힙

선행학습

우선순위 큐 :

우선순위의 개념을 큐에 도입한 자료 구조

텍스트이(가) 표시된 사진

자동 생성된 설명데이터들이 우선순위를 가지고 있고 우선순위가 높은 데이터가 먼저 나간다.

우선순위 큐의 이용 사례

1. 시뮬레이션 시스템
2. 네트워크 트래픽 제어
3. 운영 체제에서의 작업 스케쥴링
4. 수치 해석적인 계산 등등

테이블이(가) 표시된 사진

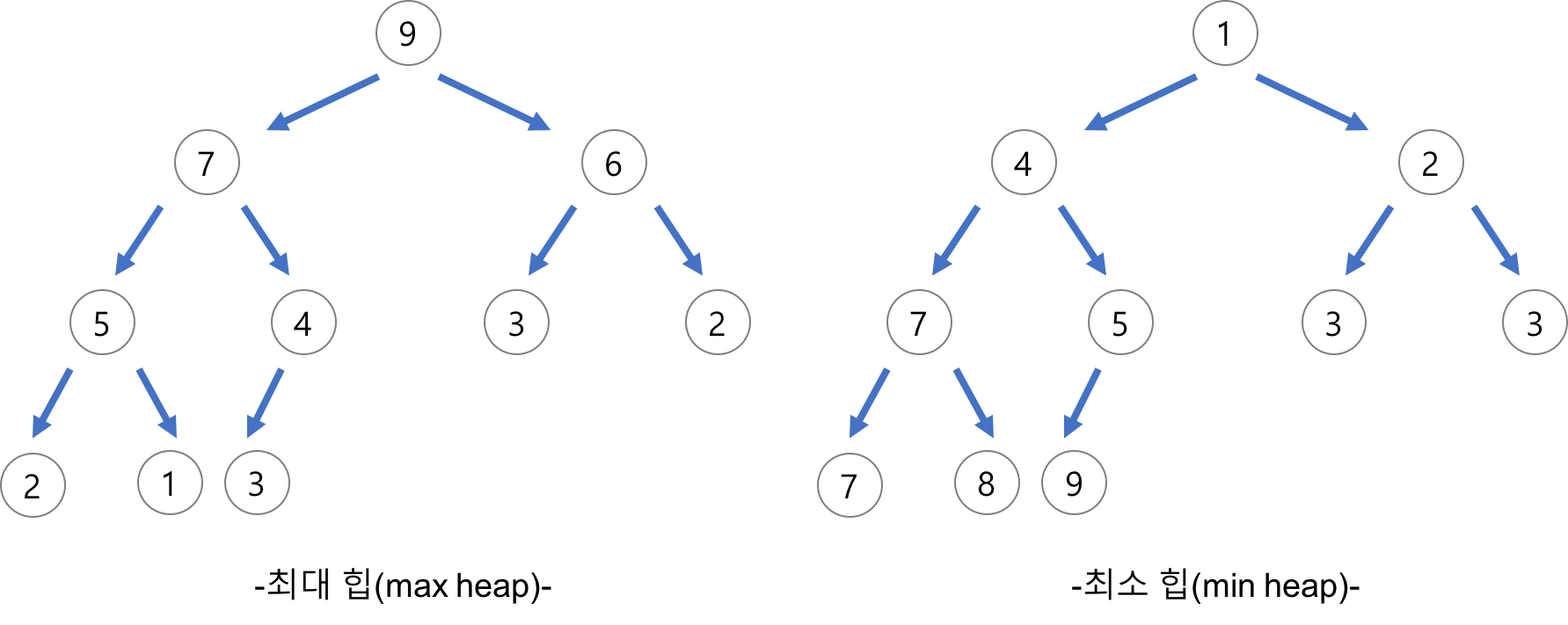
자동 생성된 설명우선순위 큐는 배열, 연결리스트, 힙으로 구현이 가능하다. 이 중 힙(heap)으로 구현하는 것이 가장 효율적이다.

