Heap

두 가지 특성 만족

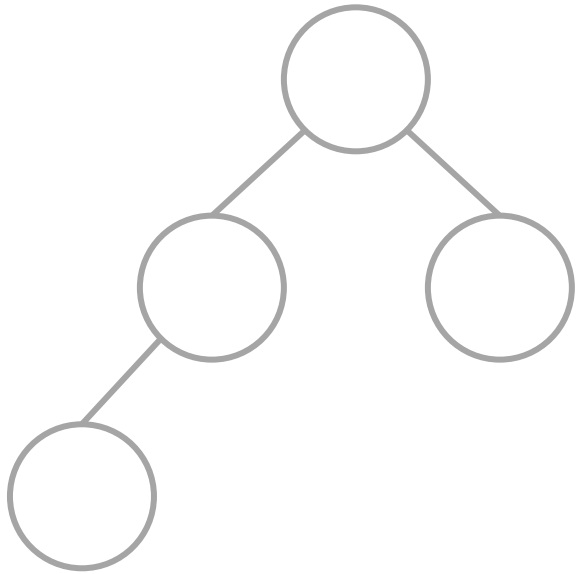
1. 완전 이진 트리
2. 부모 노드의 우선순위가 자식 노드의 우선순위보다 높다.

=> 최우선순위 노드는 루트에 존재

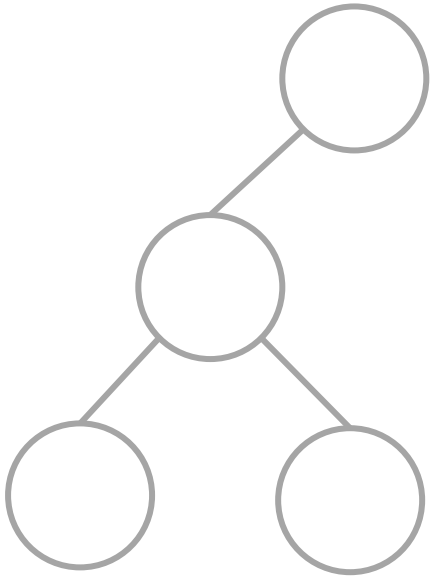
Heap

1. 완전 이진 트리

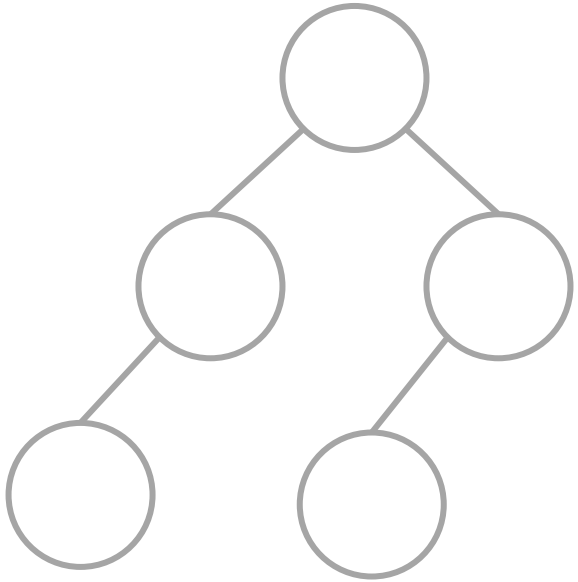
- 모든 노드의 자식 노드는 최대 2개
- 노드는 위에서부터, 왼쪽부터 꽉 채워진다.



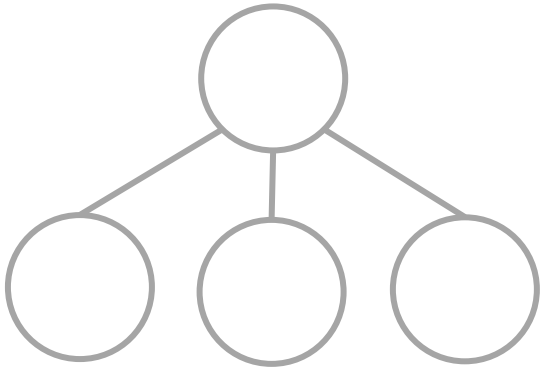
(O)



(X)



(X)

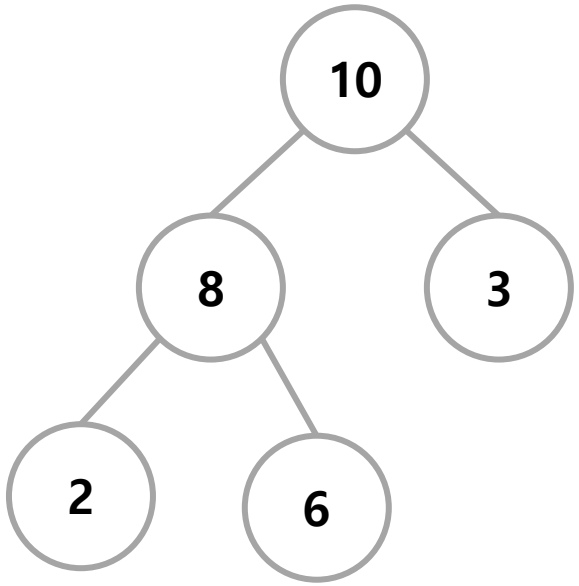


(X)

