

알튜비튜

우선순위 큐

오늘은 STL에서 제공하는 container adaptor인 priority queue에 대해 알아봅니다.
가장 최근의 데이터를 뽑는 스택, 제일 먼저 들어갔던 데이터를 뽑는 큐와 달리 우선순위가 가장 높은 데이터를 뽑는 자료구조입니다.

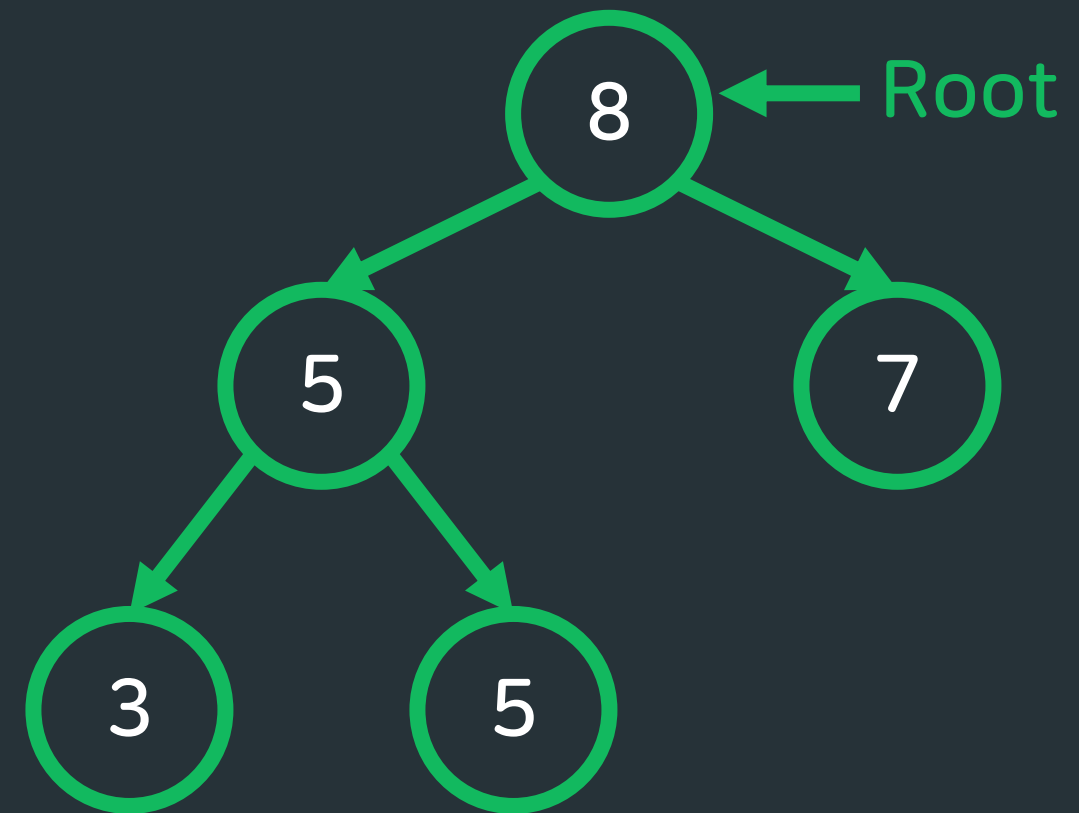


FAMILY
EMERGENCY ROOM



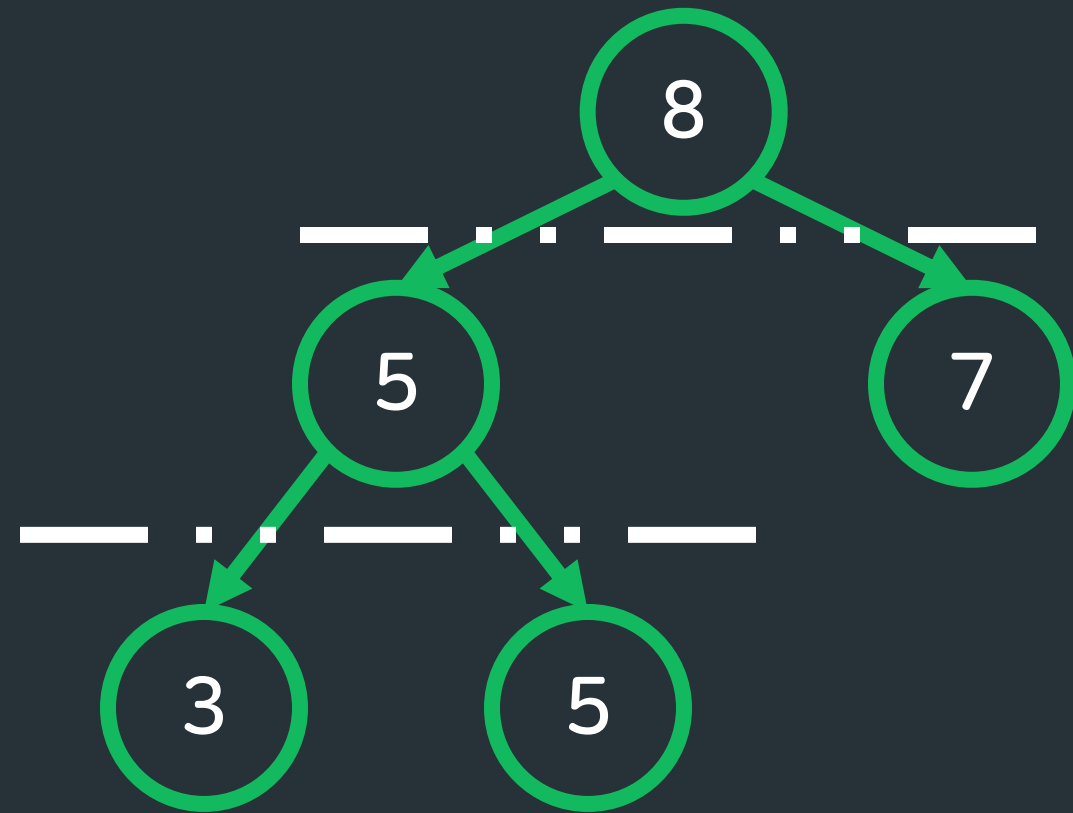
Priority Queue

- 우선순위가 높은 데이터가 먼저 나옴
- 자료의 Root 노드에서만 모든 연산이 이루어짐
- 모든 연산에 대한 시간 복잡도는 $O(\log n)$
- Heap으로 구현
- Heap의 조건
 1. 완전 이진 트리
 2. 상위 노드의 값은 모든 하위 노드의 값보다 우선순위가 크거나 같다



