#11-2. 이진 힙

나정휘

https://justiceHui.github.io/

이진 힙

우선순위 큐 (Priority Queue)

- Queue: 먼저 들어온 데이터가 먼저 나가는 자료구조
- Priority Queue: 우선순위가 높은 데이터가 먼저 나가는 자료구조
  - 원소 삽입
  - 가장 큰 원소 검색
  - 가장 큰 원소 제거

- 우선순위로 BST를 만들고 가장 오른쪽 자손을 반환?
  - BBST는 O(log N)이지만 생각보다 느림
  - log N vs 10 log N 느낌

- 더 좋은 방법은 없을까?

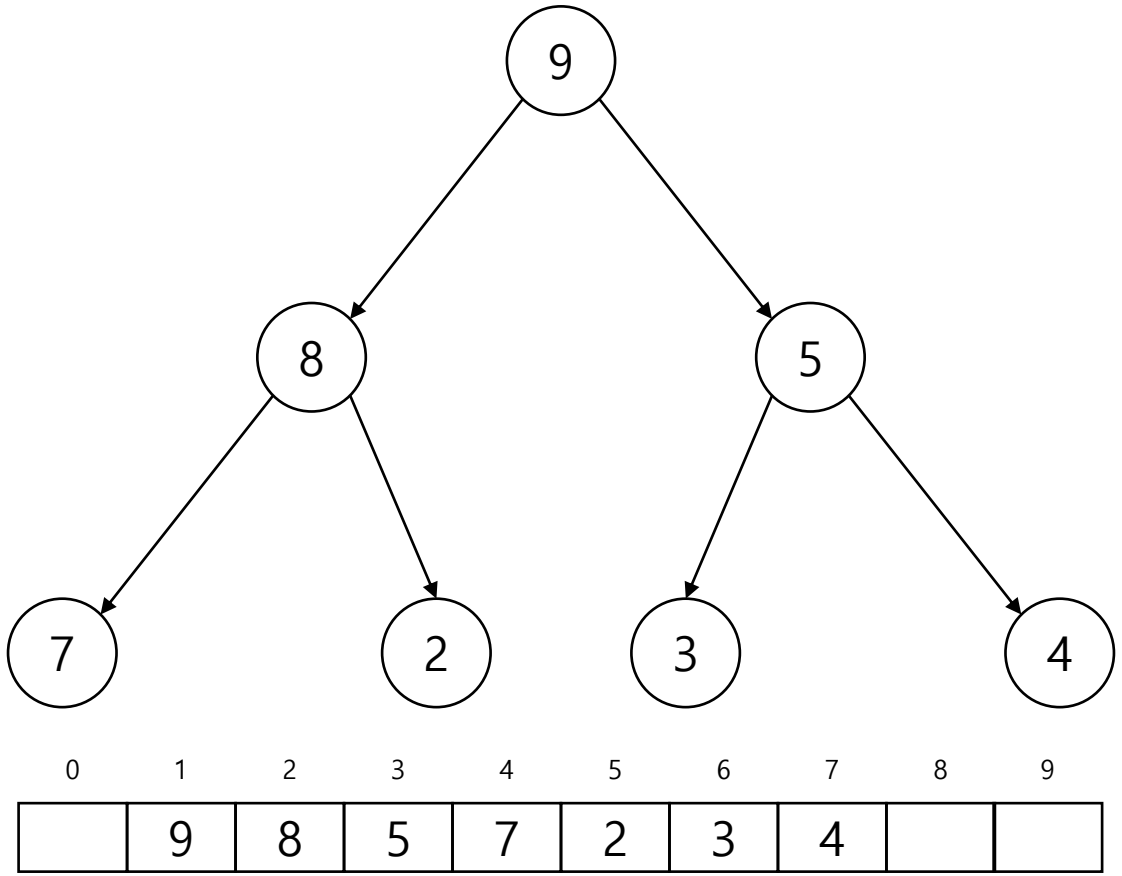
이진 힙

힙 구조

- 각 정점의 값이 자식 정점보다 큰 트리 구조
  - BST보다 제약 조건이 적음
  - 보통 제약 조건이 약하면 더 효율적으로 구현 가능

- 주로 사용하는 건 이진 힙 (Binary Heap)
  - 완전 이진 트리(Complete Binary Tree) 형태
  - 루트 : 1번 인덱스
  - x의 부모 : x / 2
  - x의 왼쪽/오른쪽 자식 : 2x, 2x+1

