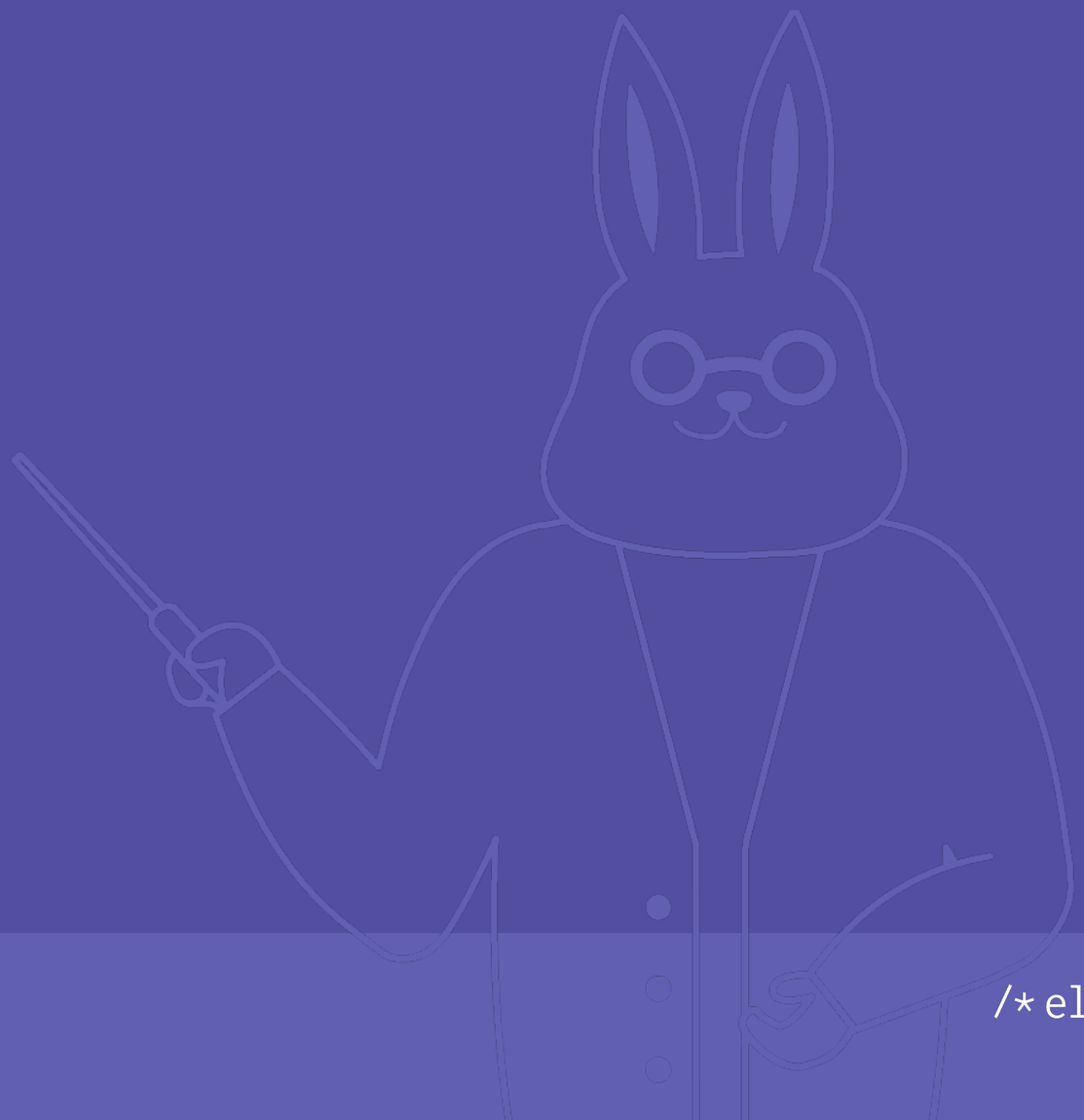
자료구조

4장 우선순위 큐와 힙

김경민 선생님



Contents

01

우선순위 큐의 구현 방법과 힙

01 우선순위 큐의 구현 방법과 힙

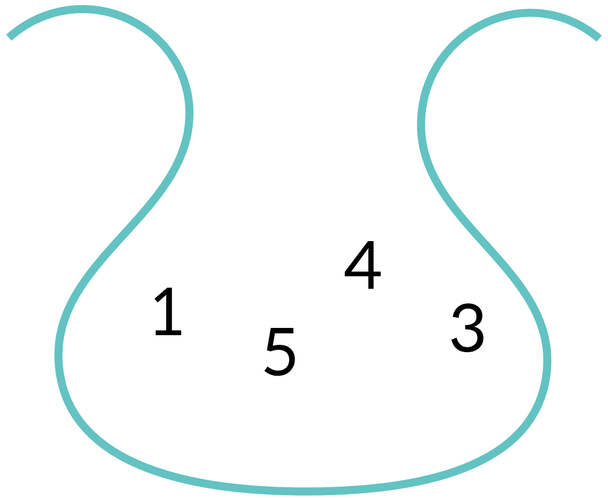
우선순위 큐란?

우선순위가 높은 **원소**가 **먼저 출력**되는 추상적 자료형

01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

출력 연산 시 가장 **우선순위가 높은** 원소를 출력한다.

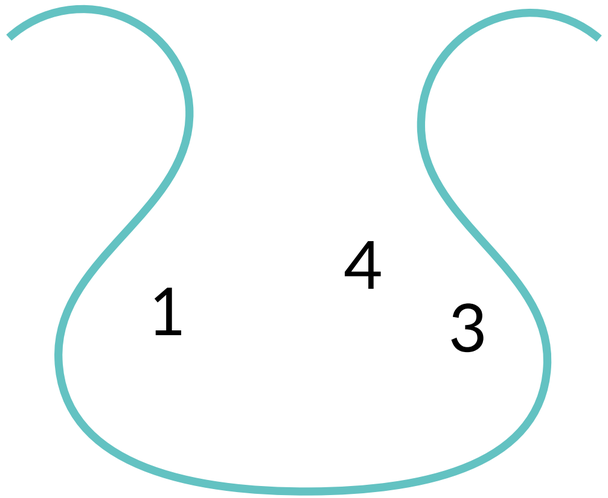


우선순위 큐

01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

출력 연산 시 가장 **우선순위가 높은** 원소를 출력한다.

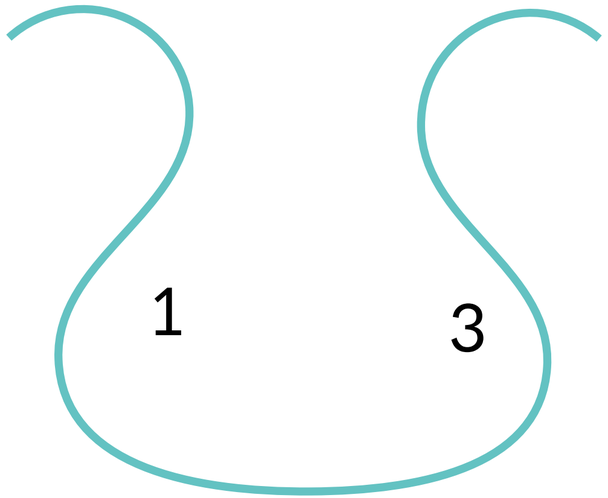


우선순위 큐

01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

출력 연산 시 가장 **우선순위가 높은** 원소를 출력한다.

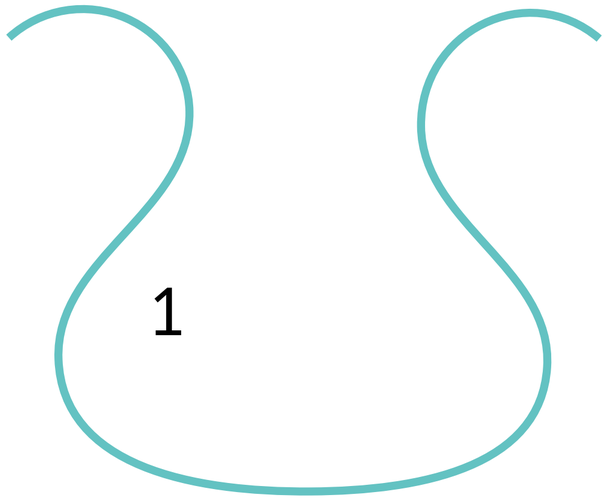


우선순위 큐

01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

출력 연산 시 가장 **우선순위가 높은** 원소를 출력한다.



우선순위 큐

01 우선순위 큐의 구현 방법과 힙

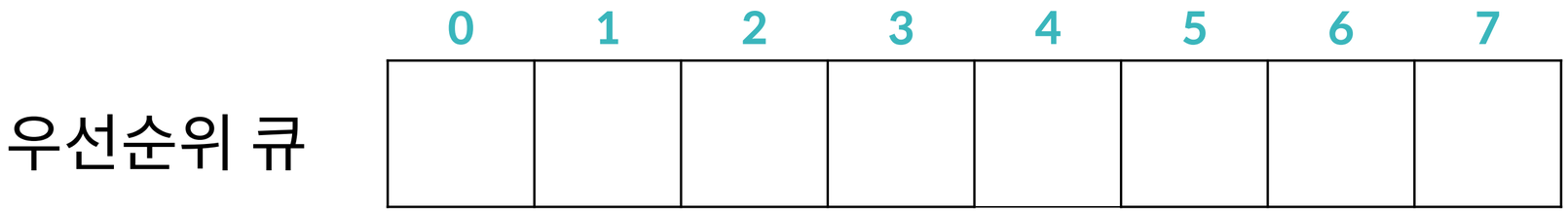
우선순위 큐란?

우선순위 큐를 배열로 단순하게 구현해보자.

