1) Vector, list, int형 배열 차이Vector는 int 형 배열과는 다르게 자동으로 메모리를 할당해준다.

List는 노드 기반으로 자료의 추가나 삭제가 편리하지만 Vector는 좀 불편하다.

텍스트이(가) 표시된 사진

자동 생성된 설명List는 노드 기반이라서 임의의 위치에 바로 접근할 수 없지만 Vector랑 int형 배열은 바로 접근 가능하다.

2) Vector, list, set, map

Vector: 자동으로 메모리를 할당해주는 배열이다.

List: 노드 기반으로 구현된 자료구조이다.

Set: 노드 기반의 균형 이진트리로 값이 있는 지 확인하기 위한 것이고.

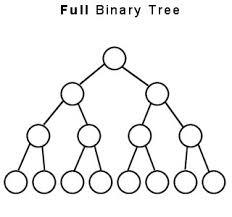
Map: 노드 기반의 균형 이진트리로 각 노드 안에 key와 value 값이 존재한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

3)set, map 내부 구현

Set과 map 둘 다 균형 이진트리를 사용한다.

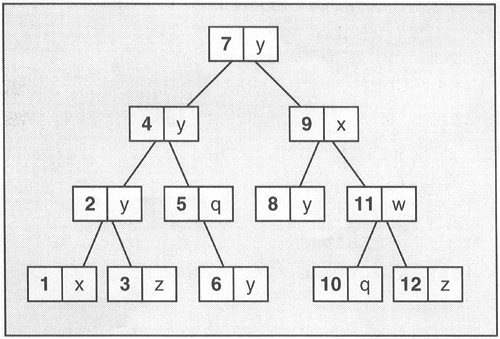
이런 식으로 균형 잡히게 만들어진 이진 트리이다.

Set 은 각 노드에 키 값만 넣고

Map은 테이블이다 딕셔너리 처럼 각 노드 안에

Key와 value값을 동시에 관리한다.

Map 형태

7과 y가 쌍을 이루고 9와 x가 쌍을 이루고 있다.

4) map에 key 값이 없을 때

실행텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명 결과는 이렇게 된다. Int 와 double 같은 경우에는 0을 반환하고 string에는 아무것도 출력하지 않는다.

검색해보니깐 operator []는 [k] 해서 k가 key랑 매치되는 게 없다면 새로 다시 만들고 그것을 반환한다. 그래서 사이즈는 증가한다.

5) sort, binary\_search, lower\_bound, upper\_bound

Sort: sort(start, end)으로 사용하고 [start, end)의 범위를 가지고 오름차순으로 정렬해준다.

Binary\_search: 정렬이 되어있다는 가정하에 숫자가 있는 지 찾는 알고리즘이다. 시간복잡도는 O(log N)이다.

Lower\_bound : lower\_bound(begin(), end(), value) 이렇게 쓰는 것으로 value 값이 나타나는 반환

Upper\_bound: value 값을 초과하는 첫번째 원소 위치를 구한다.

텍스트이(가) 표시된 사진

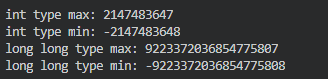
자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

6) numeric\_limits

텍스트이(가) 표시된 사진

자동 생성된 설명



7) int 형의 값의 범위는 대략 -21억부터 21억까지이다.

8) int & var는 레퍼런스 타입으로

선언을 하려면

a = 5;

int& var = a 이런 식으로 선언한다.

Int& var =5 (x), const int& var = 5(O)

텍스트이(가) 표시된 사진

자동 생성된 설명

