

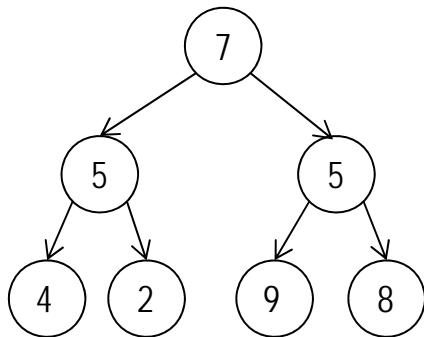
# 이진힉과 우선순위큐

# 이진 힙(binary heap)

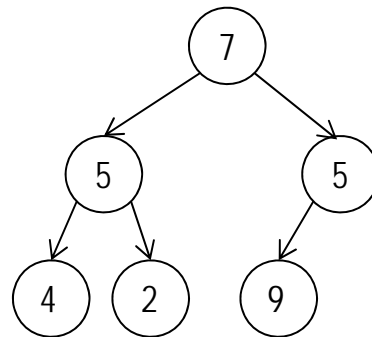
[https://en.wikipedia.org/wiki/Binary\\_heap](https://en.wikipedia.org/wiki/Binary_heap), CC-BY-SA



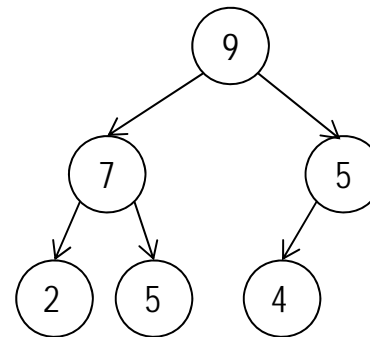
- 이진힙은 우선순위큐(priority queue)를 구현하는 자료구조이다.
- 최대이진힙(binary max heap)은 완전이진트리(complete binary tree)이면서, 각 노드의 키 값이 자식 노드들의 키 값보다 크거나 같은 조건을 만족한다.



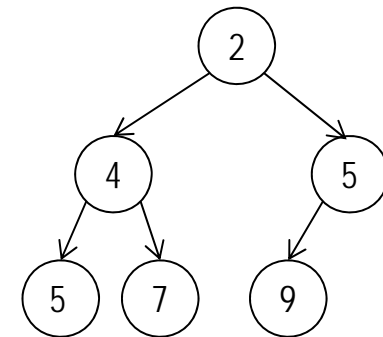
포화이진트리(full binary tree)



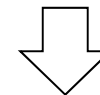
완전이진트리(complete binary tree)



최대이진힙



최소이진힙



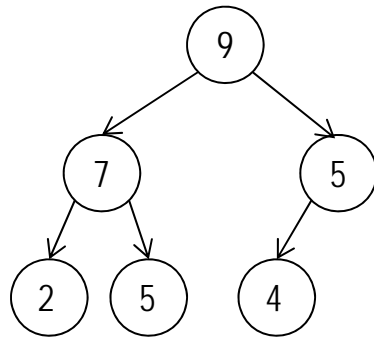
0	1	2	3	4	5
9	7	5	2	5	4

최대이진힙은 완전이진트리이므로 배열 내 효율적 저장 가능

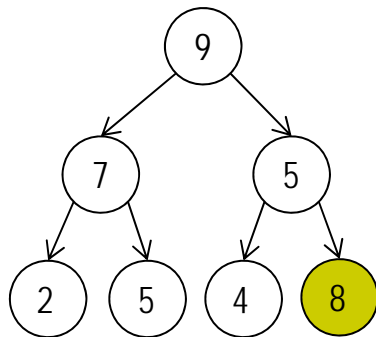
- 배열  $i$ 번째 위치 노드의 왼쪽자식노드 위치  $\rightarrow (2 * i) + 1$
- 배열  $i$ 번째 위치 노드의 오른쪽자식노드 위치  $\rightarrow (2 * i) + 2$
- 배열  $i$ 번째 위치 노드의 부모노드 위치  $\rightarrow (i - 1) / 2$