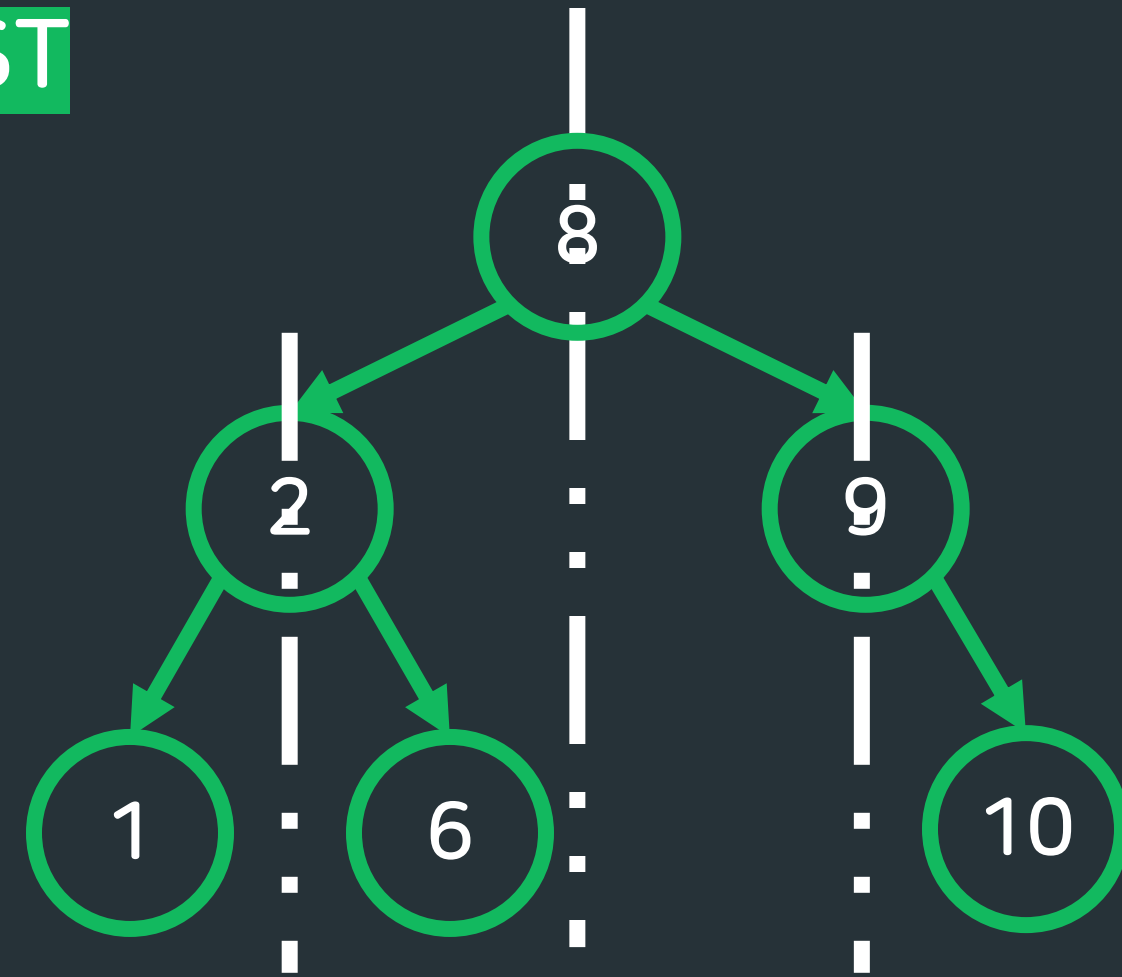
Heap과 BST의 차이

Heap



(상위) \geq (하위)
상하관계

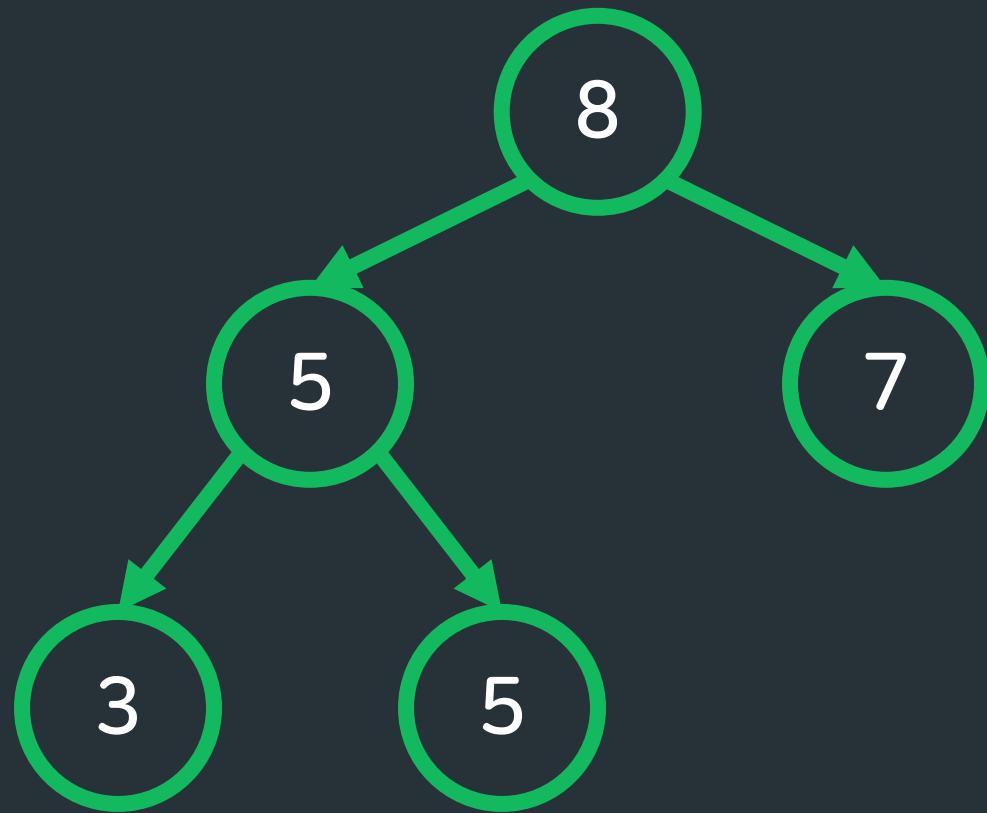
BST



(왼쪽) $<$ (루트) $<$ (오른쪽)
좌우관계

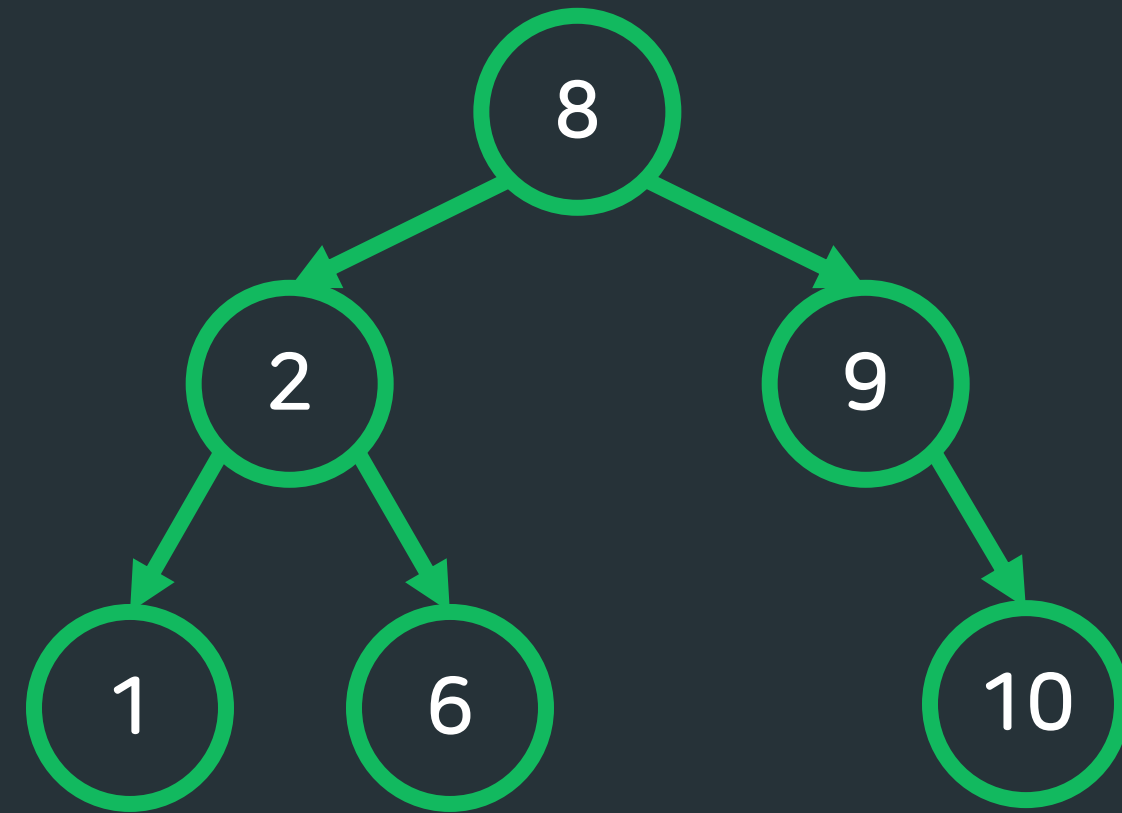
Heap과 BST의 차이

Heap

