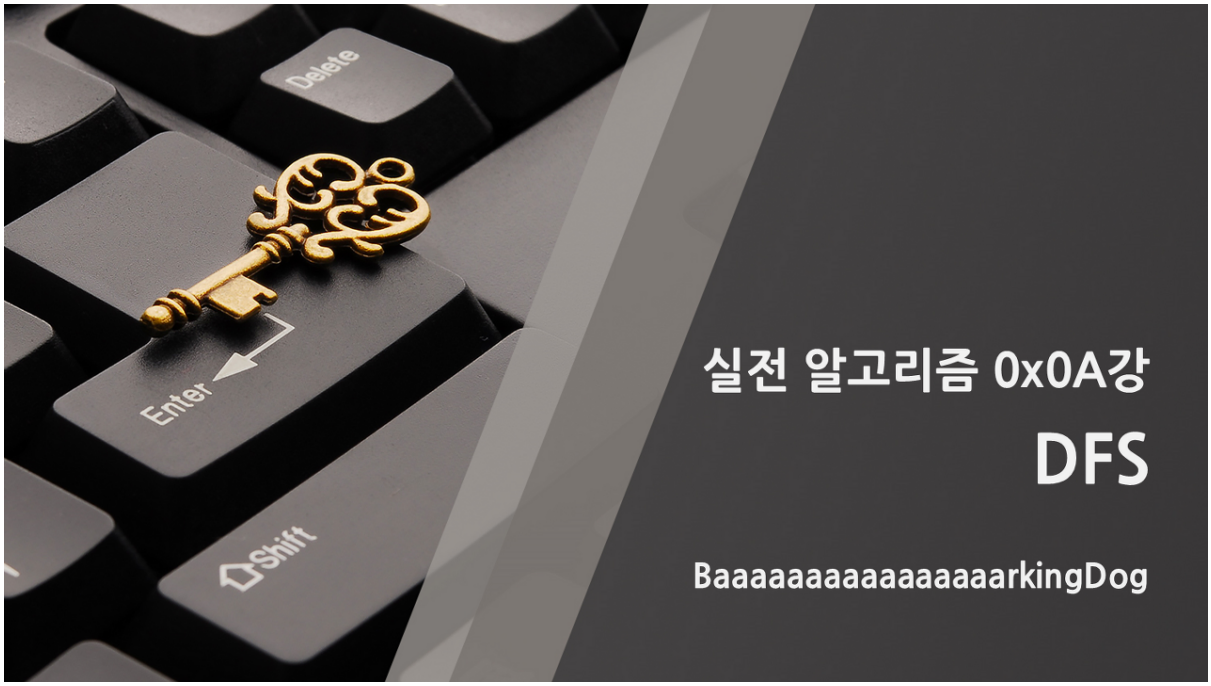
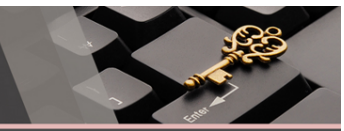
바킹독 0x0A DFS

태그	
공개여부	☑
날짜	
작성일자	



목차



0x00 알고리즘 설명

0x01 예시

0x02 BFS vs DFS

2

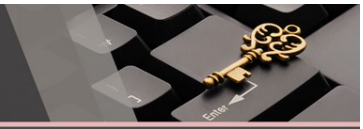
0x00 알고리즘 설명

DFS(Depth First Search)
: 다차원 배열에서 각 칸을 방문할 때 깊이를 우선으로 방문하는 알고리즘

BFS(Breadth First Search)
: 다차원 배열에서 각 칸을 방문할 때 너비를 우선으로 방문하는 알고리즘

3

0x01 예시



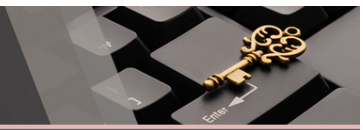
1. 시작하는 칸을 스택에 넣고 방문했다는 표시를 남김
2. 스택에서 원소를 꺼내어 그 칸과 상하좌우로 인접한 칸에 대해 3번을 진행
3. 해당 칸을 이전에 방문했다면 아무 것도 하지 않고, 처음으로 방문했다면 방문했다는 표시를 남기고 해당 칸을 스택에 삽입
4. 스택이 빌 때 까지 2번을 반복

모든 칸이 스택에 1번씩 들어가므로 시간복잡도는 칸이 N개일 때 O(N).

4

> 과정을 먼저 소개해보면 뭔가 어디서 본 것 같다는 느낌을 받을텐데, 바로 BFS의 과정에서 딴건 다 똑같고 큐만 스택으로 바뀐 것 뿐입니다. 큐를 쓰면 BFS이고 스택을 쓰면 DFS가 됩니다.