이진 힙



이진 힙

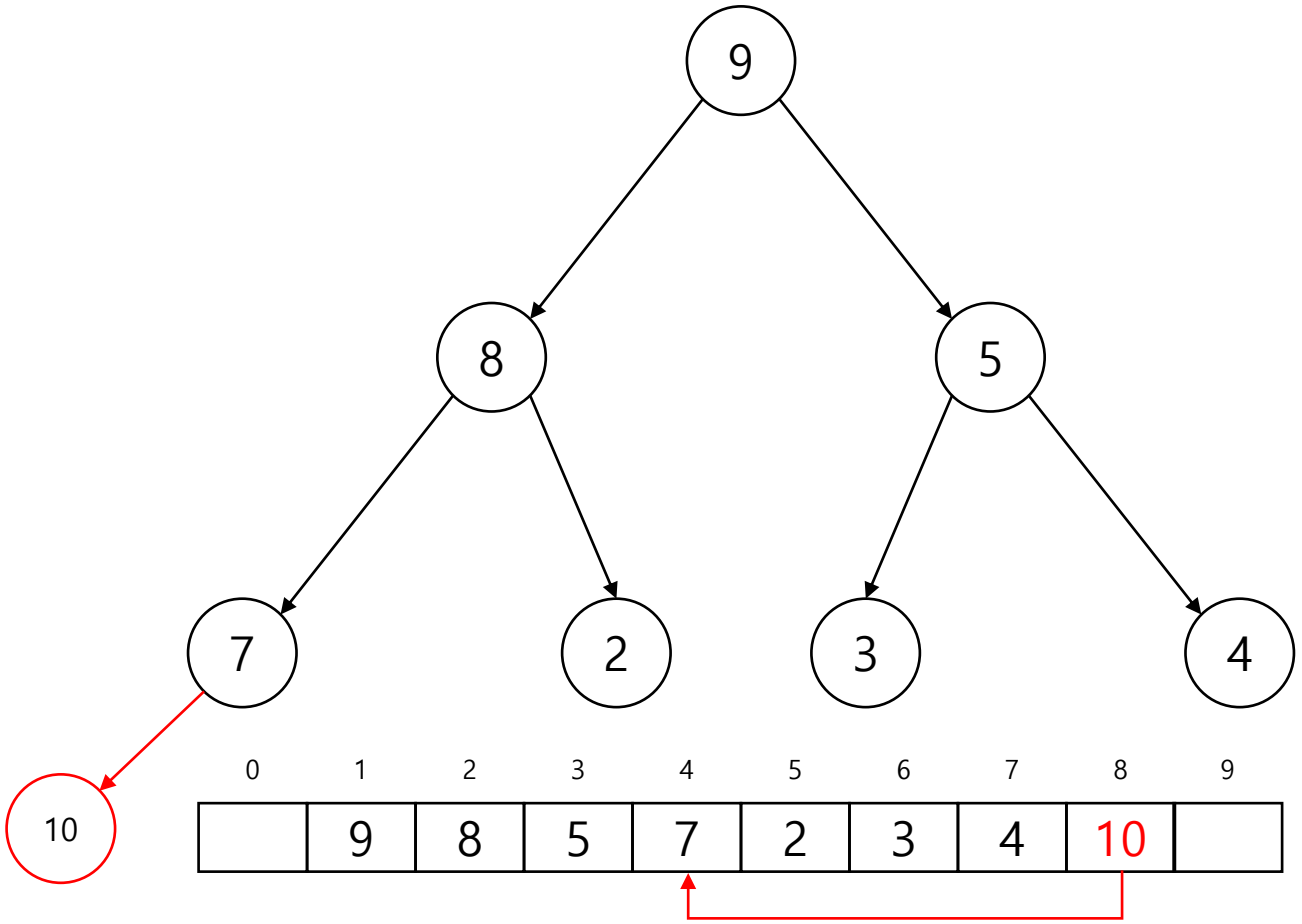
원소 삽입

- 배열의 맨 뒤에 원소 추가
- 만약 부모가 더 작으면 부모와 교환

이진 힙

원소 삽입

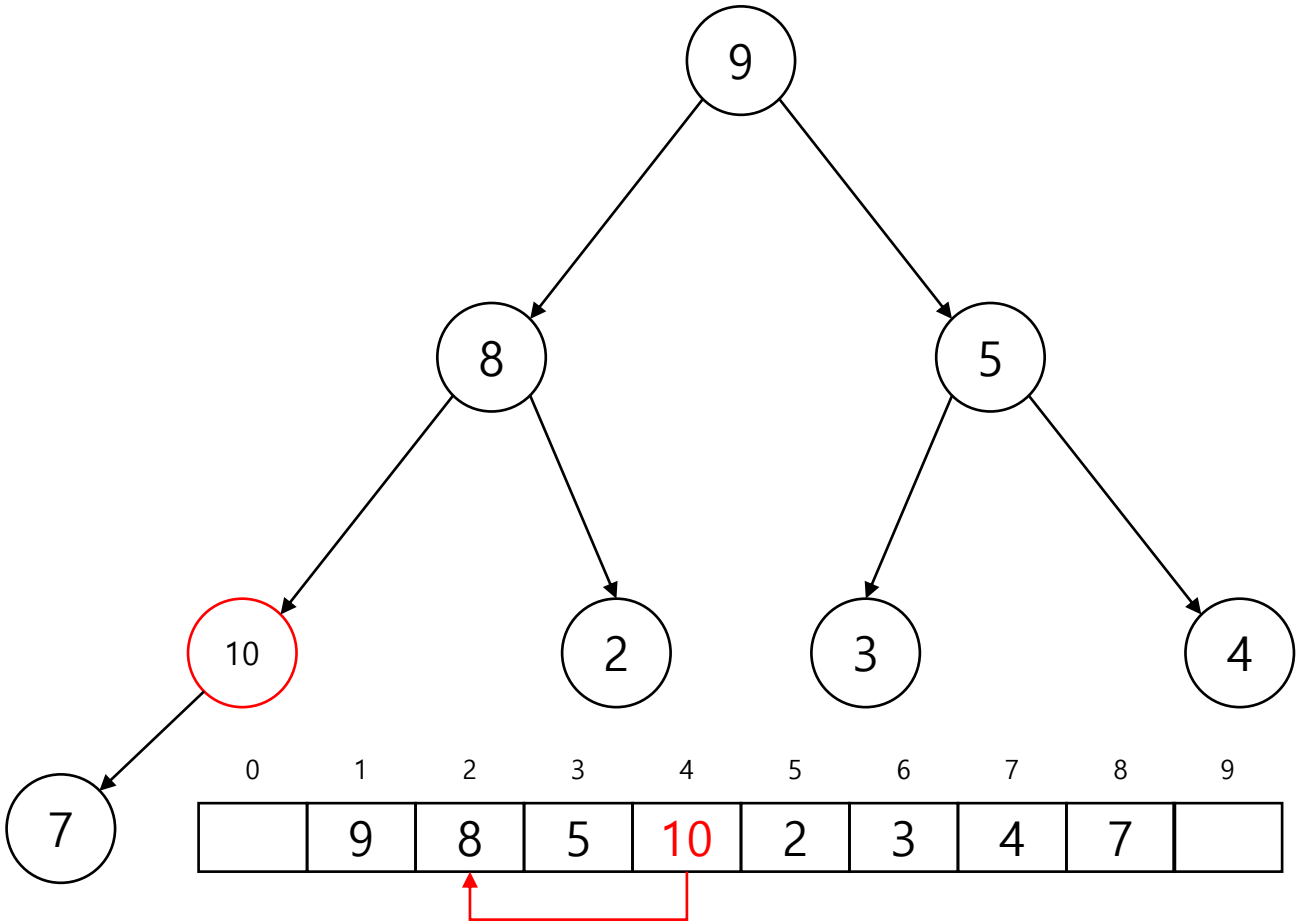
- INSERT 10



