double ended priority queues

(Single-ended) Priority Queue

- 각 원소가 연관된 우선순위(*priority*) 키를 갖고 있는 원소들의 모임

- Max-priority queue
  1) insert an element with arbitrary key
  2) delete an element with the largest key

- Min-priority queue
  1) insert an element with arbitrary key
  2) delete an element with the smallest key

Double-ended priority queue

1) insert an element with arbitrary key
2) delete an element with the largest key
3) delete an element with the smallest key

- max-heap supports operations 1) and 2)
- min-heap supports operations 1) and 3)

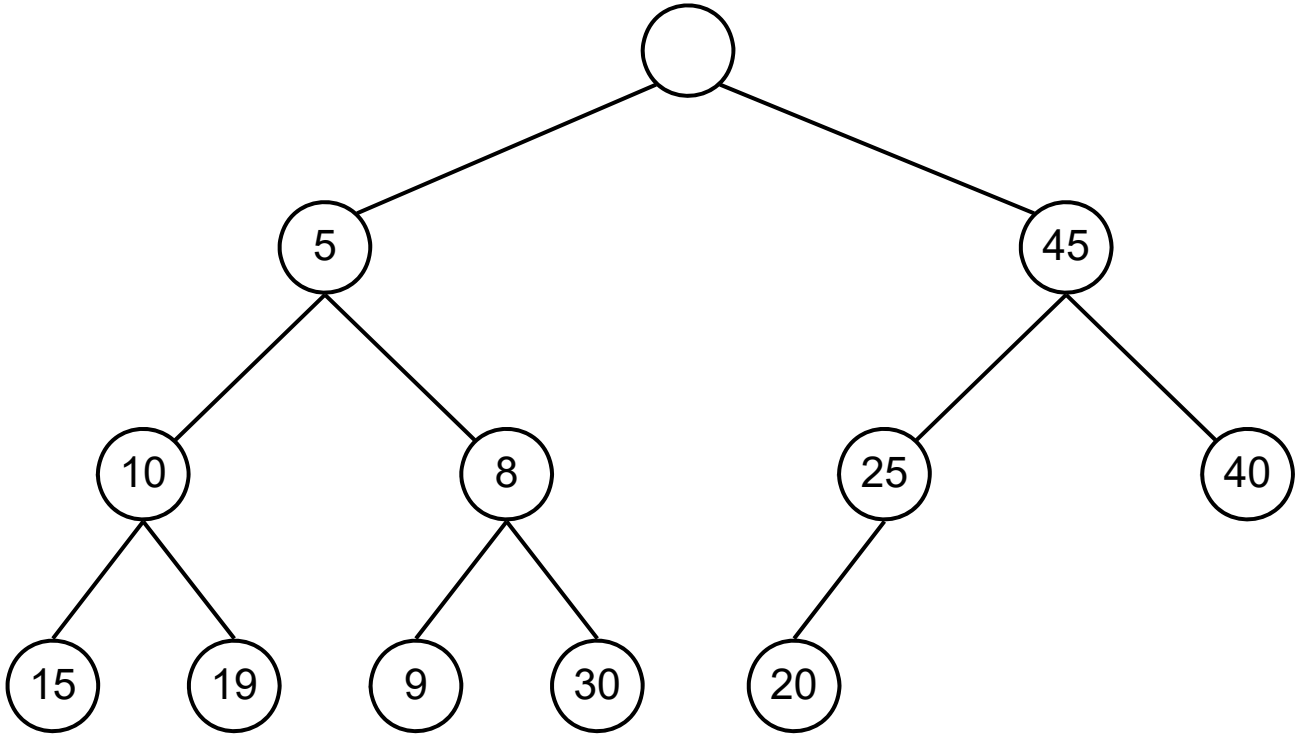
- 디프(Deap: double ended heap) supports all of the above operations

디프(Deap)

- 완전이진트리
- 공백이거나 다음의 성질을 만족한다

1. 루트에는 원소가 없다
2. 왼쪽 서브트리는 최소히프이다
3. 오른쪽 서브트리는 최대히프이다
4. 왼쪽서브트리의 임의의 노드를 i라고 하고, 이에 대응하는 오른쪽 서브트리의 노드를 j라 하자. 만약 대응되는 원소가 없으면 i의 부모와 대응되는 노드를 j라 하자. 노드 i의 키값은 노드 j의 키값보다 작거나 같다.