01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

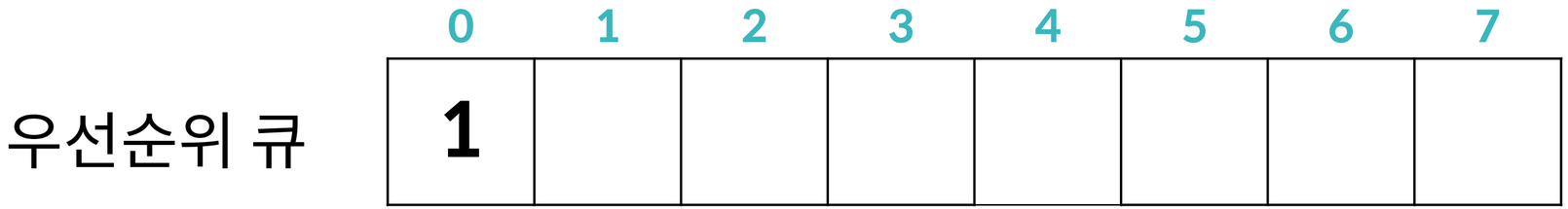
단순하게 생각해보기



01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

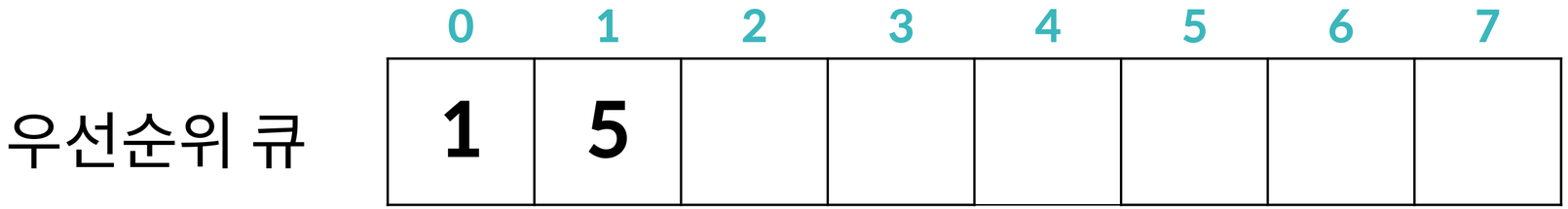
단순하게 생각해보기



01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

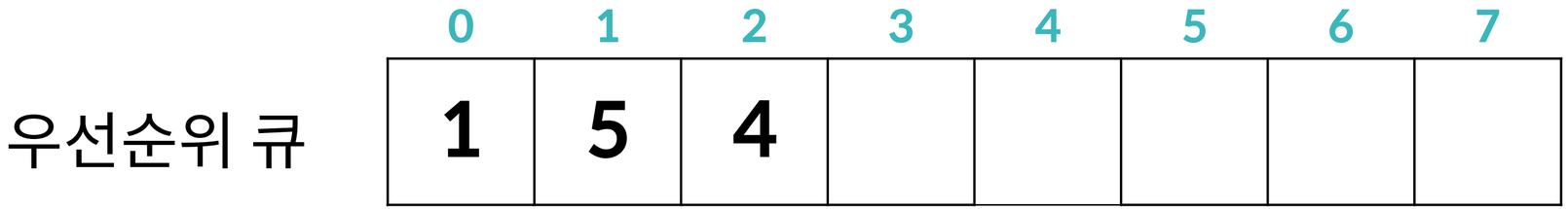
단순하게 생각해보기



01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

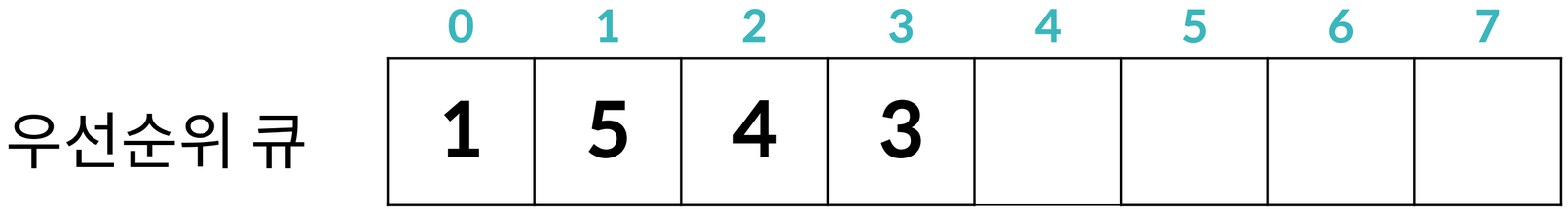
단순하게 생각해보기



01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

단순하게 생각해보기

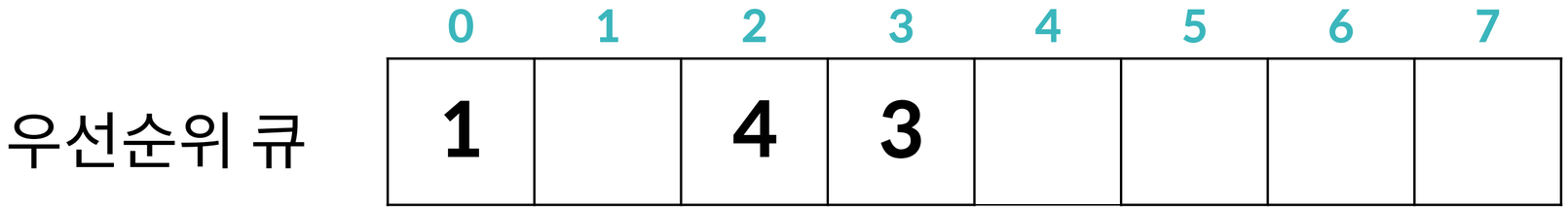


01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

단순하게 생각해보기

출력값 : 5

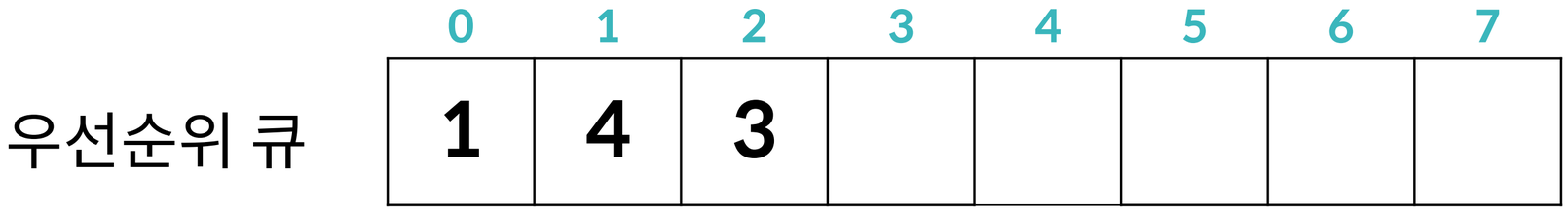


01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

단순하게 생각해보기

출력값 : 5

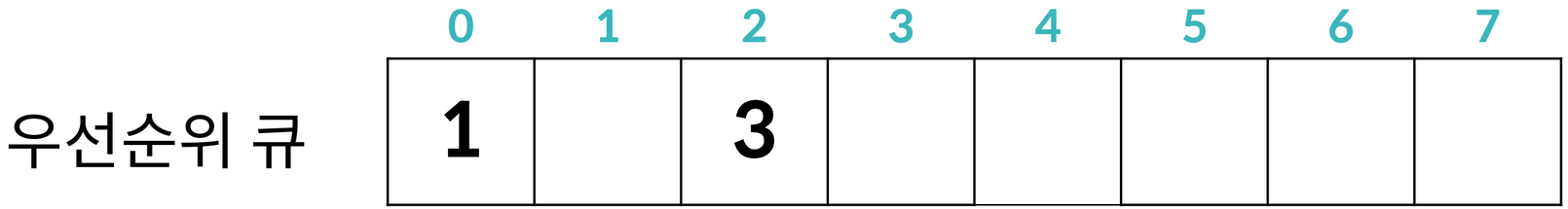


01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

단순하게 생각해보기

출력값 : 5 4

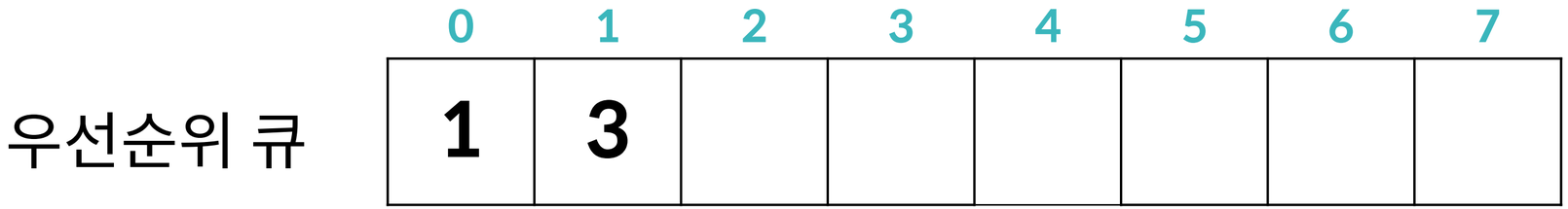


01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

단순하게 생각해보기

출력값 : 5 4

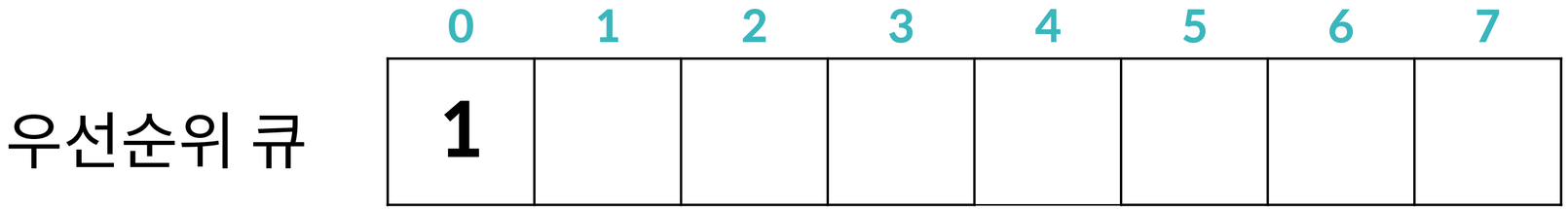


01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

단순하게 생각해보기

출력값 : 5 4 3

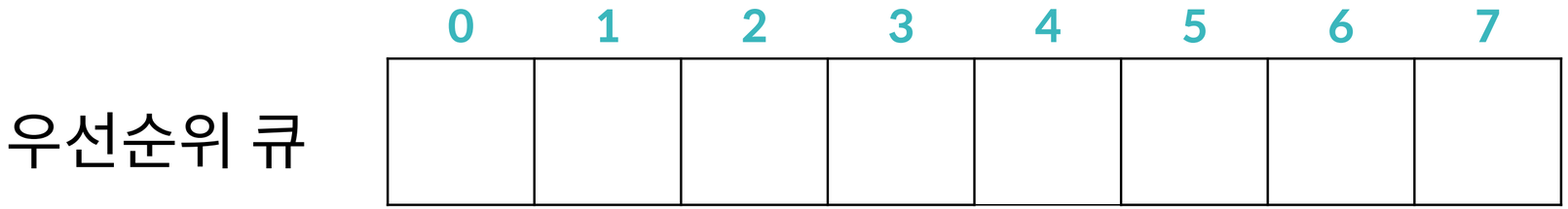


01 우선순위 큐의 구현 방법과 힙

우선순위 큐란?

단순하게 생각해보기

출력값 : 5 4 3 1



우선순위 큐란?

우선순위 큐를 단순한 접근 방법으로 구현한 경우

입력 : $O(1)$

출력 : $O(n)$

우선순위 큐란?

우선순위가 가장 높은 원소를 **찾는** 과정과,

그 원소를 **제거**하는 것이 비효율적이다.

더 나은 방법은 없을까?

힙

최솟값 또는 **최댓값**을 빠르게 찾기 위해 고안된 **완전 이진 트리**

최대 힙(Max Heap)

부모 노드는 항상 자식 노드보다 **큰** 값을 갖고 있다.

최소 힙(Min Heap)

부모 노드는 항상 자식 노드보다 **작은** 값을 갖고 있다.

힙

```
import heapq
```

파이썬의 heapq 모듈로 **최소 힙**을 사용할 수 있다.

(강의자료에서도 최소 힙으로만 설명하겠습니다.)

힙

최대 힙이 필요한 경우에는
값을 저장할 때 **-1을 곱한 값**을 저장하면 된다.

-1을 곱함으로서 최댓값과 최솟값이 **반전**된다는 점을 이용한다.

단, 이 방법은 힙이 저장하는 값이 **수(number)**일 때만 유효하다.

