힙 (Heap)

01 힙 이론

02 힙의 활용

신 제 용

01 힙 이론

우선순위 큐라고도 불리우는 힙 자료구조의 동작과,
이러한 자료구조의 장점에 대해 알아봅니다.

학습 키워드 - 힙, 우선순위 큐, 정렬

Chapter 01
힙 이론

ZEROBASE SCHOOL

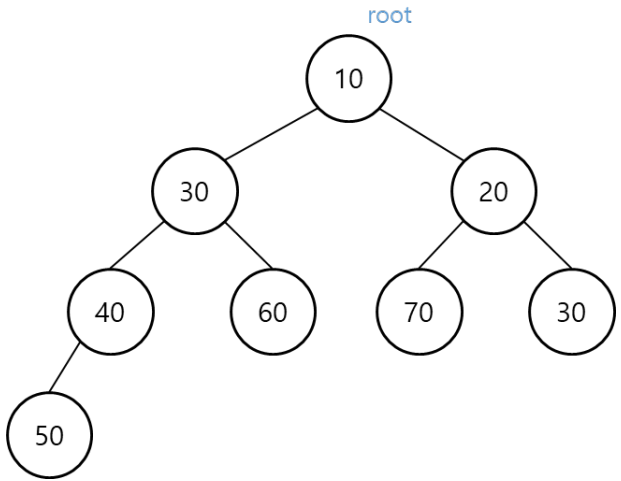
힙 (Heap)

- 완전 이진 트리 의 일종으로, 우선순위 큐(Priority queue) 라고도 한다.

- 최소값 또는 최대값을 빠르게 찾아내는데 유용한 자료구조

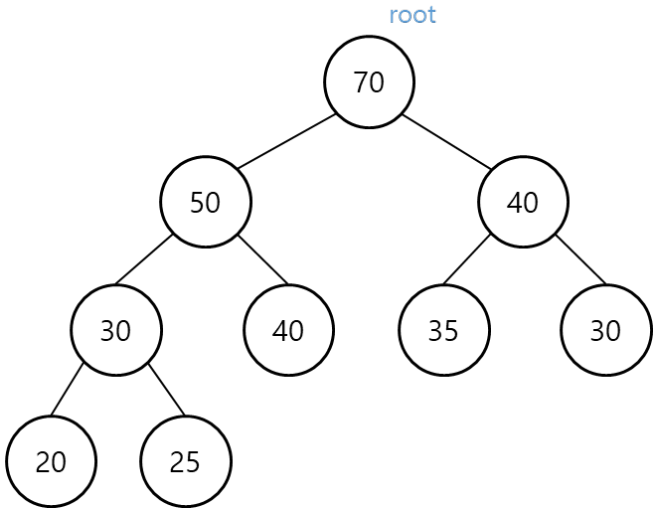
최소 힙 (Min heap)

- **부모 노드의 value ≤ 자식 노드의 value 상태**로 항상 정렬되어 있는 자료구조
- 동일한 자료에 대해서 여러가지 최소 힙이 성립한다.



최대 힙 (Max heap)

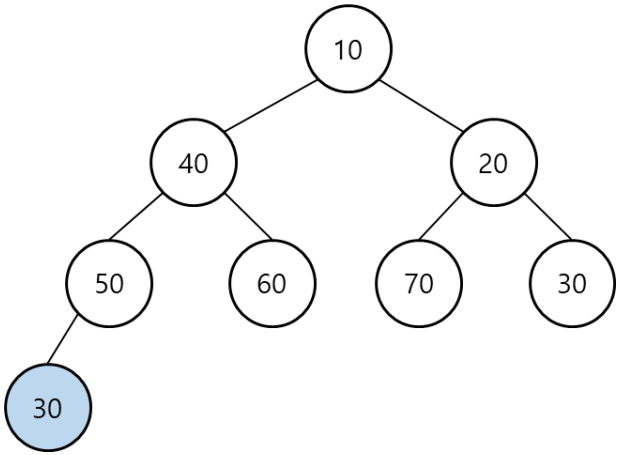
- **부모 노드의 value ≥ 자식 노드의 value 상태**로 항상 정렬되어 있는 자료구조
- 동일한 자료에 대해서 여러가지 최대 힙이 성립한다.