0x02 BFS vs DFS



			11			
18			7			
15			3			17
10	6	2	1	4	8	12
16			5		13	
			9			
			14			

BFS

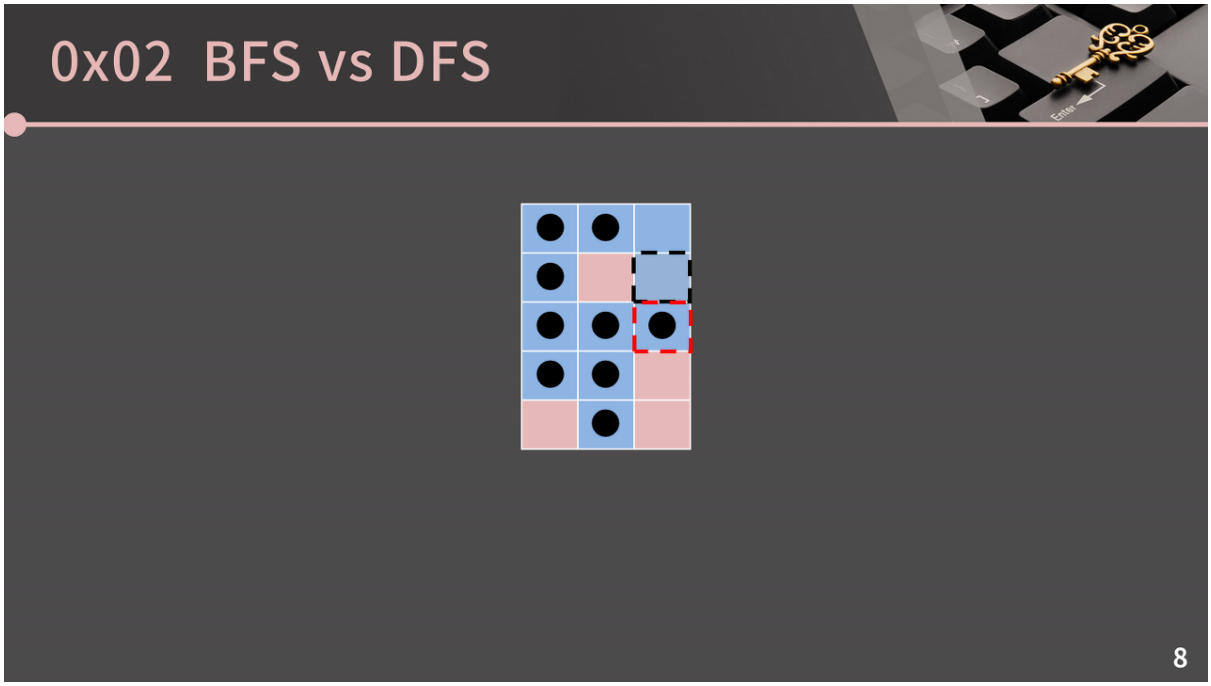
			10			
6			9			
5			8			14
4	3	2	1	11	12	13
7			16		15	
			17			
			18			

DFS

7

> BFS는 큐를 쓰고 DFS는 스택을 쓴다는 차이가 있지만 원소 하나를 빼내고 주변을 살펴본다는 알고리즘의 흐름은 똑같습니다. 하지만 둘의 **방문 순서**는 큰 차이가 있는데 우선 BFS에서의 방문 순서를 확인해보

> 겠습니다. 시작점을 중앙으로 잡았는데 보면 마치 냇가에 던진 돌로 인해 동심원이 생기는 것 처럼 **상하좌우로 퍼져나가는 것을 볼 수 있습니다.** 그리고 이전 단원에서 다룬 BFS의 성질인 **거리 순으로 방문한다는 것** 또한 잘 성립함을 알 수 있습니다. 이번에는 DFS에서의 방문 순서를 확인해보겠습니다. 이건 어떻게 비유를 해야할지 모르겠긴 한데 아무튼 뭔가 차이가 있다는건 알 수 있을 것입니다. 뭔가 **한 방향으로 막힐 때 까지 쭉 직진을 한다는걸 알 수 있습니다.** 지금 보신 것과 같이 BFS와 DFS는 방문 순서에 큰 차이가 있습니다.



그리고 **BFS에서 유용하게 썼던 "현재 보는 칸으로부터 추가되는 인접한 칸은 거리가 현재 보는 칸보다 1만큼 더 떨어져있다"는 성질**이 DFS에서는 성립하지 않습니다. 이 그림은 (0, 0)에서 DFS를 시작할 때 나오는 상황인데, 빨간색 칸은 거리가 4인 반면 검정색 칸은 거리가 3입니다. 그래서 거리를 계산할 때에는 DFS를 사용할 수 없습니다.

그러면 결국 우리는 다차원 배열에서 굳이 BFS 대신 DFS를 써야하는 일이 없습니다. Flood Fill은 BFS와 DFS 중에서 어느 것을 써도 상관없는데 거리 측정은 BFS만 할 수 있으니 BFS 대신 DFS를 쓸 일이 없습니다. 그래서 앞으로 다차원 배열에서 순회하는 문제를 풀 때 계속 BFS만 짜게 됩니다.

하지만 그렇다고 해서 DFS가 아예 무쓸모한건 아니고 나중에 **그래프와 트리라는 자료구조**

를 배울 때 DFS가 필요하게 됩니다. 그러니 아예 잊어버리지는 마시고 **DFS는 스택을 써서 다차원 배열의 순회를 처리하는 알고리즘이다, 깊이를 우선해서 탐색한다는데 깊이가 무슨 의미인지는 아직 잘 와닿지 않는다** 정도로만 기억하고 넘어가면 이번 단원에서 해야할 내용은 다 했습니다.