**힙(Heap)**

트리 형태로 쌓인 데이터로, 최댓값이나 최소값을 찾기 위한 자료구조 [완전 이진 트리 구조]

최대힙(Max heap): 부모 노드의 key 값이 자식 노드의 key 값보다 크거나 같은 완전 이진 트리

최소힙(Min heap): 부모 노드의 key 값이 자식 노드의 key 값보다 작거나 같은 완전 이진 트리

\*완전 이진 트리: 노드 삽입 시 가장 왼쪽부터 채워 나가는 트리

최대힙의 데이터 삽입:

1. 완전 이진 트리이기 때문에 트리의 남은 자리에 데이터가 삽입된다.
2. 부모 노드의 key 값과 비교하여 더 크다면 교환하고 아니라면 그 자리에 그대로 두는 방식을 계속 반복하여 최대힙의 성질을 만족하도록 한다.

최소힙의 데이터 삽입:

1. 완전 이진 트리이기 때문에 트리의 남은 자리에 데이터가 삽입된다.
2. 부모 노드의 key 값과 비교하여 더 작다면 교환하고 아니라면 그 자리에 그래도 두는 방식을 계속 반복하여 최소힙의 성질을 만족하도록 한다.

힙에서의 삭제: 루트 노드 (맨 위 노드 삭제) -> 최대힙은 최댓값 삭제, 최소힙은 최솟값 삭제

최대힙의 데이터 삭제:

1. 루트 노드를 삭제하고 가장 끝에 있는 노드를 루트 노드로 지정한다.
2. 자식 노드와 비교해가며 자식 노드보다 작다면 swap, 크거나 같다면 그대로 두며 다시 최대힙 정렬을 한다.

최소힙의 데이터 삭제:

1. 루트 노드를 삭제하고 가장 끝에 있는 노드를 루트 노드로 지정한다.
2. 자식 노드와 비교해가며 자식 노드보다 크다면 swap, 작거나 같다면 그대로 두며 다시 최소힙 정렬을 한다.

**우선 순위 큐(Priority Queue)**

데이터에 우선 순위를 부여하여, 정보가 들어온 순서와 관계 없이 높은 우선 순위를 가진 데이터가 먼저 나오는 큐

우선 순위에 따라 데이터가 출력된다는 점에서 힙의 특성을 이용하면 가장 효율적으로 자료구조를 구현할 수 있다.

만약 우선 순위 큐가 ‘배열’로 구현된다면, 우선 순위가 높은 순으로 배열에 들어가야 하므로 이미 들어가 있는 자료보다 더 우선 순위가 높은 자료를 넣으려면 배열을 밀어야 하는 단점이 있다.

만약 우선 순위 큐가 ‘연결 리스트’로 구현된다면, 우선 순위가 높은 자료의 위치를 최악의 경우 끝까지 찾아 들어가야 한다는 단점이 있다.

힙은 부모와 자식 간의 비교만으로 이루어지기 때문에 우선 순위 큐를 가장 효율적으로 구현할 수 있다.

**C++을 이용한 우선 순위 큐 구현**

1. <queue> 헤더파일을 먼저 선언한다.
2. 큐 헤더파일에는 priority\_queue 컨테이너가 있다.
3. 우선 순위 큐를 선언해주는데, 보통 priority\_queue<int, vector<int>, greater<int>>처럼 선언해준다.
4. push(n) -> 큐에 n 원소 추가하는 함수이다.
5. top() -> greater는 가장 큰 값을, less는 가장 작은 값을 반환하는 함수이다.
6. pop() -> top에 있는 원소 삭제하는 함수이다.
7. empty() -> 큐가 비어 있으면 true를 반환하고, 원소가 있다면 false를 반환하는 함수이다.
8. size() -> 큐의 사이즈를 반환하는 함수이다.