



실전 알고리즘 0x05강

BFS, DFS

BaaaaaaaaaaaaaaaaarkingDog

목차



0x00 플러드 필(Flood Fill)

0x01 BFS

0x02 DFS

0x03 문제 소개

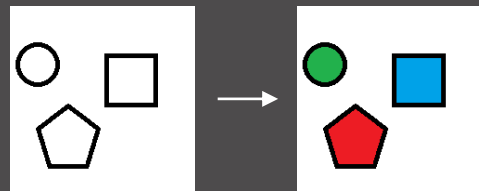
시작하기에 앞서..



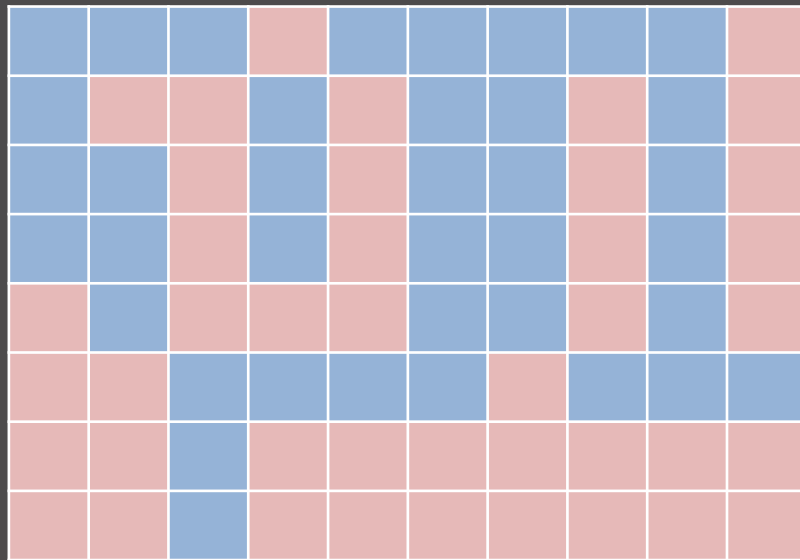
- 원래는 스택의 응용 중 하나인 전위·중위·후위 표기식을 먼저 다루려고 했습니다. 해당 내용은 학부 자료구조에서 스택을 배울 때 거의 대부분 짚고 넘어가는 내용이기 때문입니다.
- 그러나 전위·중위·후위 표기식이 난이도도 어렵고 요구되는 구현력도 다소 높은 것에 비해 다소 지엽적인 내용이기 때문에 코딩 테스트와 면접에 나올 확률은 0에 가깝다고 판단해서 전위·중위·후위 표기식을 실전 알고리즘 강의에서 다루지 않기로 결정했습니다.
- 그러나 삼성 역량 테스트 B형 정도의 수준에는 충분히 나올 수 있는 내용이니 본인이 알고리즘을 더 깊게 공부하고 싶다면 여유가 있을 때 직접 찾아서 공부해보시는 것을 추천드립니다.

0x00 플러드 필(Flood Fill)

- 다차원 배열에서 어떤 칸과 연결된 영역을 찾는 알고리즘



- 예시 문제 : 주어진 지도에서 파란색이 물, 빨간색이 육지일 때 바다의 갯수, 섬의 갯수는 각각 몇 개인가?



- 답 : 3, 6

0x00 플러드 필(Flood Fill)



- 플러드 필 문제는 BFS/DFS 알고리즘으로 해결할 수 있습니다.
- 적당한 난이도의 자료구조 지식과 구현력을 요구하는 문제이기 때문에 코딩테스트에 정말 자주 나오는 유형입니다.
- BFS는 큐로 구현할 수 있고 DFS는 스택으로 구현할 수 있습니다.

0x01 BFS



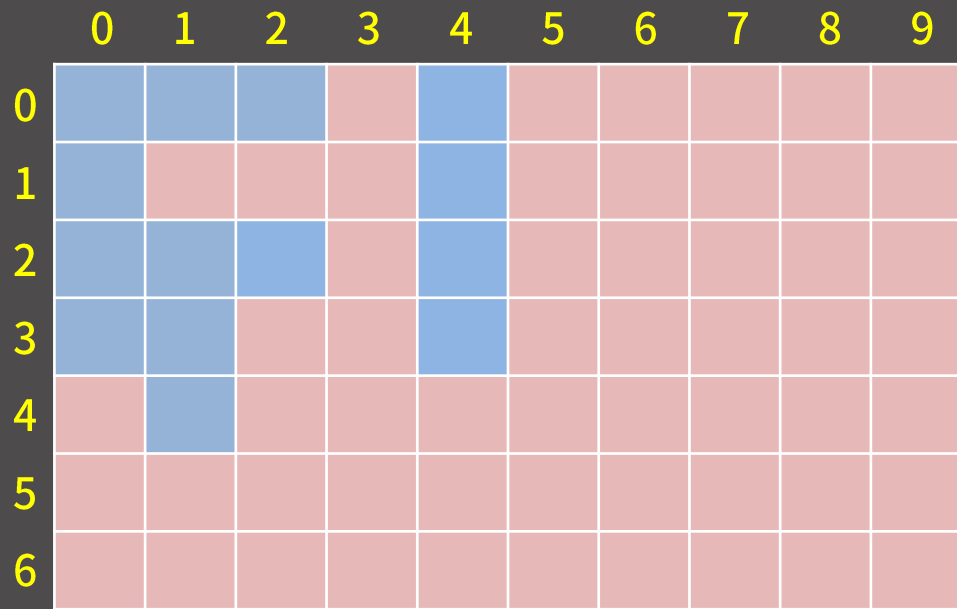
- BFS(Breadth First Search)는 다차원 배열에서 각 칸을 방문할 때 너비를 우선으로 방문하는 알고리즘입니다.
- 잘 이해가 안가시죠? 정상입니다. 원래 BFS와 DFS는 그래프 자료구조에서 모든 노드를 방문하기 위한 알고리즘으로, 다차원 배열이 그래프 자료구조의 특수한 형태이기 때문에 다차원 배열에서도 BFS와 DFS를 사용할 수 있습니다.
- 그렇기에 BFS와 DFS를 정확하게 이해하려면 그래프 자료구조에 대한 이해가 선행되어야 하나 그건 배보다 배꼽이 커지는 느낌입니다.
- 대신 구체적인 알고리즘을 보면서 다차원 배열에서의 BFS를 이해해봅시다.