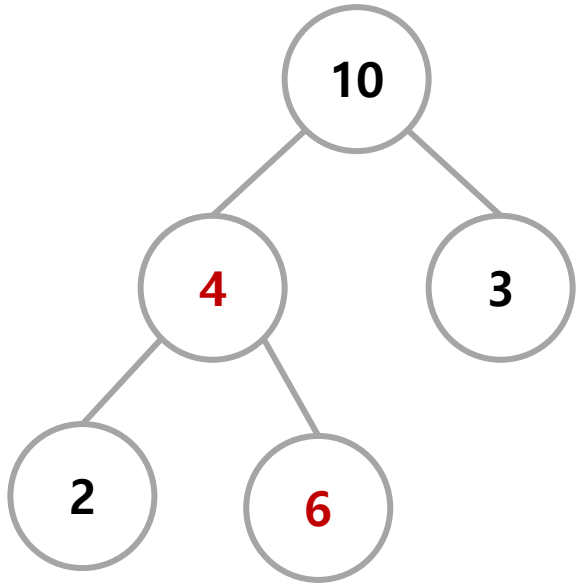
Heap

2. 부모 노드의 우선순위가 자식 노드의 우선순위보다 높다.

ex) 값이 클수록 우선순위가 높다 : MAX_HEAP



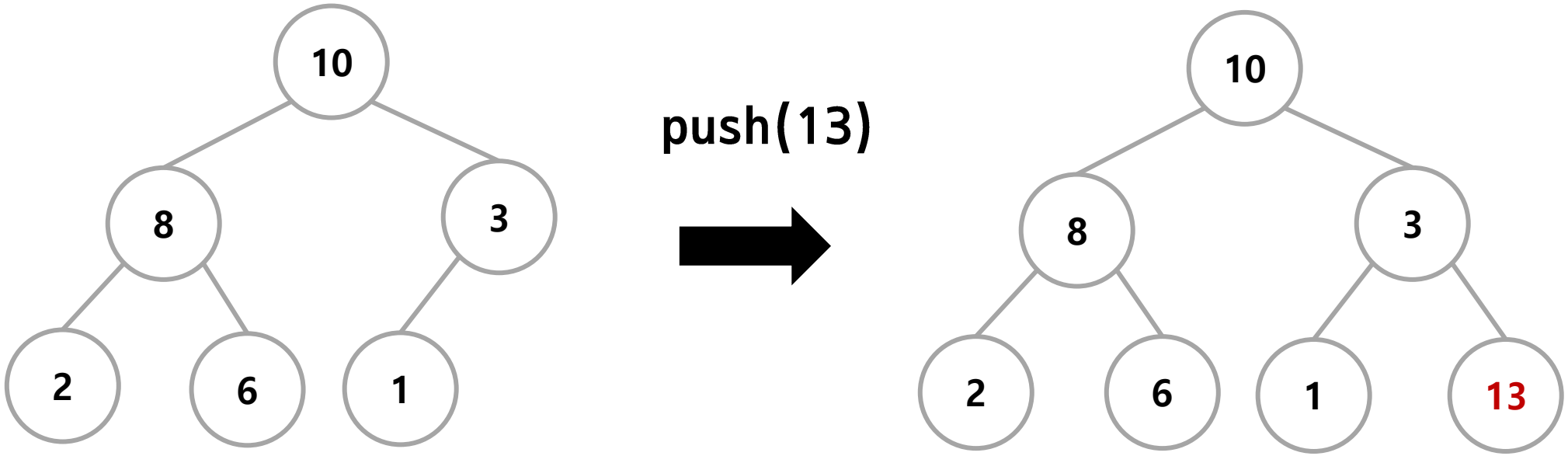
(O)



(X)

Heap : push

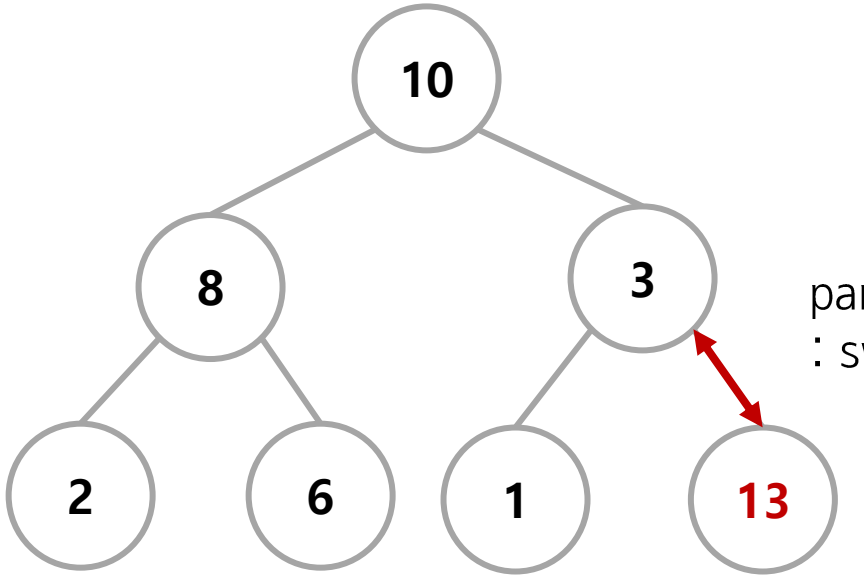
1. 마지막 위치에 추가
2. 추가한 노드를 부모 노드랑 비교하며 우선순위 높으면 바꿔 올라간다.