# 이진 힙(binary heap): 삽입

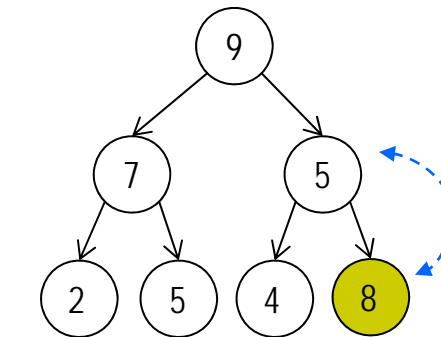
[https://en.wikipedia.org/wiki/Binary\\_heap](https://en.wikipedia.org/wiki/Binary_heap), CC-BY-SA



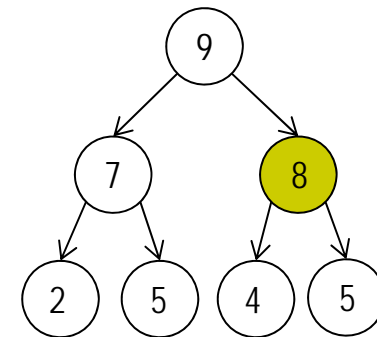
최대힙에 키 값 8 삽입



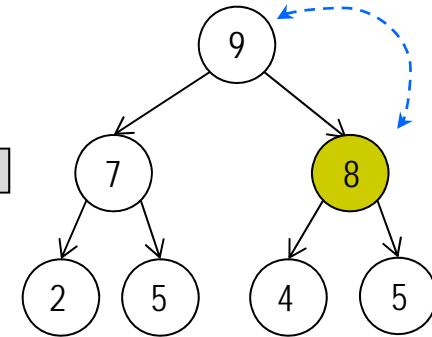
삽입할 노드를 마지막 위치에 추가



부모노드와 키 값 비교하여  
최대힙조건 불만족 시 교환



부모노드와 키 값 비교하여  
최대힙조건 불만족 시 교환

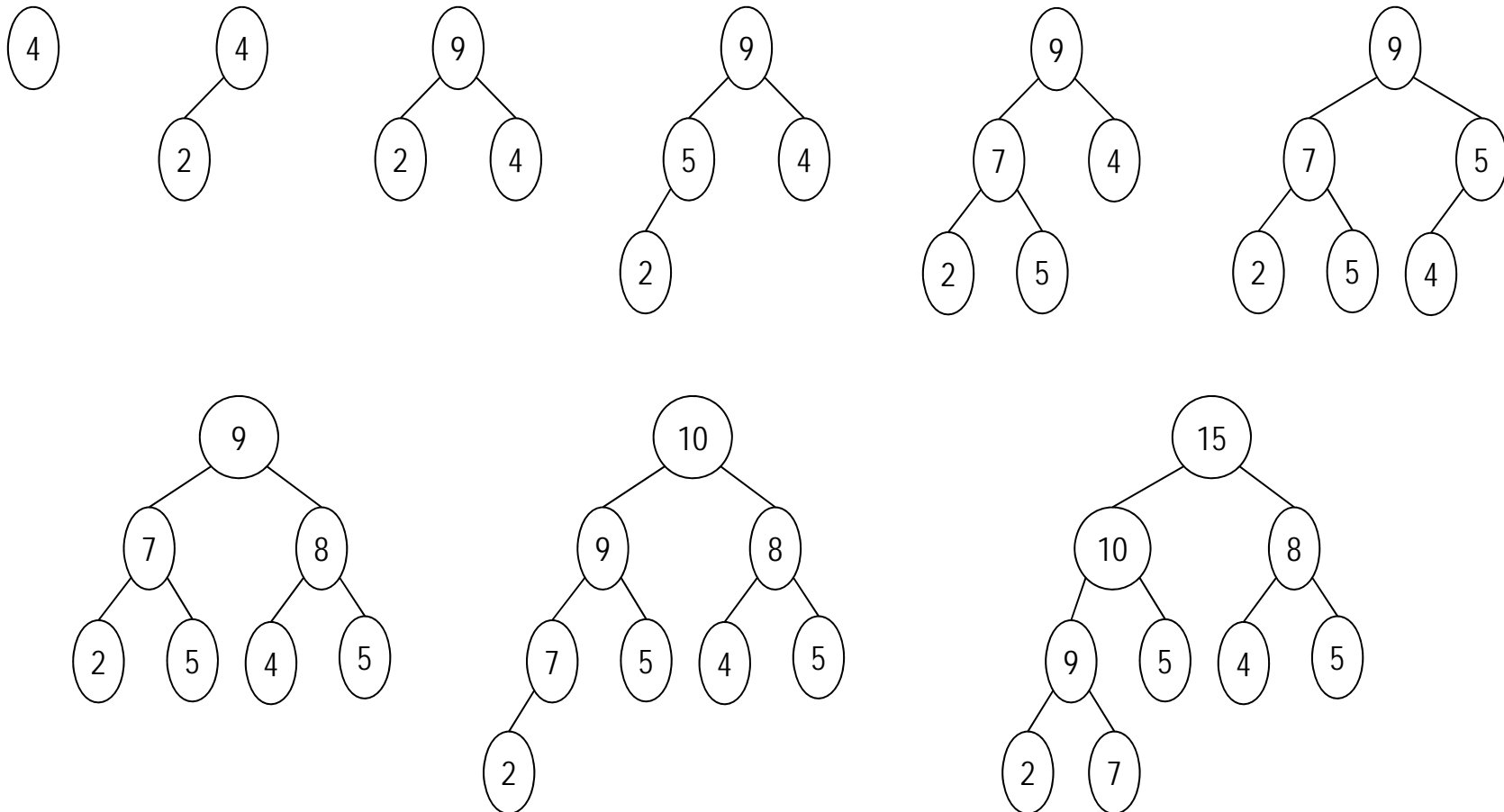


# 이진 힙(binary heap): 삽입

[https://en.wikipedia.org/wiki/Binary\\_heap](https://en.wikipedia.org/wiki/Binary_heap), CC-BY-SA



4, 2, 9, 5, 7, 5, 8, 10, 15 순으로 최대이진힙에 자료 삽입



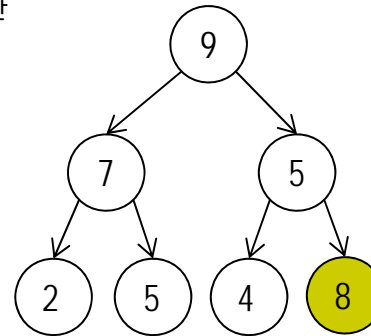
# 이진 힙(binary heap): 삽입

[https://en.wikipedia.org/wiki/Binary\\_heap](https://en.wikipedia.org/wiki/Binary_heap), CC-BY-SA  
<https://algs4.cs.princeton.edu/24pq/MaxPQ.java.html>, GPLv3

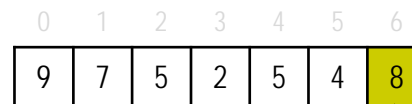


```
public class SimpleHeap {
    int last=-1, MaxHeapSize=4;
    int heap[]=new int[MaxHeapSize];
    private void swap(int m, int n) { int temp=heap[m]; heap[m]=heap[n]; heap[n]=temp; }