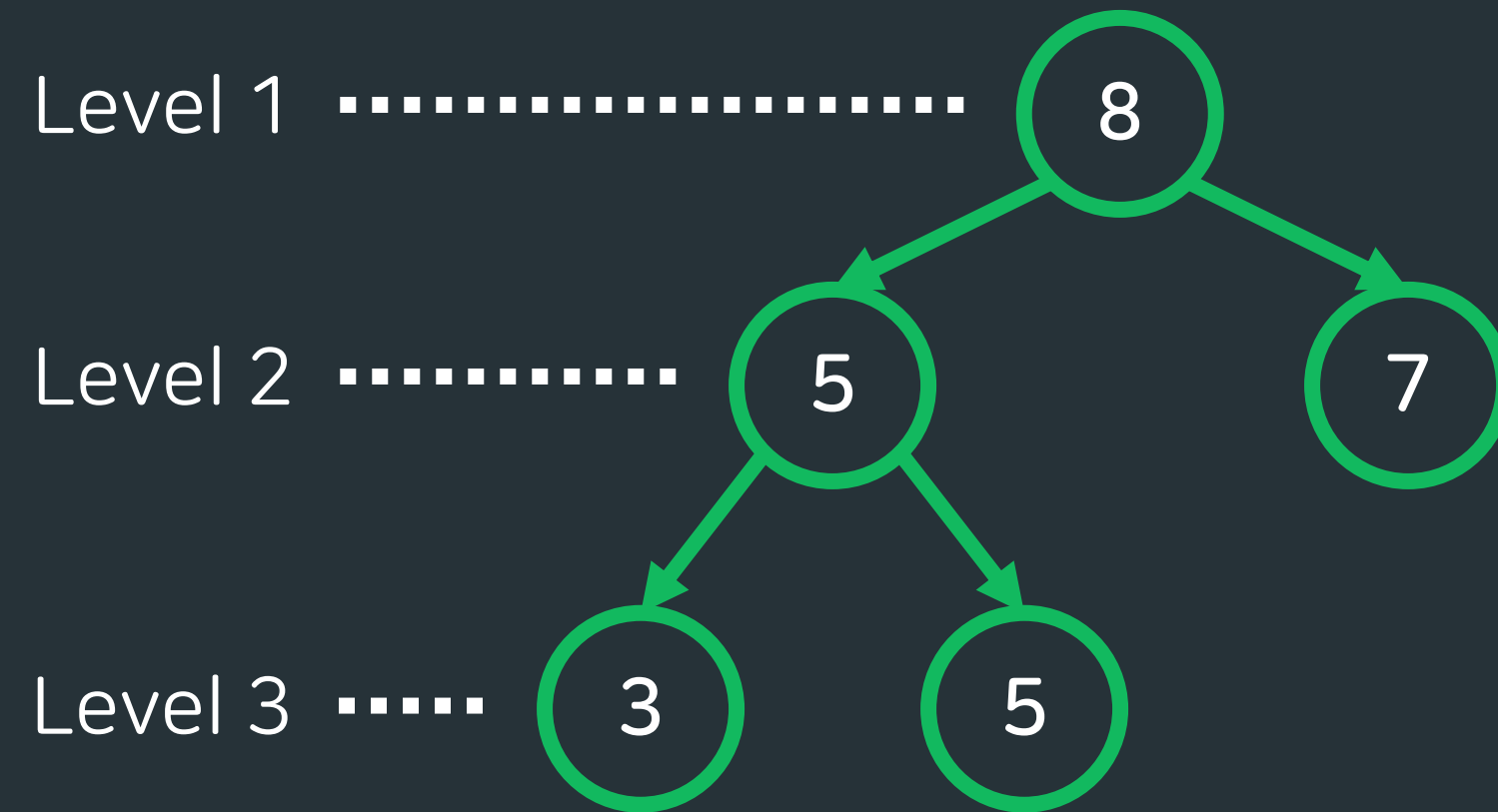


중복 0
완전 이진 트리

BST



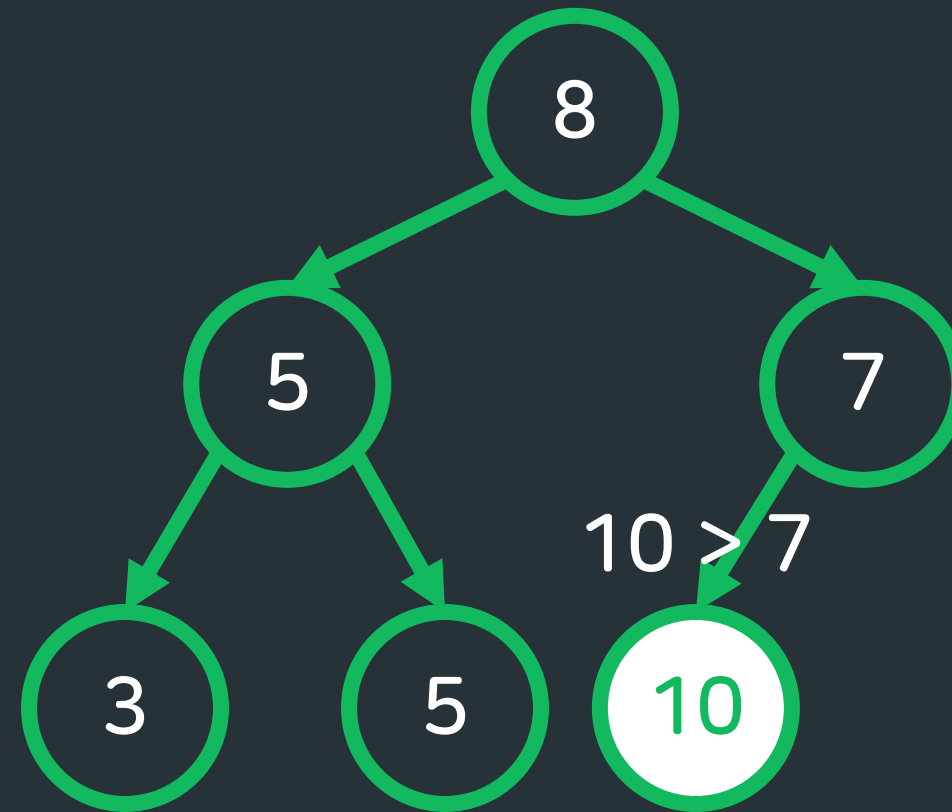
중복 X
완전 이진 트리일 필요 없음



Complete Binary Tree

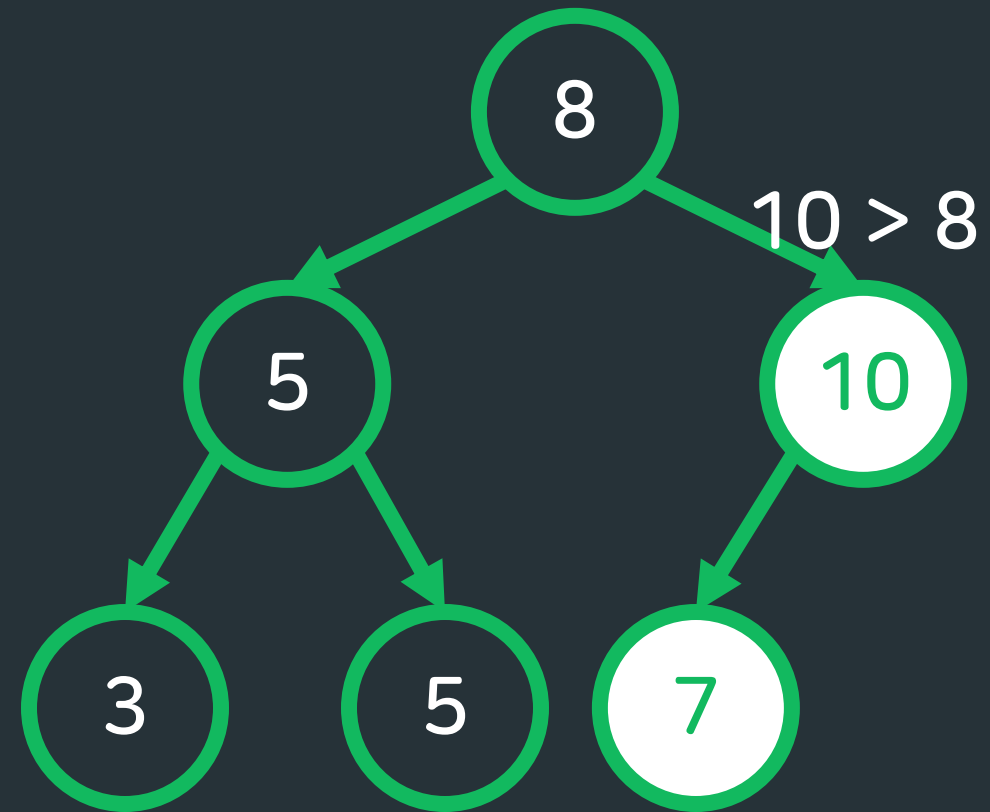
- 마지막 레벨을 제외하고 모든 레벨을 다 채움
- 마지막 레벨의 모든 노드는 왼쪽부터 빈 공간 없이 채움