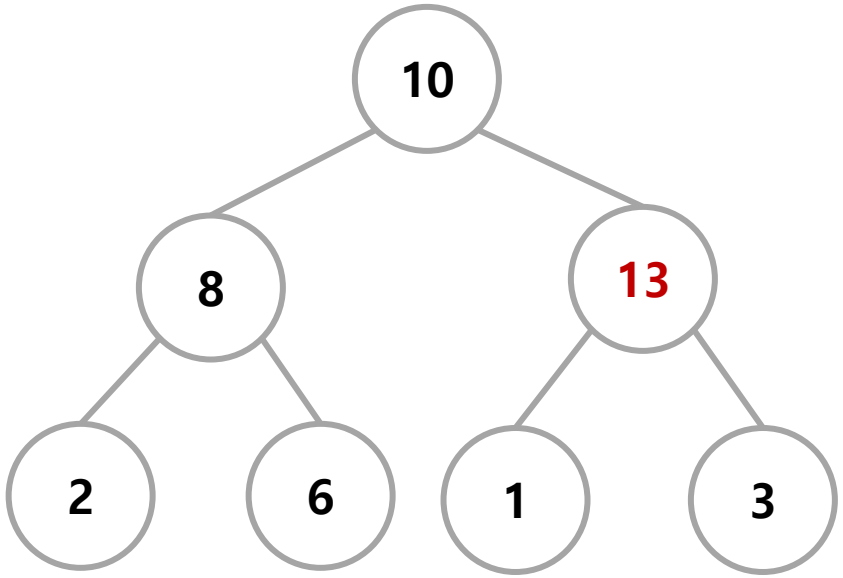
Max_Heap example

Heap : push

1. 마지막 위치에 추가
2. 추가한 노드를 부모 노드랑 비교하며 우선순위 높으면 바꿔 올라간다.



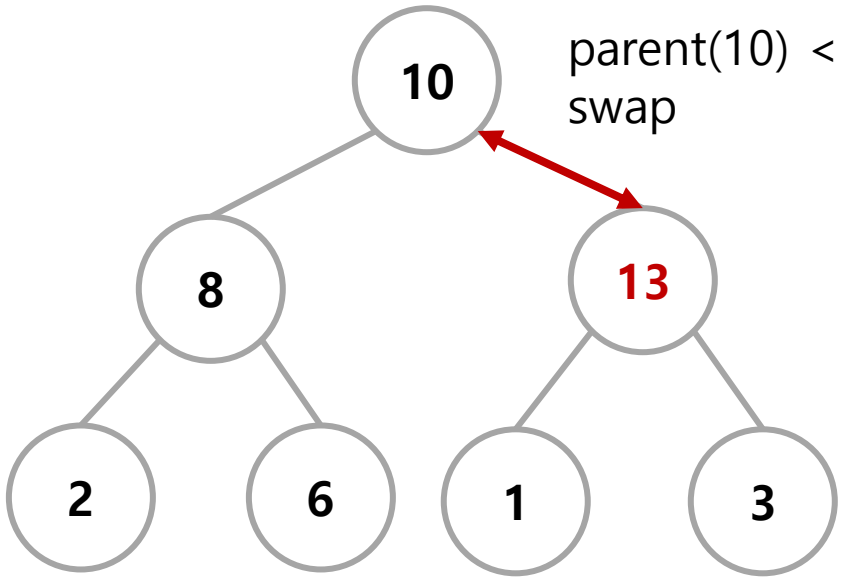
parent(3) < child(13)
: swap

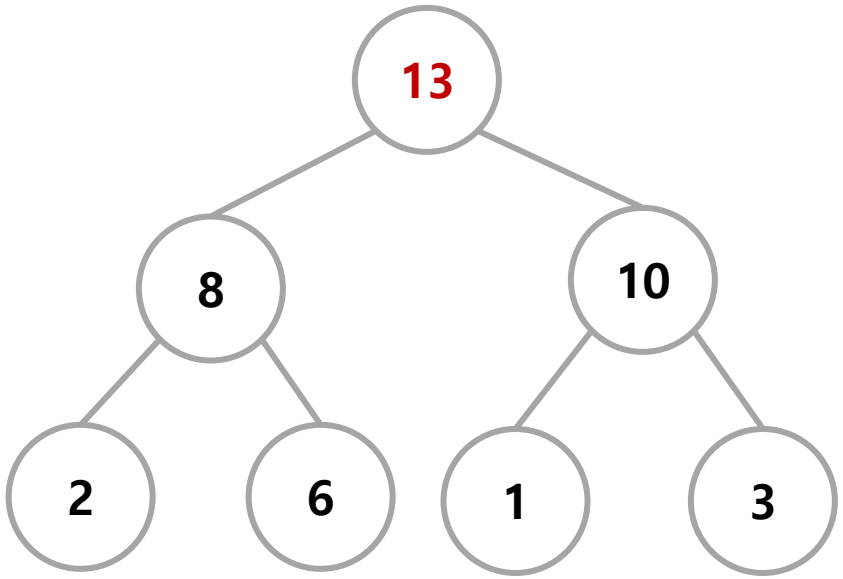


Max_Heap example

Heap : push

1. 마지막 위치에 추가
2. 추가한 노드를 부모 노드랑 비교하며 우선순위 높으면 바꿔 올라간다.