힙

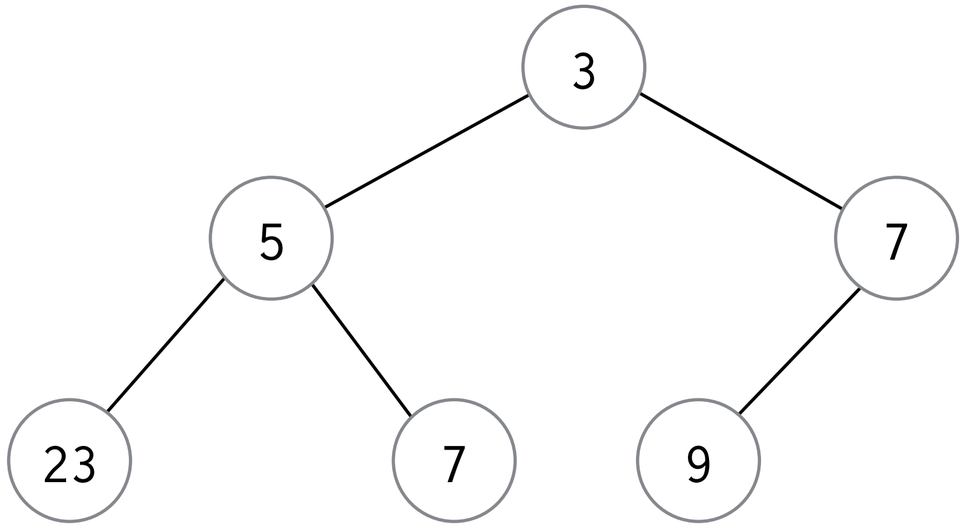
힙에 자료를 입력, 출력할 때 어떻게 연산이 이루어지는지 알아보자.

힙 : 자료 입력하기

힙은 **완전 이진 트리**의 특성을 유지해야 한다.

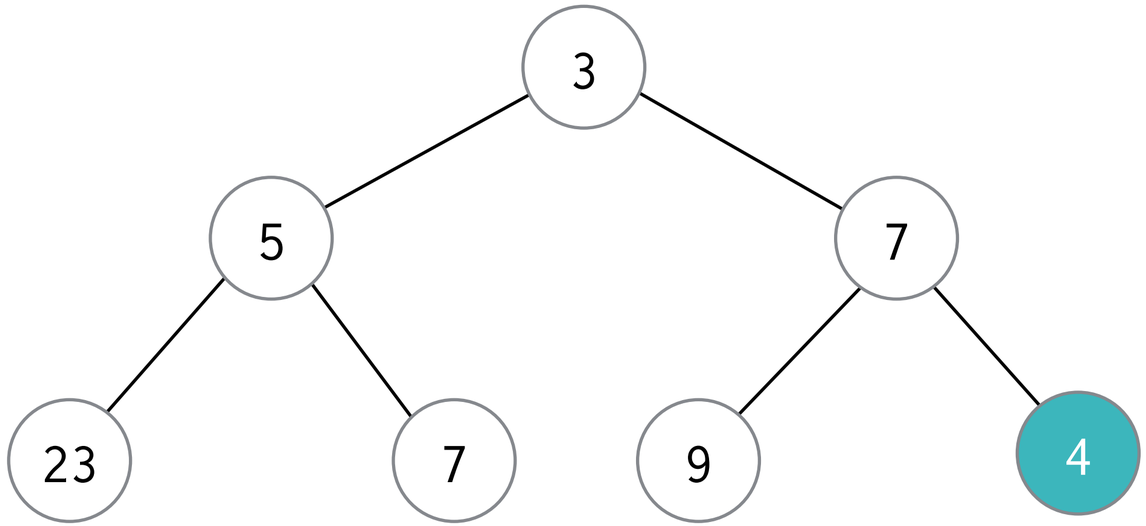
따라서 입력된 자료는 **항상 마지막 레벨의 가장 오른쪽 자리**에 채워진다.

힙 : 자료 입력하기



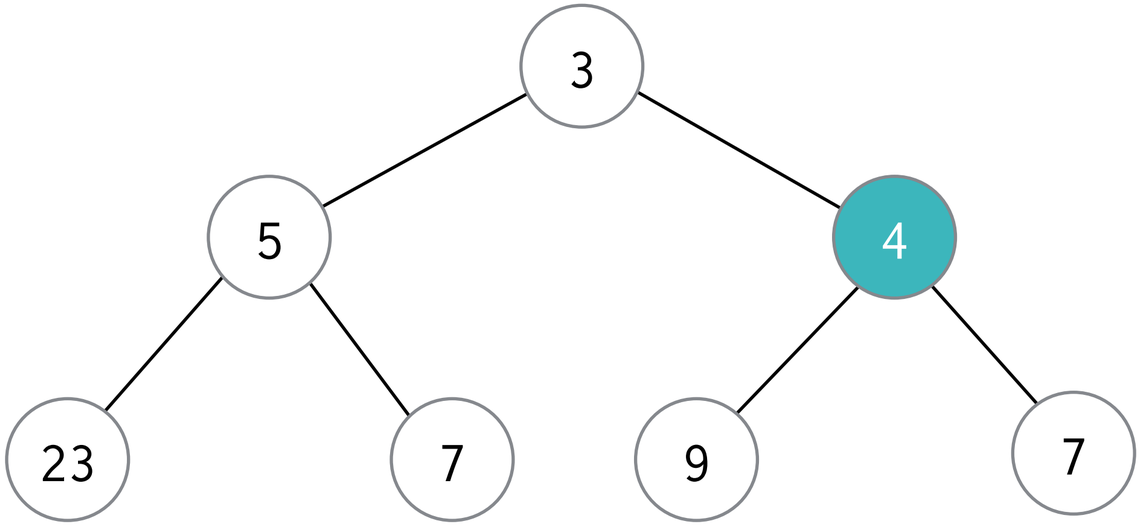
입력값 : 4

힙 : 자료 입력하기



입력값 : 4

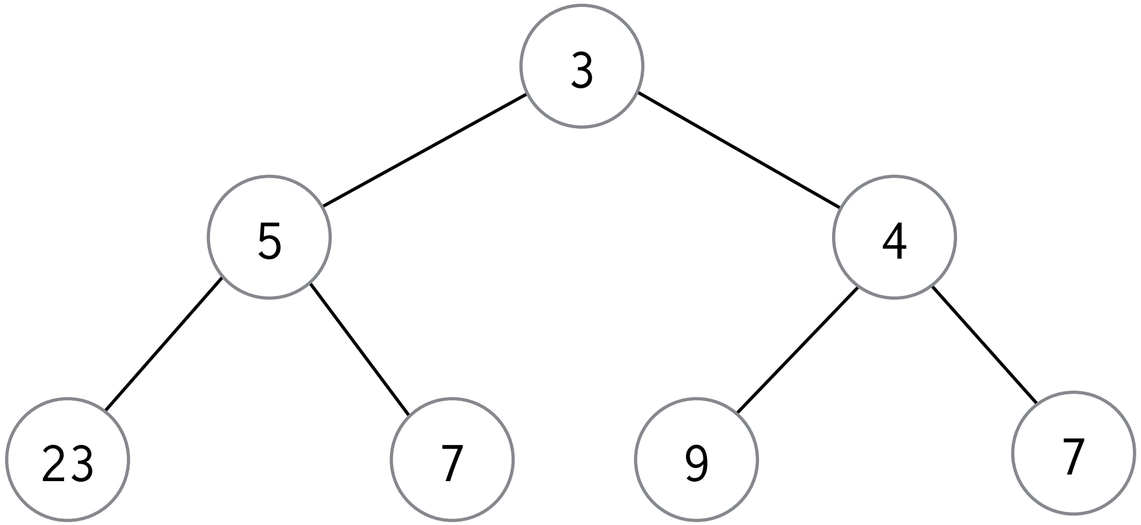
힙 : 자료 입력하기



입력값 : 4

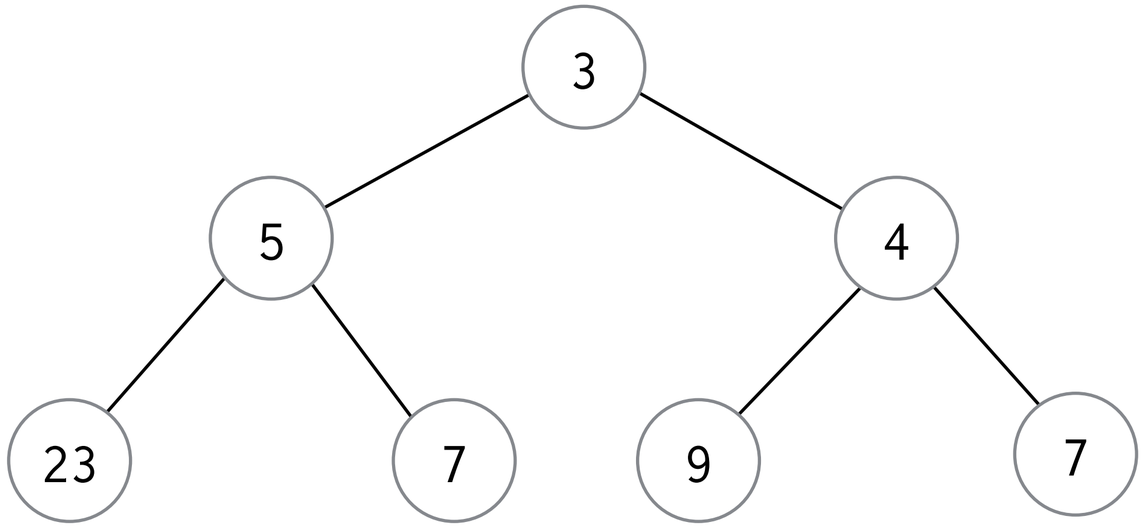
부모 노드의 값이 더 크므로
자리를 바꾼다.

힙 : 자료 입력하기



부모 노드가 자신보다 **더 작은 값**이 될 때까지
계속 자리를 변경한다.