최대 힙에 데이터 삽입



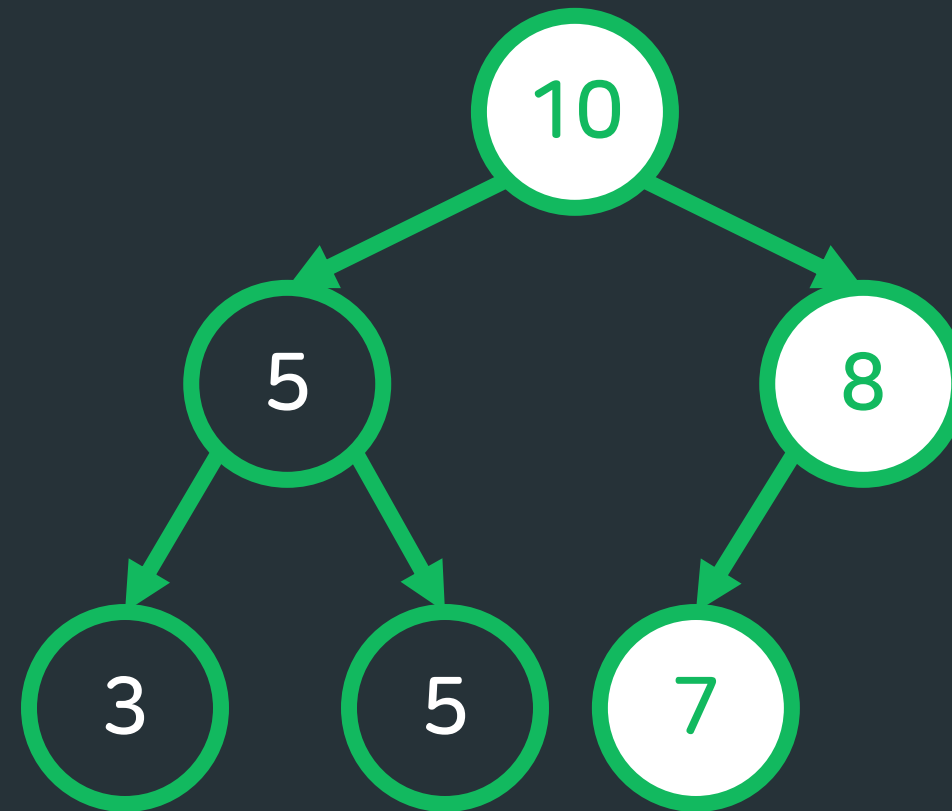
key = 10

최대 힙에 데이터 삽입



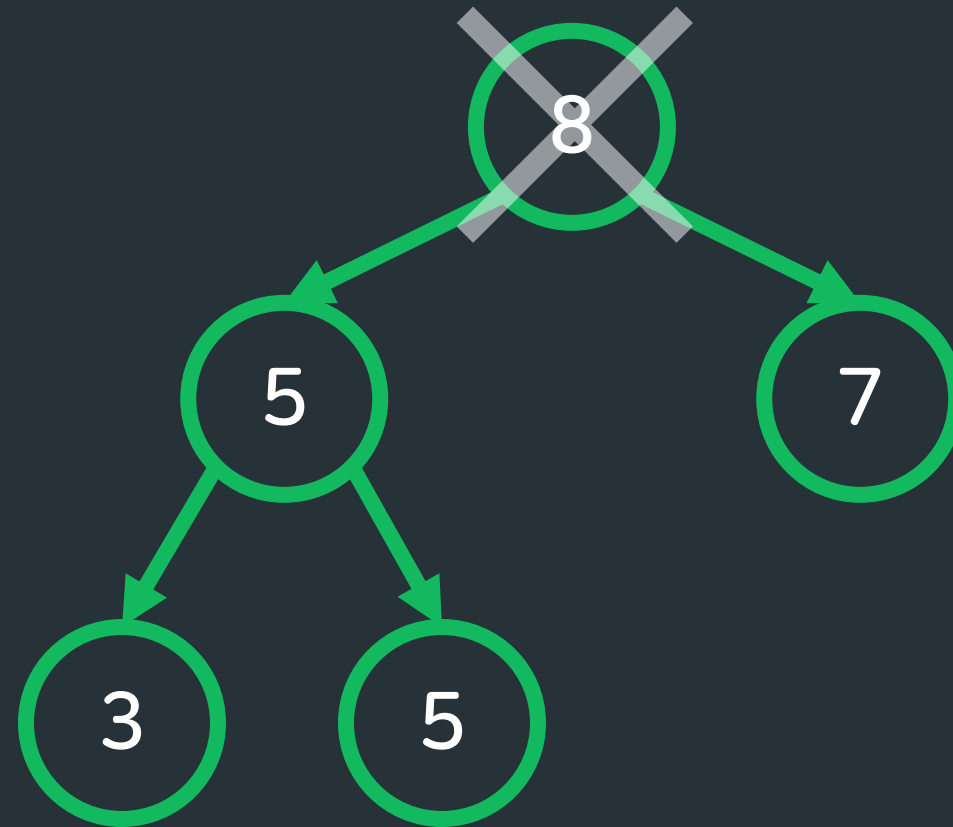
key = 10

최대 힙에 데이터 삽입

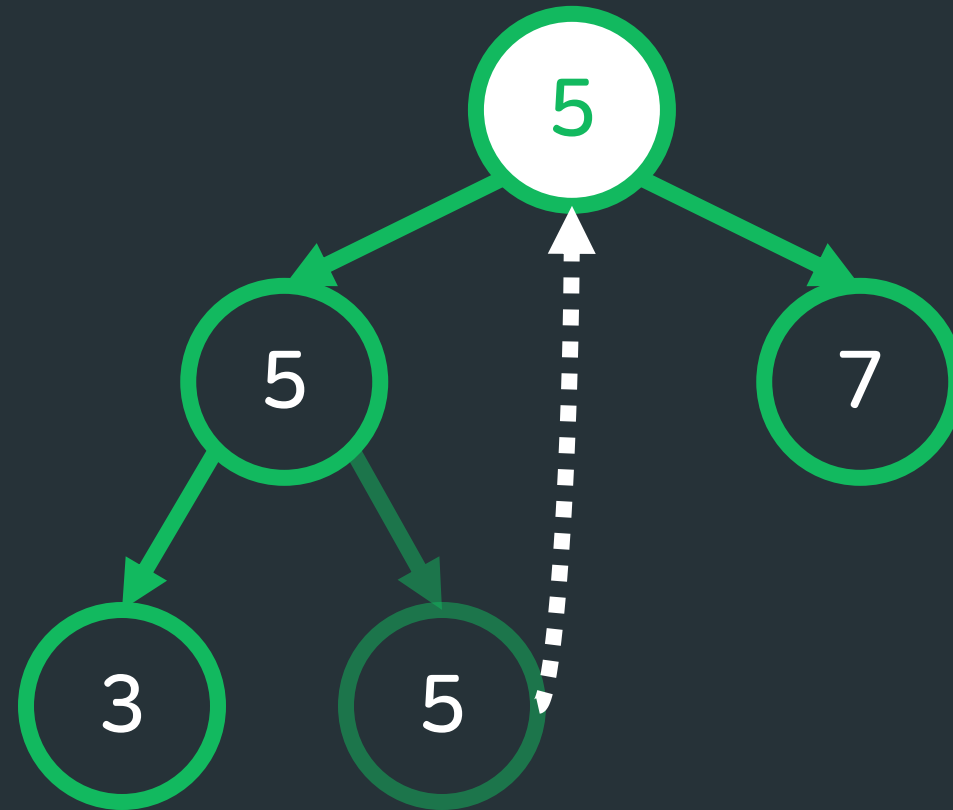


key = 10

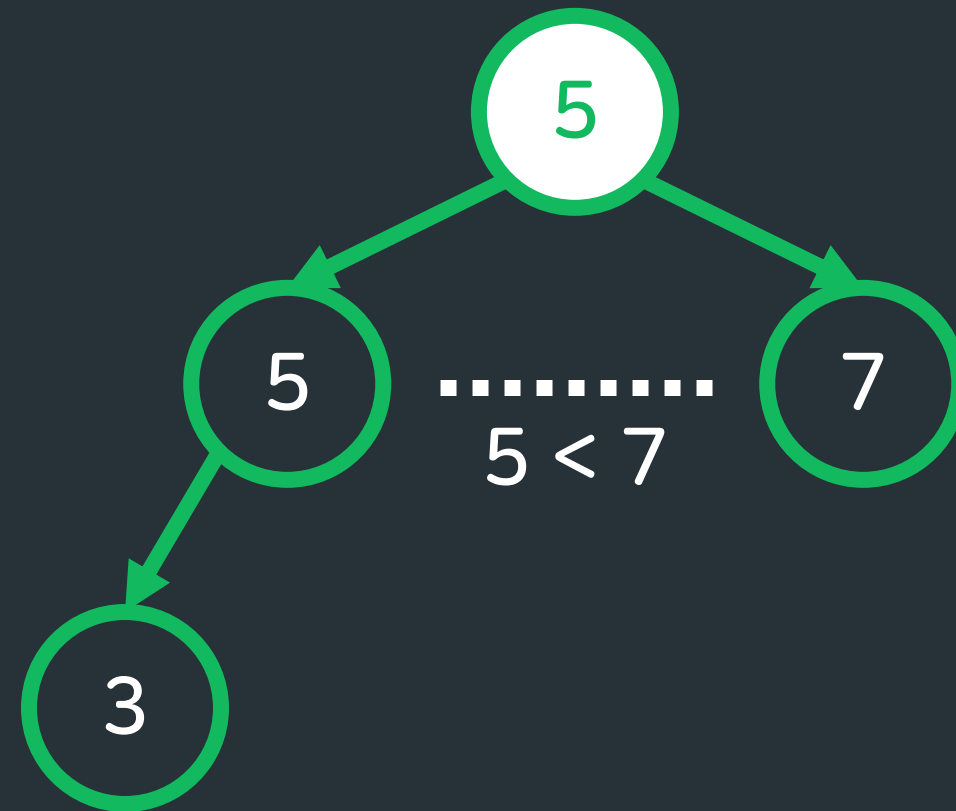
최대 힙에서 데이터 삭제



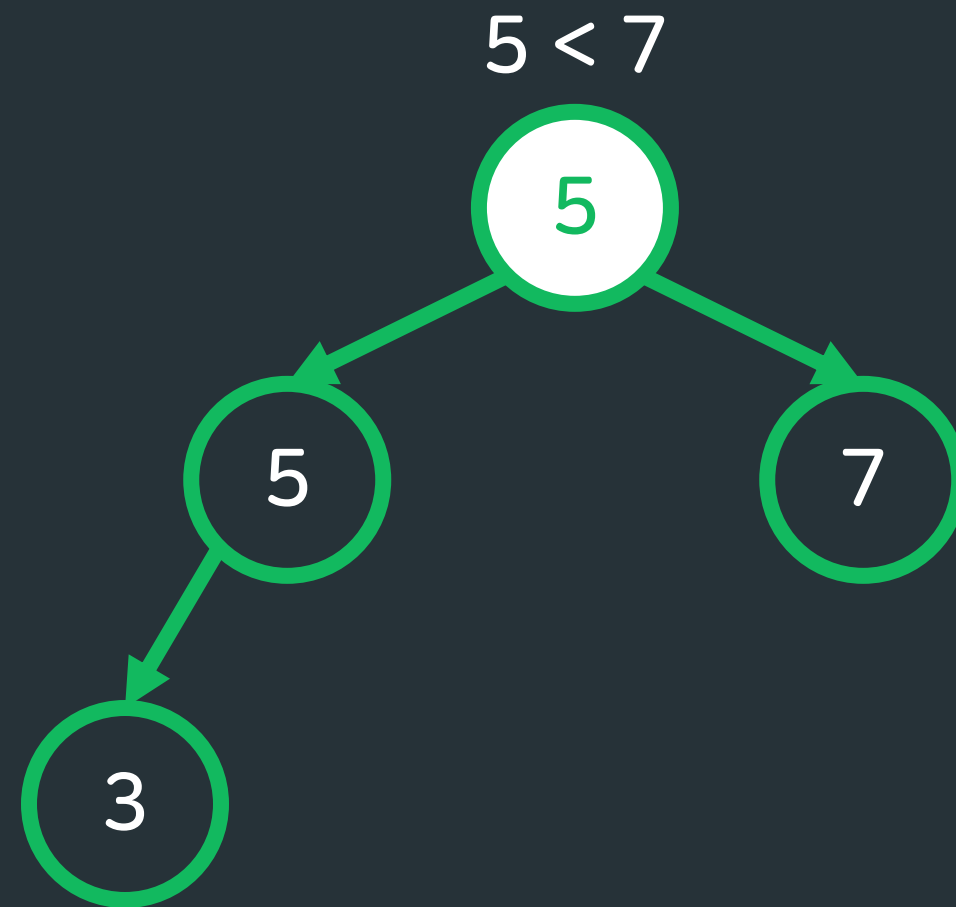
최대 힙에서 데이터 삭제



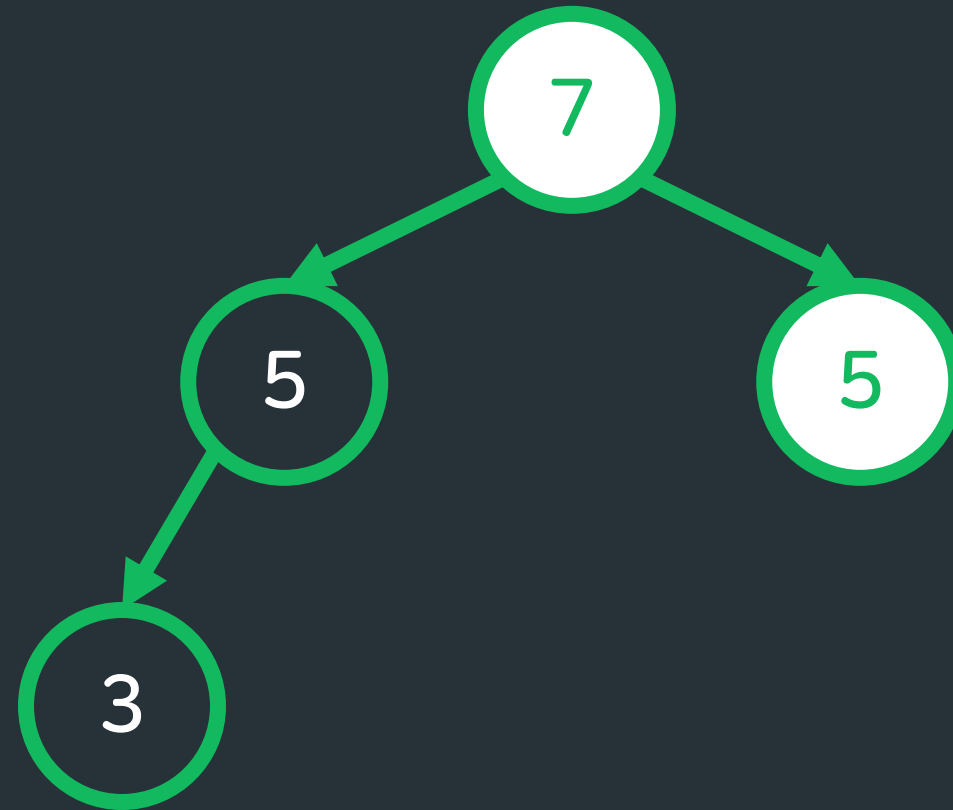
최대 힙에서 데이터 삭제



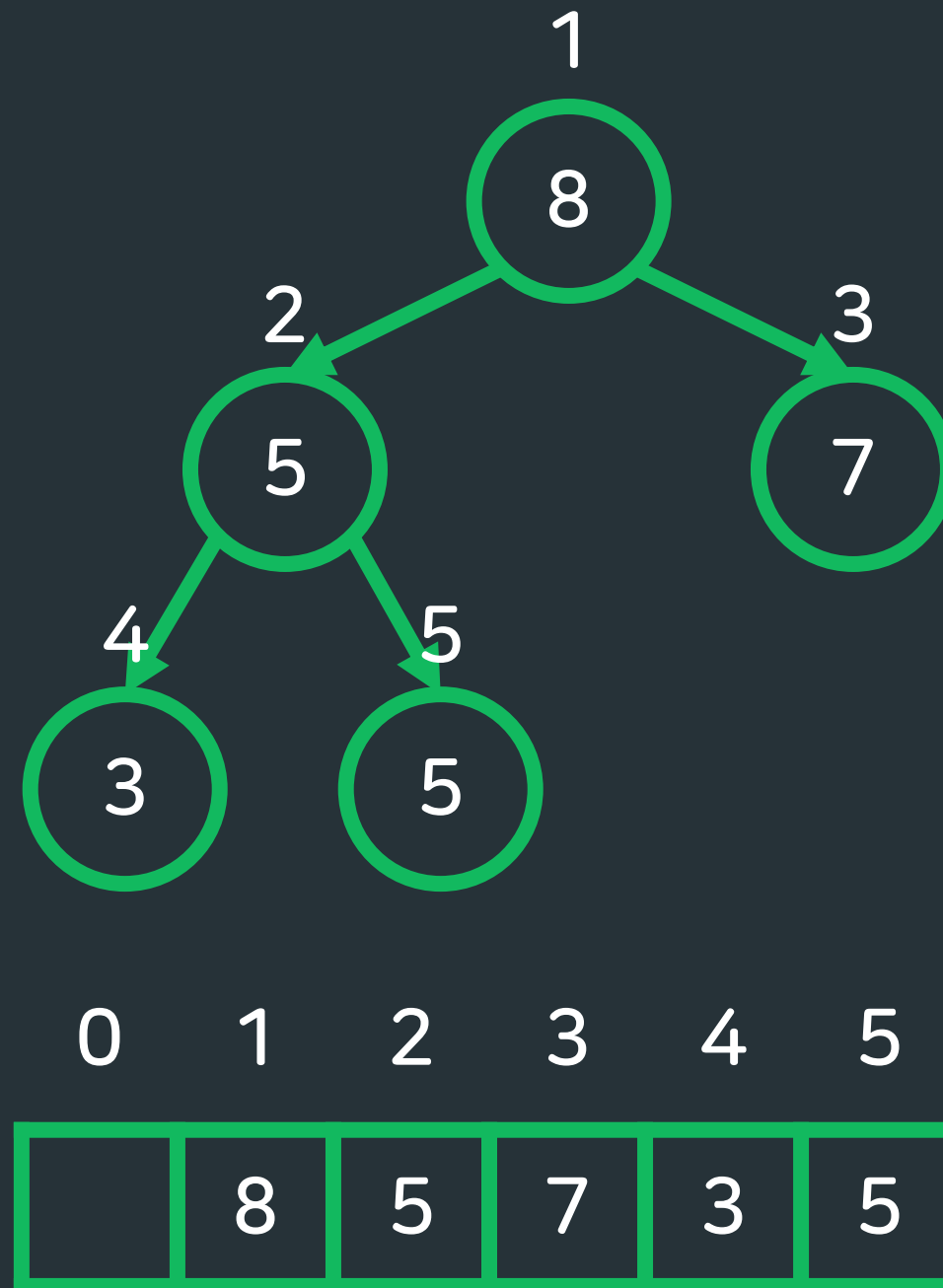
최대 힙에서 데이터 삭제



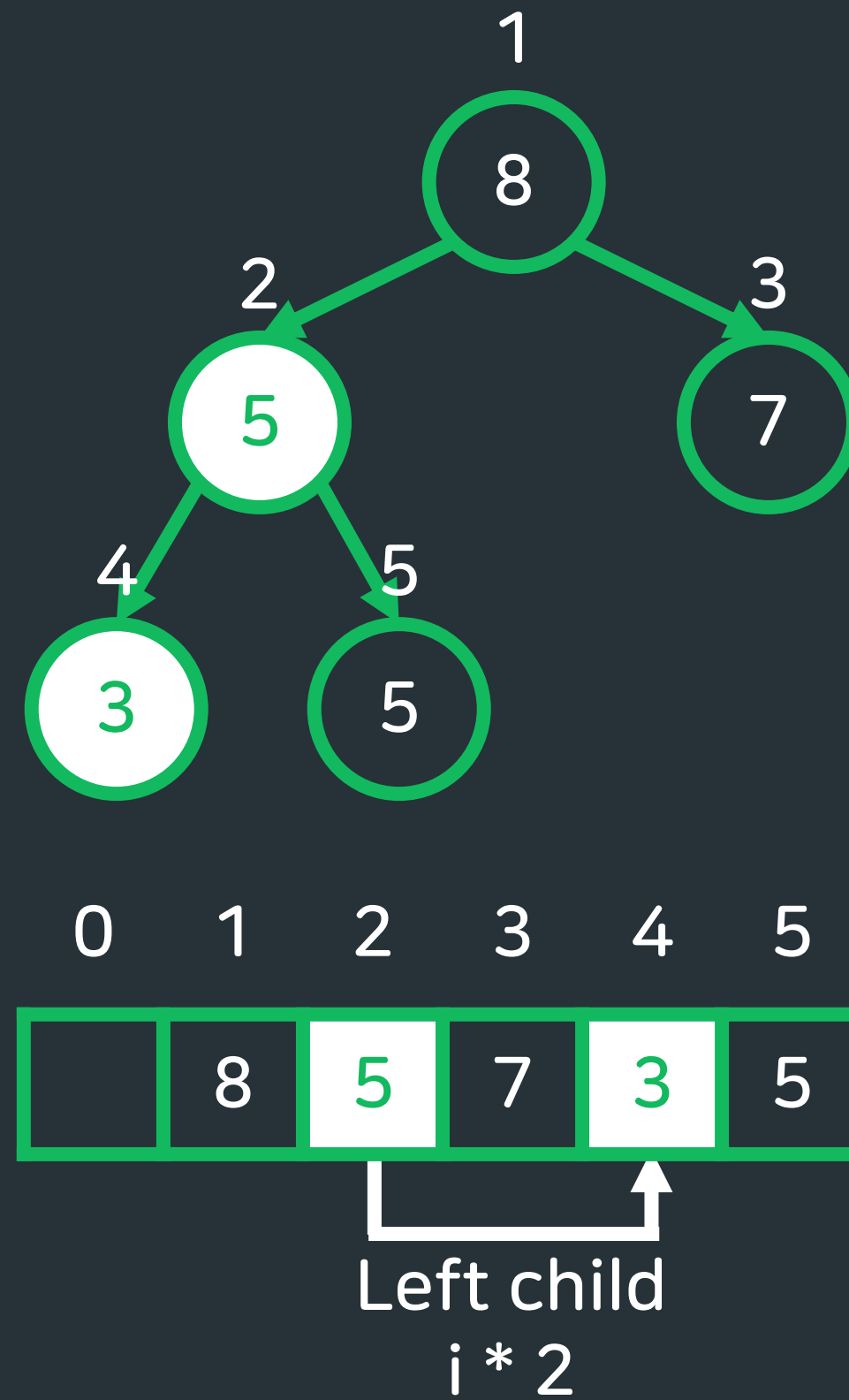
최대 힙에서 데이터 삭제



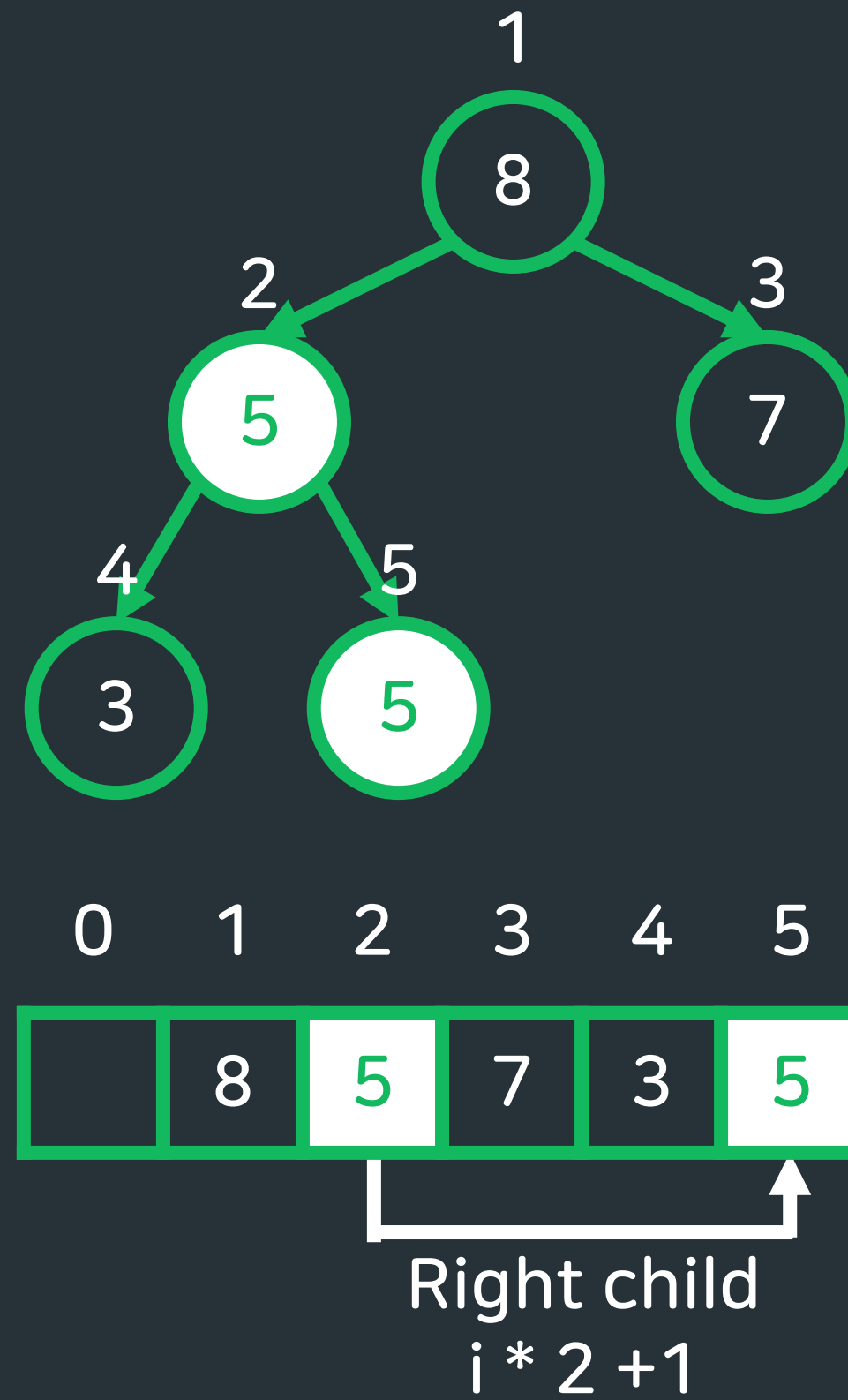
