비내림차순 = 같은것을 허용하는 오름차순

그래프/BFS

정점 vertex, 간선 edge

단순 경로, 단순 사이클 = 같은 정점을 한번만 방문하는.

특별히 언급없으면 경로 = 단순 경로

차수: 정점에 연결된 간선의 개수

인접행렬과 인접리스트중 인접 리스트를 많이 쓰는데 인접행렬은 정점의 제곱만큼 필요해서 너무 크기때문이다. E << V^2

DFS/BFS

그래프의 모든정점을 한번씩 지나려는 것.

DFS는 스택이용, 인접리스트를 사용해서 구현하는게 빠르다. O(V+E) vs O(V^2)

BFS는 큐를 이용해 지금 위치에서 갈 수 있는 것을 모두 큐에 넣는 것.(큐에 넣을때 방문이라고 체크.) 큐가 비어있으면 탐색완료

이분그래프

그래프를 A와 B로 나누는데 A에 있는 정점끼리 간선 없고 B끼리도 없다

연결구조 connected component

서로 연결된 그래프들

플러드필

어떤 위치와 연결된 모든 위치를 찾는 알고리즘(단지 번호붙이기)

BFS는 모든 가중치가 1일때 최단거리를 구하는 알고리즘

BFS는 항상 맵의 현재상태를 나타내는 행렬과 방문여부를 나타내는 check행렬이 있다.

BFS는 사람이 복제되는거라고 생각하면 편하다.

또한 BFS는 무언가를 할 수 있다, 없다를 판단하는데도 사용되는데 갈수 없다고 판명나면 못가는거니까.

가중치가 1이 아닌경우(0인경우도 있는 때)는 큐를 두개 쓰거나 deque를 쓴다.

Java의 HashSet은 집합이나 다름없어서 중복 없애기에 좋다.

백트래킹: 브루트 포스에서 재귀함수 쓸 때, 더이상 진행해도 의미없다 싶은 경우 중간에 멈춰버리는 경우. 순열은 모든 경우의 수를 다해서 별로기에 재귀를 이용한 백트래킹으로 하는게 더 좋다?

**비트마스크 = 10진수를 2진수로 만들어서 경우에 따라 사용.**

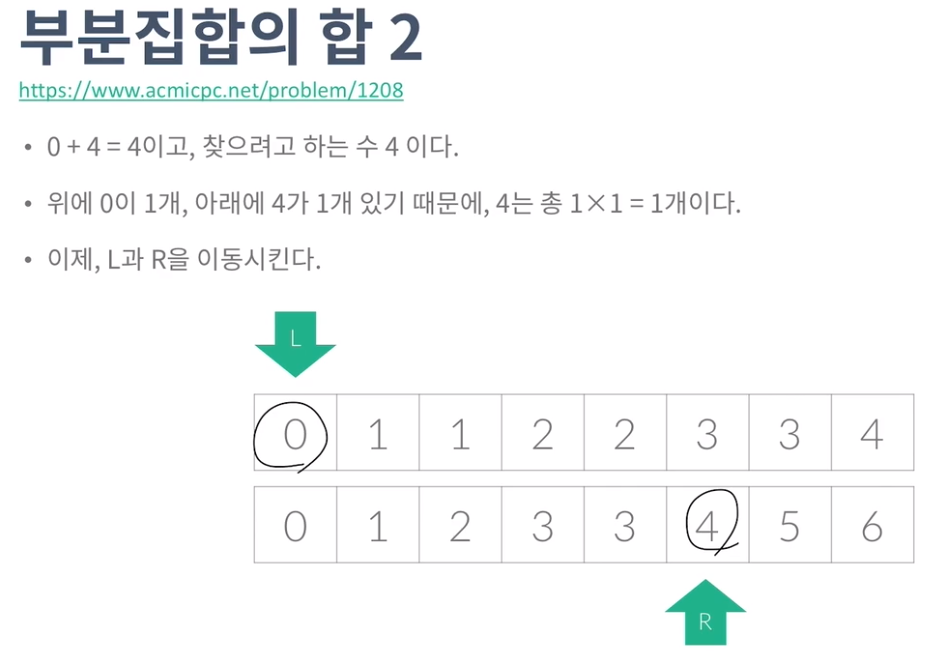
for (i=1; 1<<n; i++;) 이런식으로 비트형태로 만들고 각 자리가 0인지 1인지 등을 체크해서 사용한다.

대표적인 사용예는 2개의 팀으로 나누는 경우와 4가지 방향중 어디로 갈지를 정할때(4가지는 2^2이기에 2비트를 사용하자)

일부 경우만 해보기 = 절대 안될거는 시작도 안한다. 백트래킹과 비슷한데 백트래킹은 중간에 안될거는 더 안하는거고 일부 경우는 안될거는 시작도 안하는 차이인듯

**중간에서 만나기(Meet in the middle, MITM):** 문제가 너무 커서 문제를 예를들어 절반으로 나눠서 양쪽 절반에서 모든 경우를 해보고 합치는 것.

예를들어 부분집합의 합을 구하는 문제에서 최대 40개의 원소가 있는 집합일 경우 2^40이면 너무 큰데 이걸 반으로 쪼개서 2^20과 2^20서로 더하는 걸 하면 훨씬 계산이 적다.



정렬된 수열의 같은 수의 개수를 찾으려면 upper bound - lower bound

lower bound와 upper bound를 알아야 하는데

lower bound는 원하는 수보다 크거나 같으면서 가장 작은수,

upper bound는 원하는 수보다 크면서 가장 작은수