이진 힙

원소 삽입

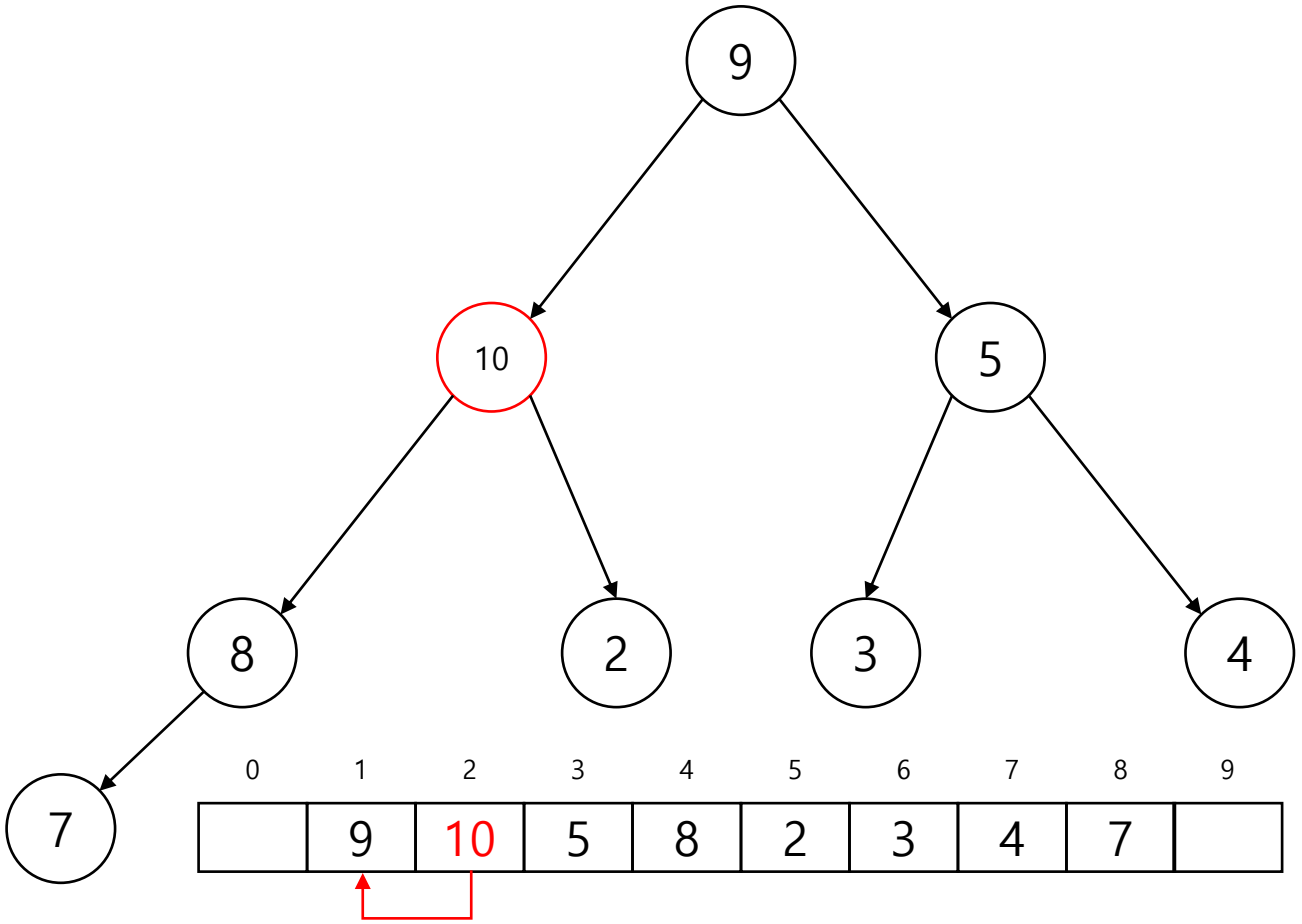
- INSERT 10



이진 힙

원소 삽입

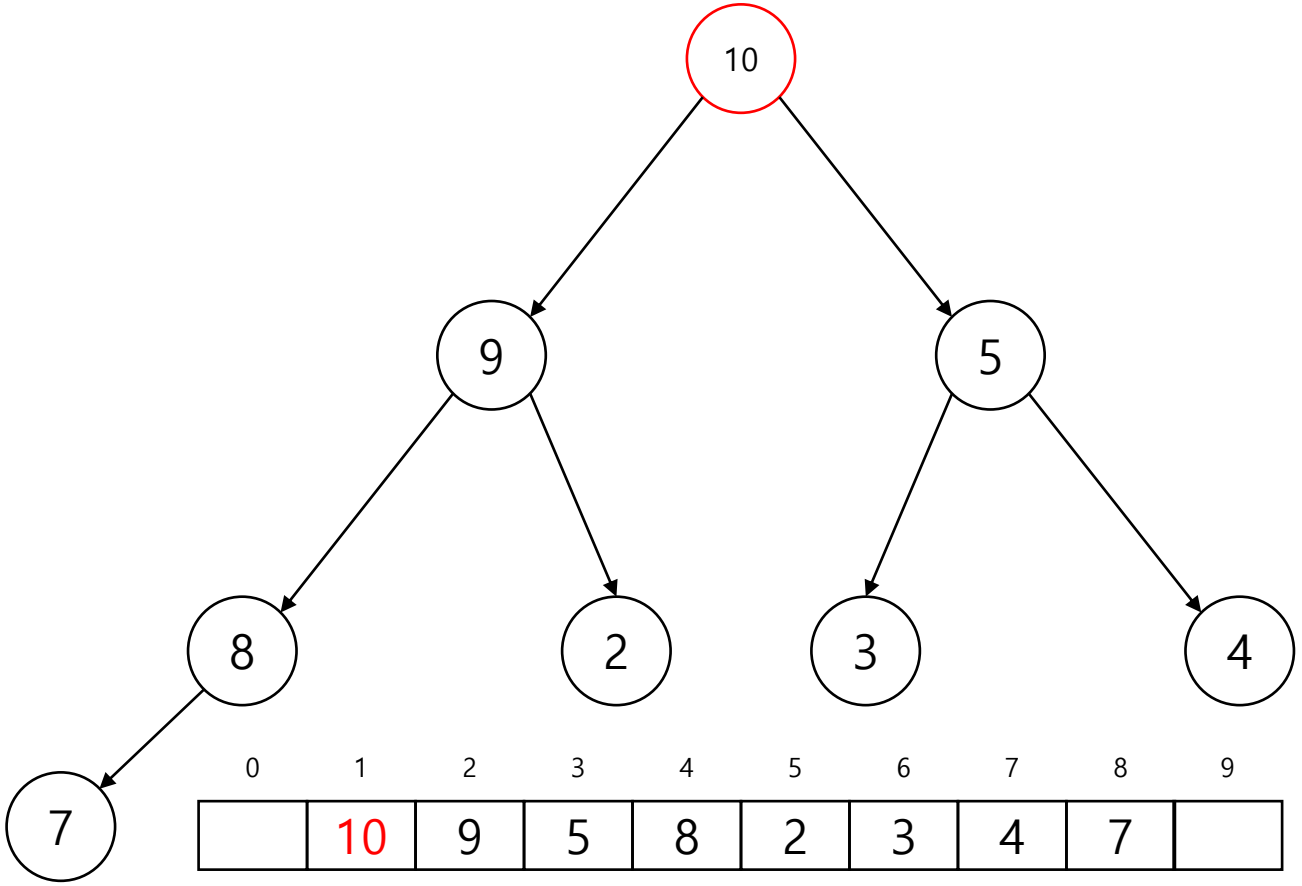
- INSERT 10



이진 힙

원소 삽입