0x01 BFS - 알고리즘 설명



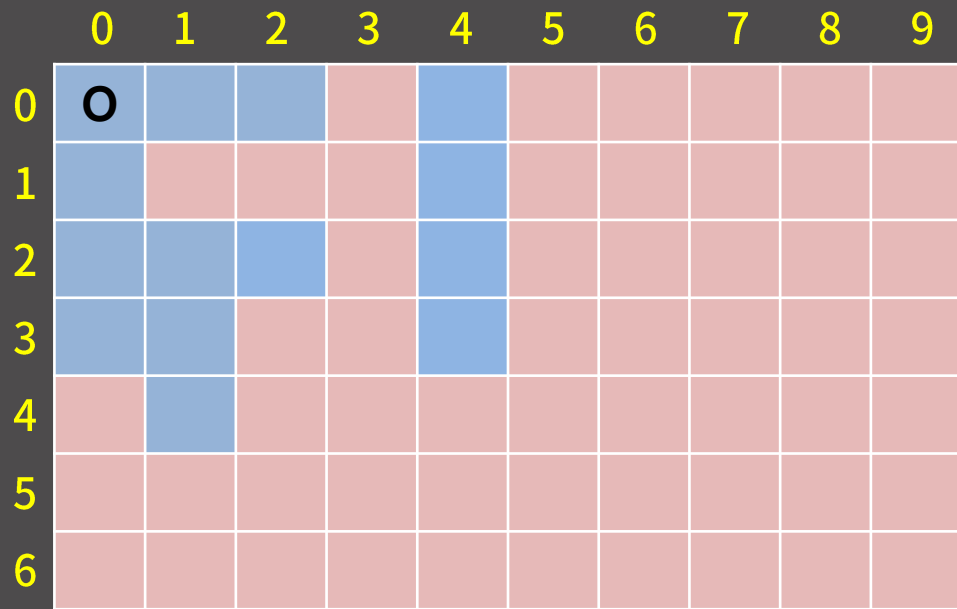
- BFS의 과정은 아래와 같습니다.
- 1. 시작하는 칸을 큐에 넣고 방문했다는 표시를 남긴다.
- 2. 큐에서 원소를 꺼내어 그 칸에 상/하/좌/우로 인접한 4개의 칸에 대해 3번 행동을 한다.
- 3. 해당 칸을 이전에 방문했다면 아무 것도 하지 않고, 처음으로 방문했다면 방문했다는 표시를 남기고 해당 칸을 큐에 넣는다.
- 4. 큐의 모든 원소가 빌 때 까지 2를 반복한다.
- 모든 칸이 큐에 1번 씩만 들어감이 보장되므로 시간복잡도는 칸이 N 개일 때 $O(N)$ 입니다.
- 직접 예시를 보면서 이해해봅시다.

0x01 BFS – 예시



- (0, 0)과 연결된 모든 파란 칸을 확인하는 것이 목적입니다. 방문했다는 표시는 칸 내에 O를 두어서 표시합니다.

0x01 BFS – 예시



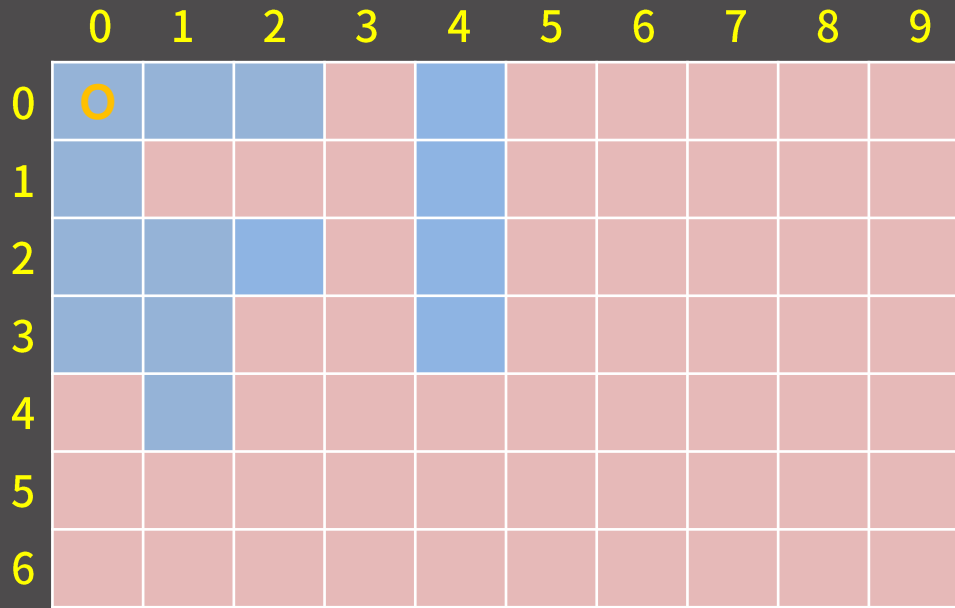
현재 보고 있는 칸



(0, 0)

- BFS를 시작합니다. 시작 칸인 (0, 0)을 큐에 넣고 방문했다는 표시를 남깁니다.

0x01 BFS - 예시

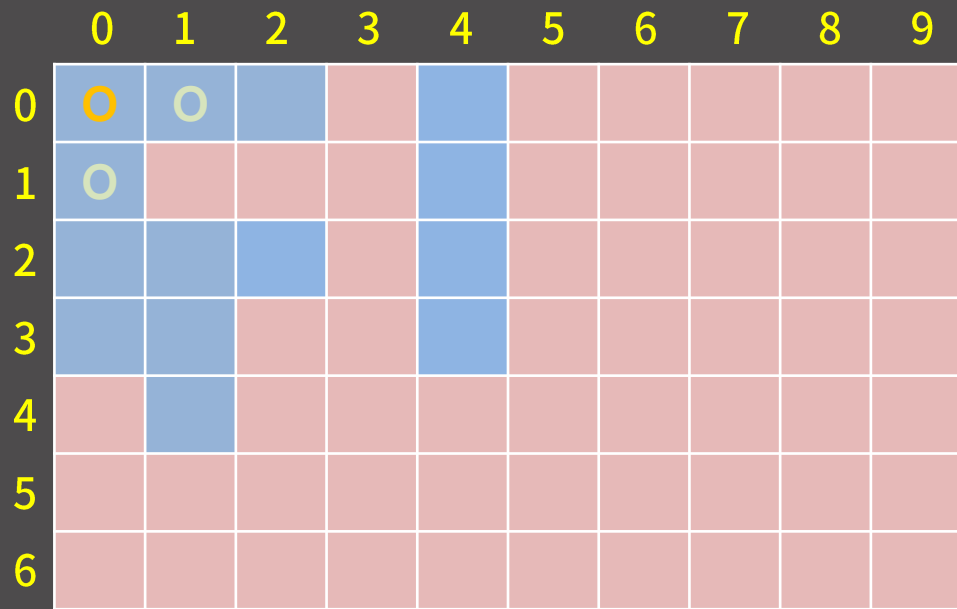


현재 보고 있는 칸

(0, 0)

- 큐가 비어있지 않으므로 큐의 front인 (0, 0)을 꺼냅니다.

0x01 BFS - 예시



현재 보고 있는 칸

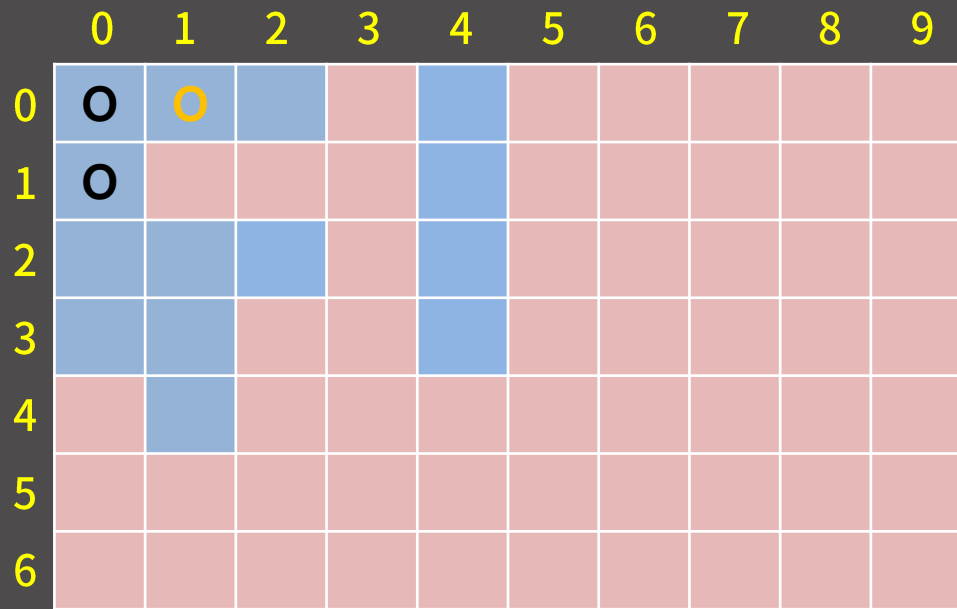
(0, 0)

(1, 0)

(0, 1)

- (0, 0)과 인접한 칸 (0, 1), (1, 0) 모두 파란 칸이면서 아직 방문하지 않았으므로 방문했다는 표시를 남기고 큐에 넣습니다.