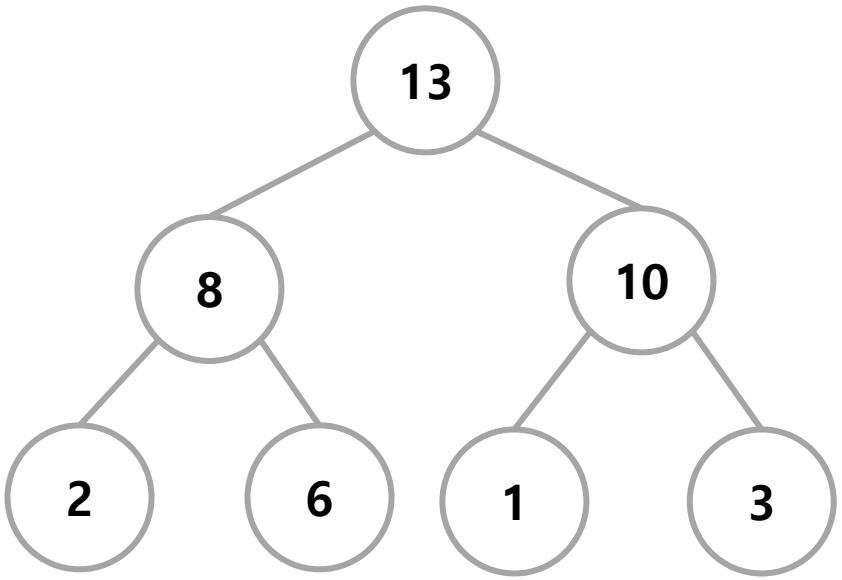


Max_Heap example

Heap : push

1. 마지막 위치에 추가
2. 추가한 노드를 부모 노드랑 비교하며 우선순위 높으면 바꿔 올라간다.

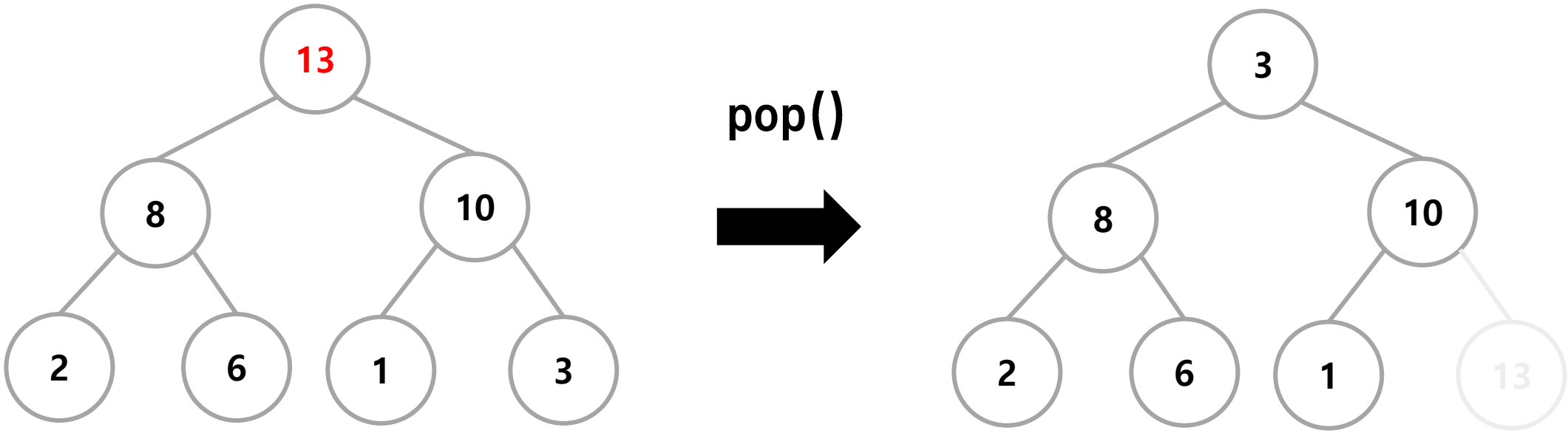
[13,8,10,2,6,1,3,X]