자료구조 힙(heap)

* 완전 이진 트리의 일종으로 우선순위 큐를 위하여 만들어진 자료구조 이다.
* 여러 개의 값들 중에서 최댓값이나 최솟값을 빠르게 찾아내도록 만들어진 자료구조이다.
* 힙은 일종의 반정렬 상태(느슨한 정렬 상태)를 유지한다.
  + 큰 값이 상위 레벨에 있고 작은 값이 하위 레벨에 있다는 정도
  + 부모 노드의 값이 자식 노드의 값보다 항상 크거나 작은 이진 트리이다.
* 힙 트리에서는 중복된 값을 허용한다. (이진 탐색 트리에서는 중복 값을 허용하지 않는다.)

힙(heap)의 종류

* 최대 힙(max heap)
  + 부모 노드의 값이 자식 노드의 값보다 크거나 같은 완전 이진 트리
  + Key(부모노드) >= key(자식노드)
* 최소 힙(min heap)
  + 부모 노드의 값이 자식 노드의 값보다 작거나 같은 완전 이진 트리
  + Key(부모노드) <= key(자식노드)



힙(heap)의 구현

* 힙을 저장하는 표준적인 자료구조는 배열이다.
* 구현을 쉽게 하기 위해 배열의 첫 번째 인덱스인 0은 사용하지 않는다.
* 특정 위치의 노드 번호는 새로운 노드가 추가되어도 변하지 않는다.

힙에서의 부모 노드와 자식 노드의 관계

* 왼쪽 자식의 인덱스 = (부모의 인덱스 \* 2)
* 오른쪽 자식의 인덱스 = (부모의 인덱스 \* 2) + 1
* 부모 인덱스 = (자식의 인덱스) / 2