0x01 BFS - 예시



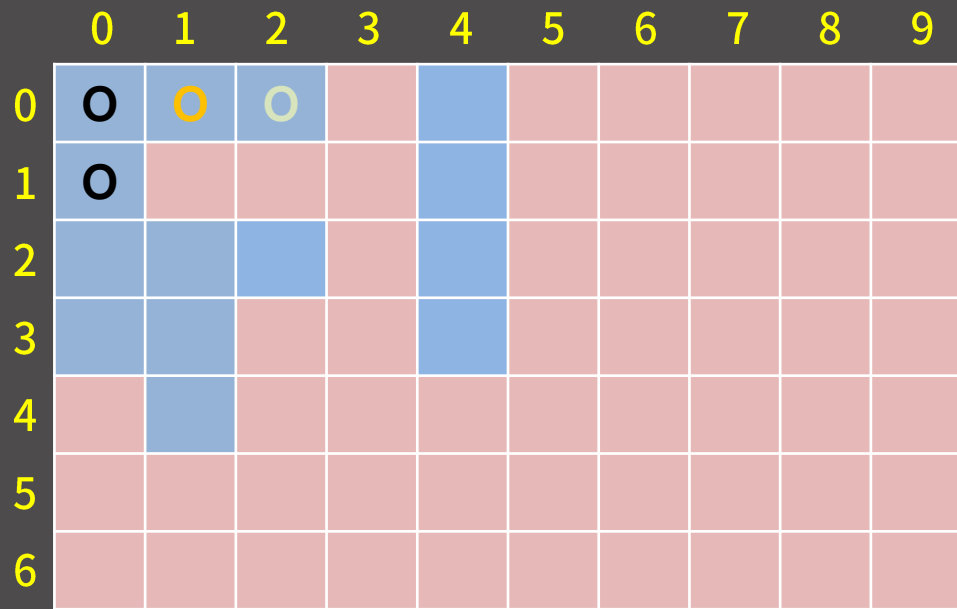
현재 보고 있는 칸

(0, 1)

(1, 0)

- 큐가 비어있지 않으므로 큐의 front인 (0, 1)을 꺼냅니다.

0x01 BFS - 예시



현재 보고 있는 칸

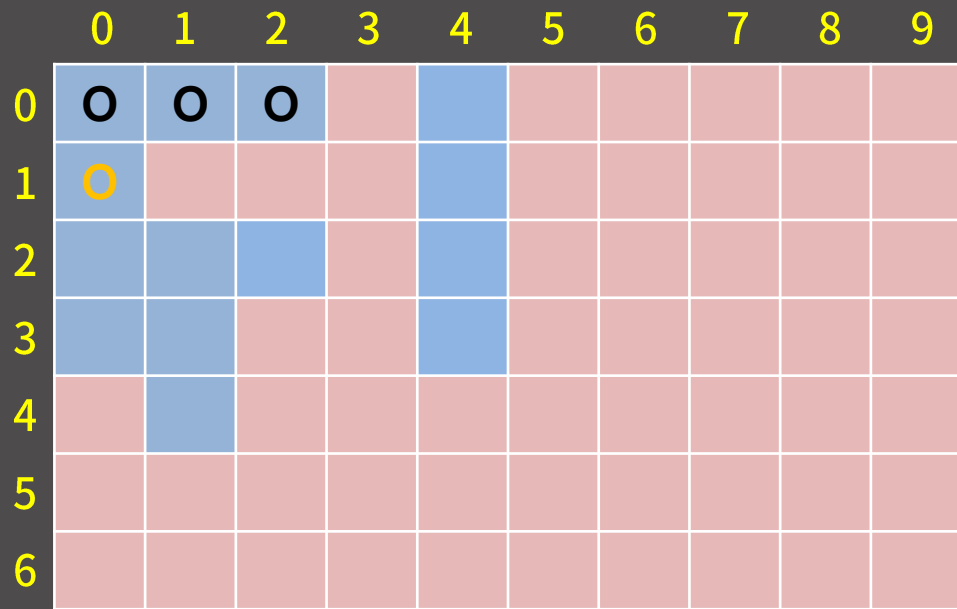
(0, 1)

(0, 2)

(1, 0)

- (0, 1)과 인접한 칸 중에서 (0, 0)은 이미 방문했고, (1, 1)은 파란 칸이 아니므로 넘어갑니다. (0, 2)는 파란 칸이면서 아직 방문하지 않았으므로 방문했다는 표시를 남기고 큐에 넣습니다.

0x01 BFS - 예시



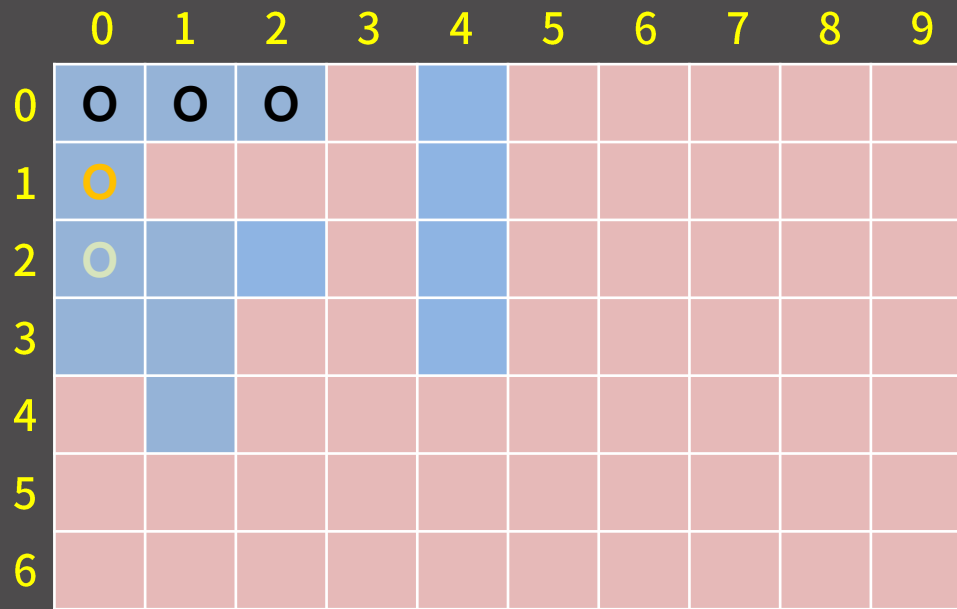
현재 보고 있는 칸

(1, 0)

(0, 2)

- 큐가 비어있지 않으므로 큐의 front인 (1, 0)을 꺼냅니다.

0x01 BFS - 예시



현재 보고 있는 칸

(1, 0)

(2, 0)

(0, 2)

- (1, 0)과 인접한 칸 중에서 (0, 0)은 이미 방문했고, (1, 1)은 파란 칸이 아니므로 넘어갑니다. (2, 0)는 파란 칸이면서 아직 방문하지 않았으므로 방문했다는 표시를 남기고 큐에 넣습니다.