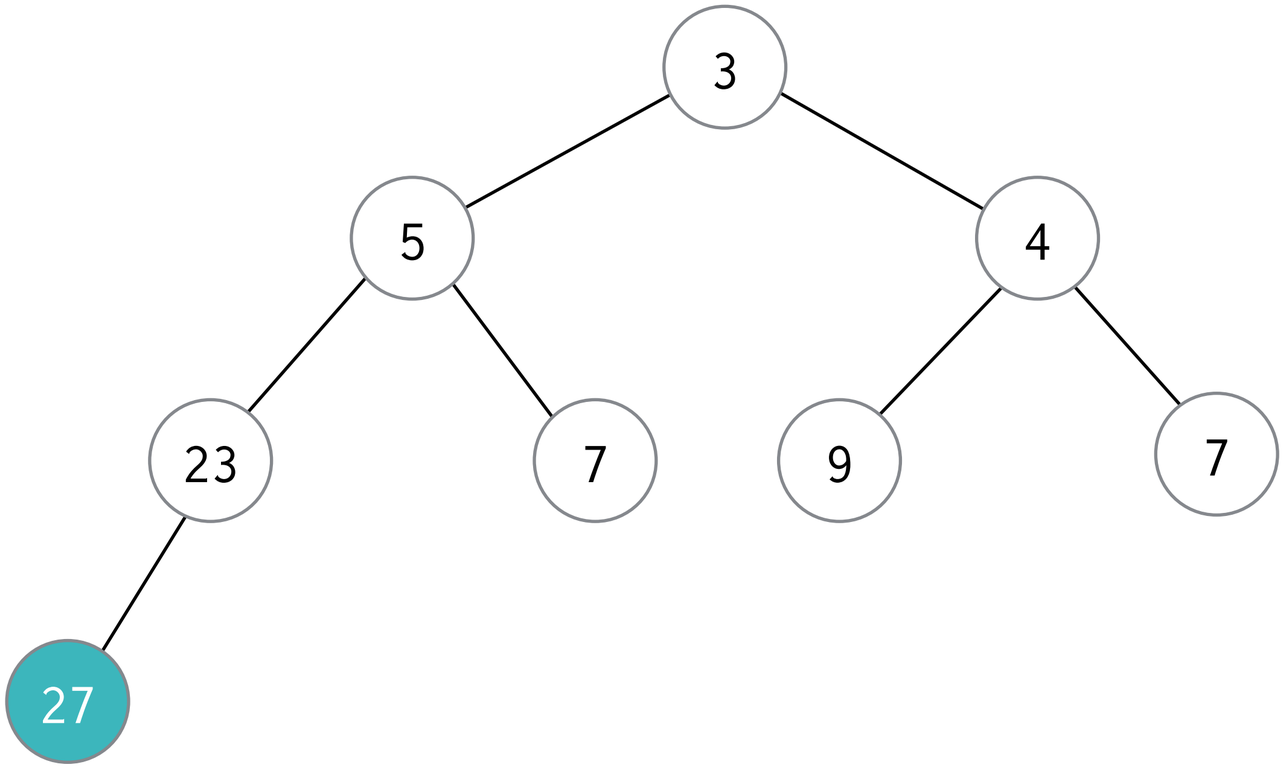
힙 : 자료 입력하기



입력값 : 27

01 우선순위 큐의 구현 방법과 힙

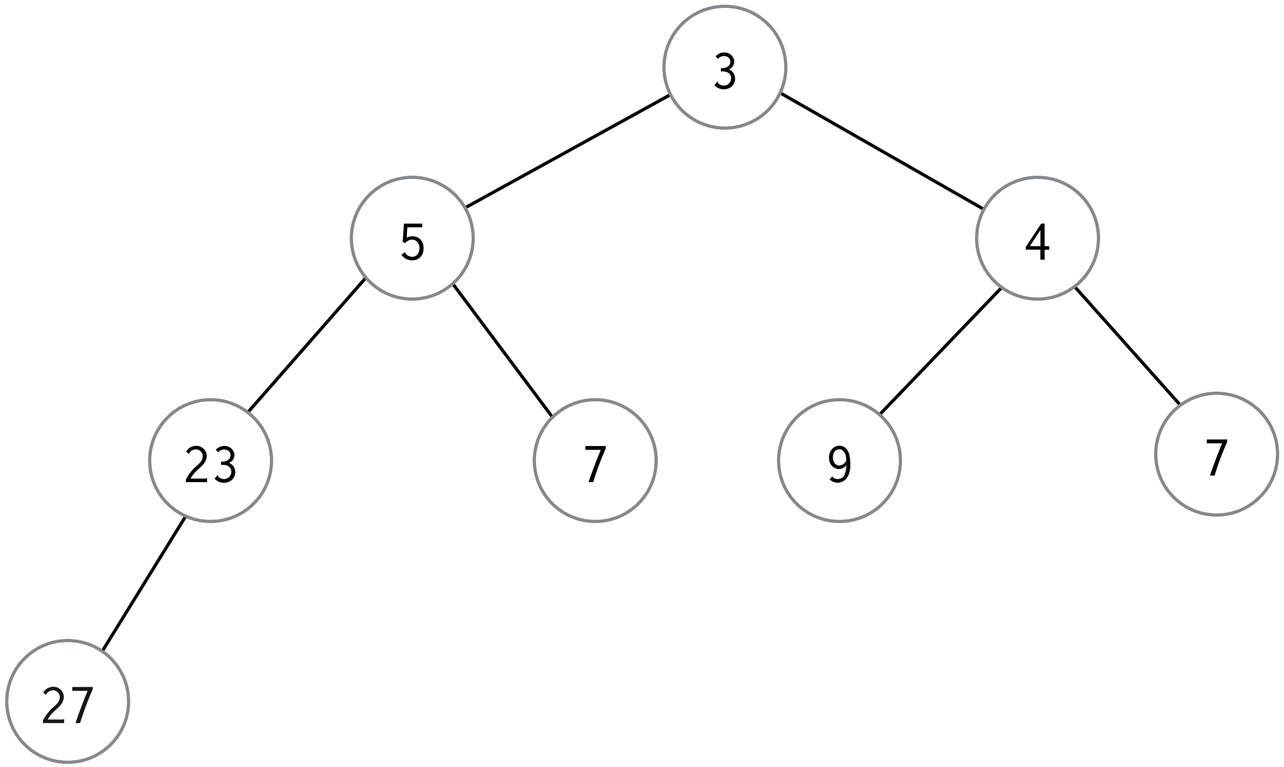
힙 : 자료 입력하기



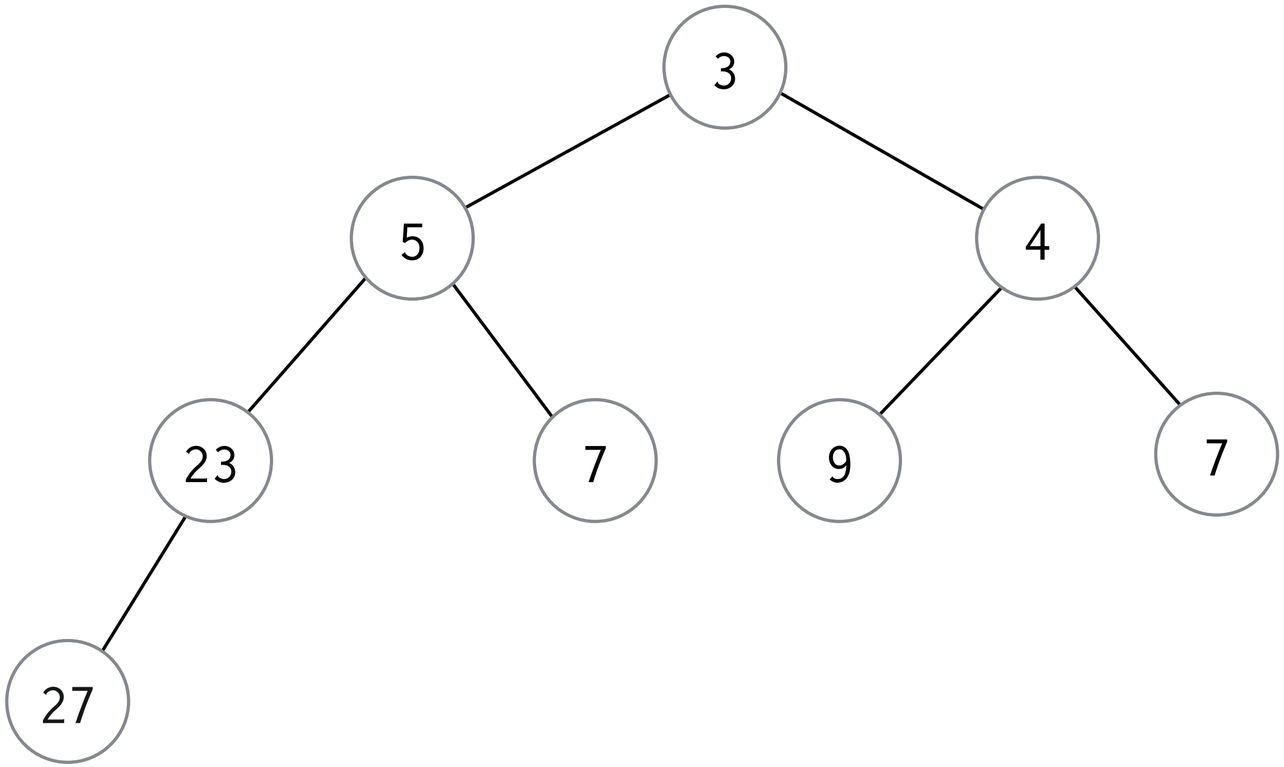
입력값 : 27

01 우선순위 큐의 구현 방법과 힙

힙 : 자료 입력하기

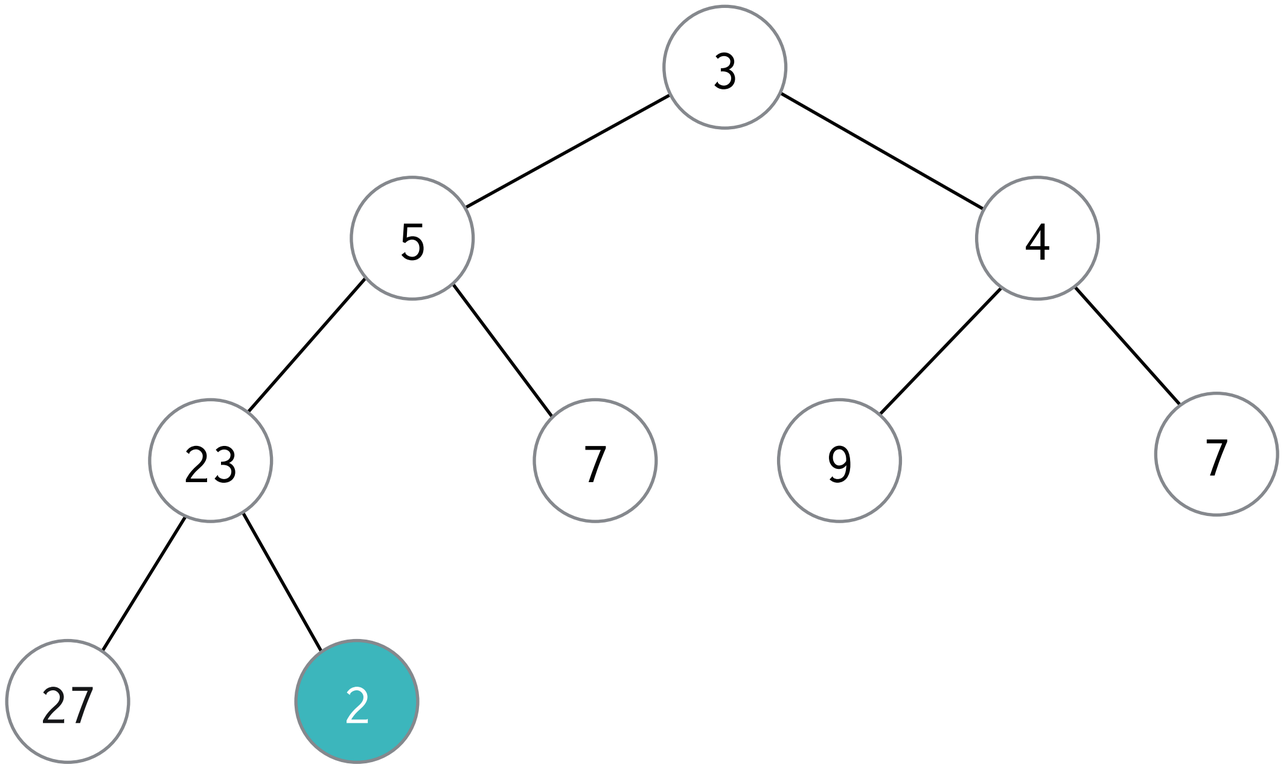


힙 : 자료 입력하기



입력값 : 2

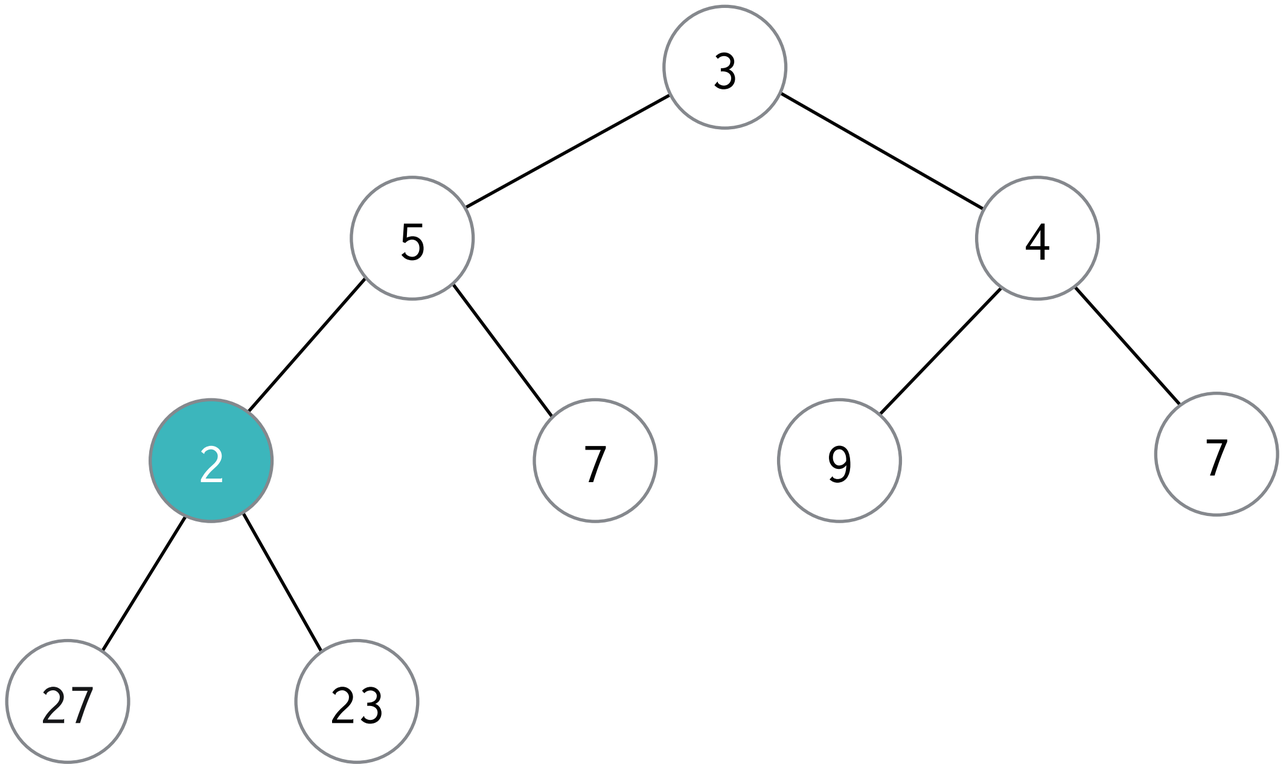
힙 : 자료 입력하기