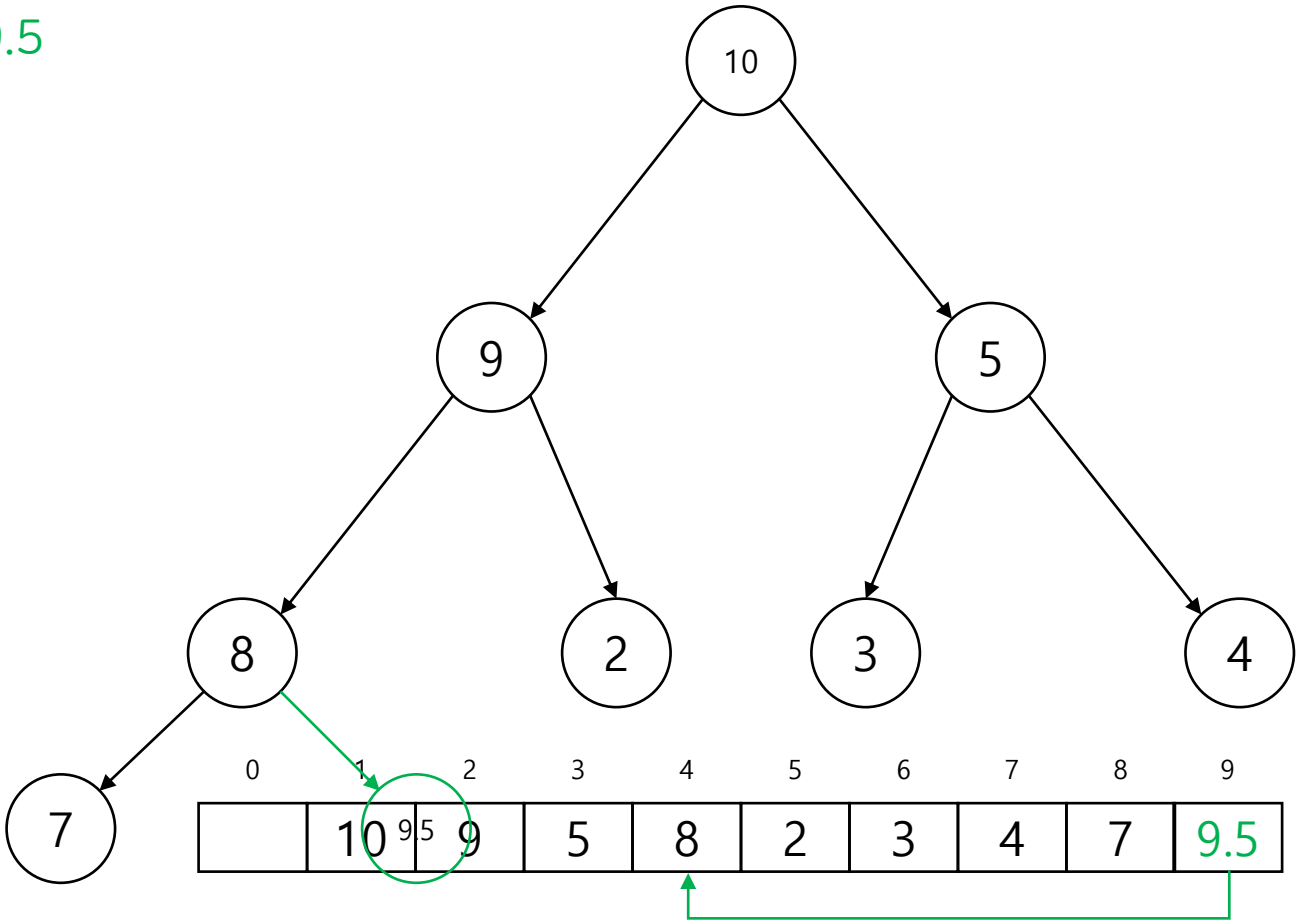
- INSERT 10



이진 힙

원소 삽입

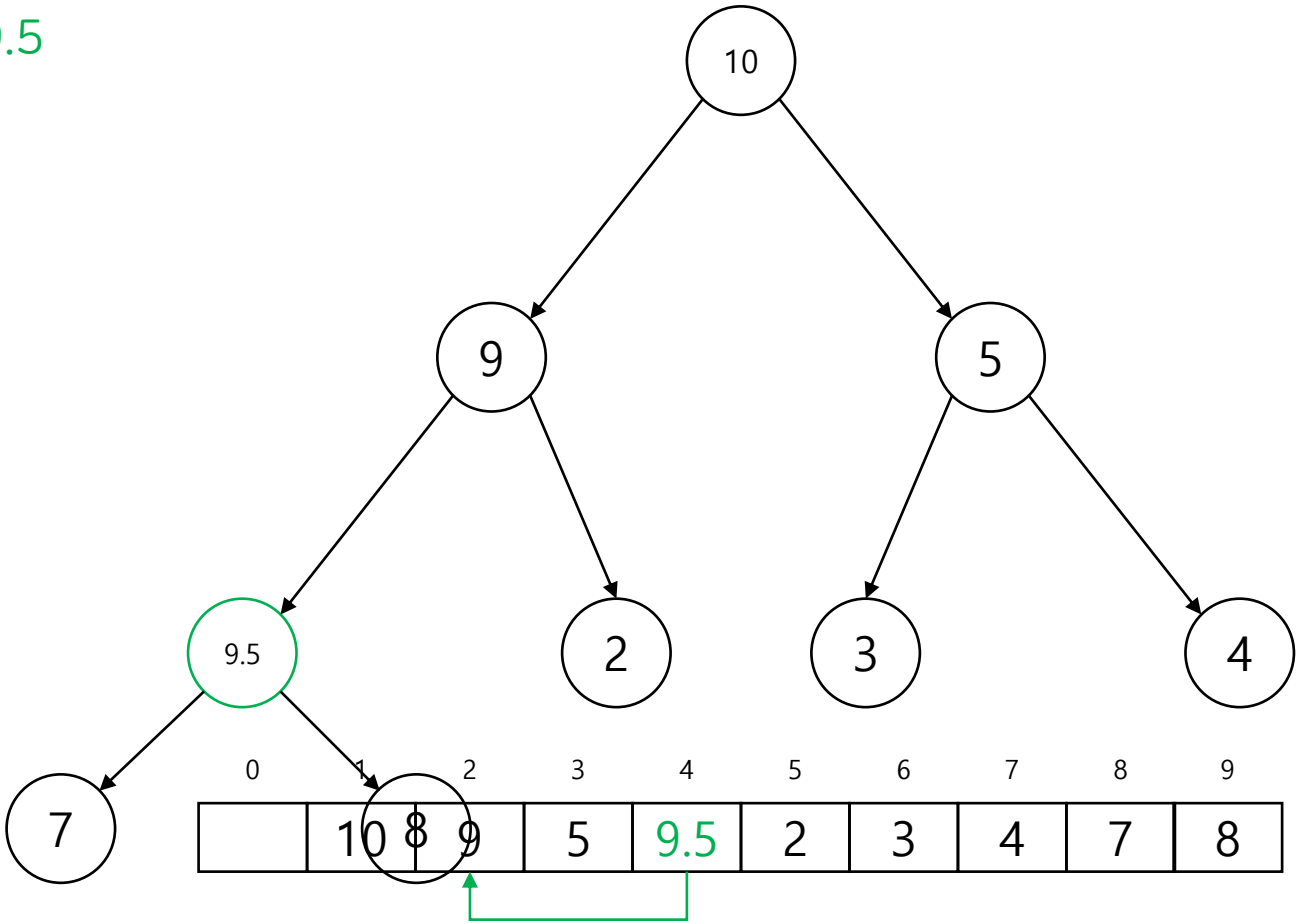
- INSERT 10
- INSERT 9.5



이진 힙

원소 삽입

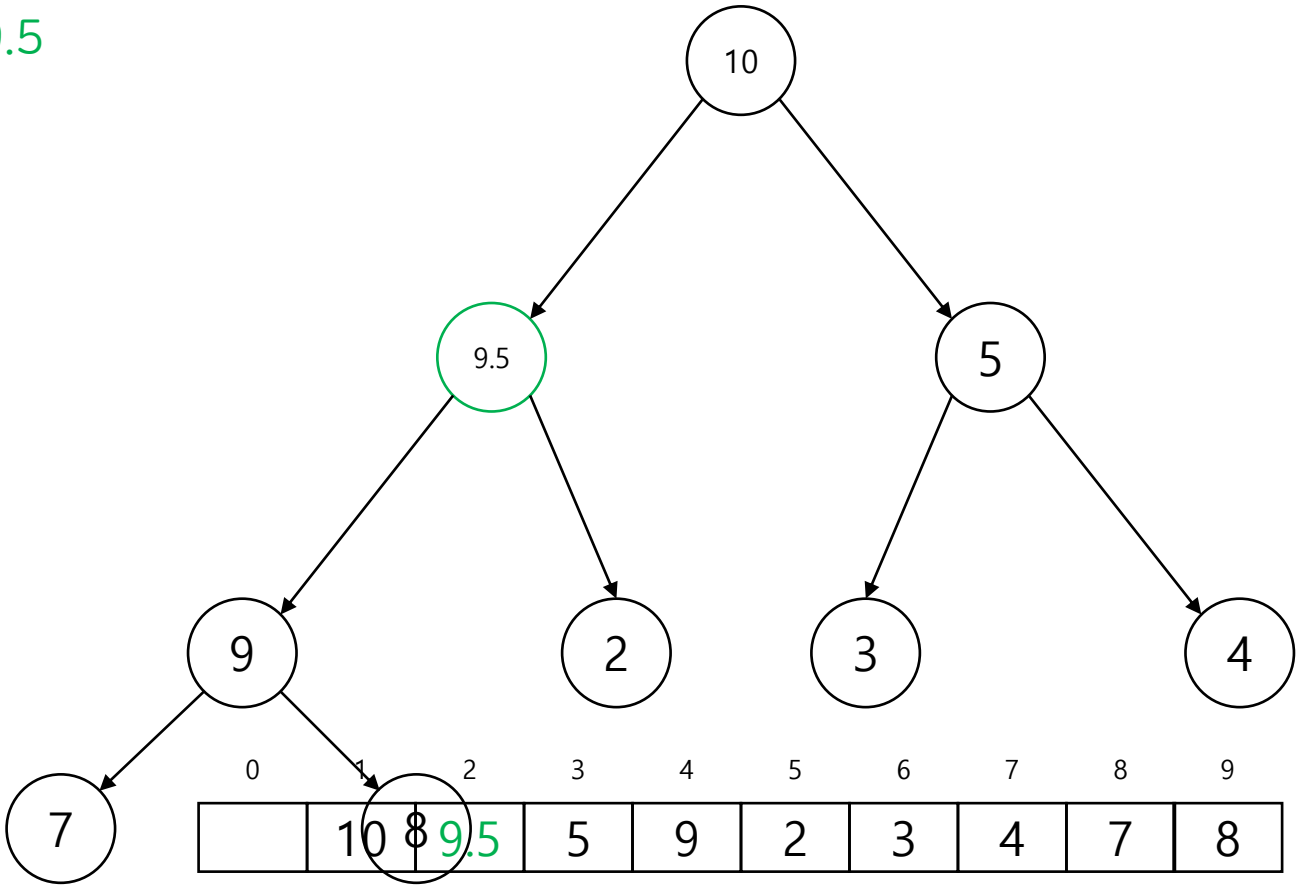