Max_Heap example

Heap : pop

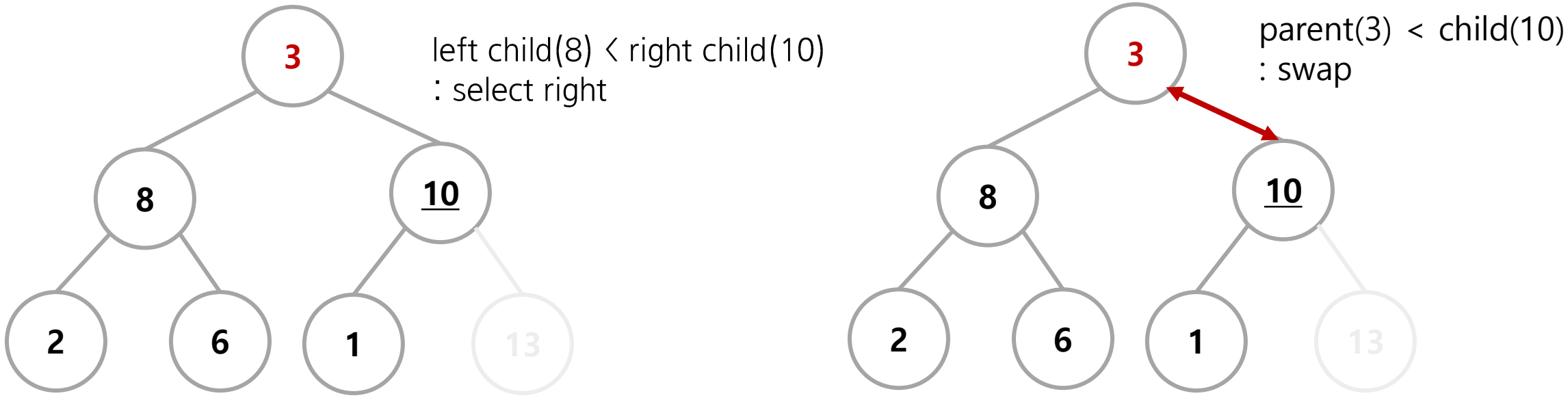
1. 루트와 마지막 노드를 바꾸고 heap size를 감소시킨다.
2. 루트 노드에서 우선순위 높은 자식 노드와 비교하여 우선순위가 낮으면 바꿔 내려간다.



Max_Heap example

Heap : pop

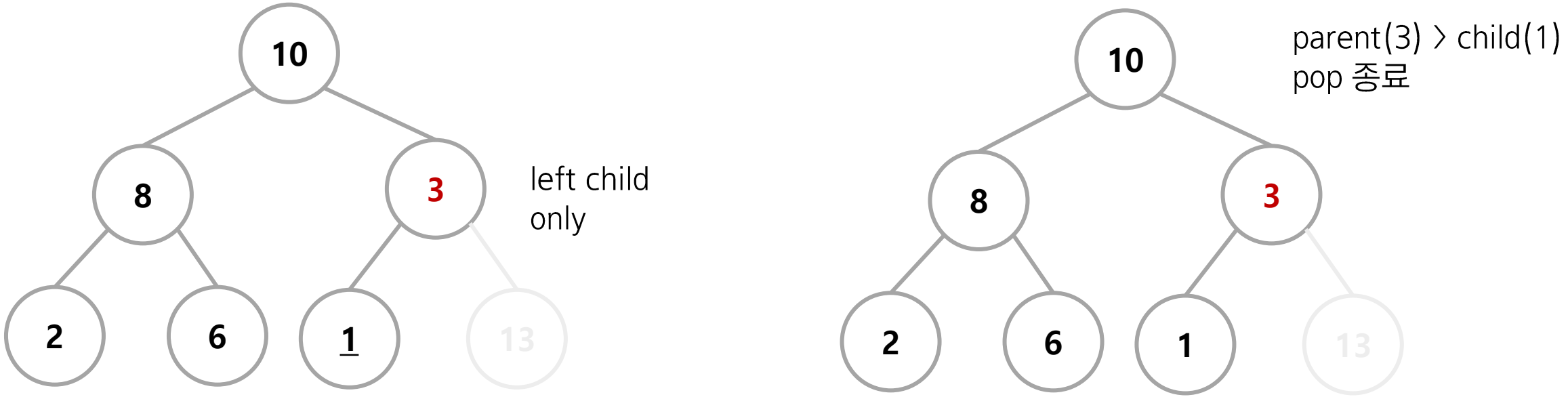
1. 루트와 마지막 노드를 바꾸고 heap size를 감소시킨다.
2. 루트 노드에서 우선순위 높은 자식 노드와 비교하여 우선순위가 낮으면 바꿔 내려간다.



Max_Heap example

Heap : pop

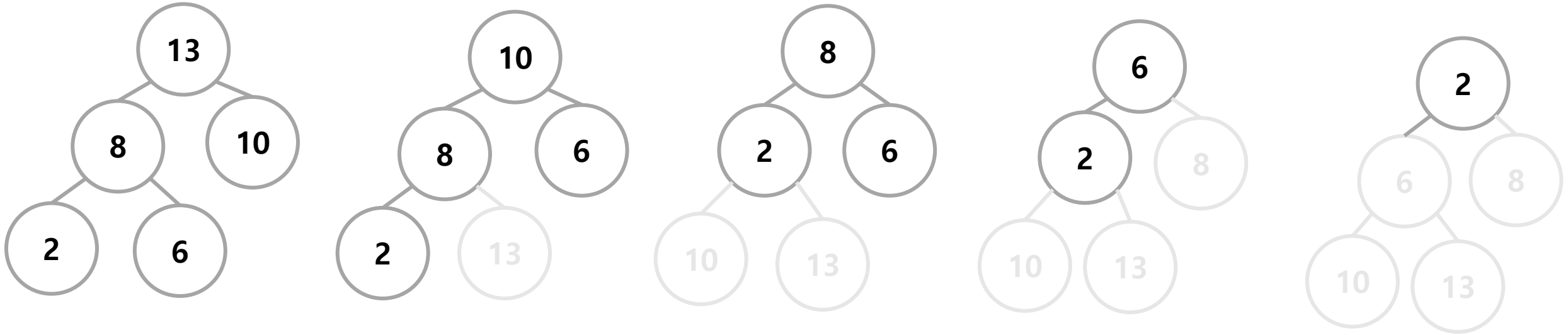
1. 루트와 마지막 노드를 바꾸고 heap size를 감소시킨다.
2. 루트 노드에서 우선순위 높은 자식 노드와 비교하여 우선순위가 낮으면 바꿔 내려간다.