입력값 : 2

01 우선순위 큐의 구현 방법과 힙

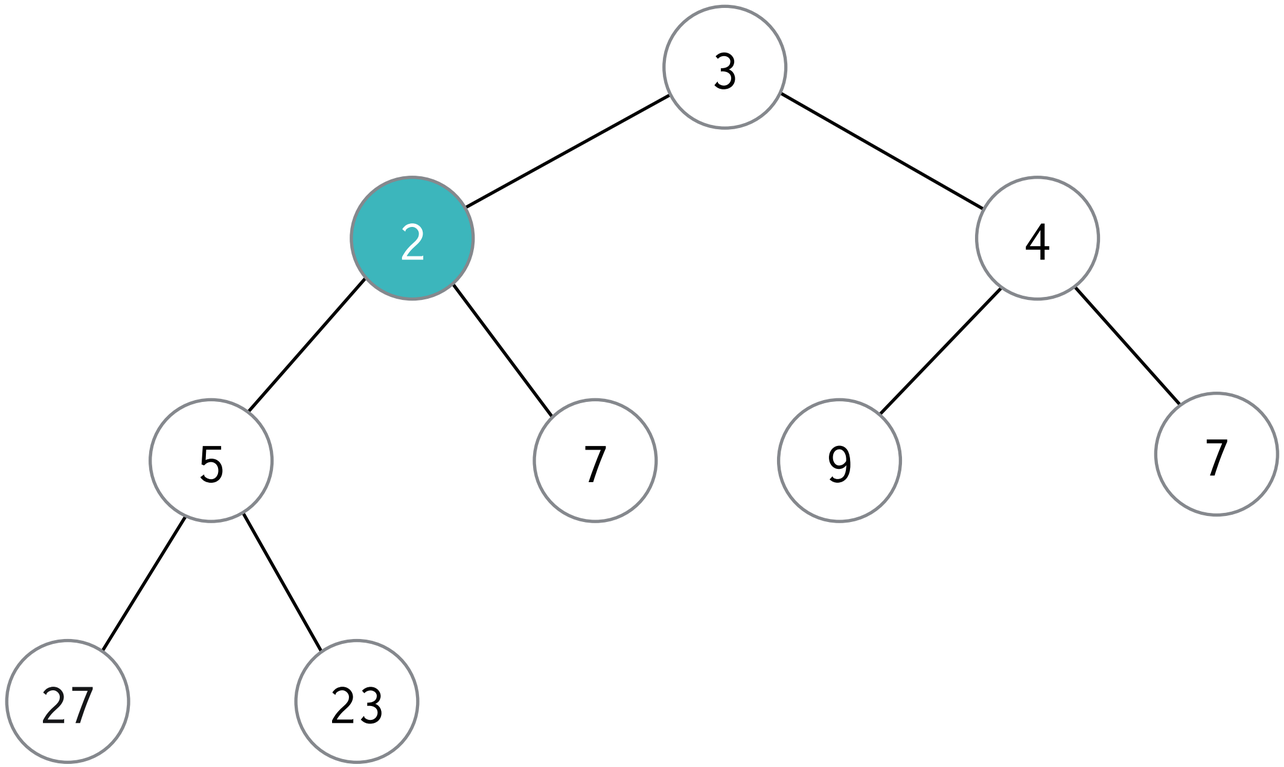
힙 : 자료 입력하기



입력값 : 2

01 우선순위 큐의 구현 방법과 힙

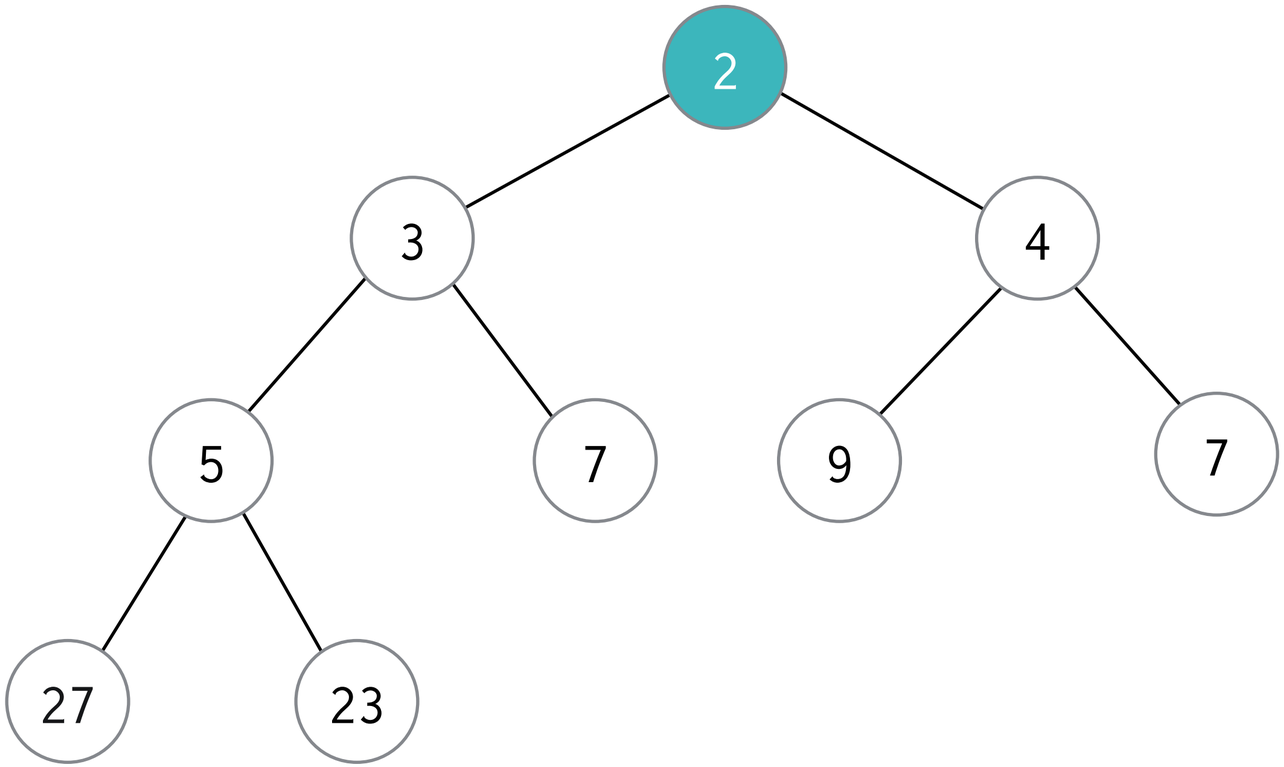
힙 : 자료 입력하기



입력값 : 2

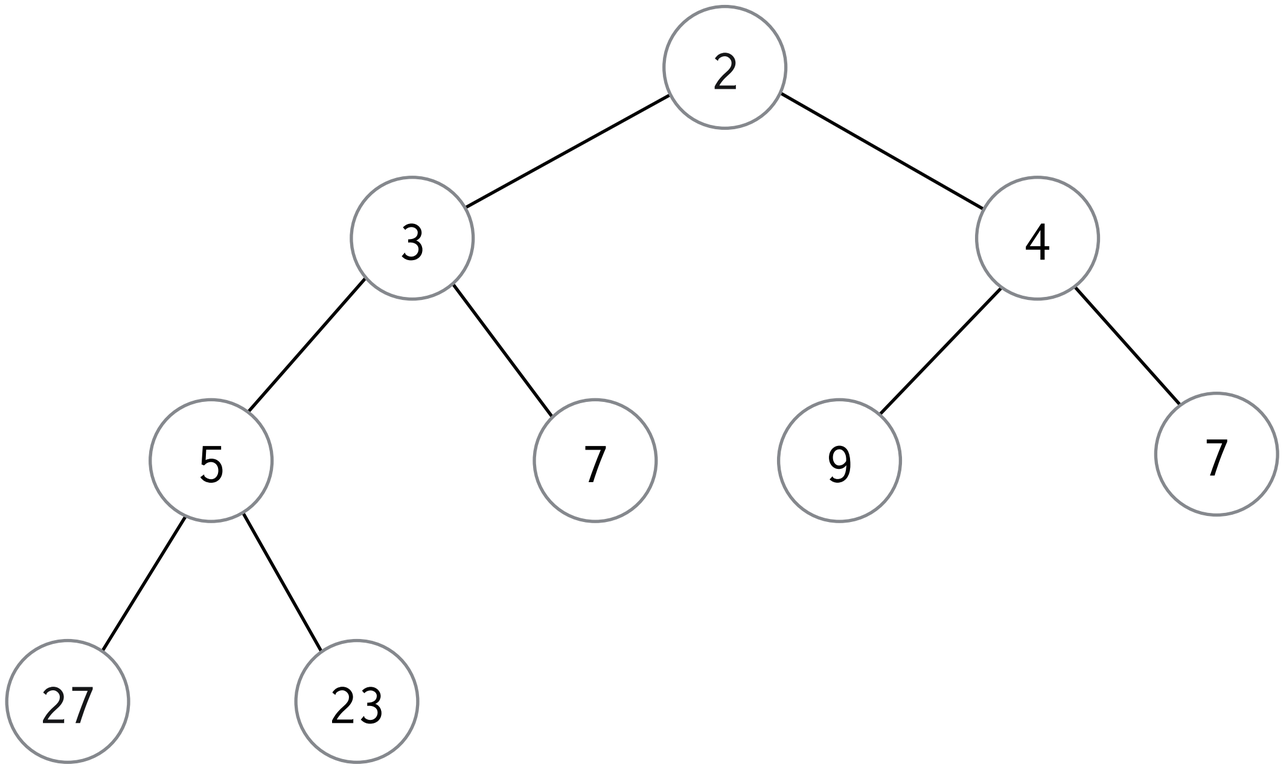
01 우선순위 큐의 구현 방법과 힙

힙 : 자료 입력하기



입력값 : 2

힙 : 자료 입력하기



입력값 : 2

힙 : 자료 입력하기

부모 노드와의 **대소관계**와

완전 이진 트리의 특성을 유지한 채

자료를 입력하면 된다.

