1. String

* #include <string>
  + Str1.length()
  + str1 = str1.substr(index, size)

1. Queue

* q.push(4)
* q.pop()
* q.front()
* 인덱스로 접근하지 못함

1. Map & set (unordered\_도 동일)

* m[‘B’] = 2
* for(auto i = m.begin(); it != m.end(); ++i)

it->first, it-> second로 접근

* m.find(‘B’) != m.end() // 찾은경우

1. deque

* #include <deque>, deque<int> d
* d.push\_back(1), d.push\_front(2)
* d.back(), d.front()
* d.pop\_back(), d.pop\_front()

1. sort, cmp : 새로운 자료형이든 뭐든, 무조건 cmp로 추가할 것

* 기본형 : 오름차순 : sort(v.begin(), v.end(), less<int>()) // 내림차순 : greater<int>()
* 주의 : less<int>만 쓰면 안되고 괄호()도 써야함 // priority\_queue는 ()안씀
* 정의형 : bool cmp (int a, int b)

return a<b;

* 객체형

typedef struct Key {

String s, int n;

bool operator < (const Key &k) const { // 해당 객체 수정불가

return n < k.n;

}Key;

sort(v.begin(), v.end())

1. priority\_queue, cmp : 새로운 자료형이면 무조건 struct형으로 정의

* #include <queue>
* Priority\_queue<int, vector<int>, less<int>) pq; // less<int>()괄호 안씀
* cmp사용시, sort처럼 bool cmp하면 안되고 struct형으로 넣어주어야함
* 우선순위 큐는 무조건 less<기본형>, greater<기본형>, struct에서 해결할 것
* Sort와 정렬 이름은 반대임

1. bfs, dfs문제
2. union-find문제
3. 다익스트라문제
4. 바이너리서치
5. DFS

* 조합 구현 : visited를 무조건 사용해야함

for(int i = start ; i < end ; i++)

if(!visited[i])

visited[i] = true; // 조합을 구현하기 위함 (전역변수에 선언)

DFS(cnt + 1, i); // 단, i대신에 i+1을 넣으면 visited 안써도 무방.

visited[i] = false;

* 순열 구현 : visited무조건 사용한 후, for문에 i가 0부터 시작임

for(int i = 0 ; i < end ; i++)

1. DP문제