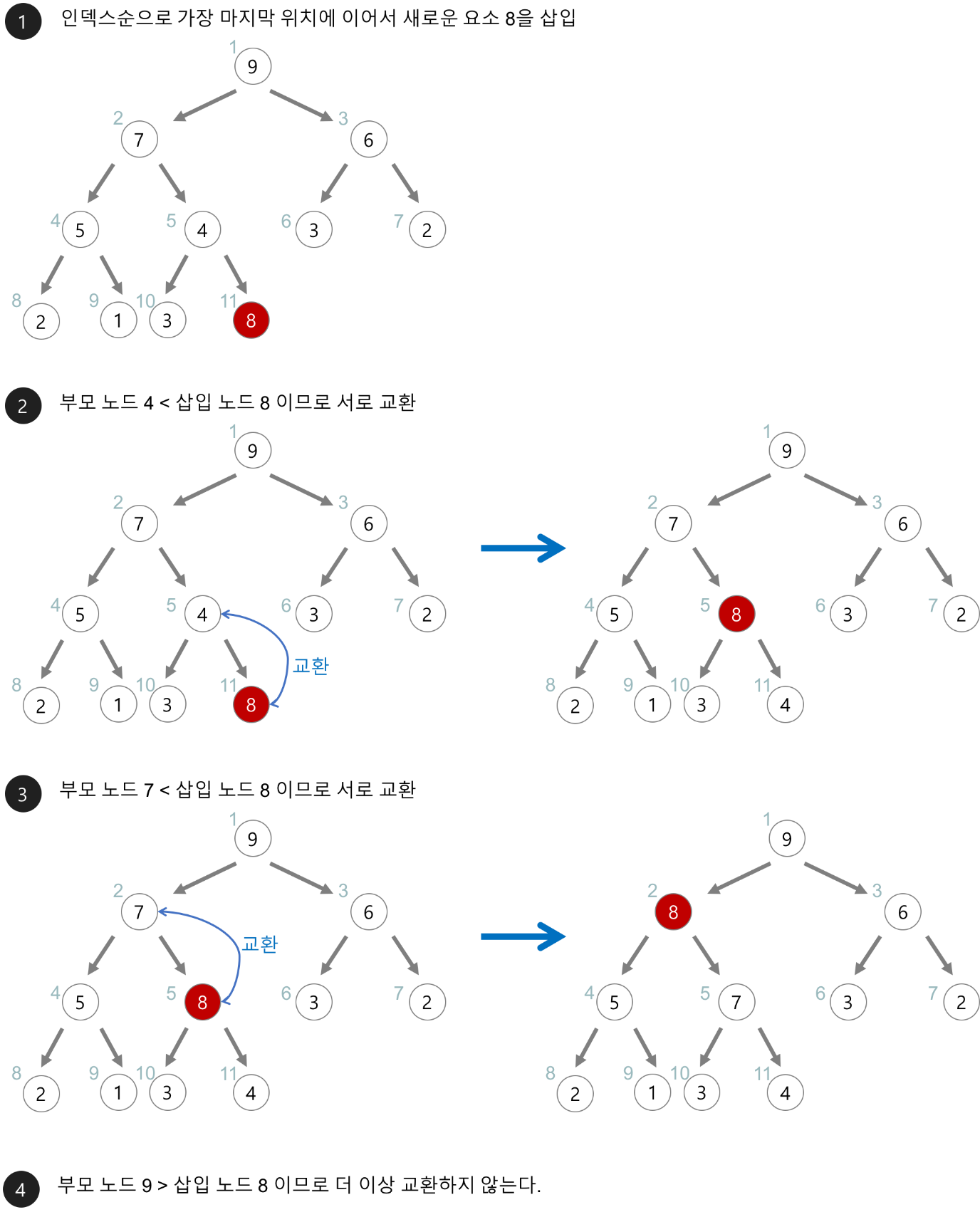
손목시계이(가) 표시된 사진

자동 생성된 설명

힙(heap)의 삽임

1. 힙에 새로운 요소가 들어오면, 새로운 노드를 힙의 마지막 노드에 이어서 삽입한다
2. 새로운 노드를 부모 노드와 비교, 교환해서 힙의 성질을 만족시킨다.

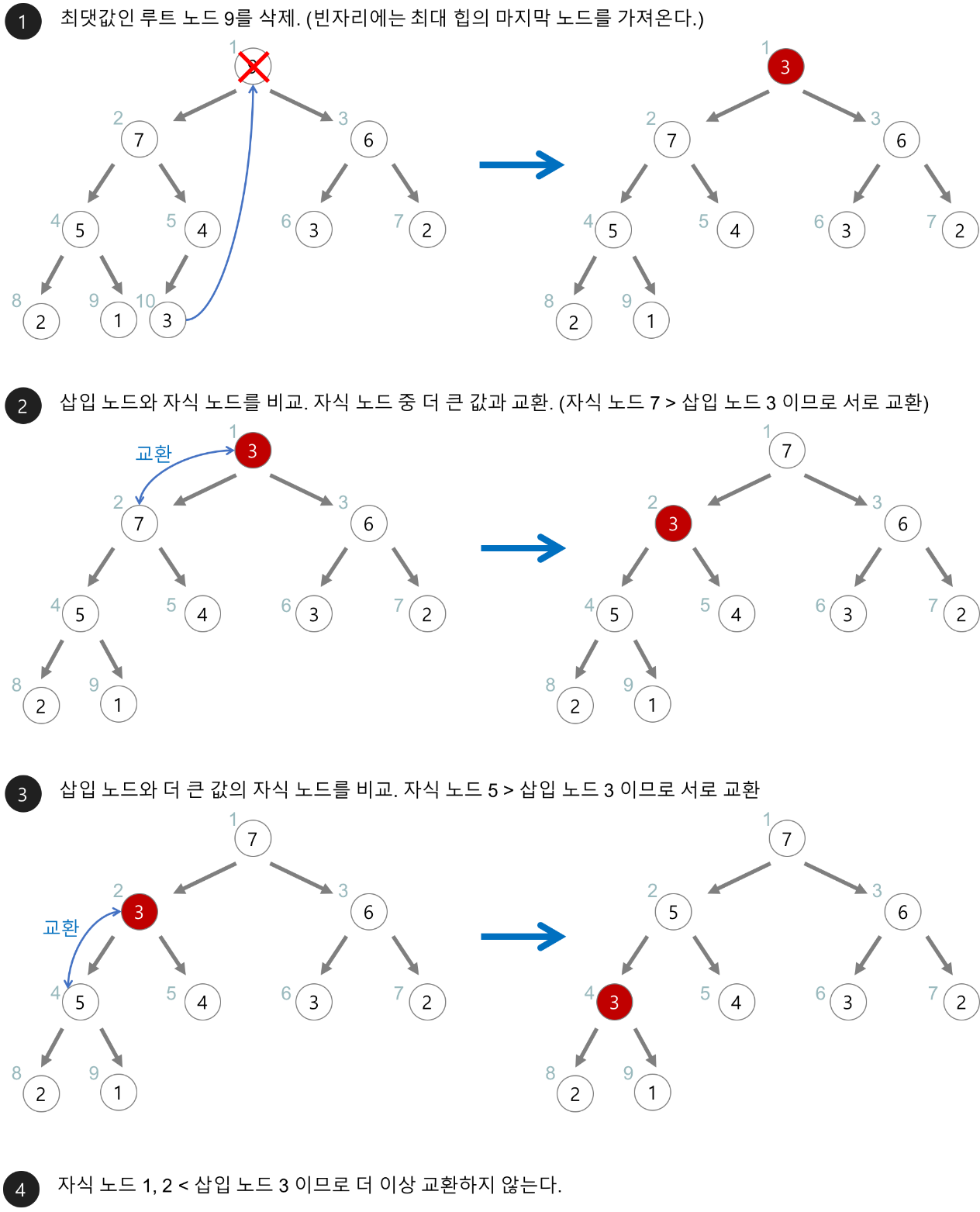
예시)