힙 : 자료 입력하기

최악의 경우는 **새로운 최솟값**이 입력되는 경우이고,

루트 노드까지 거슬러 올라가게 된다.

이때 발생하는 연산 횟수는 **트리의 높이**와 비례하므로

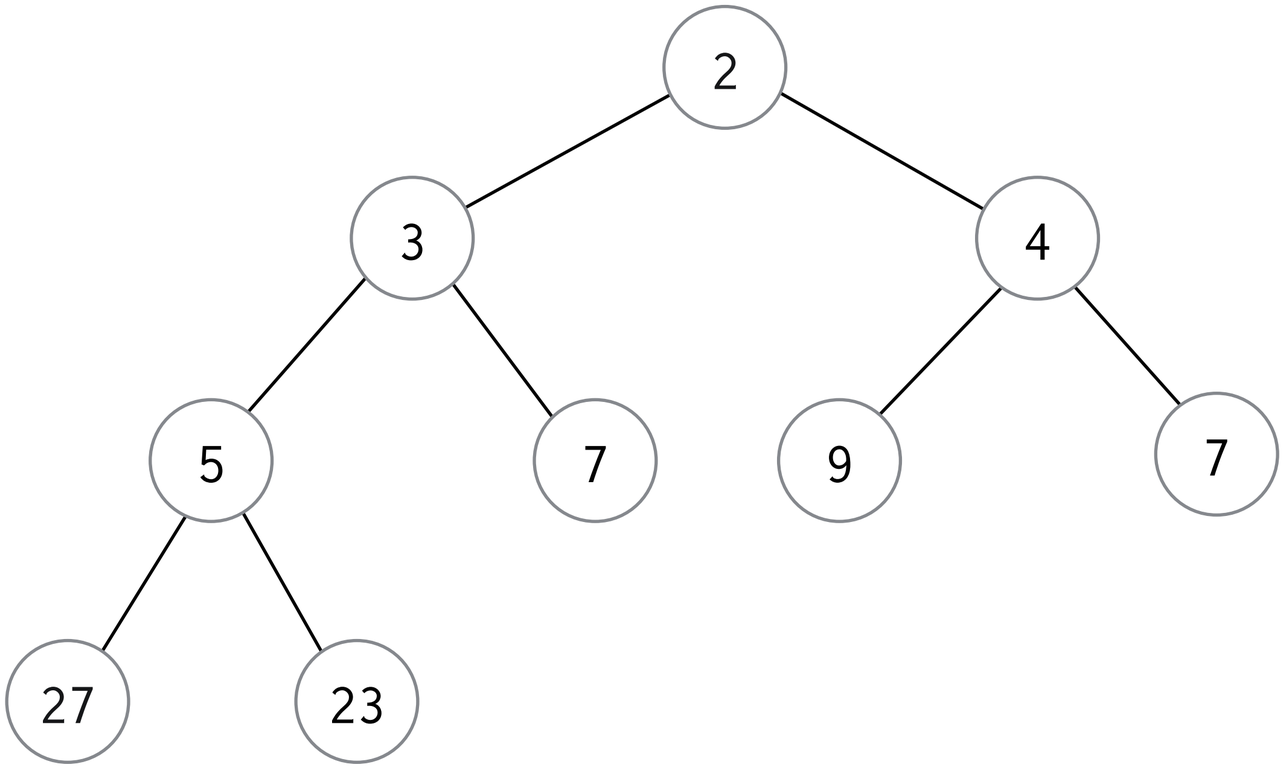
$O(logn)$의 시간 복잡도를 가진다.

힙 : 자료 출력하기

이번에는 힙에서 자료를 **출력**해보고자 한다.

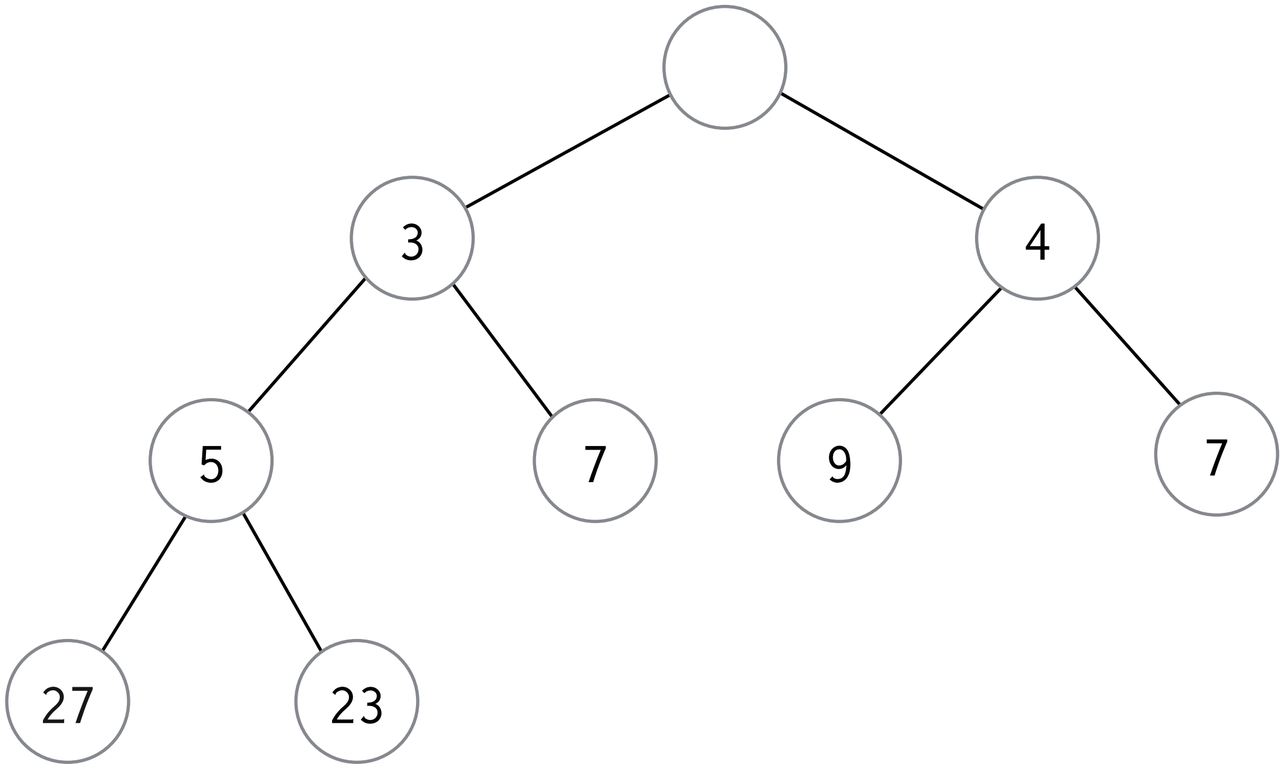
힙에서 출력되는 자료는 무조건 **루트 노드**이다.

힙 : 자료 출력하기



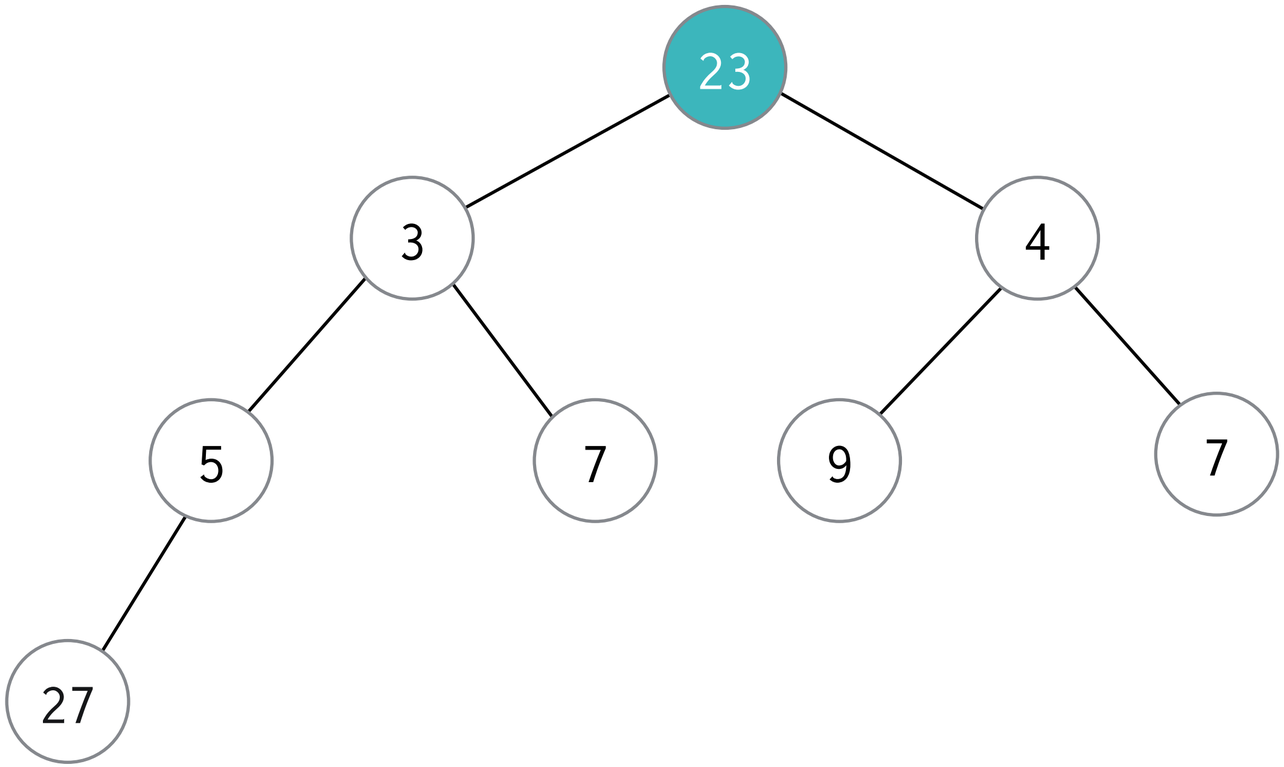
루트 노드를 출력한다.