    private void resize() {
        MaxHeapSize*=2;
        heap=Arrays.copyOf(heap, MaxHeapSize);
    }
    public void add(Integer data) {
        if(last+1==MaxHeapSize) resize();
        heap[++last]=data; // heap의 마지막 노드 다음 위치에 새 자료 삽입
        for (int child=last; child>0; ) { // 마지막 노드를 heapify-up
            int parent=(child-1)/2;
            if(heap[child]<=heap[parent]) break; // 부모 키와 최대힙조건 검사
            swap(child, parent); // 조건 불만족 시 교환
            child=parent;
        }
    }
    public int remove() { ... }
    @Override
    public String toString() { return Arrays.toString(heap); }
}
```

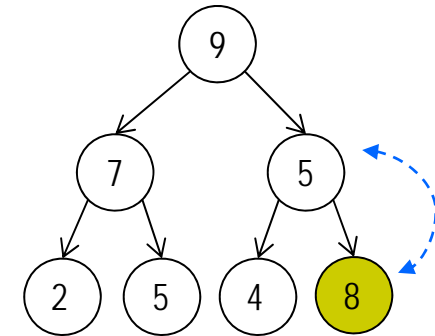
```
public class Test {
    public static void main(String[] args) {
        SimpleHeap heap=new SimpleHeap();
        for (int i = 0; i < 10; i++) {
            heap.add(i);
            System.out.println(heap);
        }
        for (int i = 0; i < 10; i++) {
            System.out.println(heap.remove()+">"+heap);
        }
    }
}
```