0x01 BFS - 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○									
2	○									
3										
4										
5										
6										

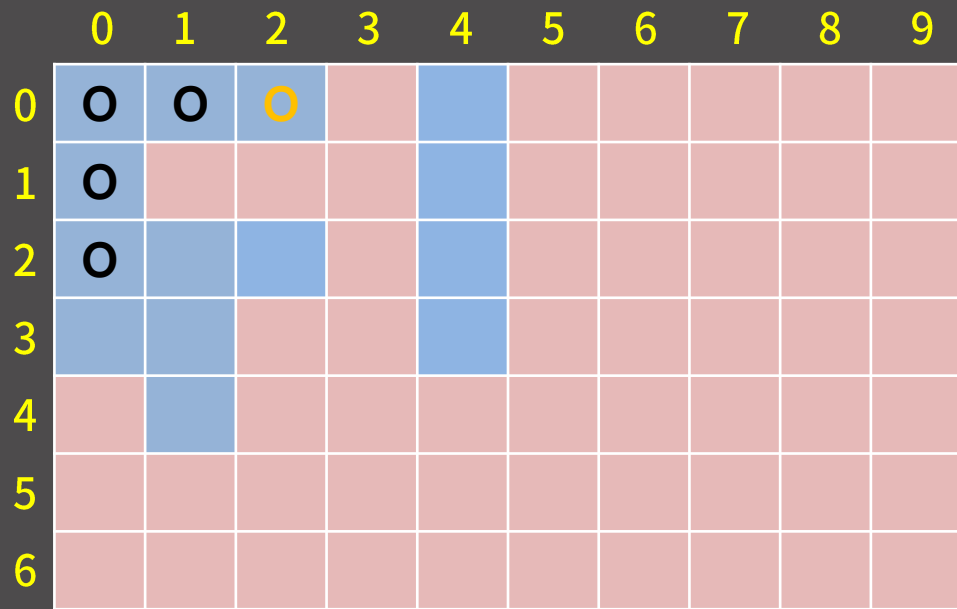
현재 보고 있는 칸

(0, 2)

(2, 0)

- 큐가 비어있지 않으므로 큐의 front인 (0, 2)를 꺼냅니다.

0x01 BFS - 예시



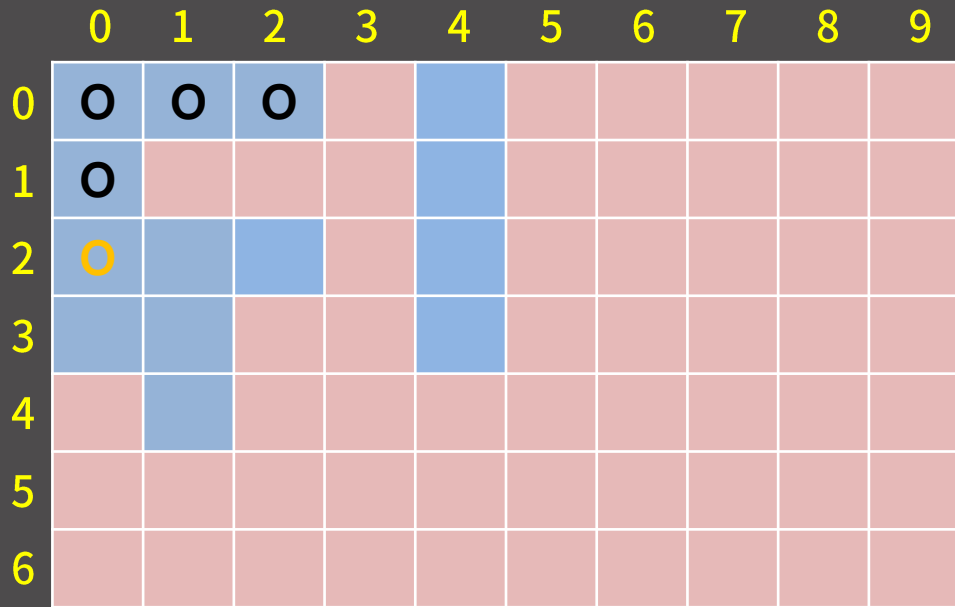
현재 보고 있는 칸

(0, 2)

(2, 0)

- (0, 2)와 인접한 칸 중에서 (0, 1)은 이미 방문했고, (0, 3)과 (1, 2)은 파란 칸이 아니기 때문에 모든 인접한 칸에 대해 작업할 것이 없습니다.

0x01 BFS - 예시

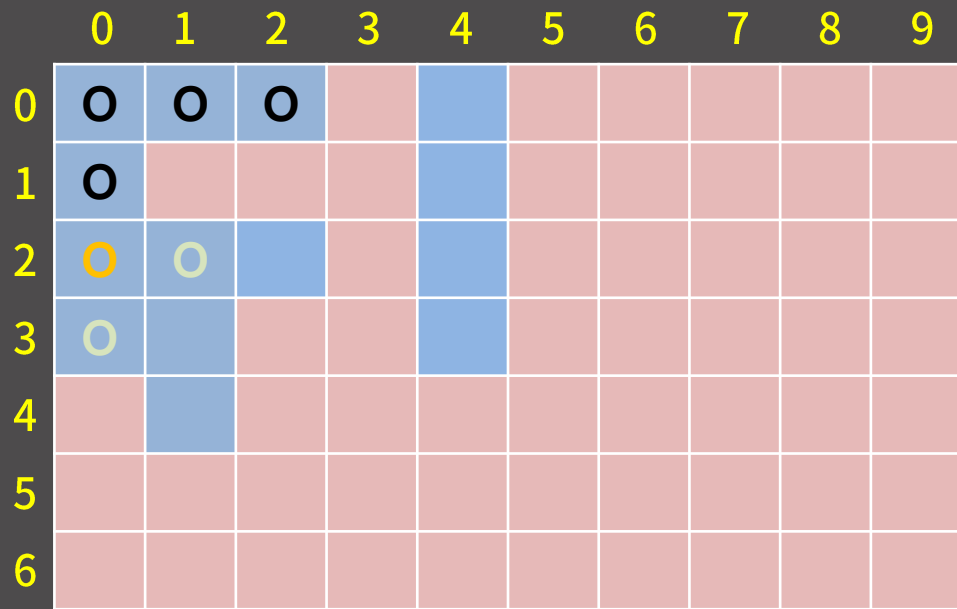


현재 보고 있는 칸

(2, 0)

- 큐가 비어있지 않으므로 큐의 front인 (2, 0)을 꺼냅니다.

0x01 BFS - 예시



현재 보고 있는 칸

(2, 0)

(3, 0)

(2, 1)

- (2, 0)와 인접한 칸 중에서 (1, 0)은 이미 방문했으니 넘어갑니다. (2, 1)과 (3, 0)은 파란 칸이면서 아직 방문하지 않았으므로 방문했다는 표시를 남기고 큐에 넣습니다.

0x01 BFS - 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○									
2	○	○								
3	○									
4										
5										
6										

현재 보고 있는 칸

(2, 1)

(3, 0)

- 큐가 비어있지 않으므로 큐의 front인 (2, 1)을 꺼냅니다.

0x01 BFS - 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○									
2	○	○	○							
3	○	○								
4										
5										
6										

현재 보고 있는 칸

(3, 0)

(2, 2)

(3, 1)

- 큐가 비어있지 않으므로 큐의 front인 (3, 0)을 꺼냅니다.

0x01 BFS - 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○									
2	○	○	○							
3	○	○								
4										
5										
6										

현재 보고 있는 칸

(3, 0)

(2, 2)

(3, 1)

- (3, 0)와 인접한 칸 중에서 (2, 0)과 (3, 1)은 이미 방문했고, (4, 0)은 파란 칸이 아니기 때문에 모든 인접한 칸에 대해 작업할 것이 없습니다.