힙 : 자료 출력하기



루트 노드를 출력한다.

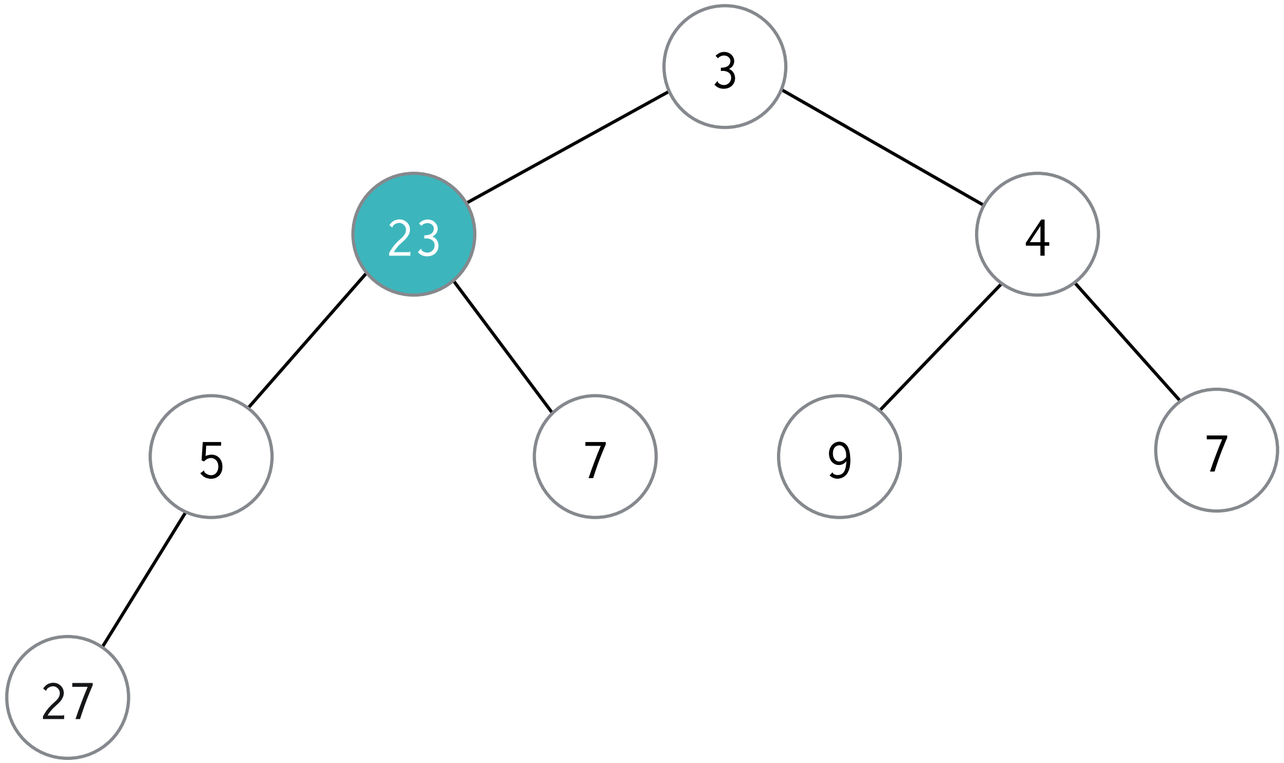
힙 : 자료 출력하기



루트 노드를 출력한다.

루트 노드의 빈 자리는
마지막 노드가 대신한다.

힙 : 자료 출력하기

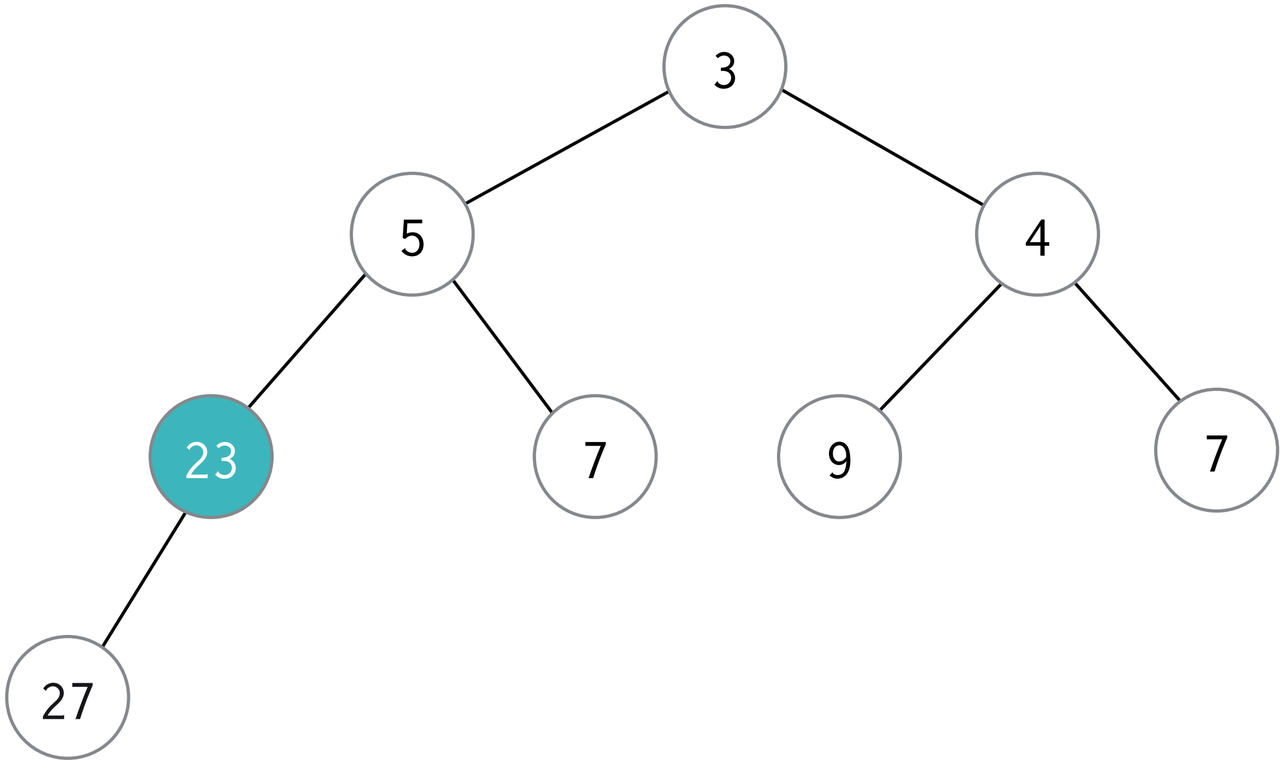


루트 노드를 출력한다.

루트 노드의 빈 자리는
마지막 노드가 대신한다.

자식 노드와 값을 비교하며
자리를 바꾼다.

힙 : 자료 출력하기

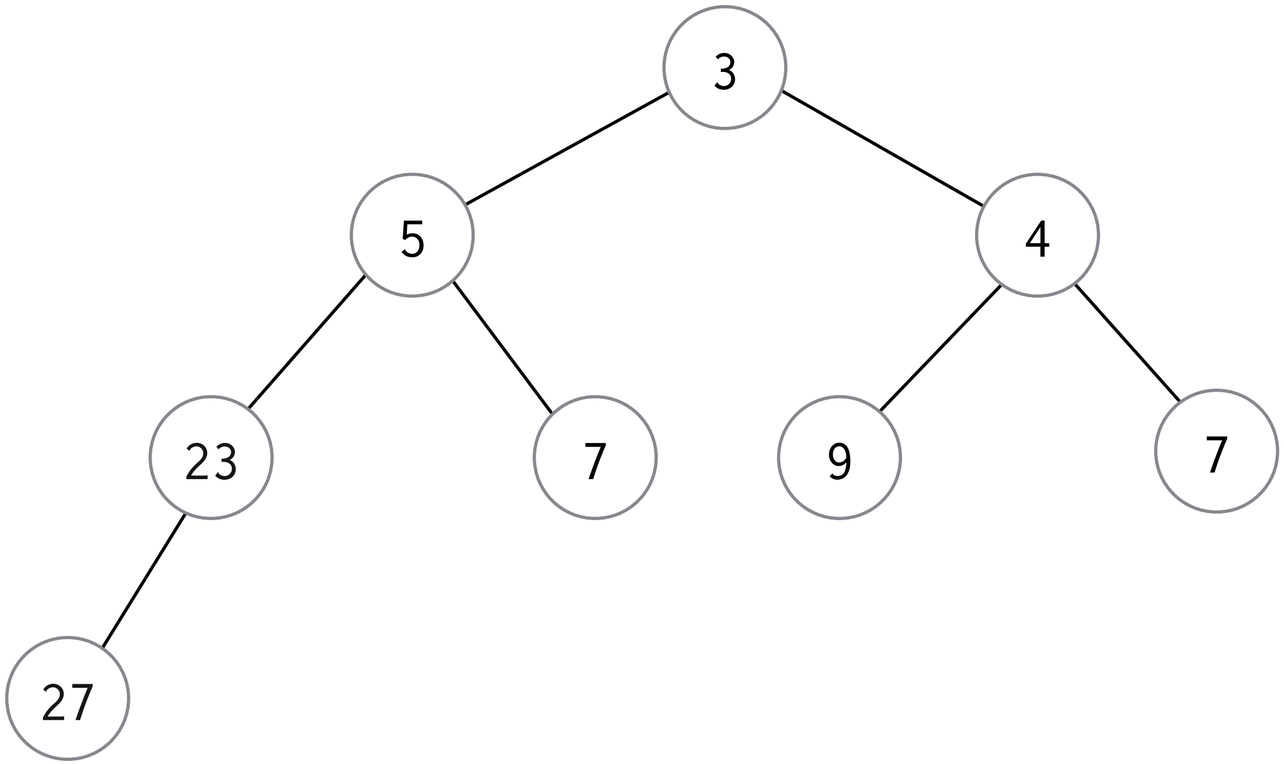


루트 노드를 출력한다.

루트 노드의 빈 자리는
마지막 노드가 대신한다.

자식 노드와 값을 비교하며
자리를 바꾼다.

힙 : 자료 출력하기



출력 완료

힙 : 자료 출력하기

최악의 경우는 힙의 **맨 마지막 원소**가 가장 **큰 값**을 가진 경우이다.

이 때 자리를 바꾸는 연산의 횟수는 트리의 높이만큼 이루어지므로

마찬가지로 $O(logn)$의 시간 복잡도를 가진다.

01 우선순위 큐의 구현 방법과 힙

우선순위 큐의 구현

	단순한 구현	힙으로 구현
입력	O(1)	O(log n)
출력	O(n)	O(log n)

02

힙을 이용한 문제 풀이

[실습1] 최소 힙 구현하기

heapq 라이브러리를 사용하지 않고, **최소 힙**을 구현해보자.

완전 이진 트리는 배열로 구현될 수 있으므로

힙도 **배열**로 구현할 수 있다.

[실습2] 최대 힙 구현하기

heapq 라이브러리를 사용하지 않고, **최대 힙**을 구현해보자.

완전 이진 트리는 배열로 구현될 수 있으므로

힙도 **배열**로 구현할 수 있다.

[실습3] 절댓값 힙

절댓값이 가장 작은 원소가 먼저 출력되는

'절댓값 힙'을 구현해보자.

02 힙을 이용한 문제 풀이

[실습4] 힙 정렬 구현하기

주어진 정수들을 **오름차순 정렬**한 결과를 출력해야 한다.
단, **힙 정렬**로 구현해야 한다.