배열로 힙 구현하기



배열로 힙 구현하기



배열로 힙 구현하기



/<> 11279번 : 최대 힙 - Silver 2

문제

- 다음의 명령을 처리하는 **최대 힙** 프로그램 만들기
 1. 정수 x 가 주어진다.
 2. x 가 **자연수**라면 최대 힙에 x **추가**
 3. x 가 **0**이라면 최대 힙에서 가장 큰 값을 출력하고 **제거**. 최대 힙이 비었다면 0 출력

제한 사항

- 명령의 수 N 의 범위는 $1 \leq N \leq 100,000$
- 명령과 함께 주어지는 정수 x 의 범위는 $0 \leq x \leq 2^{31}$

예제 입력

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

예제 출력


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

예제 입력

```
5
12 7 9 15 5
13 8 11 19 6
21 10 26 31 16
48 14 28 35 25
52 20 32 41 49
```

예제 출력

```
35
```



Search:

[Reference](#)
[<queue>](#)
[priority_queue](#)

[register](#)
[log in](#)

C++

[Information](#)
[Tutorials](#)
[Reference](#)
[Articles](#)
[Forum](#)

Reference

C library:

Containers:

<array>

<deque>

<forward_list>

<list>

<map>

<queue>

<set>

<stack>

<unordered_map>

<unordered_set>

<vector>

Input/Output:

Multi-threading:

Other:

<queue>

priority_queue

queue

priority_queue

priority_queue::priority_queue

member functions:

priority_queue::emplace

priority_queue::empty

priority_queue::pop

You were redirected to [cplusplus.com/priority_queue](#) || See search results for: "**priority_queue**"

class template

std::priority_queue

```
template <class T, class Container = vector<T>,
          class Compare = less<typename Container::value_type> > class priority_queue;
```

Priority queue

Priority queues are a type of container adaptors, specifically designed such that its first element is always the greatest of the elements it contains, according to some *strict weak ordering* criterion.