Deap



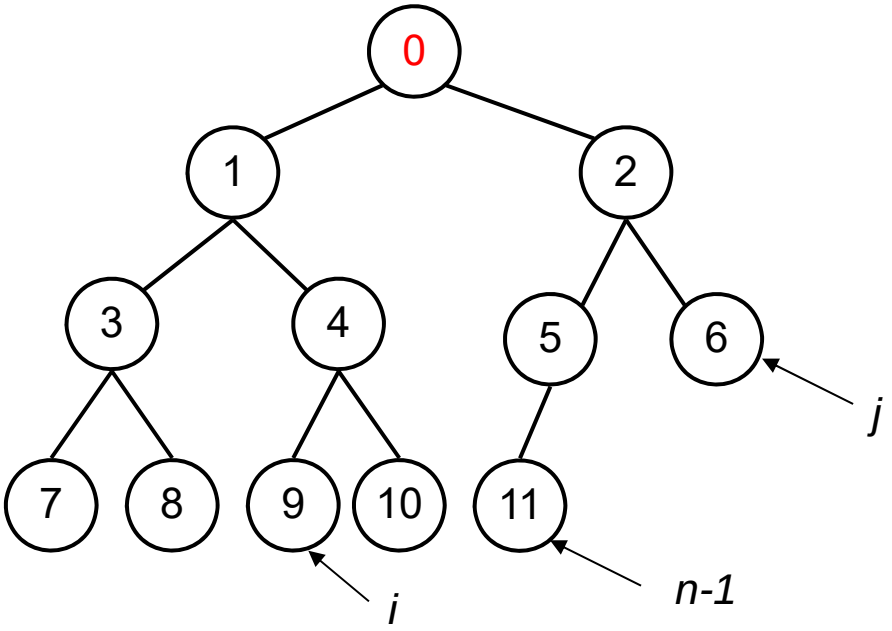
Deaps

- 노드 i와 j사이의 관계

$$j = i + 2^{\lfloor \log_2 (i+1) \rfloor - 1};$$

$$\text{if } (j > n - 1)$$

$$j = (j - 1)/2;$$



Insertion into a deap

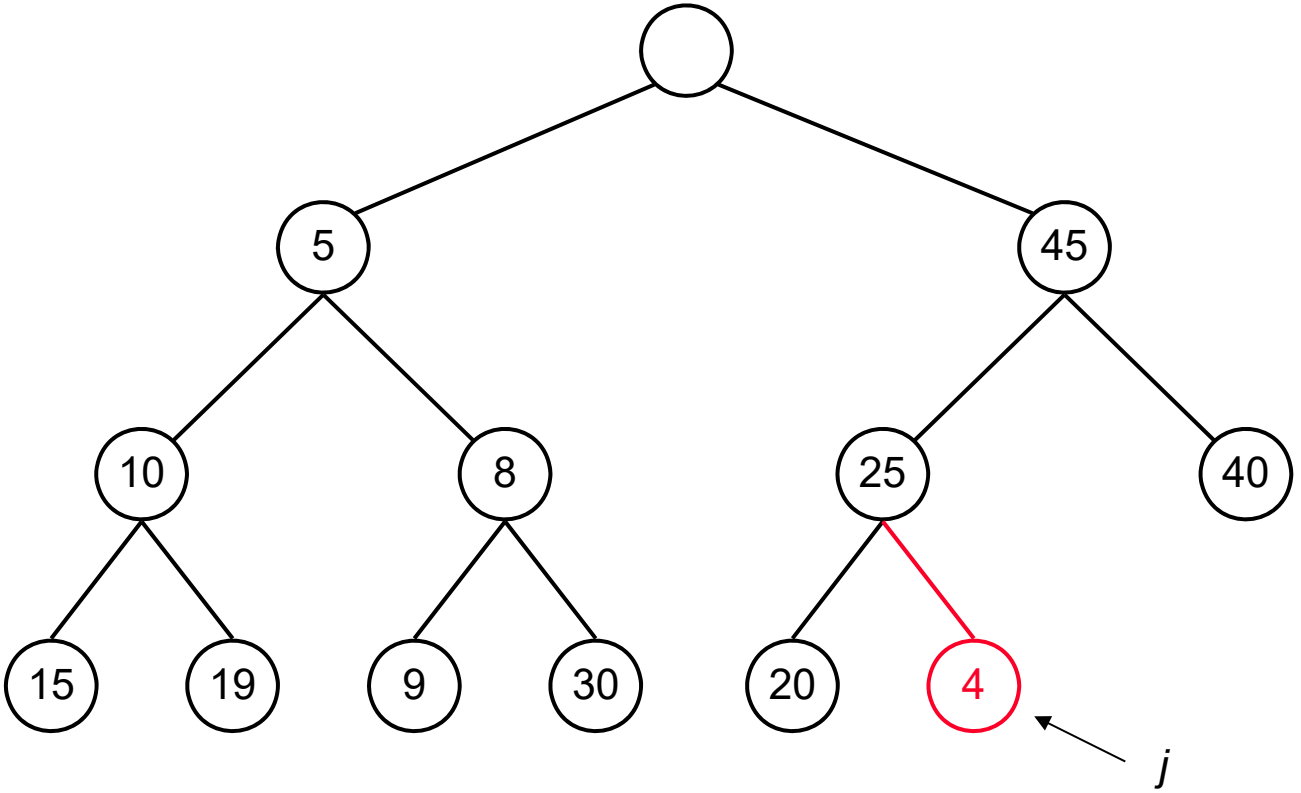
- 디프에 삽입하는 과정에서 사용하는 함수들
  1. max_heap(*i*): 이 함수는 *i* 가 디프의 최대히프에 있으면 TRUE를 돌려준다
  2. m=min_partner(i) : 이 함수는 최대히프의 위치 i에 대응하는 최소히프의 노드를 계산한다. 이 값은
     $$i - 2^{\lfloor \log_2(i+1) \rfloor - 1}$$
  3. max_partner(*i*) : 이 함수는 최소히프의 위치 i에 대응하는 최대히프의 노드를 계산한다. 이 값은
     $$j = i + 2^{\lfloor \log_2(i+1) \rfloor - 1}$$ if j>n-1 then $$j = (j-1)/2$$
  4. min_inert와 max_insert : 이 함수들은 각각 최소히프와 최대히프의 지정된 위치에 원소를 삽입한다