입력 예시

```
6 1 10 8 3 5 4 7 2 9
```

출력 예시

```
1 2 3 4 5 6 7 8 9 10
```

/* elice */

[실습4] 힙 정렬 구현하기

힙을 출력할 때는 저장하고 있는 자료 중 **최솟값**을 반환하므로

주어진 정수를 정렬하기 위해서는 **모든 정수**를 힙에 **입력**하고,

힙의 **모든 정수를 출력**하면 된다.

[실습4] 힙 정렬 구현하기

n개의 정수가 있다고 할 때,

모든 정수를 힙에 **입력**하는 연산은 $O(nlogn)$,

힙에서 모든 정수를 **출력**하는 연산도 $O(nlogn)$이다.

[실습5] 점토 놀이

가능한 한 적은 힘을 사용하여 점토를 합쳤을 때
필요한 힘이 얼마인지 구해보자.

입력 예시

```
4              # 점토의 개수
1 5 7 3        # 각 점토의 무게
```

```
4
5 8 9 10
```

출력 예시

```
29
```

```
64
```

/* elice */

[실습5] 점토 놀이

무게가 **가벼운** 점토끼리 합칠수록 드는 힘도 적다.

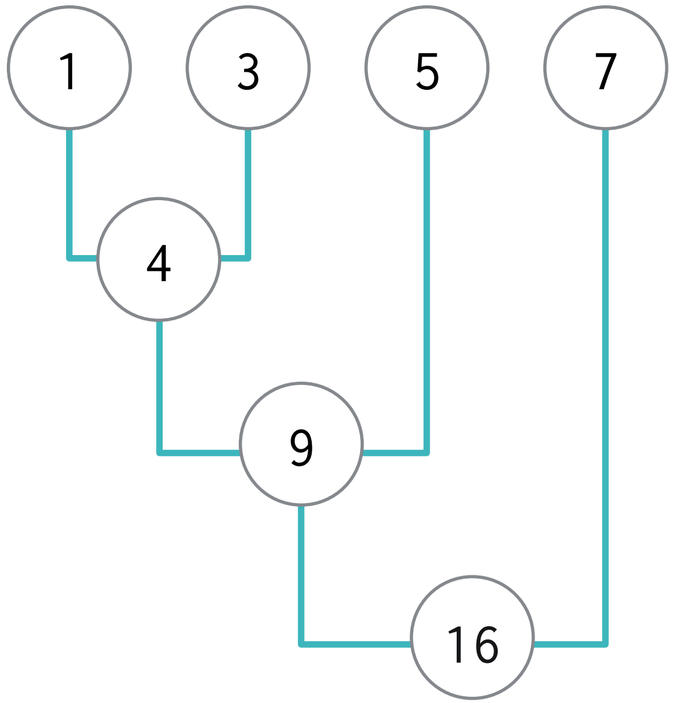
따라서 **무거운 점토는 가장 나중에 합쳐야** 전체 힘을 적게 만들 수 있다.

[실습5] 점토 놀이

이러한 가설을 세워볼 수 있다.

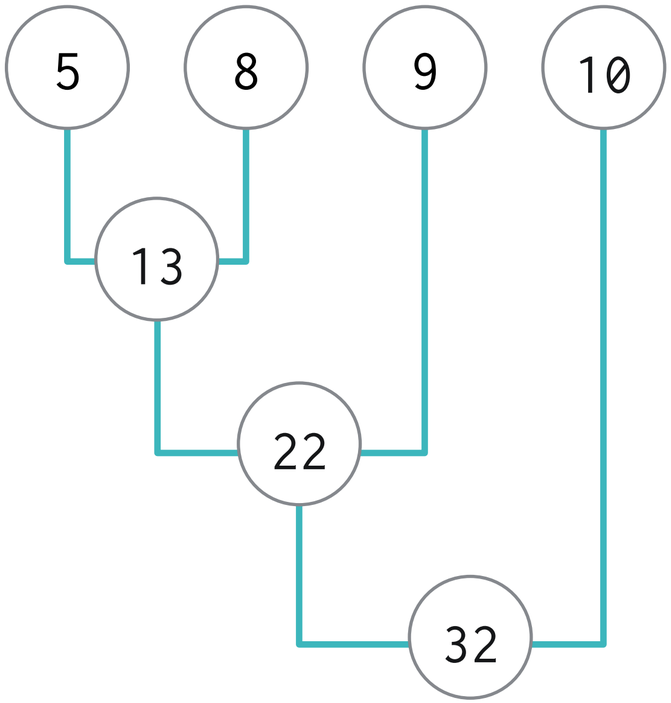
1. 점토를 **가벼운 순서**대로 정렬한다.
2. 앞에서부터 차례대로 점토를 하나씩 합쳐본다.

[실습5] 점토 놀이



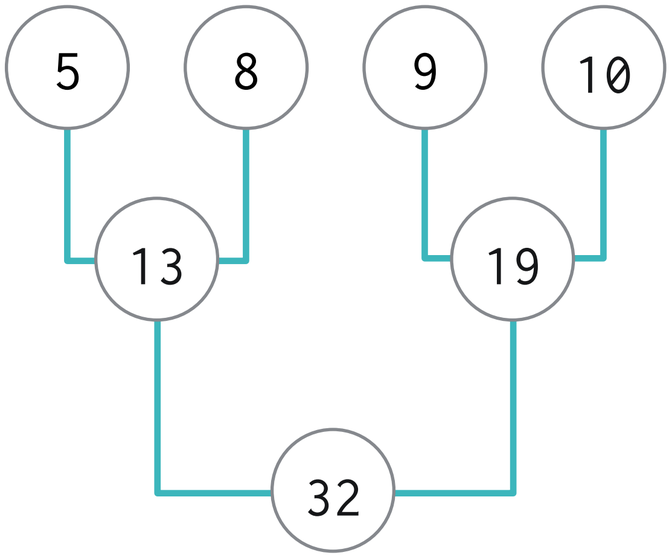
4 + 9 + 16 = 29
정답이다.

[실습5] 점토 놀이



13 + 22 + 32 = 67
정답일까?

[실습5] 점토 놀이



13 + 19 + 32 = 64
이렇게 되어야 정답이다.

[실습5] 점토 놀이

무조건 가벼운 점토부터 합쳐야 한다.

앞서 설명한 가설은 **합치는 과정에서 만들어진 점토**가 **기존의 점토**보다 가볍지 않아

오답이 발생하였다.

[실습5] 점토 놀이

1. 점토들을 모두 **힙**에 저장한다.
2. 가장 가벼운 **두 점토**를 꺼내서 합치고, **다시 힙에 저장**한다.
3. 점토가 하나가 될 때까지 **1번부터 반복**한다.

[실습6] 중간값 찾기

n개의 수가 차례대로 주어질 때, 매 순간마다
지금까지 입력된 수 중 **중간값**을 출력해야 한다.

입력 예시

```
10
1 2 3 4 5 6 7 8 9 10
```

출력 예시

```
1 1 2 2 3 3 4 4 5 5
```

```
7
1 -2 -5 10 4 7 5
```

```
1 -2 -2 -2 1 1 4
```

/* elice */

[실습6] 중간값 찾기

수를 배열에 입력할 때마다 중간값을 찾는다면
매번 정렬해야 하기 때문에
시간 복잡도는 $O(n^2 logn)$이 된다.

n이 입력될 수 있는 범위가 넓기 때문에
위 알고리즘으로는 제한 시간 내에 문제를 해결할 수 없다.

[실습6] 중간값 찾기

힙에서 항상 **최솟값(최댓값)**을

$O(logn)$의 시간에 찾아낼 수 있는 것처럼

중간값을 빠르게 찾아낼 방법이 필요하다.

[실습6] 중간값 찾기

중간값 mid **이하**의 값들을 $less$, **초과**의 값들을 $greater$ 라고 할 때

mid 는 $less$ 의 모든 수보다 항상 같거나 크고,

$greater$ 의 모든 수보다 항상 작다.

[실습6] 중간값 찾기

전체 수를 절반씩 나누어서 **최대 힙**과 **최소 힙,**

두 개의 힙에 나누어 저장할 수 있다.

전체 수가 홀수라면,

최대 힙이 1개의 수를 더 저장하게 한다.

[실습6] 중간값 찾기

최대 힙에 $less$ 를 저장하고,

최소 힙에 $greater$ 를 저장하면

항상 **최대 힙**의 루트 노드가 **중간값**이 된다.

[실습6] 중간값 찾기

입력될 수가 **최소 힙**의 루트 노드 이상의 값이라면 **최소 힙**에,

그렇지 않다면 **최대 힙**에 입력한다.

힙에 입력하고 나서, 두 힙이 저장하고 있는 수의 양을 조정해준다.

[실습6] 중간값 찾기

입력값 : 1, -2, -5, 10, 4, 7, 5

최대 힙

최소 힙

[실습6] 중간값 찾기

입력값 : **1,** -2, -5, 10, 4, 7, 5

최대 힙

1

최소 힙

중간값 : **1**

/* elice */

[실습6] 중간값 찾기

입력값 : 1, -2, -5, 10, 4, 7, 5

최대 힙

1, -2

최소 힙

[실습6] 중간값 찾기

입력값 : 1, -2, -5, 10, 4, 7, 5

최대 힙	최소 힙
-2	1

중간값 : 1, -2

/* elice */

[실습6] 중간값 찾기

입력값 : **1, -2, -5,** 10, 4, 7, 5

최대 힙	최소 힙
-2, -5	1

중간값 : 1, -2, **-2**

/* elice */

[실습6] 중간값 찾기

입력값 : **1, -2, -5, 10,** 4, 7, 5

최대 힙	최소 힙
-2, -5	1, 10

중간값 : 1, -2, -2, **-2**

/* elice */

[실습6] 중간값 찾기

입력값 : 1, -2, -5, 10, 4, 7, 5

최대 힙	최소 힙
-2, -5	1, 10, 4

[실습6] 중간값 찾기

입력값 : **1, -2, -5, 10, 4,** 7, 5

최대 힙

-2, -5, 1

최소 힙

10, 4

중간값 : 1, -2, -2, -2, **1**

/* elice */

[실습6] 중간값 찾기

입력값 : **1, -2, -5, 10, 4, 7,** 5

최대 힙	최소 힙
-2, -5, 1	10, 4, 7

중간값 : 1, -2, -2, -2, 1, **1**

/* elice */

[실습6] 중간값 찾기

입력값 : **1, -2, -5, 10, 4, 7, 5**

최대 힙	최소 힙
-2, -5, 1	10, 4, 7, 5

[실습6] 중간값 찾기

입력값 : **1, -2, -5, 10, 4, 7, 5**

최대 힙	최소 힙
-2, -5, 1, 4	10, 7, 5

중간값 : 1, -2, -2, -2, 1, 1, **4**

[실습6] 중간값 찾기

N **개**의 정수를 힙에 넣어야 하고,

최악의 경우 N **번** 힙에서 출력되어야 하므로

시간 복잡도는 $O(NlogN)$이다.

[실습6] 중간값 찾기

최대 힙과 최소 힙을 이용하여
수가 입력될 때마다 입력된 수 집합의 중간값을
빠른 시간 내에 구할 수 있다.

Credit

/* elice */

코스 매니저

김경민

콘텐츠 제작자

김경민

강사

김경민

감수자

이형민

디자인

강혜정

Contact

TEL
070-4633-2015

WEB
https://elice.io

E-MAIL
contact@elice.io