0x01 BFS - 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○									
2	○	○	○							
3	○	○								
4										
5										
6										

현재 보고 있는 칸

(3, 1)

(2, 2)

- 큐가 비어있지 않으므로 큐의 front인 (3, 1)을 꺼냅니다.

0x01 BFS - 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○									
2	○	○	○							
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(3, 1)

(4, 1)

(2, 2)

- (3, 1)과 인접한 칸 중에서 (3, 0)과 (2, 1)은 이미 방문했으니 넘어갑니다. (3, 2)는 빨간 칸이므로 넘어갑니다. (4, 1)은 파란 칸이면서 아직 방문하지 않았으므로 방문했다는 표시를 남기고 큐에 넣습니다.

0x01 BFS - 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○									
2	○	○	○							
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(2, 2)

(4, 1)

- 큐가 비어있지 않으므로 큐의 front인 (2, 2)을 꺼냅니다.
- 인접한 모든 칸이 이미 방문했거나 파란 칸이 아니므로 넘어갑니다.

0x01 BFS - 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○									
2	○	○	○							
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(4, 1)

- 큐가 비어있지 않으므로 큐의 front인 (4, 1)을 꺼냅니다.
- 인접한 모든 칸이 이미 방문했거나 파란 칸이 아니므로 넘어갑니다.

0x01 BFS - 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○									
2	○	○	○							
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸



- 큐가 비어있으므로 알고리즘을 종료합니다.
- 방문한 기록으로부터 어느 칸이 (0, 0)과 이어진 파란 칸인지 알 수 있습니다.

0x01 BFS - 예시 코드



- 가로와 세로를 표현하기 위해 STL의 pair를 활용했습니다.
- pair 레퍼런스 : <http://www.cplusplus.com/reference/utility/pair/>
- 만약 STL을 제공하지 않는다면 직접 구조체를 만들거나 그냥 큐에서 2개의 data 배열을 두면 됩니다.
- <https://bit.ly/2CLUjk0> (출력되는 방문 순서는 상/하/좌/우 중에 어디를 먼저 큐에 넣냐에 따라 다를 수 있음.)

0x01 BFS - 응용

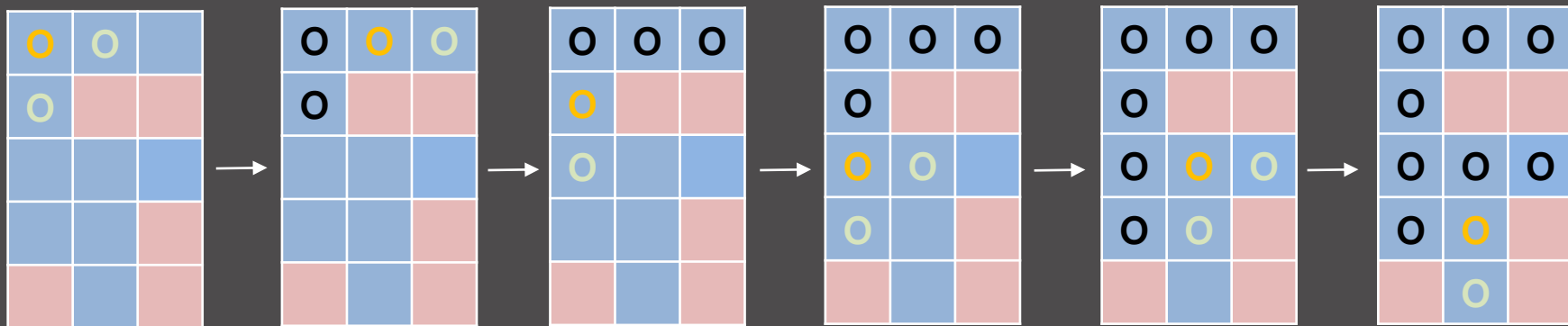


- BOJ 1926번 : 그림(icpc.me/1926), 도화지 내에 모든 연결된 1들에 대한 처리를 해주고 각각의 넓이를 구해야 합니다.
- 모든 1을 처리하기 위해서는 $(0, 0)$ 에서 시작하는 것이 아니라 이중 for문을 돌면서 아직 방문하지 않은 시작점을 찾으면 됩니다. 넓이를 구하기 위해서는 각 시작점에 대해 내부적인 변수를 두어 큐에서 원소를 꺼낼 때 마다 그 변수를 1 증가시키면 됩니다.
- 정답 코드 : <http://boj.kr/6a8e108a6ce046cab0ac278c03517be1>

0x01 BFS - 응용



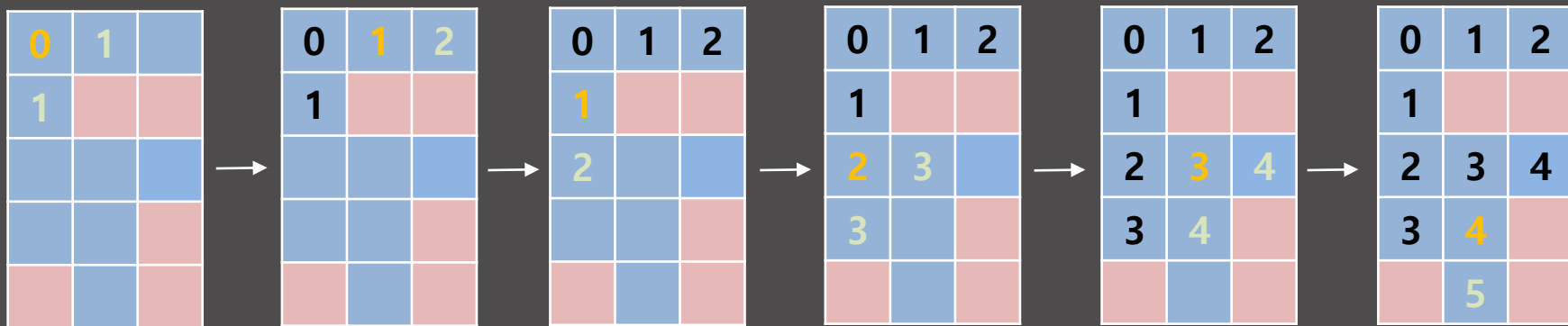
- BOJ 2178번 : 미로 탐색(icpc.me/2178), 미로의 시작점에서 끝점으로 가는 최단 경로의 길이를 찾는 것이 문제입니다.
- BFS를 이용해 시작점에서 다른 모든 점으로의 최단 경로를 찾을 수 있습니다.
- BFS의 과정을 다시 살펴봅시다.



0x01 BFS - 응용



- 뭔가 시작점 (0, 0)에서부터 퍼져나가는 느낌이 들지 않나요?
- (0, 0)에서부터의 거리를 적어보면 더 명확해집니다.
- 현재 보는 칸(주황)로부터 추가되는 인접한 칸(초록)은 반드시 현재 보는 칸 보다 시작점으로부터 1만큼 더 떨어져 있습니다.