Insertion into a deap

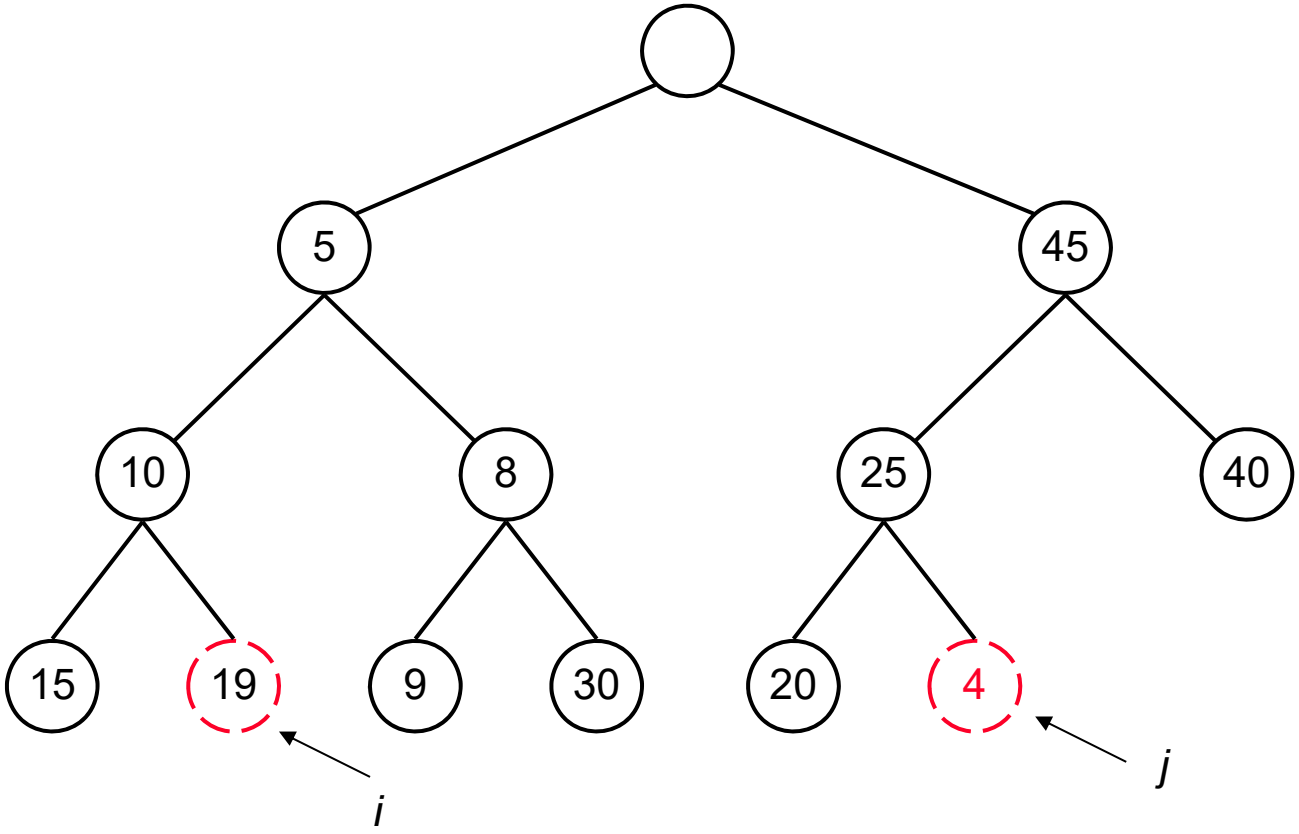
- 디프에 삽입하는 과정

1. j가 디프에 생기는 새로운 노드라고 하자. j가 최소히프의 노드인지 최대히프의 노드인지 체크한다
2. 대응되는 히프의 노드 위치 i를 구한다.
3. i와 j의 키가 디프의 조건을 만족하는지 비교한다. 만족하지 않으면 i와 j의 키값을 교환한다
4. min_insert() 또는 max_insert()를 수행한다

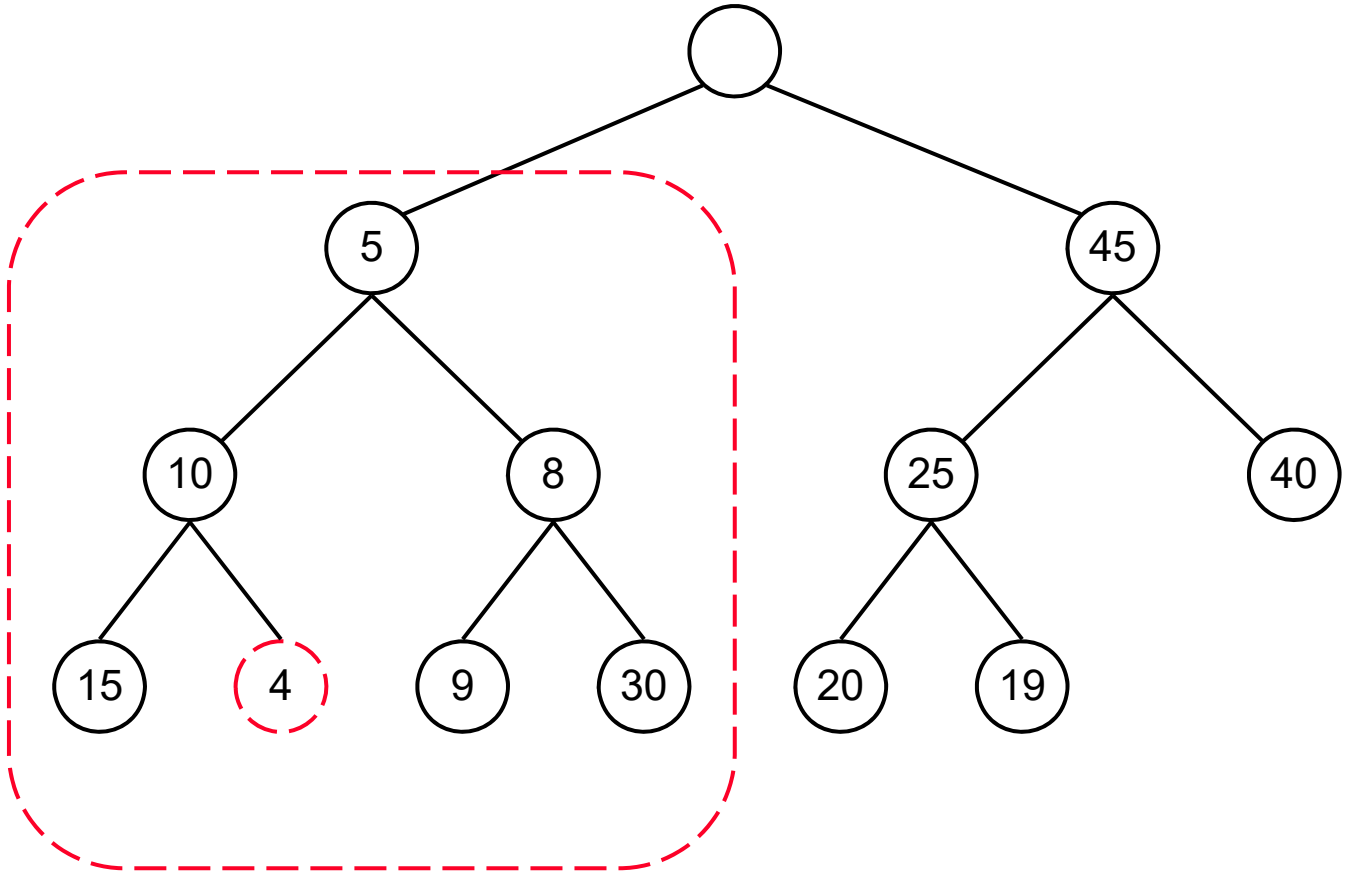
Insertion into a deap (예제 1)