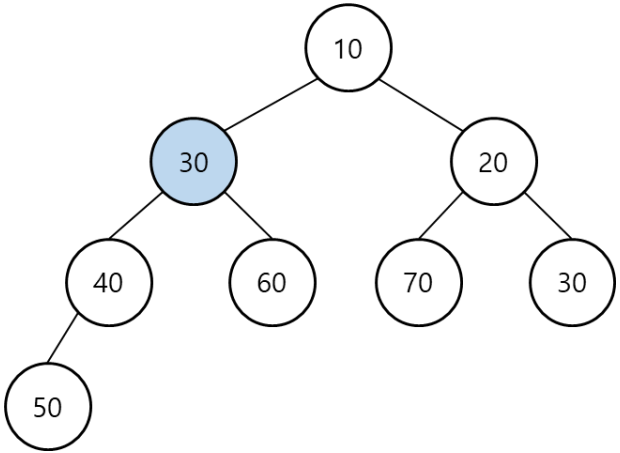
최소 힙 – 삭제

- 트리의 가장 끝 위치에 데이터 삽입
- 부모 노드와 키 비교한 후 작을 경우 부모 자리와 교체 (반복)

< 데이터 30 삽입 >



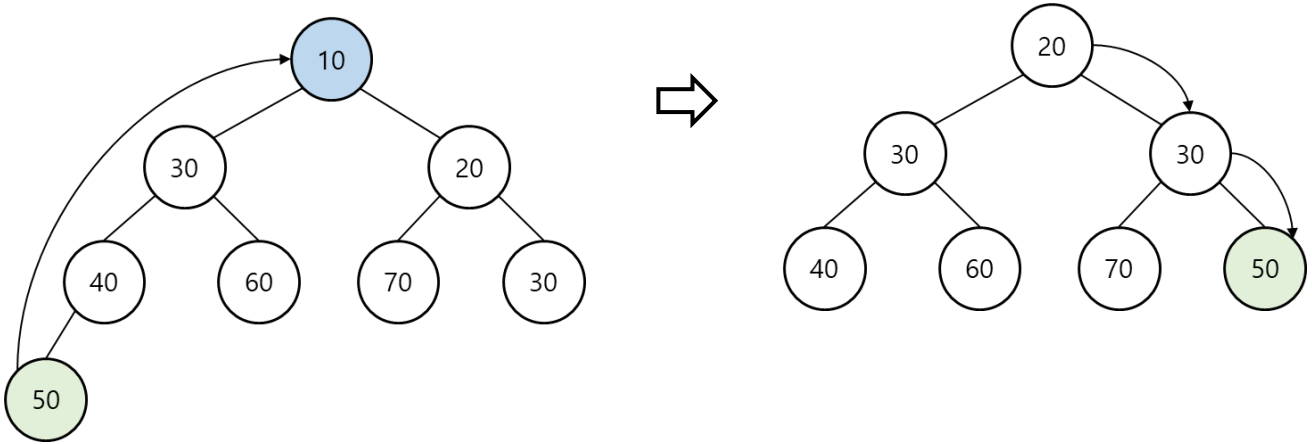
< 삽입 완료 후 >



최소 힙 – 삽입

- 최상위 노드 반환 및 삭제
- 가장 마지막 위치의 노드를 최상위 노드로 위치 시킴
- 자식 노드 중 작은 값과 비교 후 부모 노드가 더 크면 자리 교체 (반복)