- INSERT 10
- INSERT 9.5



이진 힙

원소 삽입

- INSERT 10
- INSERT 9.5



이진 힙

원소 삽입

- 시간 복잡도 : O(h) = O(log N)

이진 힙

가장 큰 원소 탐색

- return heap[1];
- 시간 복잡도 : O(1)

이진 힙

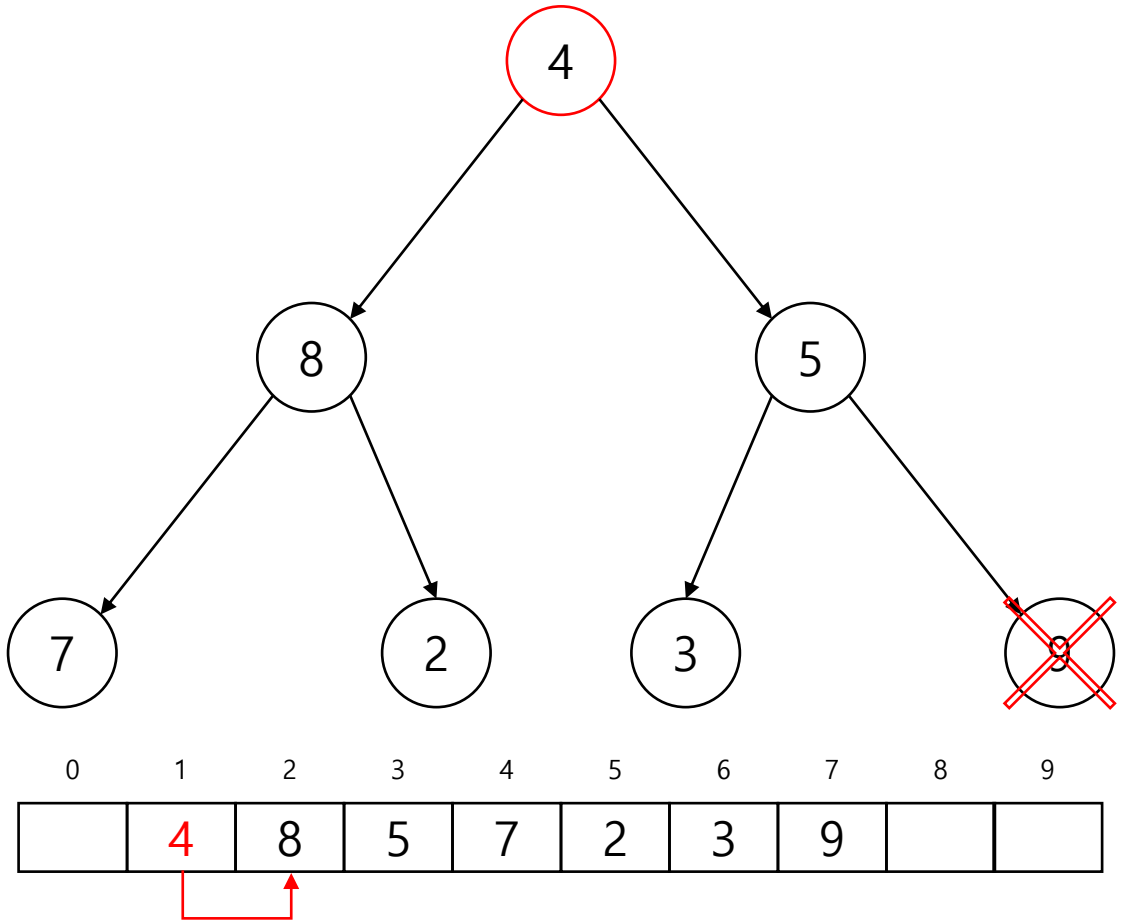
가장 큰 원소 제거

- 루트(가장 큰 원소)와 마지막 정점의 값을 바꿈
- 마지막 정점 제거
- 현재 루트에 있는 값이 자식보다 작으면 밑으로 내림
  - 두 자식 모두 현재 정점보다 크면 더 큰 방향으로 이동