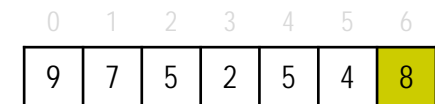
삽입할 노드를 마지막 위치에 추가



last



부모노드와 최대힙조건 검사 & 교환



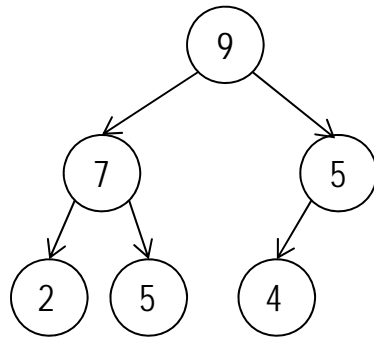
parent = (child-1)/2

child

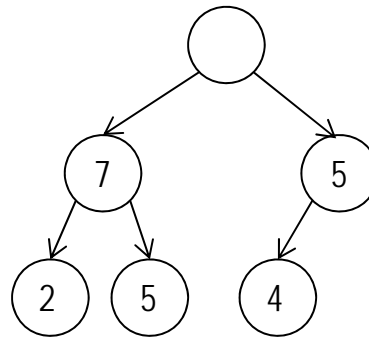
시간복잡도  $O(\log n)$

# 이진 힙(binary heap): 삭제

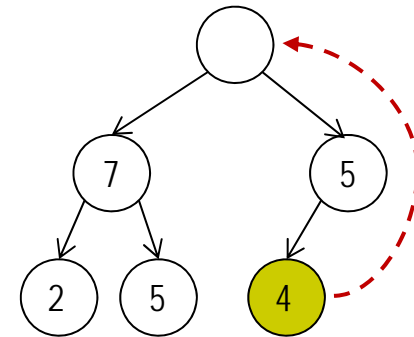
[https://en.wikipedia.org/wiki/Binary\\_heap](https://en.wikipedia.org/wiki/Binary_heap), CC-BY-SA  
<https://algs4.cs.princeton.edu/24pq/MaxPQ.java.html>, GPLv3