This context is similar to a *heap*, where elements can be inserted at any moment, and only the *max heap* element can be retrieved (the one at the top in the *priority queue*).

Priority queues are implemented as *container adaptors*, which are classes that use an encapsulated object of a specific container class as its *underlying container*, providing a specific set of member functions to access its elements. Elements are *popped* from the "back" of the specific container, which is known as the *top* of the priority queue.

The underlying container may be any of the standard container class templates or some other specifically designed container class. The container shall be accessible through *random access iterators* and support the following operations:

- empty()
- size()
- front()
- push_back()
- pop_back()

The standard container classes `vector` and `deque` fulfill these requirements. By default, if no container class is specified for a particular `priority_queue` class instantiation, the standard container `vector` is used.

Support of *random access iterators* is required to keep a heap structure internally at all times. This is done automatically by the container adaptor by automatically calling the algorithm functions `make_heap`, `push_heap` and `pop_heap` when needed.

/<> 2075번 : N번째 큰 수 - Gold 5

문제

- $N \times N$ 크기의 표에서 N번째 큰 수를 찾아라
- 표 안의 모든 수는 자신의 한 칸 위에 있는 수보다 크다.

제한 사항

- N의 범위 $1 \leq N \leq 1,500$
- 표에 적힌 수 k의 범위 $-1,000,000,000 \leq k \leq 1,000,000,000$
- 메모리 제한 : 12MB

Hint

1. 도움 안되는 **조건**이 하나 있어요
2. 모든 정보를 다 담기엔 **메모리**가 부족해요

자신의 한 칸 위에 있는 수보다 크다?

12	7	9	15	5
13	8	11	19	6
21	10	26	31	16
48	14	28	35	25
52	20	32	41	49

좌우 관계를 모르기 때문에 쓸데없는 정보
(최대 힙에서 현재의 최솟값을 찾을 수 없는 것과 같음)

자신의 한 칸 위에 있는 수보다 크다?

12	7	9	15	5
13	8	11	19	6
21	10	26	31	16
48	14	28	35	25
52	20	32	41	49

좌우 관계를 모르기 때문에 쓸데없는 정보
(최대 힙에서 현재의 최솟값을 찾을 수 없는 것과 같음)

-> 그냥 우선순위 큐에 다 넣어버리자!

- int형 변수 하나의 크기 = 4 바이트
- 1024 바이트 = 1 KB
- 1024 KB = 1 MB
- 12MB에 담을 수 있는 int형 변수의 개수 = $12 * 1024 * 1024 / 4 = 3,145,728$ 개
- N이 최대일 때 필요한 int형 변수의 개수 = $1,500 * 1,500 = 2,250,000$ 개

-> 여기까지만 계산하면 메모리가 충분해 보이지만... 헤더, 라이브러리 등등의 메모리로 12MB 초과

최대 힙이 아니라 최소 힙

~~최대 힙에 다 넣은 뒤 N번 pop하자!~~
최소 힙의 크기를 N으로 유지하며 입력을 처리하자!

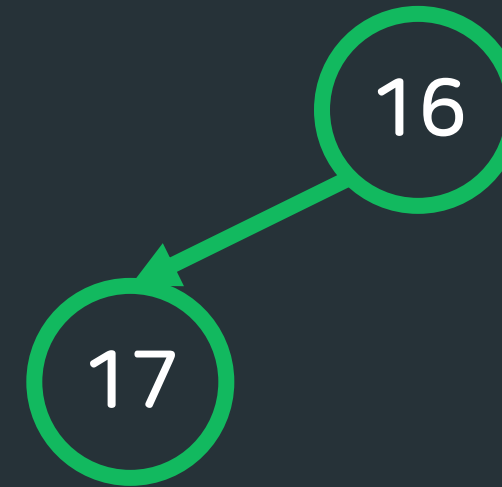
N = 2일 때

17	16
12	20

17

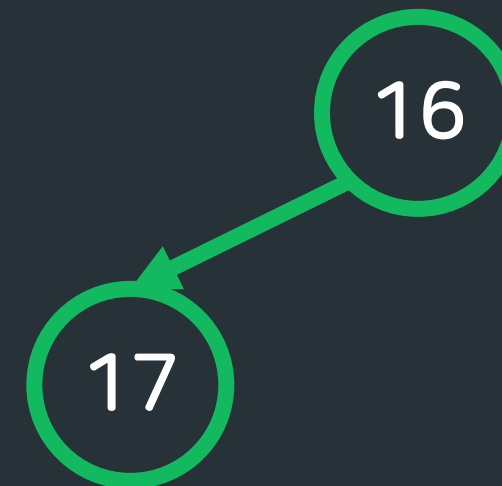
N = 2일 때

17	16
12	20



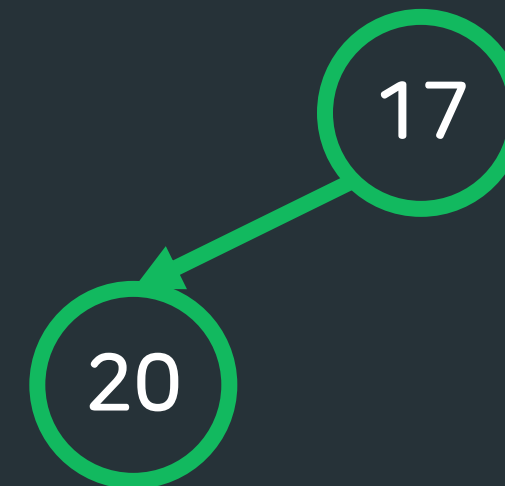
N = 2일 때

17	16
12	20



N = 2일 때

17	16
12	20



정리

- 우선순위 큐는 힙으로 구현하고, 시간 복잡도가 $O(\log n)$ 인 자료구조
- 효율성을 보는 문제에 사용되는 경우가 많음
- 그리디, 최단 경로 알고리즘 풀이에 활용되기도 함
- comp 정의할 때는 헛갈리지 말기! priority queue는 comp가 true를 반환해야 swap됨! (sort와 반대)
- 무한 루프 (pop을 하지 않음), 런타임 에러 (empty 체크 안하고 조회 or 삭제 시도) 조심!!

이것도 알아보세요!

- 힙을 배열로 구현할 때 왜 인덱스를 1부터 시작했을까요? 0부터 시작한다면 어떻게 될까요?

필수

- /<> 11723번 : 집합 - Silver 5
- /<> 3613번 : java vs C++ - Silver 5

3문제 이상 선택

- /<> 2493번 : 탑 - Gold 5
- /<> 7662번 : 이중 우선순위 큐 - Gold 5
- /<> 11000번 : 강의실 배정 - Gold 5
- /<> 11286번 : 절댓값 힙 - Silver 1
- /<> 12018번 : Yonsei TOTO - Silver 3
- /<> 15903번 : 카드 합체 놀이 - Silver 2