이진 힙

가장 큰 원소 제거

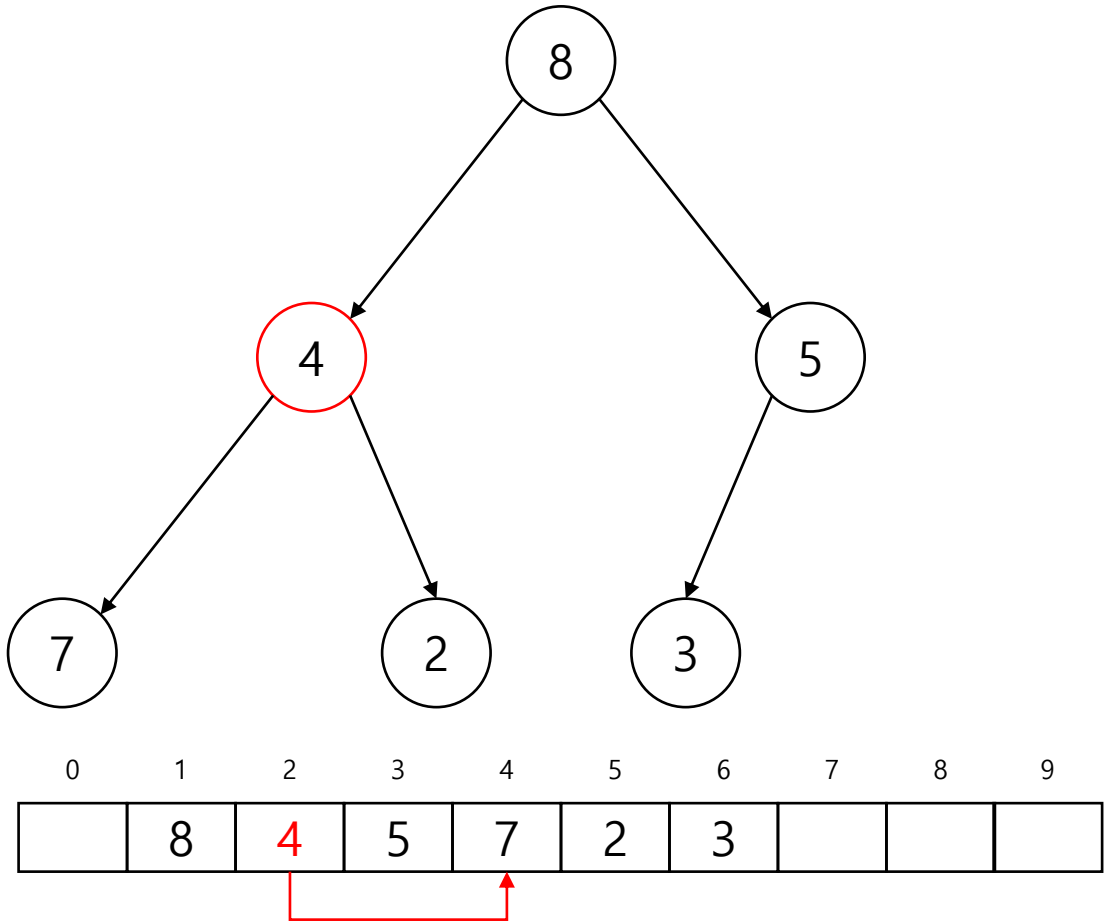
- POP 9



이진 힙

가장 큰 원소 제거

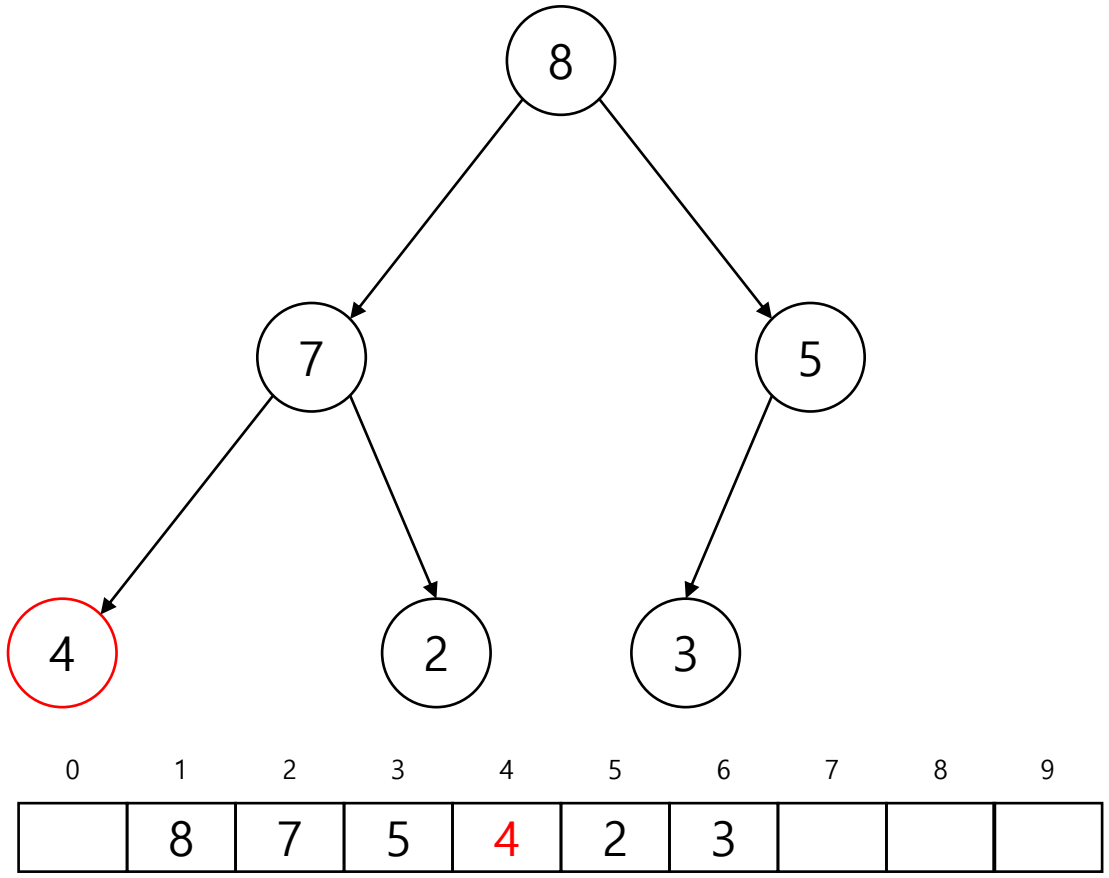
- POP 9



이진 힙

가장 큰 원소 제거

- POP 9



이진 힙

가장 큰 원소 제거

- 시간 복잡도 : O(h) = O(log N)

이진 힙

구현

```cpp
int Heap[101010], sz = 0;

void push(int v){
    Heap[++sz] = v;
    for(int i=sz; i>1; i>>=1){
        if(Heap[i] > Heap[i/2]) swap(Heap[i], Heap[i/2]);
        else break;
    }
}

int pop(){
    swap(Heap[1], Heap[sz--]);
    for(int i=1; i*2<=sz; ){
        int ch = i * 2;
        if(ch+1 <= sz && Heap[ch+1] > Heap[ch]) ch += 1;
        if(Heap[ch] > Heap[i]) swap(Heap[ch], Heap[i]), i = ch;
        else break;
    }
    return Heap[sz+1];
}
```

질문?