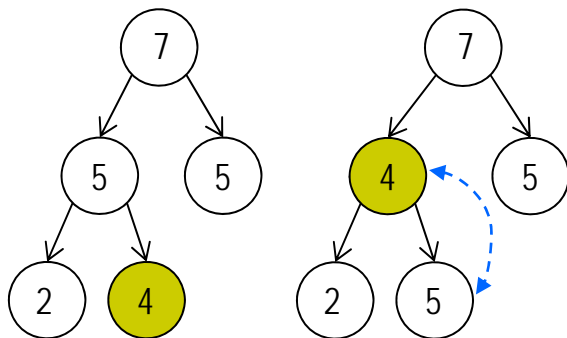
최대힙에서 최대값 삭제



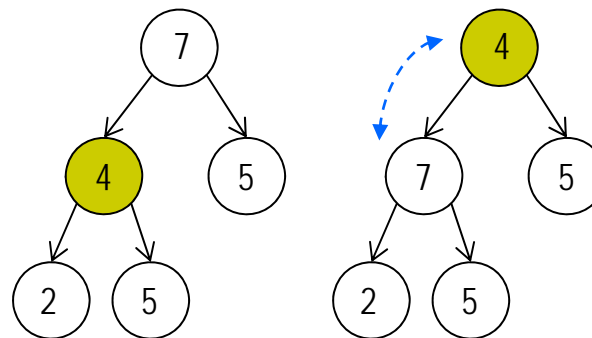
루트 노드 값 삭제



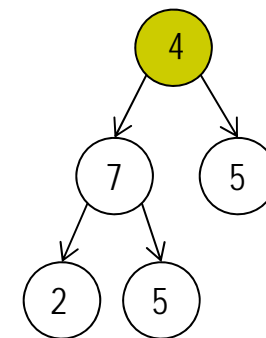
마지막 노드를 루트로 이동



두 자식노드 키 값 중 큰 값과 비교 후  
최대힙조건 불만족 시 교환



두 자식노드 키 값 중 큰 값과 비교 후  
최대힙조건 불만족 시 교환



마지막 노드를 루트로 이동



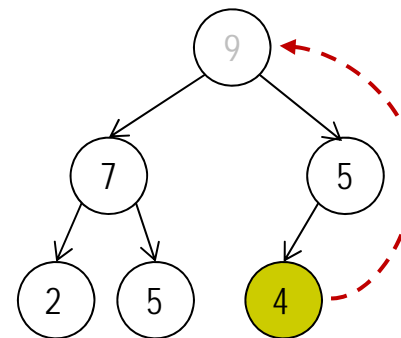
# 이진 힙(binary heap): 삭제

[https://en.wikipedia.org/wiki/Binary\\_heap](https://en.wikipedia.org/wiki/Binary_heap), CC-BY-SA  
<https://algs4.cs.princeton.edu/24pq/MaxPQ.java.html>, GPLv3

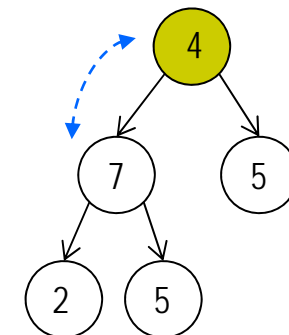
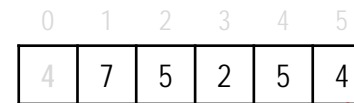


```
public class SimpleHeap {
    int last=-1, MaxHeapSize=4;
    int heap[]=new int[MaxHeapSize];
    private void swap(int m, int n) { int temp=heap[m]; heap[m]=heap[n]; heap[n]=temp; }
    private void resize() { ... }
    public void add(Integer data) { ... }
    public int remove() {
        if(last<0) throw new RuntimeException("heap empty");
        int data=heap[0]; // root 노드 자료 추출
        heap[0]=heap[last--]; // 마지막 노드 루트로 이동 & 크기 1 감소
        for (int parent=0, child=2*parent+1; child<=last; parent=child, child=2*parent+1) { // 루트를 heapify-down
            if(child<last && heap[child]<heap[child+1]) child++;
            if(heap[child]<=heap[parent]) break;
            swap(child,parent);
        }
        return data;
    }
    @Override
    public String toString() { return Arrays.toString(heap); }
}
```

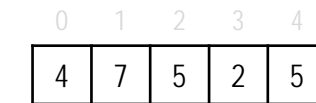
시간복잡도  $O(\log n)$



마지막 노드를 루트로 이동



키 값이 큰 자식노드와 비교 & 교환



parent      child = (2\*parent)+1

# 이진 힙(binary heap): 생성

[https://en.wikipedia.org/wiki/Binary\\_heap](https://en.wikipedia.org/wiki/Binary_heap), CC-BY-SA  
<https://algs4.cs.princeton.edu/24pq/MaxPQ.java.html>, GPLv3

