<https://gmlwjd9405.github.io/2018/08/15/algorithm-bfs.html>

그래프 탐색의 한종류 -> DFS VS BFS

BFS -> 너비 우선 탐색 ->

한번에 루트를 완성시키는게 아니라 하나의 vertex에 대해 인접한 모든 점들을 탐색해서 큐에 저장해놓고 큐에 들어간 점을 하나씩 빼면서 다시 그 점에 인접한 모든 점들을 큐에 넣고 반복한다.

이해가 안됐던 점 -> a와 인접한 b,c,d를 큐에 저장한 뒤에 b가 아무도 인접하지 않는다는 것을 알게 된 뒤에 어떻게 c로 돌아와 탐색을 진행하는가? -> 선입선출인 queue의 특성에 따라 탐색하는 모든 점들은 들어간 순서대로 pop되어 탐색을 진행하게 된다. 즉 a가 가장 먼저 queue에 들어가 있었고, a가 queue에서 pop된 후에 뒤로 b,c,d가 queue로 push, b가 pop되어 탐색을 시작했다. 이때 b에 인접한 점이 아무것도 없었으므로 현재의 queue는 c,d가 저장된 상태에서 b의 인접한 점 탐색이 종료, c가 pop되어 c에 인접한 g,h,j 등이 queue에 push되며 즉 c에 인접한 점까지 탐색을 완료한 이후 queue는 [d , g , h , j]가 된다.

다만 이전 점에 인접해있어 이미 탐색이 완료된 점을 유지하기 위해 visited array 등을 두어 한번 탐색했던 점을 다시 탐색하지 않도록 해야한다.

시간 복잡도 -> 인접리스트 O(V+E), 인접 행렬 O(N^2)

인접 리스트는 bfs는 모든 vertex에 대해서 한번만 탐색하기 때문에 처음에 시작점 vertex에서 그와 연결된 모든 edge를 탐색, 그 edge로 root vertex와 연결된 vertex를 다음 탐색 지점으로 edge를 탐색함으로 모든 vertex와 edge의 수만큼 탐색이 진행되게 된다.

인접 행렬의 경우 어떠한 하나의 vertex v에 인접한 vertex를 구하기 위해 1차원 행렬을 모두 탐색해야 함으로(v = 0이라하면 (n,m) = vertex n과 vertex m이 인접하는가 인접하면 1, 아니면 0 이런식으로 저장되기 때문에 탐색하는 범위는 (0,1),(0,2)...(0,n) 으로 O(n)이 소요된다.) 모든 vertex에 대해 O(n)번 탐색하므로 O(n^2)이 소요된다.