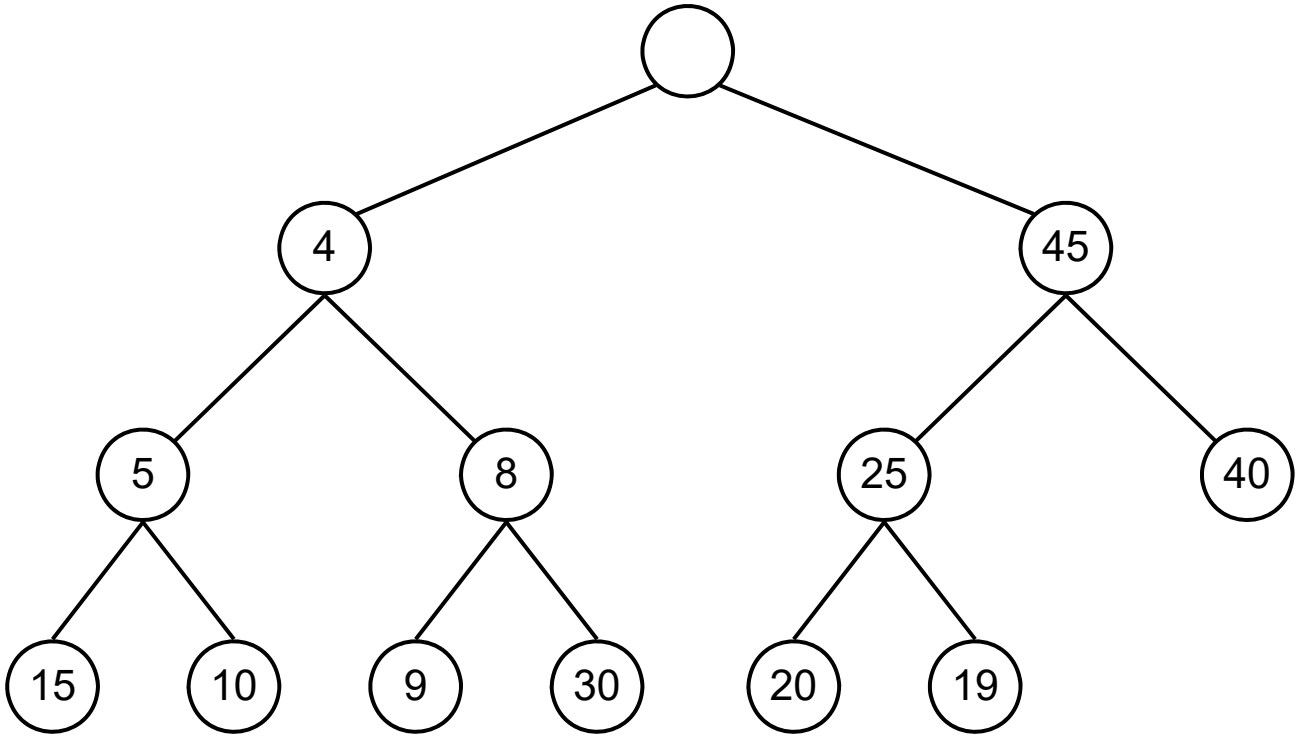
Insertion into a deap



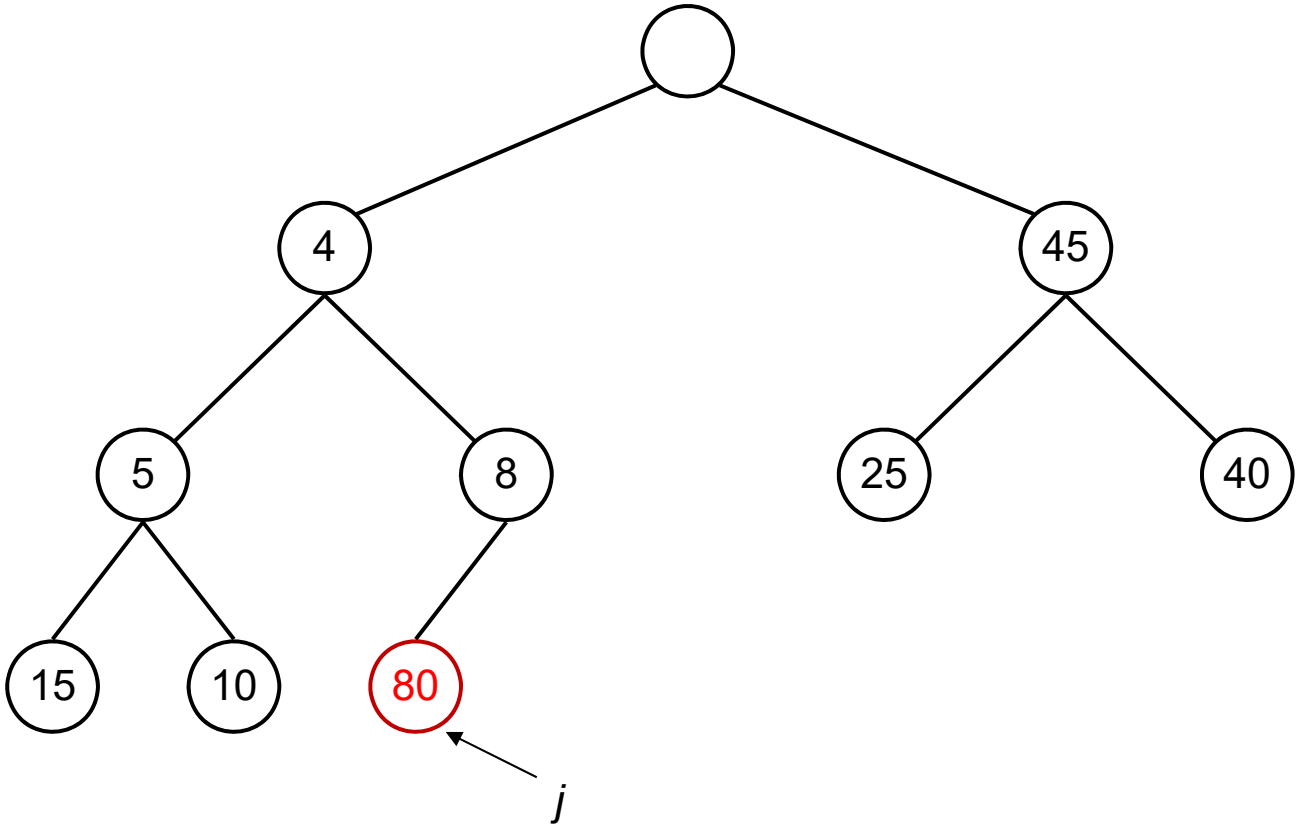
Insertion into a deap



Insertion into a deap



Insertion into a deap (예제2)



