자료구조

이진 탐색 트리(map, set)와 우선순위 큐 (Heap)

이상현_ 2020. 1. 6. 10:35 수정 삭제

이진 탐색 트리

- 이진 트리의 일종
- 자신보다 작은 값을 왼쪽 노드에 위치시키고, 자신보다 큰 값을 오른쪽 노드에 위치시킴
- C++ 컨테이너인 map, set Java의 TreeSet, TreeMap은 이진 탐색 트리로 이루어져 있다.

Map, Set

- 중복을 허용하지 않는 자료구조
- 대부분의 언어에서 Red-Black Tree를 사용함 (이진 탐색 트리의 한 종류)
- 특정한 기준에 의해서 정렬되어있음(JAVA의 경우 TreeMap, TreeSet만 정렬되어있고 HashMap, HashSet은 정렬 X)

자료구조	key	value
Map	중복될 수 없음	중복 가능
Set	중복될 수 없음	X(존재하지 않음)

Heap

-

우선순위 큐를 위한 자료구조

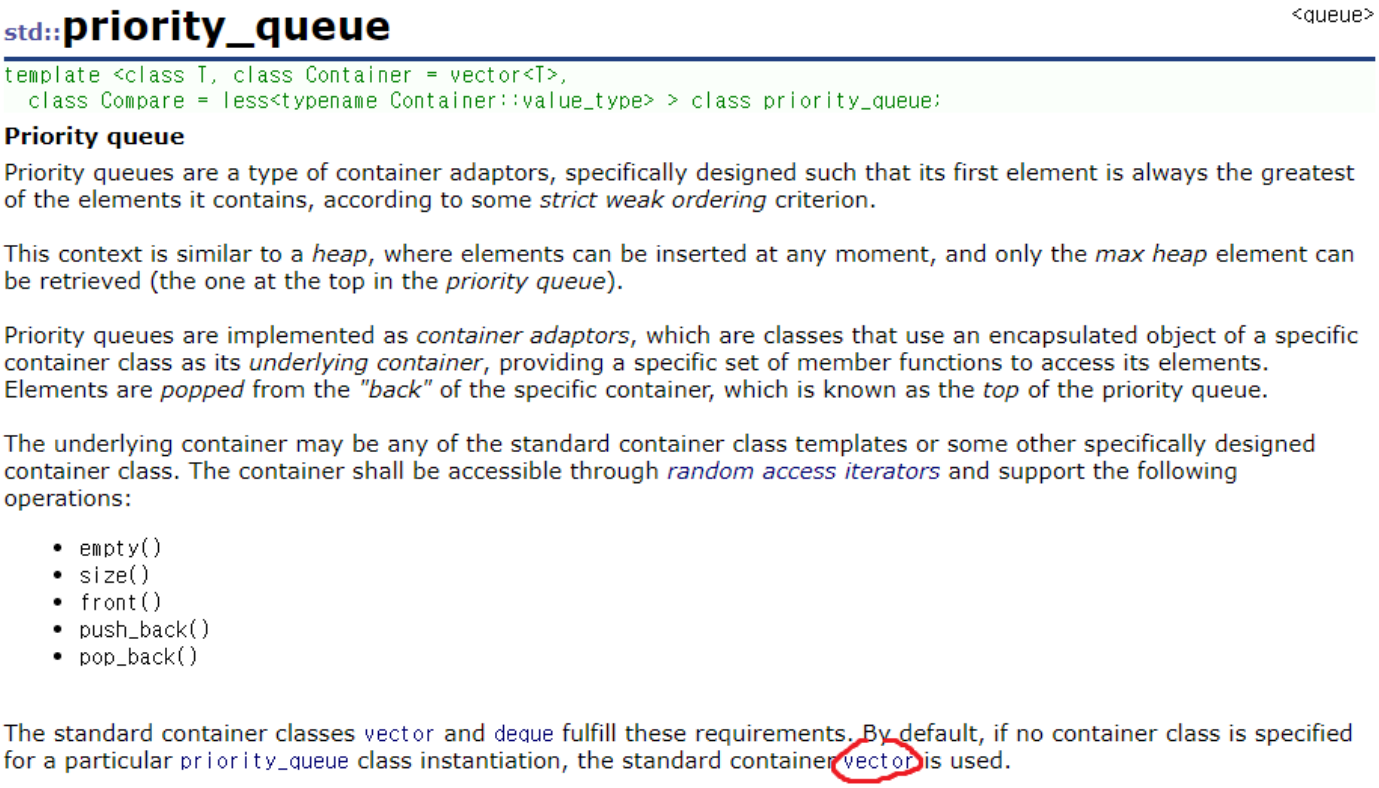
-

heap은 우선순위 큐를 구현하기 위한 자료구조로서, ***완전 이진트리*** 이다.

-

두 가지 규칙에 의해서 구현된다.

- 삽입 규칙 - 트리의 맨 마지막에 노드를 추가하고, 자신보다 우선순위가 높은 부모 노드를 만날 때 까지 부모 노드와 교환 한다.
- 삭제 규칙 - 트리의 루트 노드를 삭제하고, 트리의 맨 마지막에 위치한 노드를 루트 노드에 위치시킨 후 자신보다 우선순위가 낮은 자식 노드를 만날 때 까지 자식 노드와 교환 한다. (자식노드가 2개 존재 할 경우 더 우선순위가 낮은 자식 노드와 교환)



std::priority_queue <queue>

```
template <class T, class Container = vector<T>,
  class Compare = less<typename Container::value_type> > class priority_queue;
```

Priority queue

Priority queues are a type of container adaptors, specifically designed such that its first element is always the greatest of the elements it contains, according to some *strict weak ordering* criterion.

This context is similar to a *heap*, where elements can be inserted at any moment, and only the *max heap* element can be retrieved (the one at the top in the *priority queue*).

Priority queues are implemented as *container adaptors*, which are classes that use an encapsulated object of a specific container class as its *underlying container*, providing a specific set of member functions to access its elements. Elements are *popped* from the *"back"* of the specific container, which is known as the *top* of the priority queue.

The underlying container may be any of the standard container class templates or some other specifically designed container class. The container shall be accessible through *random access iterators* and support the following operations:

- empty()
- size()
- front()
- push_back()
- pop_back()

The standard container classes vector and deque fulfill these requirements. By default, if no container class is specified for a particular priority_queue class instantiation, the standard container vector is used.

이 두가지 룰을 통해서 삽입과 삭제가 발생 할 때마다 정렬을 반복하기 때문에
항상 루트 노드가 가장 높은 우선순위를 가지고 있고, 완전 이진트리의 조건을 만족시킨다.
그러므로 Heap은 C++ STL에서 Vector로 구현되어있다.

Heap과 Set의 차이

- Heap과 Set은 둘 다 정렬 기능을 가진 Tree형태의 자료구조이지만 내부 구조는 명확하게 다르다.
  Heap은 삽입과 삭제에 의해서 부모, 자식간의 우선순위는 확실하게 보장되지만, 형제&삼촌 노드와의 우선순위는 정의할 수 없다.
  즉, 부모 노드는 자식 노드보다 우선순위가 높기 때문에 ***루트 노드가 최고 우선순위의 노드*** 라는 것 만 보장할 수 있지만,
  Set은 이진 트리 중 이진 '탐색' 트리 이기 때문에 노드의 위치만 파악할 수 있다면 형제, 삼촌 또는 어떤 노드와도 우선순위를 비교할 수 있다.