이진 힙

std::priority_queue

- #include <queue>
- 이진 힙으로 구현되어 있음
- 기본값은 max heap – 가장 큰 원소가 맨 위에 있음
  - min heap: priority_queue<int, vector<int>, greater<>> pq;
  - max heap: priority_queue<int, vector<int>, less<>> pq;

- pq.push(x)
- pq.pop()
- pq.top()
- pq.size()
- pq.empty()

이진 힙 – 예시 1

BOJ 2075 N번째 큰 수

- N^2개의 수가 주어지면 그중 N번째로 큰 수를 구하는 문제
- 다 정렬하는 건 재미없으니 공간 복잡도 O(N)에 풀어보자.

- 주어지는 수들을 하나씩 입력받으면서
- 지금까지 받은 수 중 가장 큰 N개만 저장하는 방식으로 해결

- min heap 이용
  - 입력받은 수를 힙에 넣은 다음
  - 힙의 크기가 N+1이 되면 가장 작은 원소 하나 제거

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
    ios_base::sync_with_stdio(false); cin.tie(nullptr);
    int N; cin >> N;
    priority_queue<int, vector<int>, greater<>> Q;
    for(int i=1; i<=N*N; i++){
        int t; cin >> t;
        Q.push(t);
        if(Q.size() > N) Q.pop();
    }
    cout << Q.top();
}
```

이진 힙 – 예시 2

BOJ 2696 중앙값 구하기

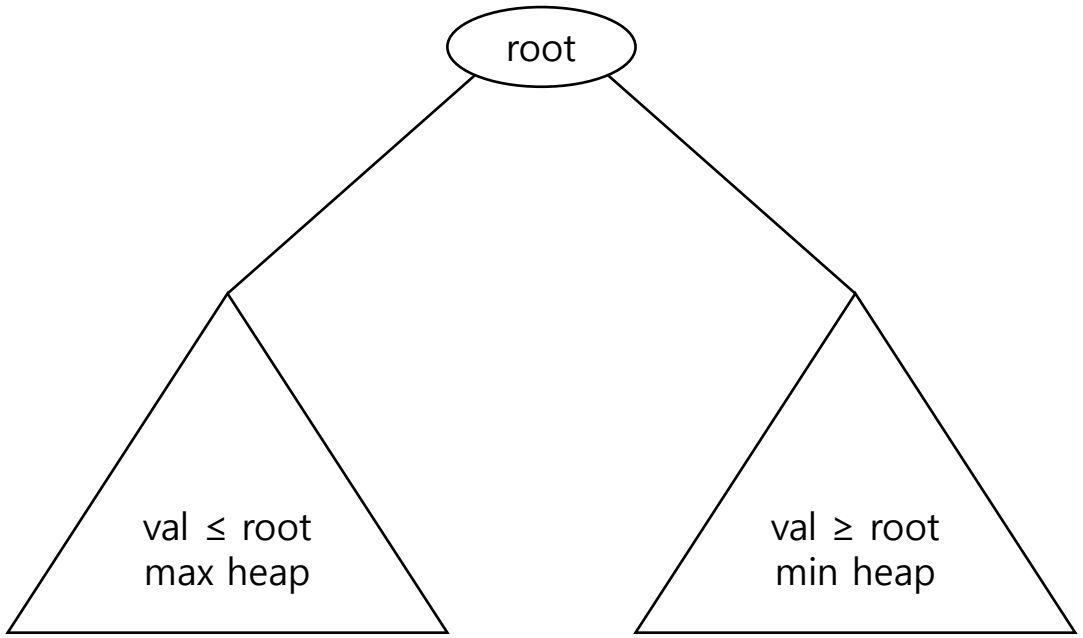
- 모든 홀수 i에 대해 A[1..i]의 중앙값을 구하는 문제

- 원소가 2개 추가될 때마다 중앙값을 구하는 문제
  - 현재의 중앙값을 x라고 하자.
    - 새로 추가된 값이 모두 x 이하 : x 이하인 가장 큰 값이 새로운 중앙값
    - 새로 추가된 값이 모두 x 이상 : x 이상인 가장 작은 값이 새로운 중앙값
    - 하나는 x 이하, 하나는 x 이상 : x가 중앙값
  - x 이하인 원소를 관리하는 max heap과 x 이상인 원소를 관리하는 min heap

이진 힙 – 예시 2

BOJ 2696 중앙값 구하기

- x 이하인 원소를 관리하는 max heap
- x 이상인 원소를 관리하는 min heap




```cpp
void Solve(){
    int N; cin >> N;
    priority_queue<int, vector<int>, less<>> max_heap;
    priority_queue<int, vector<int>, greater<>> min_heap;
    int root; cin >> root;

    cout << (N + 1) / 2 << "\n";
    cout << root << " ";
    for(int i=0; i<N/2; i++){
        int a, b; cin >> a >> b;
        if(a > b) swap(a, b);
        if(a <= root && root <= b){
            max_heap.push(a); min_heap.push(b);
        }
        else if(b <= root){
            max_heap.push(a); max_heap.push(b);
            min_heap.push(root);
            root = max_heap.top(); max_heap.pop();
        }
        else{
            min_heap.push(a); min_heap.push(b);
            max_heap.push(root);
            root = min_heap.top(); min_heap.pop();
        }
        cout << root << " ";
    }
    cout << "\n";
}
```

질문?