Insertion into a deap

- 디프의 높이가 O(log n)이기 때문에 복잡도는 O(log n)이다

Deletion of min in a deap

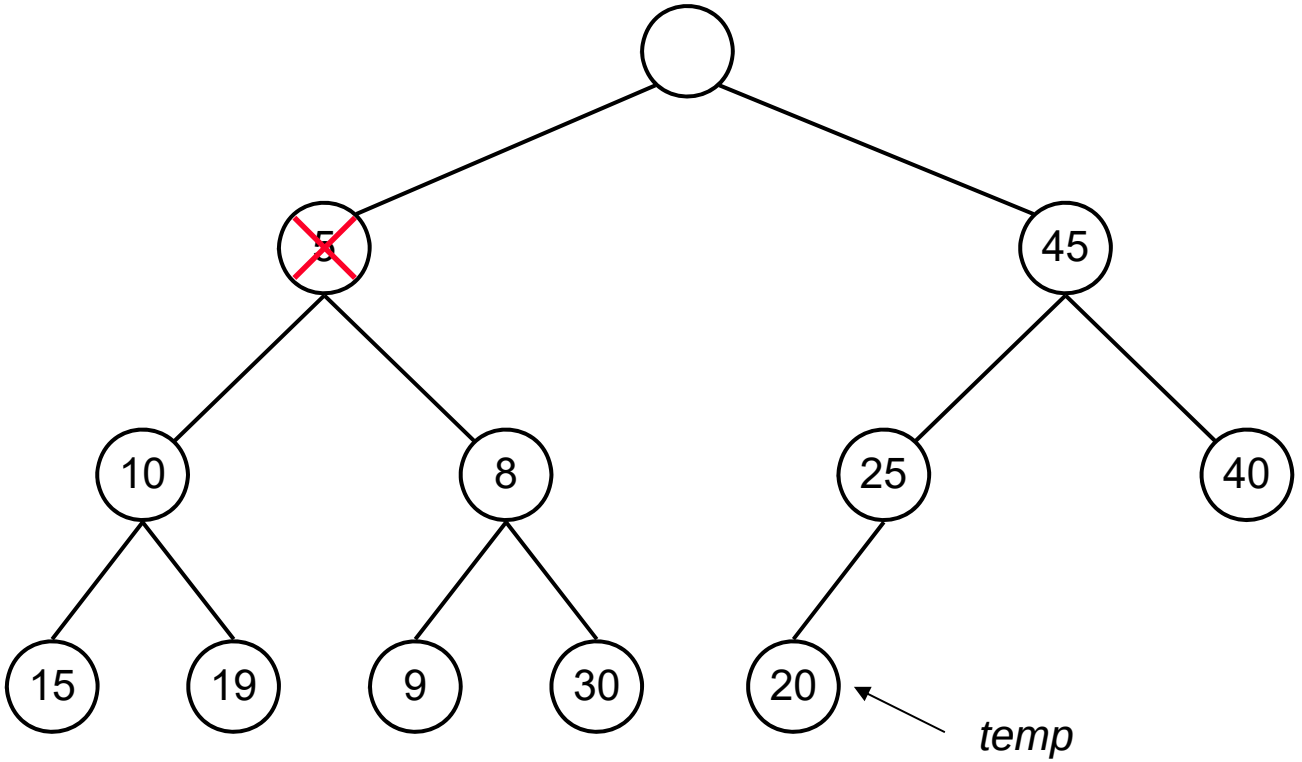
1. Save the last element as temp and remove this node from deap.