Max_Heap example

Heap Sort

1. n개의 data를 heap에 push
2. n번 pop 진행 (pop 할 때, 루트와 마지막 값 swap)



Max Heap example

0	1	2	3	4
2	6	8	10	13

Max Heap => 오름차순 Min Heap => 내림차순

Update / Erase

- **Lazy Update**
  - 실제 업데이트 되는 정보를 별도의 객체(배열)에 저장해 놓는다.
  - heap에는 push 되는 시점의 정보가 들어간다.
  - top을 확인할 때, 유효하지 않는 값들은 버린다.
  - 유효하지 않은 값이란 최신 데이터랑 push된 시점의 데이터가 같지 않은 값이다.
  - heap에는 유효하지 않은 값들도 들어가 있다.

- **Real-time Update**
  - 각 원소들의 heap index를 기록하는 별도의 배열(pos[])이 필요하다.
  - heap의 위치가 swap 될때마다 pos[] 배열도 같이 swap된다.
  - 특정 id가 update되거나 erase 될 때,
    pos[id]로 바뀐 id의 heap index에 접근한다.
  - up, down을 통해 본인 자리를 찾아간다.

Lazy Update

- **원소가 erase될 때**
  실제 데이터를 저장하는 객체(배열)의 값을 삭제됐다고 표시한다.
  top을 확인할 때 실제 정보와 같은지 판별하여 그렇지 않다면 pop하고
  유효한 정보가 나올 때까지 반복한다.

- **원소가 update 될 때**
  update된 정보를 heap에 push 한다.
  그리고 top을 확인할 때 실제 정보와 같은지 판별하여 그렇지 않다면
  pop하고 유효한 정보가 나올 때까지 반복한다.

Lazy Update Example

heap에는 유효하지 않은 정보들도 들어가 있을것이므로
실제 유효한 정보들을 저장하는 별도의 객체(배열)가 필요하다.

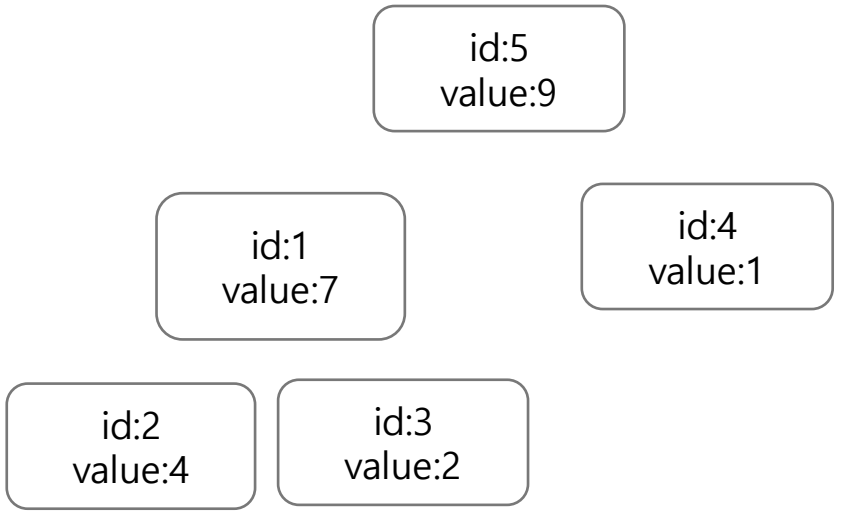
유효한지 판단해주기 위한 식별 값들이 heap data에 포함되어야 한다.
(실제 객체의 id, 변하는 value)

실제 유효한 정보를 저장하는 배열

S

1	2	3	4	5
7	4	2	1	9

Max Heap



Lazy Update Example

1번이 삭제된 경우

value 범위를 벗어나는 값을 설정하여 지워졌음을 표시한다.

예제에서는 value가 양수라고 가정하고 -1로 표현해 볼 수 있다.

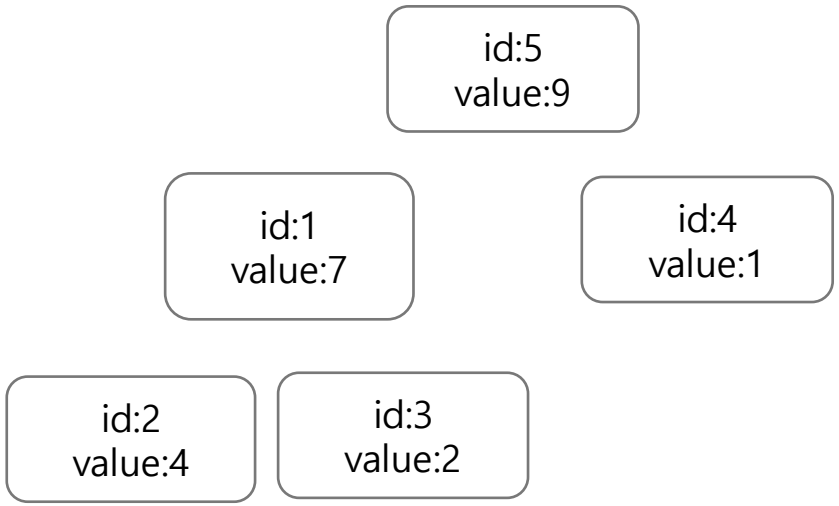
힙은 따로 처리해줄 필요가 없다.

