0x09 BFS

상태	완료
담당자	㉮ 서연

BFS란?

💡 그래프 자료구조 상에서 모든 노드를 방문하기 위해 사용하고, 각 칸을 방문할 때 너비를 우선으로 방문하는 알고리즘

> 그렇다면 그래프란?
> → 그래프는 정점과 간선으로 이루어진 비선형 자료구조이다!

BFS의 목표

시작점에서 출발해서 상하좌우로 이어진 모든 노드를 확인하는 것!

그렇다면 어떻게 할 수 있을까?

💡 BFS 탐색이 시작되면 시작점을 방문했다는 표시를 남기고 해당 칸을 큐에 넣음.

이 세팅이 끝난 후에는 큐가 빌때까지 큐의 front를 빼고 이때 상하좌우를 살피면서 인접한 정점을 큐에 넣어주는 역할을 반복하게 됨.

- 단, 방문하지 않았으면서, 반드시 인접한 노드만을 방문해야 함.
    - 탐색하면서 a 좌표와 b 좌표가 특정 노드와 같이 간선으로 연결 되어있다고 할 때는 나중에 들어온 정점은 이미 앞서 해당 좌표가 들어왔기 때문에 또 큐에 넣지 않음.

BFS 코드 흐름 정리

1. 전체 보드를 선언함
2. 방문 여부를 저장할 배열 선언
3. 보드(그래프)의 행의 수, 열의 수 변수에 할당
4. 상하좌우 값을 담을 1차원 델타 배열 선언 및 초기화
5. 인접한 노드를 담을 Queue 선언
6. 0, 0 시작 좌표를 Queue의 rear에 push
7. 방문 배열에 true 처리
8. Queue가 비기 전까지 탐색을 진행함.
    9. 현재 front 값을 빼고 1차원 배열 curr에 저장
    10. 상하좌우 탐색
        10-1. dr, dc의 값을 동시에 현 좌표에 더하여 탐색함.
        10-2. board 범위를 넘으면 continue
        10-3. 이미 방문한 노드이거나 인접한 노드가 없으면 인접한 다른 좌표를 검사하기 위해 continue
        10-4. dnl 10-2, 10-3을 제외하면 모두 탐색 가능하므로 Queue에 담고,
        해당 좌표 값을 visited 배열에 방문했음을 표시함.

-> 큐에 원소가 들어있으면 계속해서 반복됨.

코드 구현

```
import java.util.ArrayDeque;
import java.util.Arrays;
import java.util.Deque;

public class BFS_예제 {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

        int n = 9;
        int m = 6;

        int[][] board = new int[][] {
            { 1, 1, 1, 0, 1, 0, 0, 0, 0, 0 },
```

```
        { 1, 0, 0, 0, 1, 0, 0, 0, 0, 0 },
        { 1, 1, 1, 0, 1, 0, 0, 0, 0, 0 },
        { 1, 1, 0, 0, 1, 0, 0, 0, 0, 0 },
        { 0, 1, 0, 0, 0, 0, 0, 0, 0, 0 },
        { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },
        { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 } };

    // 방문 여부를 저장할 배열 선언
    boolean[][] visited = new boolean[n][m];

    // 상하좌우 값을 담을 배열 선언
    int[] dr = { -1, 0, 0, 1 };
    int[] dc = { 0, -1, 1, 0 };

    // 방문 여부를 저장할 큐 선언
    Deque<int[]> queue = new ArrayDeque<>();

    int nr = 0;
    int nc = 0;
    queue.addLast(new int[] { nr, nc });
    visited[nr][nc] = true;

    // 큐가 빌 때까지 탐색함
    while (!queue.isEmpty()) {
        int[] curr = queue.pop();
        System.out.print(Arrays.toString(curr)+"→");

        for (int i = 0; i < dr.length; i++) {
            nr = curr[0] + dr[i];
            nc = curr[1] + dc[i];

            if (nr < 0 || nr >= n || nc < 0 || nc >= m) {
                continue;
            }

            if (visited[nr][nc] || board[nr][nc] != 1) {
                continue;
            }
```

```
                queue.addLast(new int[] { nr, nc });
                visited[nr][nc] = true;
//              System.out.println(nr+" "+nc);

            }
        }
    }

}
```

▼ 나름대로 느낀 중요한 점

1. 어떤 것을 큐에 담고 탐색을 할 지가 중요한 것 같다!