힙의 삭제

1. 최대 힙에서 최댓값은 루트 노드이므로 루트 노드가 삭제된다.
   1. 최대 힙에서 삭제 연산은 최댓값을 가진 요소를 삭제하는 것 이다.
2. 삭제된 루트 노드에는 힙의 마지막 노드를 가져온다.
3. 힙을 재구성한다.

예시)



Python 힙(heap) 구현

파이썬에서는 heapq (priority queue) 알고리즘을 제공한다.

heapq 를 사용하면 최소 힙의 형태로 정렬된다.

heapq는 내장 모듈로 별도의 설치 없이 사용할 수 있다.

힙 함수

heapq.heappush(heap, item) : item을 heap에 추가한다.

heapq.heappop(heap) : heap에서 가장 작은 원소를 pop, return. 비어 있는 경우 IndexError가 호출됨

heapq.heapify(x) : 리스트 x를 heap으로 변환한다. 시간복잡도 : O(N)

힙 생성, 원소 추가

heapq 모듈은 리스트를 최소힙처럼 다룰 수 있기 때문에, 빈 리스트를 생성한 후 heapq의 함수를 호출할 때마다 리스트의 인자에 넘겨야 한다.

힙 생성

import heapq

heap = []

heapq.heappush(heap, 50)

heapq.heappush(heap, 10)

heapq.heappush(heap, 20)

Print(heap)

[10, 50, 20]

이미 생성해둔 리스트가 있다면 heapify 함수를 통해 힙 자료형으로 변환할 수 있다.

heap2 = [50, 10, 20]

heapq.heapify(heap2)

print(heap2)

[10, 50, 20]

힙에서 원소 삭제

heappop 함수는 가장 작은 원소를 제거하고 결과값으로 리턴한다.

result = heapq.heappop(heap)

print(result)

print(heap)

10

[20, 50]

원소를 삭제하지 않고 가져오려면 [0] 인덱싱을 통해 접근한다.

result2 = heap[0]

print(result2)

print(heap)

20

[20, 50]

최대힙 만들기

파이썬의 heapq 모듈은 최소힙으로 구현되기 때문에 코드가 더 필요하다.

y = -x 변환을 하면 최솟값 정렬이 최댓값 정렬로 바뀌게 된다.

힙에 원소를 추가할 때 (-item, item) 형태로 넣어주면 첫 번째 원소를 우선순위로 힙을 구성하게 된다. 이때 원소값의 부호를 바꿨기 때문에, 최소힙으로 구현된 heapq 모듈은 최대힙을 구현하게 된다.

heap\_items = [1,3,5,7,9]

max\_heap = []

for item in heap\_items :

heapq.heappush(max\_heap, (-item, item))

print(max\_heap)

[(-9,9, (-7,7), (-3,3), (-1,1) , (-5,5)]

실제 원소 값은 두번째 자리에 지정되어 있으므로 [1] 인덱싱을 통해 접근한다.

max\_item = heapq.heappop(max\_heap)[1]

print(max\_item)

9

도움을 준 사이트

https://gmlwjd9405.github.io/2018/05/10/data-structure-heap.html

https://littlefoxdiary.tistory.com/3