0x01 BFS - 응용



- 이 성질을 이용해 시작 점과의 거리를 저장할 `dist` 배열을 하나 뒀으로서 시작 점에서 다른 모든 점으로 가는데 필요한 거리를 계산할 수 있습니다.
- `dist` 배열을 미리 -1로 초기화 시켜두면 굳이 `vis` 배열을 따로 둘 필요 없이 `dist` 배열의 값이 0 이상인지로 방문 여부를 확인할 수 있습니다.
- 정답 코드 : <http://boj.kr/17777c88f0af4ce38eff1cca825c66d3>

0x01 BFS – 자주 실수하는 요소



- 1. 시작 점에 방문했다는 표시를 남기지 않는다.
- 2. 큐에 넣을 때 방문했다는 표시를 하지 않고, 큐에서 빼낼 때 방문했다는 표시를 남긴다.
- 3. 문자열로 지도를 입력받았을 때 '1'과 비교를 해야 하는데 1과 비교를 한다.
- 4. 이웃한 원소가 범위를 벗어났는지에 대한 체크를 잘못했다.
- 5. 배열로 큐를 구현한 경우, 큐의 크기를 충분하게 주지 않았다.
- 6. 가로와 세로를 헷갈렸다.
- 7. 여러 테스트 케이스를 처리해야 하는 경우, 변수를 제대로 초기화하지 않았다.

0x02 DFS



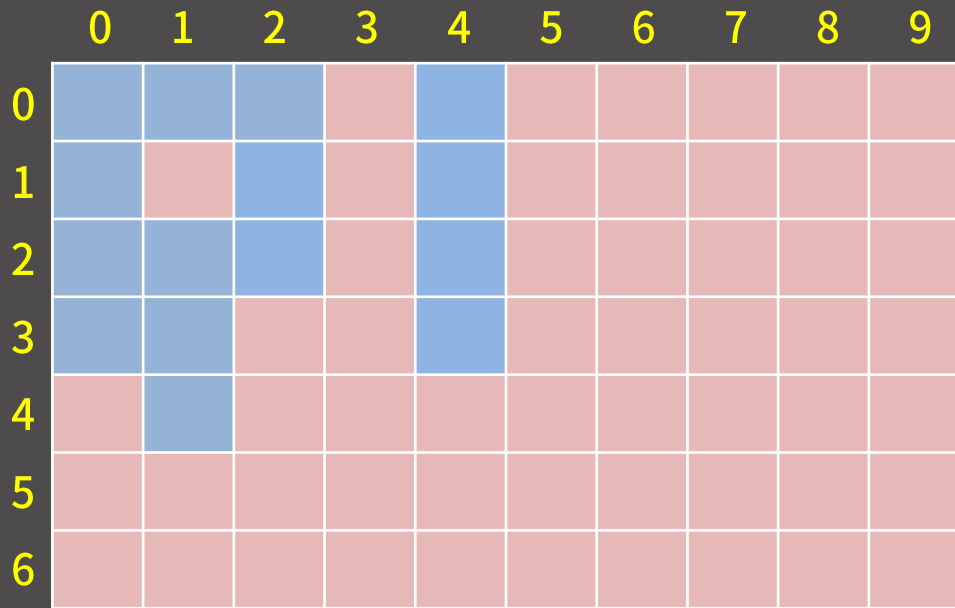
- DFS(Depth First Search)는 다차원 배열에서 각 칸을 방문할 때 깊이를 우선으로 방문하는 알고리즘입니다.
- BFS와 거의 동일한 흐름이나 큐 대신 스택을 쓴다는 차이점만 있습니다.
- 구체적인 알고리즘을 보면서 다차원 배열에서의 DFS를 이해해봅시다.

0x02 DFS – 알고리즘 설명



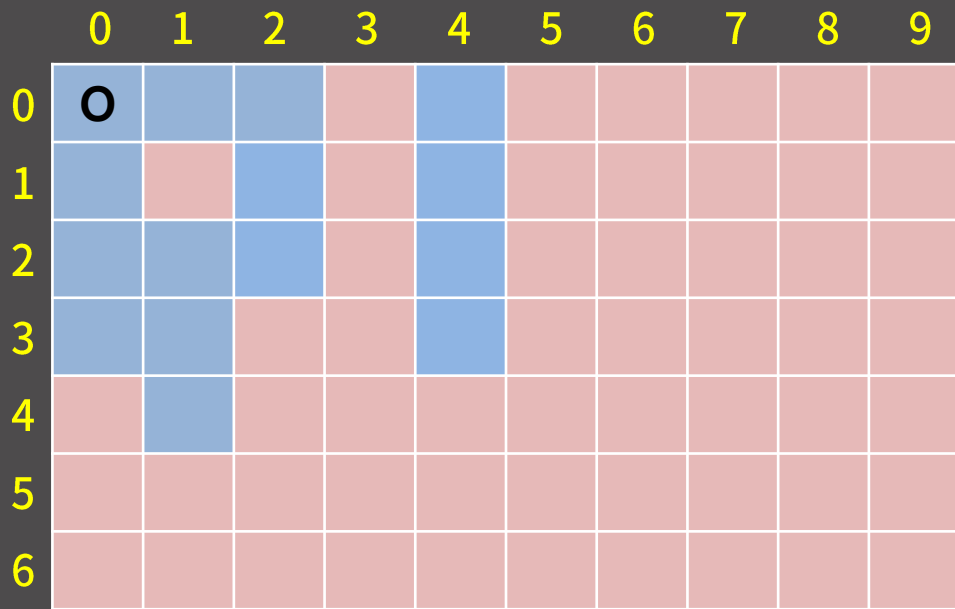
- DFS의 과정은 아래와 같습니다.
- 1. 시작하는 칸을 스택에 넣고 방문했다는 표시를 남긴다.
- 2. 스택에서 원소를 꺼내 그 칸에 상/하/좌/우로 인접한 4개의 칸에 대해 3번 행동을 한다.
- 3. 해당 칸을 이전에 방문했다면 아무 것도 하지 않고, 처음으로 방문했다면 방문했다는 표시를 남기고 해당 칸을 스택에 넣는다.
- 4. 스택의 모든 원소가 빌 때 까지 2를 반복한다.
- 모든 칸이 스택에 1번씩 들어감이 보장되므로 시간복잡도는 칸이 N 개일 때 $O(N)$ 입니다.
- 직접 예시를 보면서 이해해봅시다.

0x02 DFS – 예시



- (0, 0)과 연결된 모든 파란 칸을 확인하는 것이 목적입니다. 방문했다는 표시는 칸 내에 O를 두어서 표시합니다.

0x02 DFS – 예시



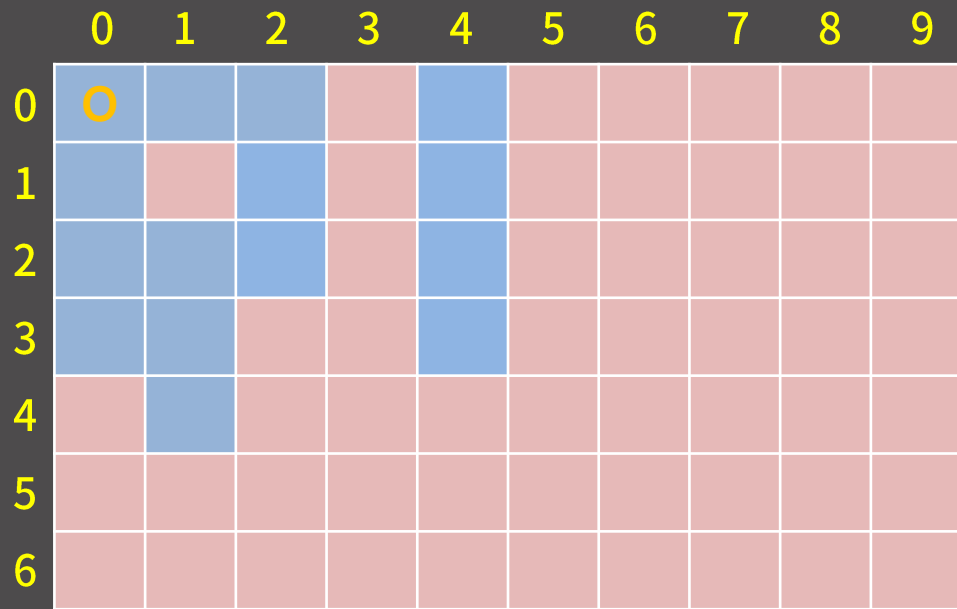
현재 보고 있는 칸



(0, 0)

- DFS를 시작합니다. 시작 칸인 (0, 0)을 스택에 넣고 방문했다는 표시를 남깁니다.

