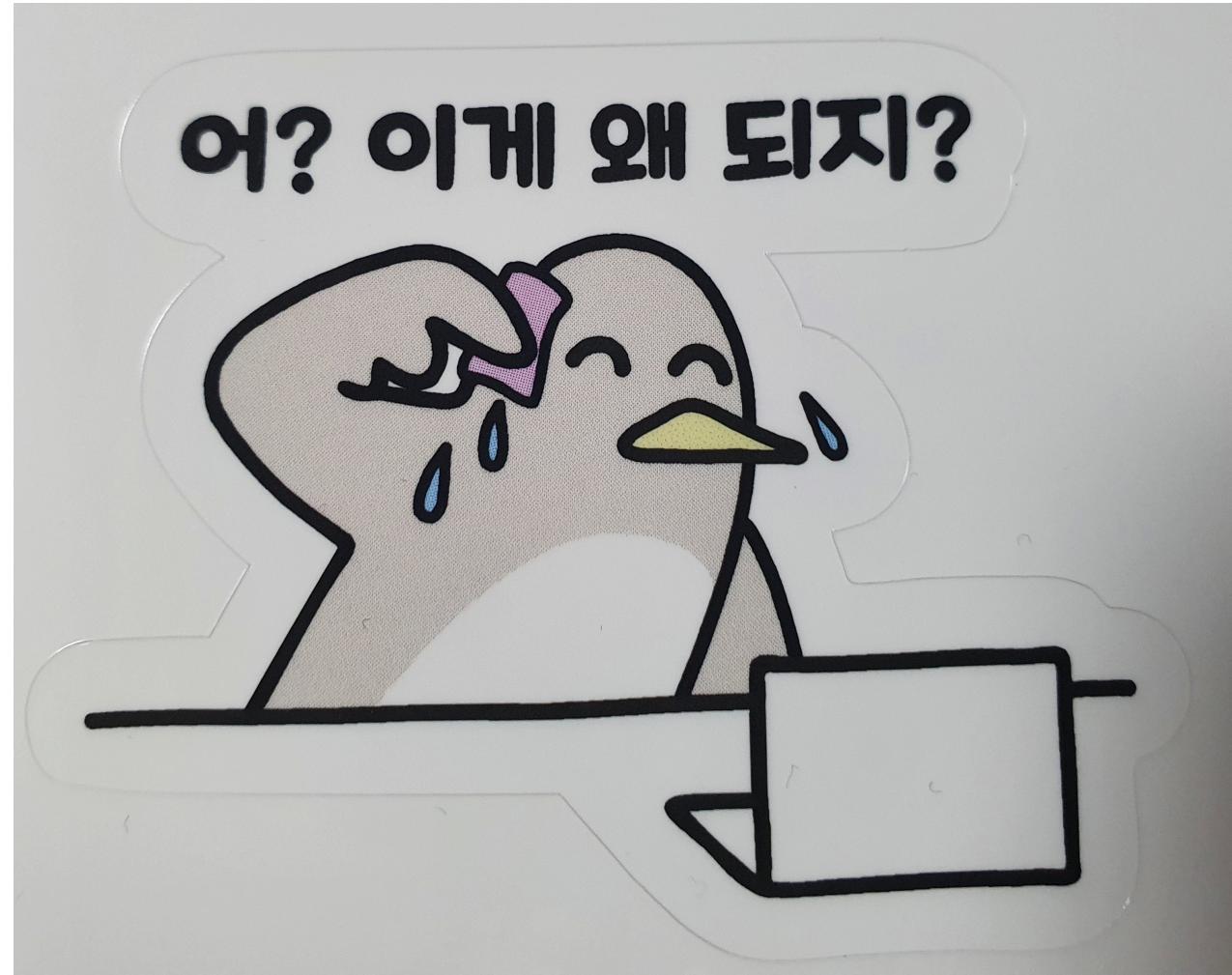
⭐ 3

⭐ 2

⭐ 1

```
1 #include <iostream>
2 #include <queue>
3
4 using namespace std;
5
6 int main(){
7     ios::sync_with_stdio(false);
8     cin.tie(NULL);
9     cout.tie(NULL);
10
11     int N;
12     int input[100000];
13
14     priority_queue<int> heap;
15
16     cin >> N;
17
18     for(int i = 0; i < N; i++){
19         cin >> input[i];
20     }
21
22     for(int i = 0; i < N; i++){
23         if(input[i] == 0){
24             if(heap.empty())
25                 cout << 0 << "\n";
26             else {
27                 cout << heap.top() << "\n";
28                 heap.pop();
29             }
30         }
31         else {
32             heap.push(input[i]);
33         }
34     }
35
36     return 0;
37 }
```

45100971	euije	11279	맞았습니다!!	3184 KB	16 ms	C++17 / 수정	672 B	3일 전
----------	-------	-------	---------	---------	-------	------------	-------	------



Baekjoon - #11286

👉 #11286, ‘절댓값 힙’

⭐ 문제 조건



	시간 제한	메모리 제한	정답 비율
1 초 (추가 시간 없음) (하단 참고)	256 MB	57.151%	

⭐ 문제

절댓값 힙은 다음과 같은 연산을 지원하는 자료구조이다.

1. 배열에 정수 x ($x \neq 0$)를 넣는다.
2. 배열에서 절댓값이 가장 작은 값을 출력하고, 그 값을 배열에서 제거한다. 절댓값이 가장 작은 값이 여러개일 때는, 가장 작은 수를 출력하고, 그 값을 배열에서 제거한다.

프로그램은 처음에 비어있는 배열에서 시작하게 된다.

Baekjoon - #11286

👉 #11286, ‘절댓값 합’

⭐️ 입력

첫째 줄에 연산의 개수 $N(1 \leq N \leq 100,000)$ 이 주어진다. 다음 N 개의 줄에는 연산에 대한 정보를 나타내는 정수 x 가 주어진다. 만약 x 가 0이 아니라면 배열에 x 라는 값을 넣는(추가하는) 연산이고, x 가 0이라면 배열에서 절댓값이 가장 작은 값을 출력하고 그 값을 배열에서 제거하는 경우이다. 입력되는 정수는 -2^{31} 보다 크고, 2^{31} 보다 작다.

⭐️ 출력

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

Baekjoon - #11286

⭐ 입력, 출력 예시

⭐ 입력 ⭐ 출력

18	-1
1	1
-1	0
0	-1
0	-1
0	1
1	1
1	-2
-1	2
-1 0	0
2 0	
-2 0	
0 0	
0 0	

고생하셨어요