실제 유효한 정보를 저장하는 배열

S

1	2	3	4	5
-1	4	2	1	9

Max Heap



Lazy Update Example

우선순위 값을 구하는 경우

Lazy update의 경우, top에 왔을 때 유효성 검증을 하여 사용해야 한다.

heap[1]의 value 와 S[5]의 값이 같은지 비교한다.

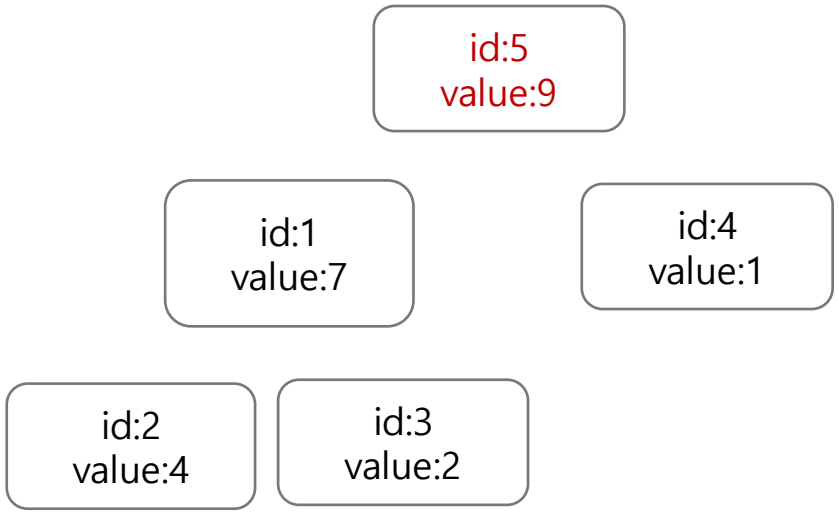
같으므로 유효한 값임을 확인 하고 처리해준 뒤 pop 한다.

실제 유효한 정보를 저장하는 배열

S

1	2	3	4	5
-1	4	2	1	9

Max Heap



Lazy Update Example

또 우선순위 값을 구하는 경우

heap[1]의 value 와 S[1]의 값이 같은지 비교한다.

다르므로 해당 heap[1]의 정보는 유효하지 않다고 판별하여 그냥 pop()해준다.

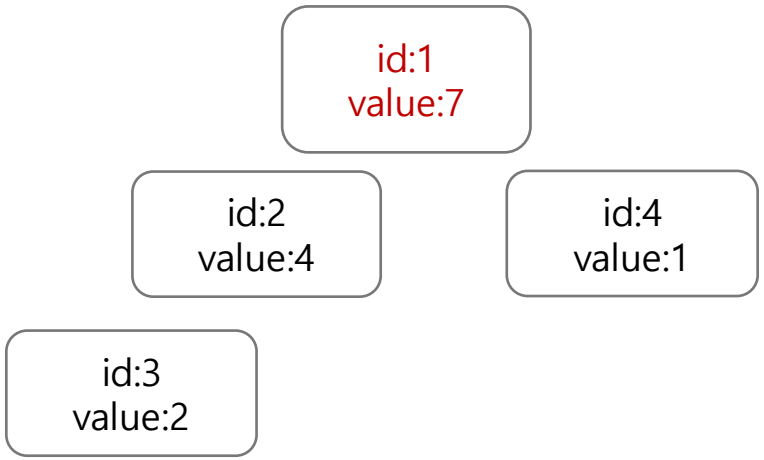
새로운 heap[1]에 대해서도 똑같이 확인하며 유효한 값이 나올 때까지 반복한다.

실제 유효한 정보를 저장하는 배열

S

1	2	3	4	5
-1	4	2	1	-1

Max Heap



Lazy Update Example

1번 지우기 전 상태에서
2번을 6으로 update 하는 경우

S에는 실시간으로 값을 바꿔준다.

heap에는 업데이트된 값으로 push 해 준다.

heap에 똑같은 index 2에 대한 정보가 중복되어 들어간다.