0x02 DFS – 예시



현재 보고 있는 칸

(0, 0)

- 스택이 비어있지 않으므로 스택의 top인 (0, 0)을 꺼냅니다.

0x02 DFS – 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○									
2										
3										
4										
5										
6										

현재 보고 있는 칸

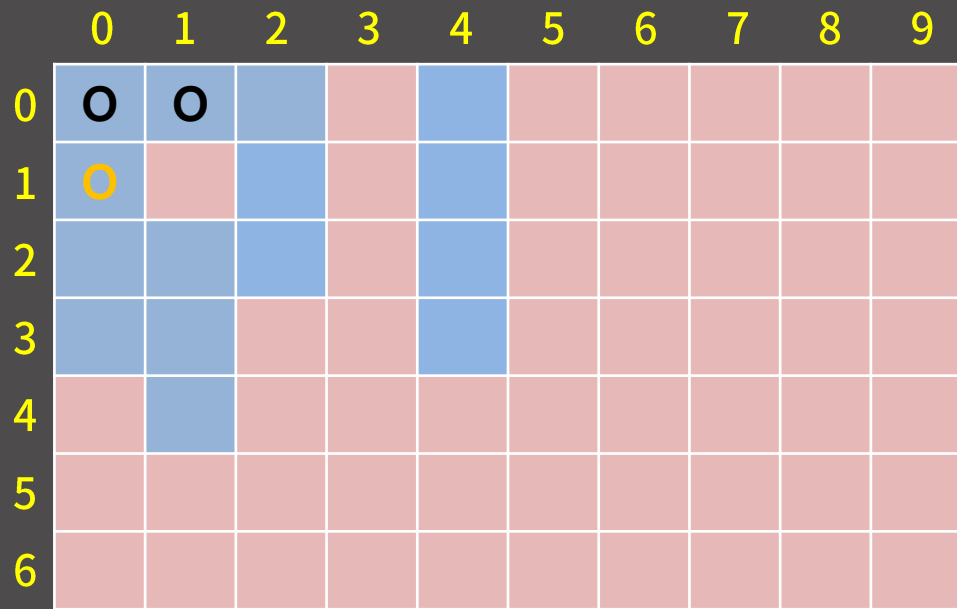
(0, 0)

(1, 0)

(0, 1)

- (0, 0)과 인접한 칸인 (0, 1)과 (1, 0) 모두 파란 칸이면서 아직 방문하지 않았으므로 방문했다는 표시를 남기고 스택에 넣습니다.

0x02 DFS – 예시



	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○									
2										
3										
4										
5										
6										

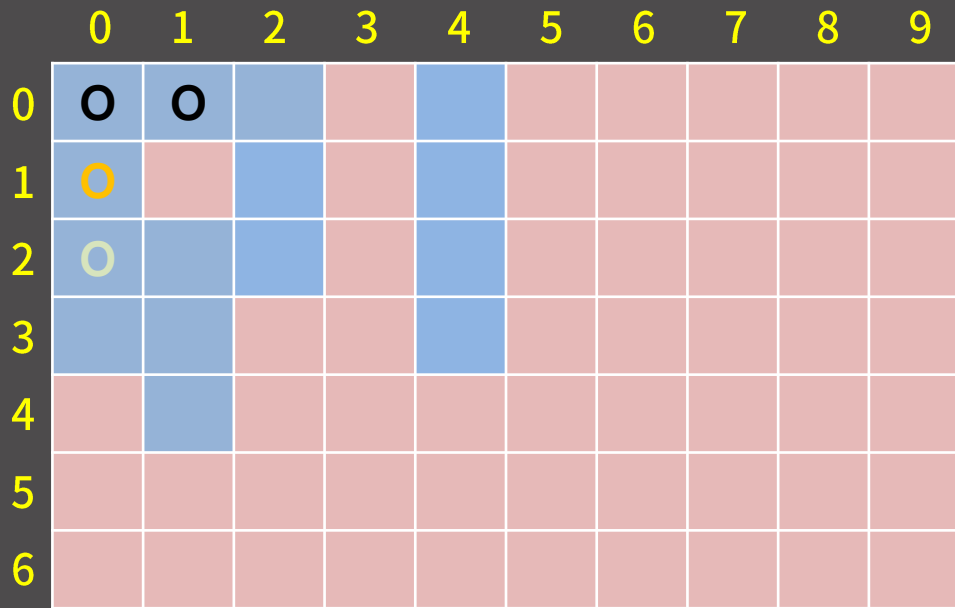
현재 보고 있는 칸

(1, 0)

(0, 1)

- 스택이 비어있지 않으므로 스택의 top인 (1, 0)을 꺼냅니다.

0x02 DFS – 예시



현재 보고 있는 칸

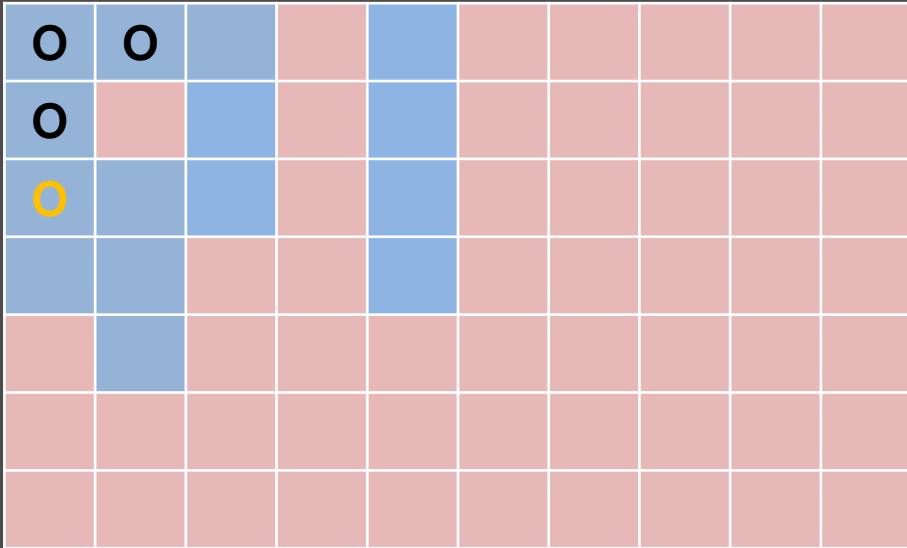
(1, 0)

(2, 0)

(0, 1)

- (1, 0)과 인접한 칸 중에서 (1, 1)은 빨간 칸이므로 넘어갑니다. (0, 0)은 이미 방문한 칸이므로 넘어갑니다. (2, 0)은 파란 칸이면서 아직 방문하지 않았으므로 방문했다는 표시를 남기고 큐에 넣습니다.

0x02 DFS – 예시



	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○									
2	○									
3										
4										
5										
6										

현재 보고 있는 칸

(2, 0)

(0, 1)

- 스택이 비어있지 않으므로 스택의 top인 (2, 0)을 꺼냅니다.