2. Find the node with smaller key from the children of removed minimum element and loop down until reaching leaves.

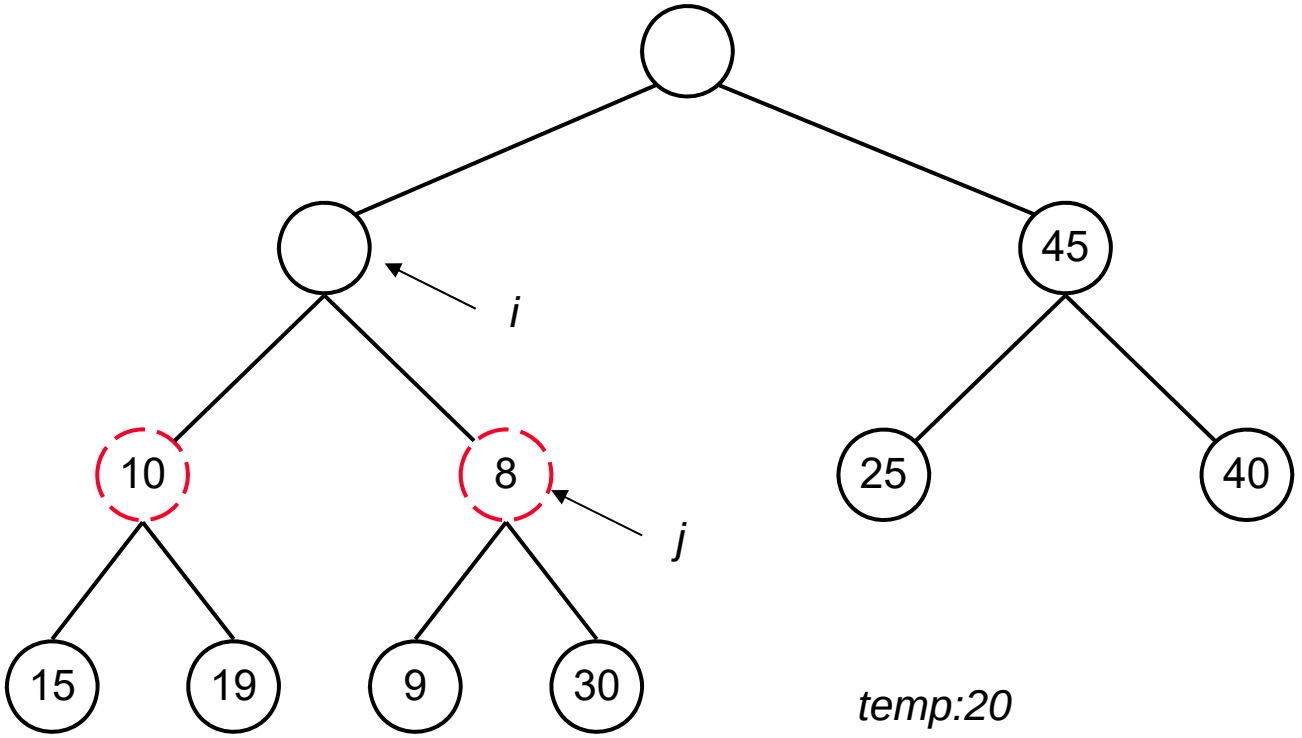
3. Insert temp into the left subtree of deap.