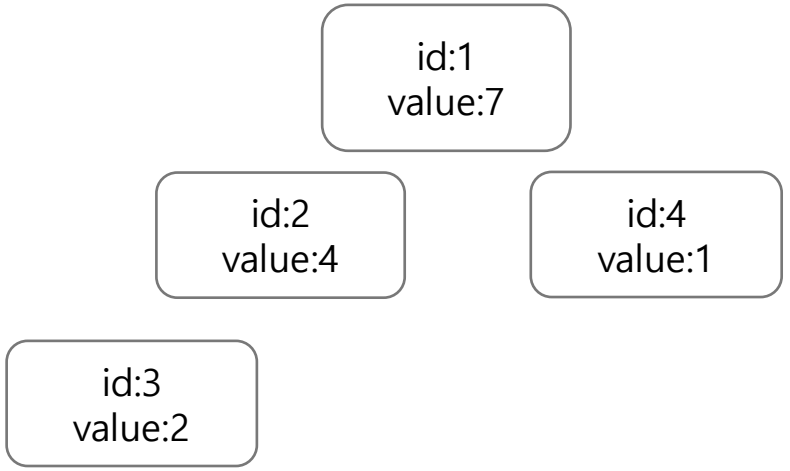
해당 정보들은 heap[1]에 왔을 때, erase와 마찬가지로 실제 정보와 비교하여 유효성 판별이 가능하다.

실제 유효한 정보를 저장하는 배열

S

1	2	3	4	5
7	4	2	1	-1

Max Heap



Lazy Update Example

2번을 6으로 update

S에는 실시간으로 값을 바꿔준다.

heap에는 업데이트된 값으로 push 해 준다.

heap에 똑같은 index 2에 대한 정보가 중복되어 들어간다.

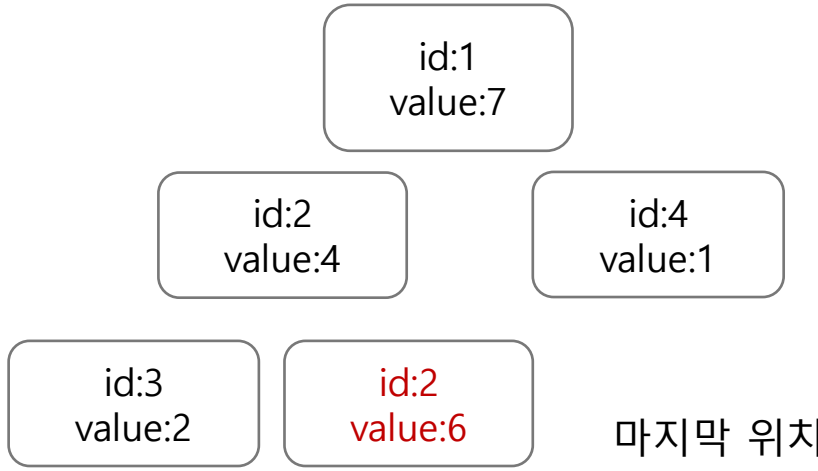
해당 정보들은 heap[1]에 왔을 때, erase와 마찬가지로 실제 정보와 비교하여 유효성 판별이 가능하다.

실제 유효한 정보를 저장하는 배열

S

1	2	3	4	5
7	6	2	1	-1

Max Heap



마지막 위치에 push

Lazy Update Example

2번을 6으로 update

S에는 실시간으로 값을 바꿔준다.

heap에는 업데이트된 값으로 push 해 준다.

heap에 똑같은 index 2에 대한 정보가 중복되어 들어간다.

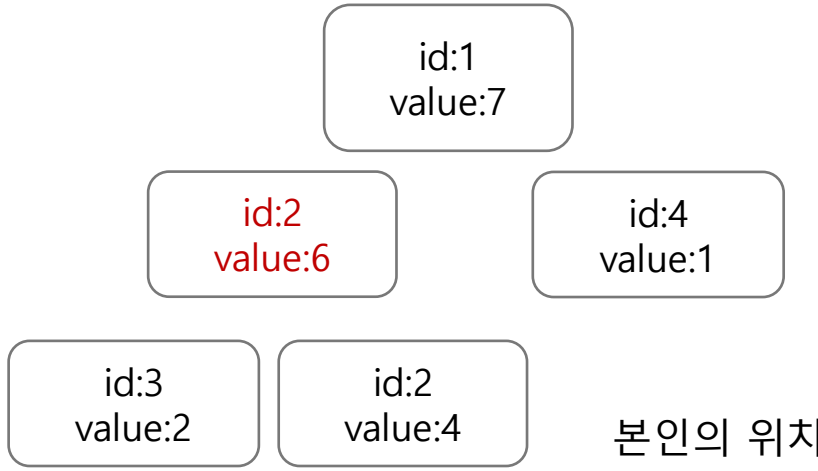
해당 정보들은 heap[1]에 왔을 때, erase와 마찬가지로 실제 정보와 비교하여 유효성 판별이 가능하다.

실제 유효한 정보를 저장하는 배열

S

1	2	3	4	5
7	6	2	1	-1

Max Heap



본인의 위치를 찾아감

Lazy Update 중복 처리

앞 example에서 우선순위 값 3개를 구해야 할 때,

1. (유효한 top 구한 뒤, 기록하고 pop) 과정을 3번 반복
2. 처리 후에 3개 다시 push

but, id:5 인 값의 value가 9->5->9 로 바뀐 경우라면,
{5, 9} 가 2번 들어가 있으므로 유효한 top 3개를 구하는 과정에서 {5, 9} 가 두 번 포함된다.

따라서, 유효한 top을 구하더라도 직전 뽑아낸 top이랑 중복되는 경우는 버려줘야 한다.

실제 유효한 정보를 저장하는 배열

S

1	2	3	4	5
7	4	2	1	9

Max Heap