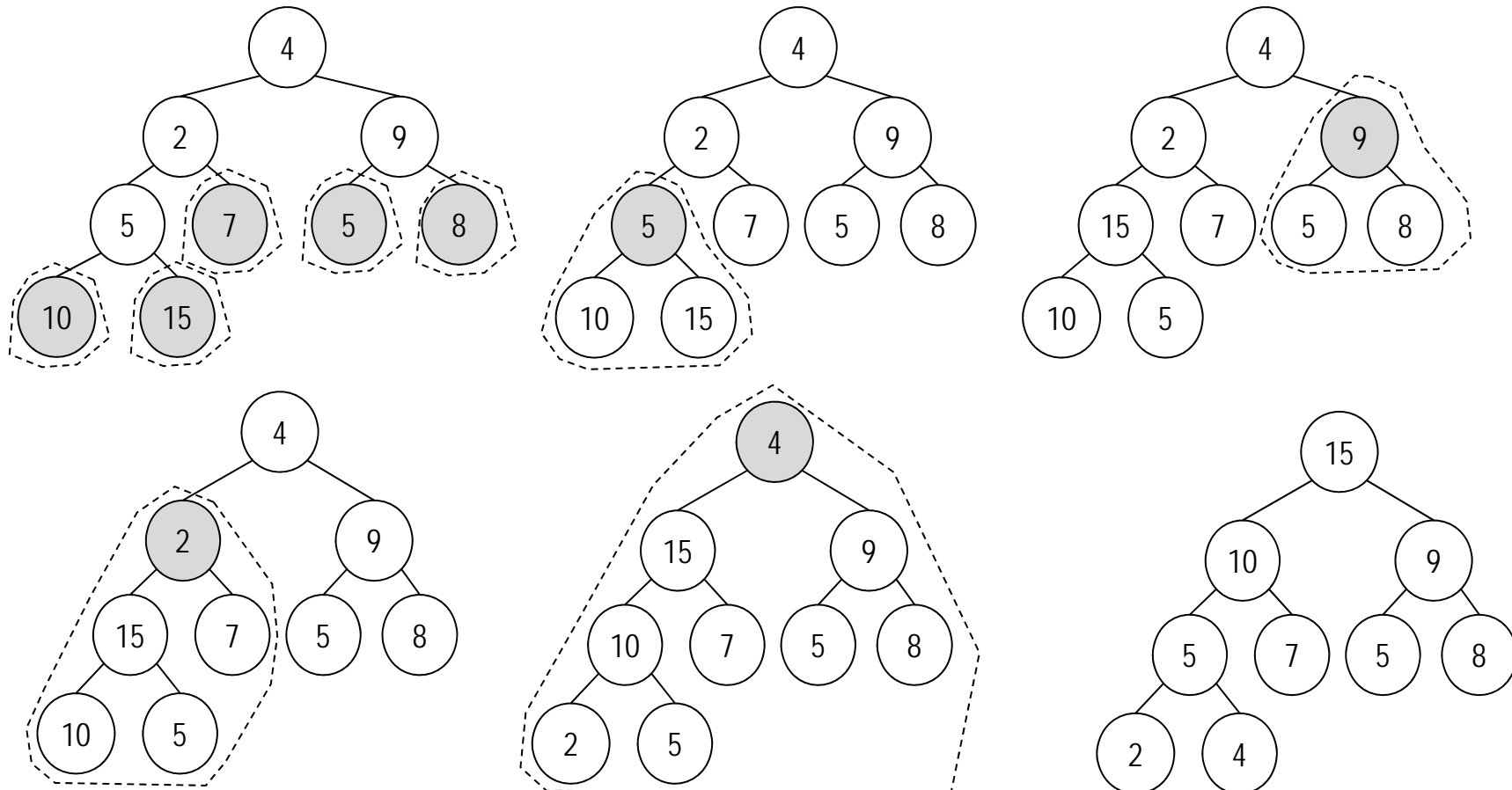


최초 배열 

4	2	9	5	7	5	8	10	15
---	---	---	---	---	---	---	----	----

- 단말노드들은 이미 그 노드를 루트로 하는 이진힙 조건 만족함
- 각 비단말노드에 대해 그 노드를 루트로 하는 이진힙 생성 절차 반복

시간복잡도  $O(n)$





# 이진 힙 vs. 균형이진트리

## 힙(이진)

[https://en.wikipedia.org/wiki/Binary\\_heap](https://en.wikipedia.org/wiki/Binary_heap), CC-BY-SA  
[https://en.wikipedia.org/wiki/Binary\\_search\\_tree](https://en.wikipedia.org/wiki/Binary_search_tree), CC-BY-SA

- 생성:  $O(n)$
- 최소값 탐색:  $O(1)$
- 최소값 삭제:  $O(\log n)$
- 삽입:  $O(\log n)$
- 균형이진트리보다 공간 효율적 (child link 불필요)

## 균형이진트리

- 생성:  $O(n \log n)$
- 삽입:  $O(\log n)$
- 삭제:  $O(\log n)$
- 탐색:  $O(\log n)$

## References

- ✚ C로 쓴 자료구조론 (Fundamentals of Data Structures in C, Horowitz et al.). 이석호 역. 사이텍미디어. 1993.
- ✚ 쉽게 배우는 알고리즘: 관계 중심의 사고법. 문병로. 한빛아카데미. 2013.
- ✚ C언어로 쉽게 풀어 쓴 자료구조. 천인국 외 2인. 생능출판사. 2017.
- ✚ 프로그래밍 콘테스트 챌린징, Akiba 등 공저, 로드북, 2011.
- ✚ <https://introcs.cs.princeton.edu/>
- ✚ Introduction to Algorithms, Cormen et al., 3rd Edition (The MIT Press)