4. After comparing temp with its max_partner, perform modified_deap_insert if needed

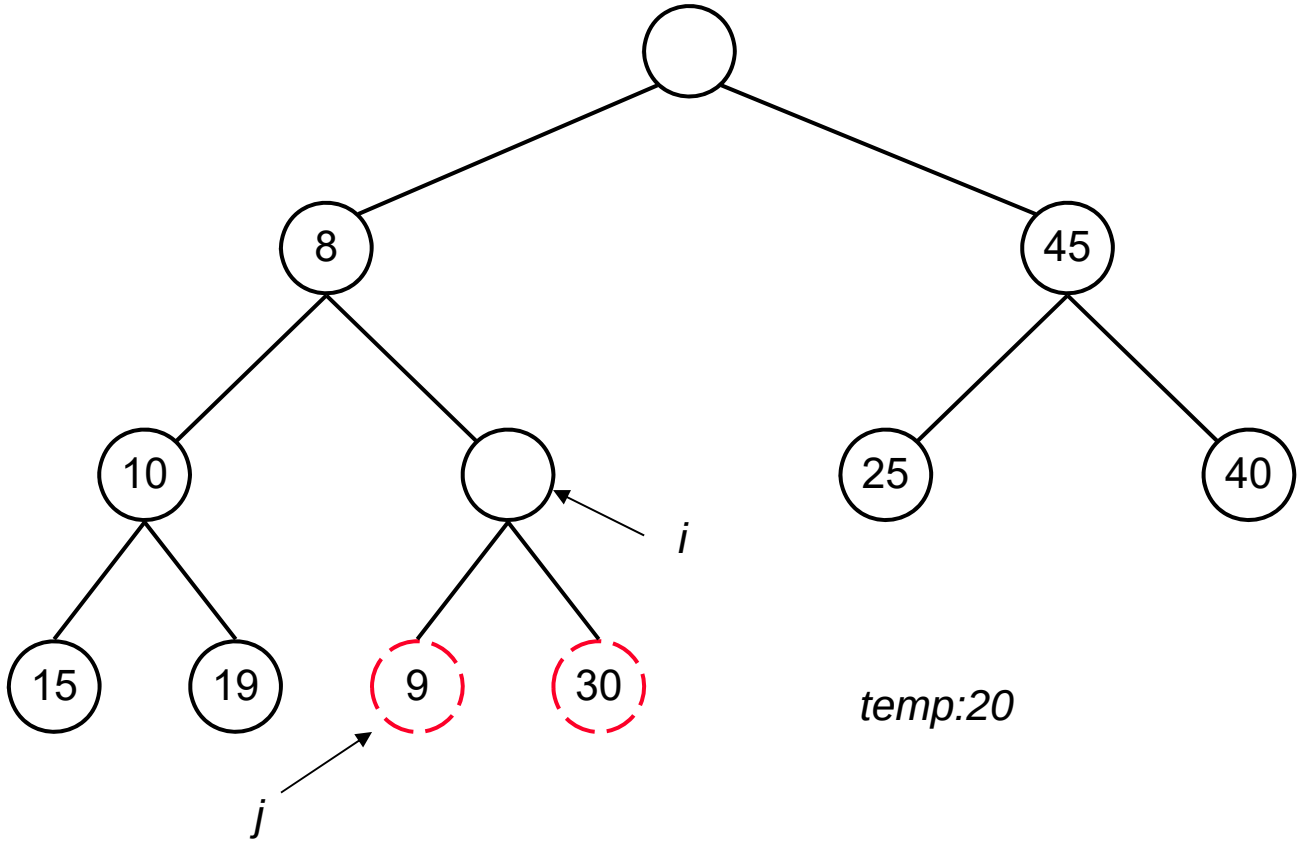
Deletion of min element



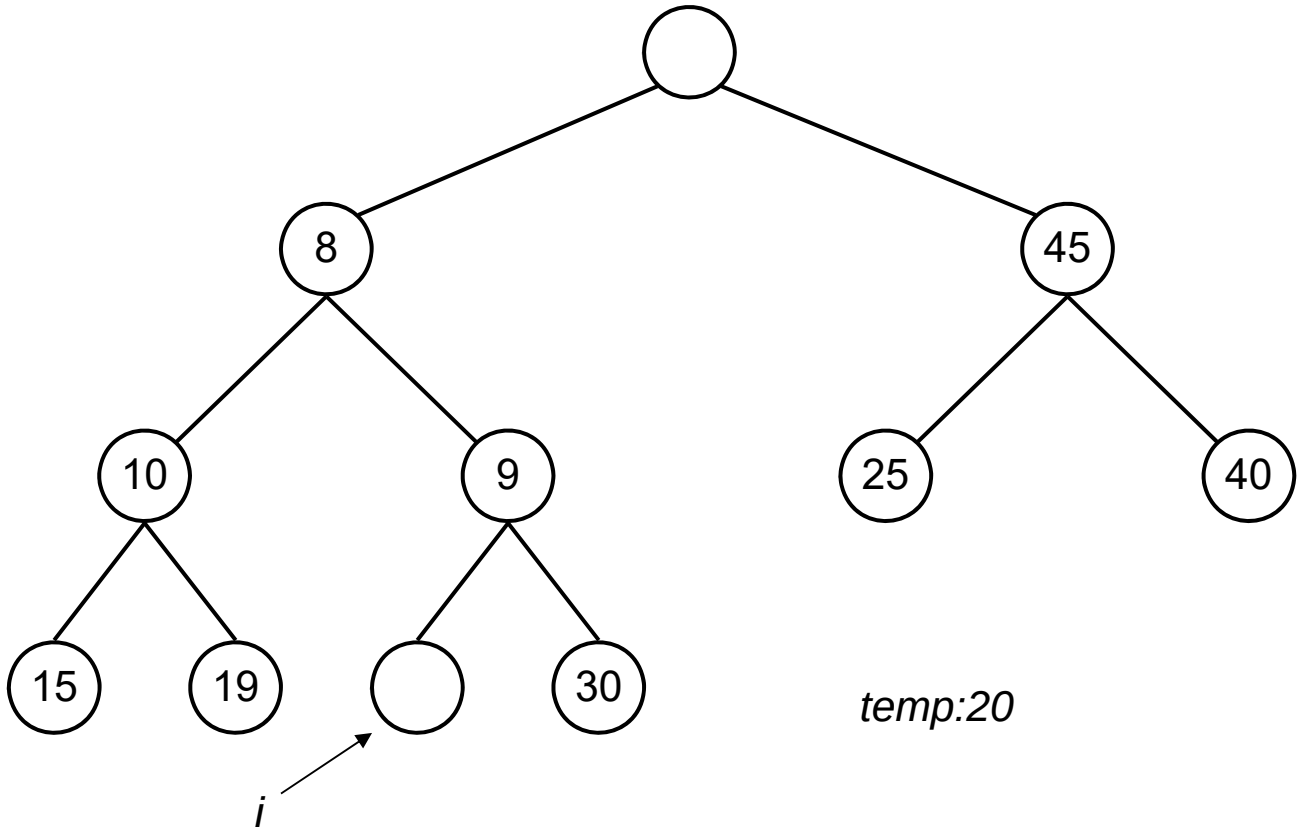
Deletion of min element



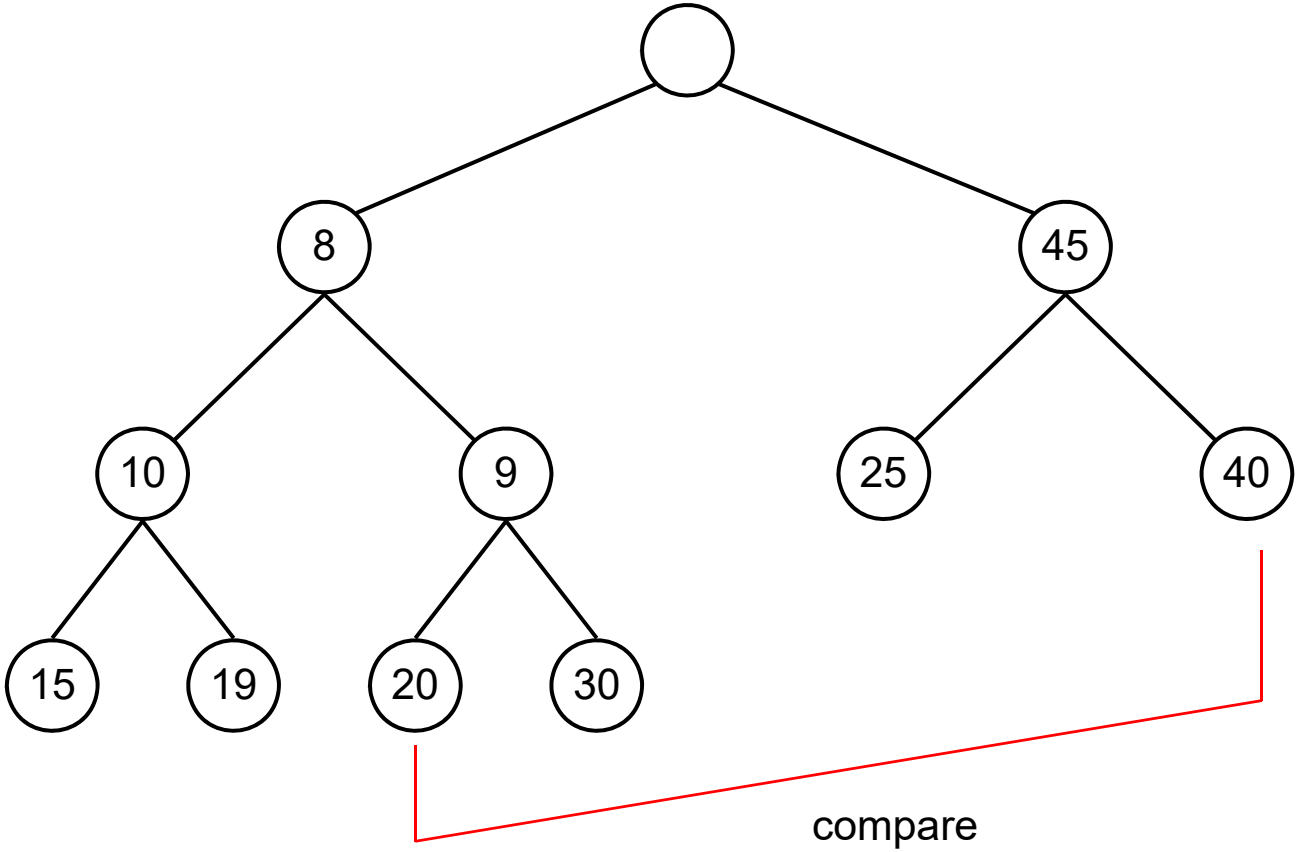
Deletion of min element



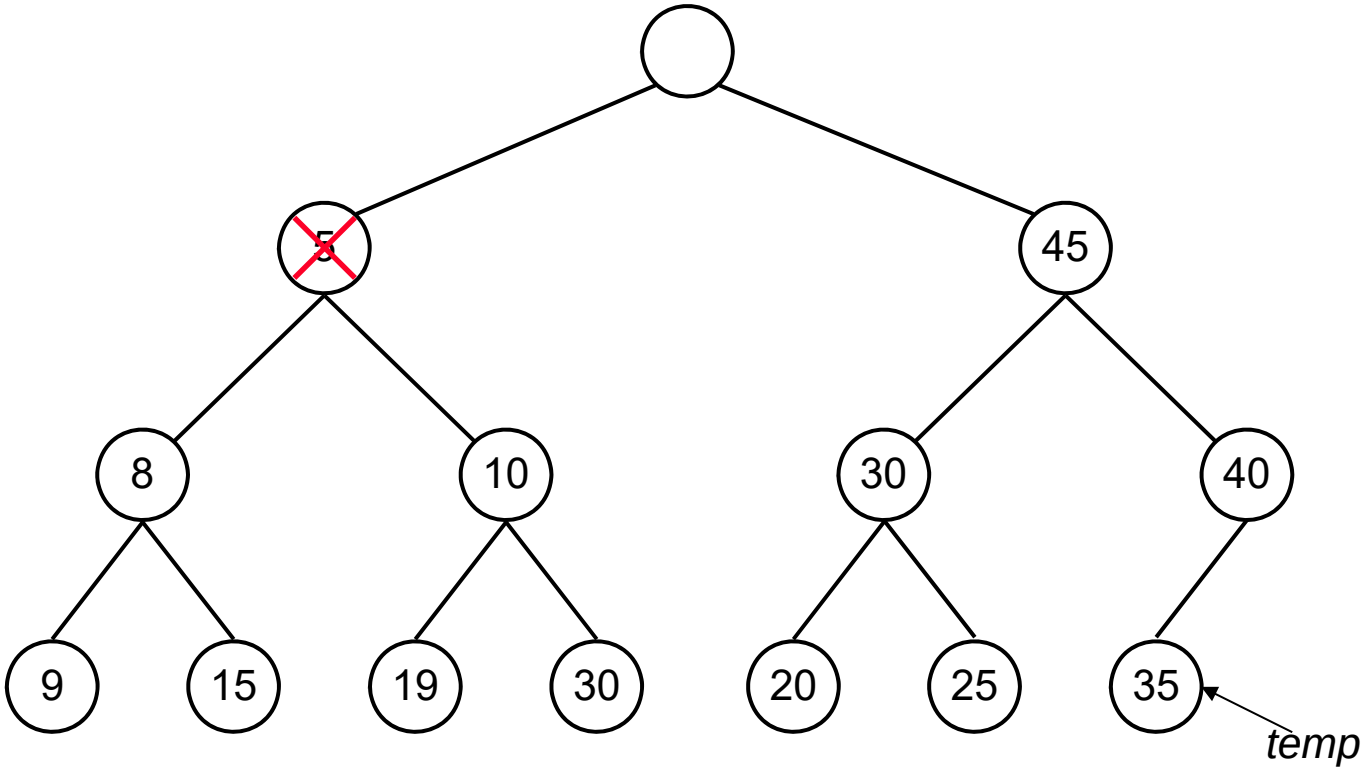
Deletion of min element



Deletion of min element



Exercise: Deletion of min element



Deletion of min element

- 디프의 높이가 O(log n)이기 때문에 복잡도는 O(log n)이다

Deletion of max in a deap

1. 마지막 원소를 제거하여 temp에 임시로 저장한다

2. max heap에서 루트를 제거 후 자식들 중 큰 것을 부모 자리에 올린다. 이 과정을 반복하여 리프 노드까지 내려간다.

3. 리프 노드의 min-partner에 해당하는 원소와 그 자식들 중에서 가장 큰 원소를 x라고 할 때, temp에 저장된 원소를 x와 비교한다.

4. x가 더 크면 x를 리프노드 자리로 옮기고 temp에 있던 원소를 x가 있던 자리에서 min-insert를 한다. temp에 저장된 원소가 더 크면 그 원소를 리프노드 자리에 max-insert 한다

Deletion of max element (예제)