0x02 DFS - 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○									
2	○	○								
3	○									
4										
5										
6										

현재 보고 있는 칸

(2, 0)

(3, 0)

(2, 1)

(0, 1)

- (2, 0)과 인접한 칸 중에서 (1, 0)은 방문한 칸이므로 넘어갑니다. (3, 0)과 (2, 1)은 파란 칸이면서 아직 방문하지 않았으므로 방문했다는 표시를 남기고 큐에 넣습니다.

0x02 DFS – 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○									
2	○	○								
3	○									
4										
5										
6										

현재 보고 있는 칸

(3, 0)

(2, 1)

(0, 1)

- 스택이 비어있지 않으므로 스택의 top인 (3, 0)을 꺼냅니다.

0x02 DFS - 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○									
2	○	○								
3	○	○								
4										
5										
6										

현재 보고 있는 칸

(3, 0)

(3, 1)

(2, 1)

(0, 1)

- (3, 0)과 인접한 칸 중에서 (2, 0)은 방문한 칸이므로 넘어갑니다. (4, 0)은 빨간 칸이므로 넘어갑니다. (3, 1)은 파란 칸이면서 아직 방문하지 않았으므로 방문했다는 표시를 남기고 큐에 넣습니다.

0x02 DFS – 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○									
2	○	○								
3	○	○								
4										
5										
6										

현재 보고 있는 칸

(3, 1)

(2, 1)

(0, 1)

- 스택이 비어있지 않으므로 스택의 top인 (3, 1)을 꺼냅니다.

0x02 DFS - 예시



	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○									
2	○	○								
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(3, 1)

(4, 1)

(2, 1)

(0, 1)

- (3, 1)과 인접한 칸 중에서 (3, 0)은 방문한 칸이므로 넘어갑니다. (3, 2)은 빨간 칸이므로 넘어갑니다. (4, 1)은 파란 칸이면서 아직 방문하지 않았으므로 방문했다는 표시를 남기고 큐에 넣습니다.

0x02 DFS – 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○									
2	○	○								
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(4, 1)

(2, 1)

(0, 1)

- 스택이 비어있지 않으므로 스택의 top인 (4, 1)을 꺼냅니다.

0x02 DFS – 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○									
2	○	○								
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(4, 1)

(2, 1)

(0, 1)

- (4, 1)과 인접한 칸 중 (3, 0)은 이미 방문한 칸이고 (4, 0), (5, 1), (4, 2)는 모두 빨간 칸이므로 넘어갑니다.

0x02 DFS – 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○									
2	○	○								
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(2, 1)

(0, 1)

- 스택이 비어있지 않으므로 스택의 top인 (2, 1)을 꺼냅니다.

0x02 DFS - 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○									
2	○	○	○							
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(4, 1)

(2, 2)

(0, 1)

- (2, 1)과 인접한 칸 중에서 (2, 0), (3, 1)은 방문한 칸이므로 넘어갑니다. (1, 1)은 빨간 칸이므로 넘어갑니다. (2, 2)은 파란 칸이면서 아직 방문하지 않았으므로 방문했다는 표시를 남기고 큐에 넣습니다.

0x02 DFS – 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○									
2	○	○	○							
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(2, 2)

(0, 1)

- 스택이 비어있지 않으므로 스택의 top인 (2, 2)을 꺼냅니다.

0x02 DFS – 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○		○							
2	○	○	○							
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(2, 2)

(1, 2)

(0, 1)

- (2, 2)과 인접한 칸 중에서 (2, 1)은 방문한 칸이므로 넘어갑니다. (3, 2)와 (2, 3)은 빨간 칸이므로 넘어갑니다. (1, 2)는 파란 칸이면서 아직 방문하지 않았으므로 방문했다는 표시를 남기고 큐에 넣습니다.

0x02 DFS – 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○								
1	○		○							
2	○	○	○							
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(1, 2)

(0, 1)

- 스택이 비어있지 않으므로 스택의 top인 (1, 2)을 꺼냅니다.

0x02 DFS - 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○		○							
2	○	○	○							
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(1, 2)

(0, 2)

(0, 1)

- (1, 2)와 인접한 칸 중에서 (2, 2)는 방문한 칸이므로 넘어갑니다. (1, 1)과 (1, 3)은 빨간 칸이므로 넘어갑니다. (0, 2)은 파란 칸이면서 아직 방문하지 않았으므로 방문했다는 표시를 남기고 큐에 넣습니다.

0x02 DFS – 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○		○							
2	○	○	○							
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(0, 2)

(0, 1)

- 스택이 비어있지 않으므로 스택의 top인 (0, 2)을 꺼냅니다.

0x02 DFS – 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○		○							
2	○	○	○							
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(0, 2)

(0, 1)

- (0, 2)와 인접한 칸 중에서 (0, 1)과 (1, 2)는 방문한 칸이므로 넘어갑니다. (0, 3)은 빨간 칸이므로 넘어갑니다.

0x02 DFS – 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○		○							
2	○	○	○							
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(0, 1)

- 스택이 비어있지 않으므로 스택의 top인 (0, 1)을 꺼냅니다.

0x02 DFS – 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○		○							
2	○	○	○							
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸

(0, 1)

- (0, 1)과 인접한 모든 칸이 이미 방문한 칸이거나 빨간 칸이므로 넘어갑니다.

0x02 DFS - 예시

	0	1	2	3	4	5	6	7	8	9
0	○	○	○							
1	○		○							
2	○	○	○							
3	○	○								
4		○								
5										
6										

현재 보고 있는 칸



- 스택이 비어있으므로 알고리즘을 종료합니다.
- 방문한 기록으로부터 어느 칸이 (0, 0)과 이어진 파란 칸인지 알 수 있습니다.

0x02 DFS – 예시 코드



- DFS와 마찬가지로 가로와 세로를 표현하기 위해 STL의 pair를 활용했습니다.
- 코드 : <https://bit.ly/2ReIrlj> (출력되는 방문 순서는 상/하/좌/우 중에 어디를 먼저 스택에 넣냐에 따라 다를 수 있음.)

0x02 DFS – 주의사항



- BFS와는 다르게 DFS는 현재 보는 칸(주황)로부터 추가되는 인접한 칸(초록)은 반드시 현재 보는 칸 보다 시작점으로부터 1만큼 더 떨어져 있다는 성질이 성립되지 않습니다.
- 그렇기 때문에 시작점으로부터의 거리를 잴 때는 DFS 대신 BFS를 사용해야 합니다.

0	1	
1		3
2	3	4
3	4	
	5	

0x02 DFS – 주의사항



- 다차원 배열에서 DFS는 딱히 쓸 일이 없습니다. 어차피 Flood Fill은 BFS에서도 할 수 있고, DFS에서는 시작 점에서 다른 모든 점으로의 거리를 썰 수 없기 때문입니다.
- 그러나 나중에 그래프를 배우고 난 뒤에 DFS를 사용해야 하는 상황이 생기므로 DFS에 대한 개념은 가지고 있는 것이 좋습니다.

0x03 문제 소개



- Flood Fill / BFS / DFS 관련 문제는 BOJ에 정말 끝도 없이 쌓여 있습니다.
- 그 중에서 그래프 개념이 필요없는 문제를 뽑아 그룹에 난이도 순으로 담아두었으니 꾸준히 풀어보세요.
- 코딩 테스트에 단골로 출제되는 유형이기 때문에 반복 숙달을 통해 코드의 흐름을 손에 익히는 것이 중요합니다.

강의 정리



- Flood Fill / BFS / DFS를 익혔습니다.
- BFS의 응용 사례들과 자주 실수하는 부분을 살펴보았습니다.