< 데이터 삭제 >



힙의 장점

- 일반 배열에서 최대값/최소값을 찾아 반환하는 연산: O(N)
- 일반 배열에 새로운 자료를 추가하는 연산: O(1)

- 정렬된 배열에서 최대값/최소값을 찾아 반환하는 연산: O(1)
  (단, **정렬하는 연산 O(NlogN)이 선행**되어야 함)
- 정렬된 배열에 새로운 자료를 추가하는 연산: O(N)

- **Heap에서 최대값/최소값을 찾아 반환하는 연산: O(logN)**
- **Heap에 새로운 자료를 추가하는 연산: O(logN)**

02 힙의 활용

다음 챕터에서는 힙을 활용하는 예시를 확인하고 구현해봅니다.

Chapter 01
힙 이론

ZEROBASE SCHOOL

02 힙의 활용

힙을 활용하는 기초 유형 문제를 확인하고 직접 풀어봅시다.

학습 키워드 - 힙, 활용, 구현

Chapter 02
힙의 활용

ZEROBASE SCHOOL

```python
from heapq import heappop, heappush, heapify

# heapq 패키지를 이용한 min heap 사용법
heap = []
for elem in [4, 10, 8, 0, 10, 9, 2]:
    heappush(heap, elem)
print(heap)

while heap:
    data = heappop(heap)
    print(data, end=' ')
print()


# heapq 패키지를 이용한 max heap 사용법
heap = []
for elem in [4, 10, 8, 0, 10, 9, 2]:
    heappush(heap, -elem)
print(heap)

while heap:
    data = -heappop(heap)
    print(data, end=' ')
print()
```

Chapter 02
힙의 활용

ZEROBASE SCHOOL

```python
from heapq import heappop, heappush, heapify

# heapq 패키지를 이용한 priority queue 사용법
pq = []
            # (우선순위(작을수록 높음), 값)
for elem in [(1, 10), (4, 14), (-2, 5), (10, 6)]:
    heappush(pq, elem)

while pq:
    data = heappop(pq)
    print(data[1], end=' ')
print()


# heapify를 사용하는 방법: 일반적인 배열(리스트)를 힙으로 변경
heap = [4, 10, 8, 0, 10, 9, 2]
heapify(heap)
while heap:
    data = heappop(heap)
    print(data, end=' ')
print()
```

Chapter 02
힙의 활용

ZEROBASE SCHOOL

```python
# Prob1.
#
# 슬라이딩 최댓값
# 리스트 arr에 정수로 이루어진 자료가 주어져 있다.
# 이 때, 연속된 k개 중 최댓값을 구하고 싶다.
# 예를 들어, arr = [1, 3, 5, 2, 7, 4]이고 k = 3이라면
#
# 첫번째 최댓값은 max([1, 3, 5]) -> 5
# 두번째 최댓값은 max([3, 5, 2]) -> 5
# 세번째 최댓값은 max([5, 2, 7]) -> 7
# 네번째 최댓값은 max([2, 7, 4]) -> 7
#
# 따라서 결과는 [5, 5, 7, 7]이 된다.
#
# 이 프로그램을 구현하시오.

def solution(arr, k):
    pass

if __name__ == '__main__':
    arr = [1, 3, 5, 2, 7, 4]
    k = 3
    sol = solution(arr, k)
    print(sol)
```

```python
# Prob2.
#
# 누적 중간값
# 리스트 arr에 정수로 이루어진 자료가 주어져 있다.
# 출력 역시 정수로 이루어진 동일한 길이의 리스트로,
# 이 리스트의 i번째 값은 arr의 처음부터 i번째까지 값의 중간값이다.
# 단, 짝수개의 값의 중간값은 중간값 2개의 평균으로 계산한다.
# 이 프로그램을 구현하시오.


def solution(arr):
    pass


if __name__ == '__main__':
    arr = [1, 3, 5, 2, 7, 4, 5, 23, -4, 52, 45]
    sol = solution(arr)
    print(sol)
```