

BOJ 16929 : Two Dots

beberiche · 약 2시간 전

통계 수정 삭제

DFS

algorithms

백준

algorithms

▼ 목록 보기

10/12



문제

16929번

제출

맞힌 사람

숏코딩

재제출 결과

채점 현황

내 제출

난이도 기어

강의

질문 게시판

Two Dots 완료 ☆

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	512 MB	7651	3501	2223	42.940%

문제

Two Dots는 Playdots, Inc.에서 만든 게임이다. 게임의 기초 단계는 크기가 N×M인 게임판 위에서 진행된다.

11 MOVES

12 / 15

13 / 15

9 / 15

⚙️

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

11 MOVES

12 / 15

13 / 15

9 / 15

⚙️

11 MOVES

12 / 15

13 / 15

9 / 15

⚙️

각각의 칸은 색이 칠해진 공이 하나씩 있다. 이 게임의 핵심은 같은 색으로 이루어진 사이클을 찾는 것이다.

다음은 위의 게임판에서 만들 수 있는 사이클의 예시이다.

11 MOVES

12 / 15

13 / 15

9 / 15

⚙️

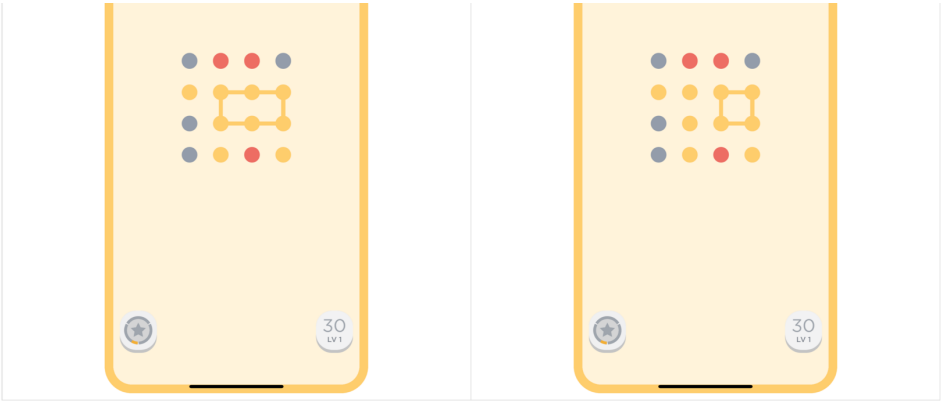
11 MOVES

12 / 15

13 / 15

9 / 15

⚙️



점 k 개 d_1, d_2, \dots, d_k 로 이루어진 사이클의 정의는 아래와 같다.

- 모든 k 개의 점은 서로 다르다.
- k 는 4보다 크거나 같다.
- 모든 점의 색은 같다.
- 모든 $1 \leq i \leq k-1$ 에 대해서, d_i 와 d_{i+1} 은 인접하다. 또, d_k 와 d_1 도 인접해야 한다. 두 점이 인접하다는 것은 각각의 점이 들어있는 칸이 변을 공유한다는 의미이다.

게임판의 상태가 주어졌을 때, 사이클이 존재하는지 아닌지 구해보자.

입력

첫째 줄에 게임판의 크기 N, M 이 주어진다. 둘째 줄부터 N 개의 줄에 게임판의 상태가 주어진다. 게임판은 모두 점으로 가득차 있고, 게임판의 상태는 점의 색을 의미한다. 점의 색은 알파벳 대문자 한 글자이다.

출력

사이클이 존재하는 경우에는 "Yes", 없는 경우에는 "No"를 출력한다.

제한

- $2 \leq N, M \leq 50$

예제 입력 1 복사

3 4
AAAA
ABCA
AAAA

예제 출력 1 복사

Yes

예제 입력 2 복사

3 4
AAAA
ABCA
AADA

예제 출력 2 복사

No

예제 입력 3 복사

4 4
YYYY
BYBY
BBBY
BBBY

예제 출력 3 복사

Yes

예제 입력 4 복사

7 6
AAAAAB
ABBBAB
ABAAAAB
ABABBB
ABAAAAB
ABBBAB
AAAAAB

예제 출력 4 복사

Yes

예제 입력 5 복사

2 13
ABCDEFGHIJKLM
NOPQRSTUVWXYZ

예제 출력 5 복사

No

풀이과정

사이클 찾기 문제.
이전에도 사이클 문제를 푼 적이 있는데, 그때는 그래프 형태로 이어졌다면 이번에는 모든 노드가 배열형태로

이루어져있다. 형태만 배열이지 결국 사방으로 이어진 그래프라고 판단하면 된다.

<https://velog.io/@beberiche/BOJ-16947-서울-지하철-2호선>

정답

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.*;

public class Main {
    static int N, M;
    static char[][] map;
    static boolean[][] visited, isCheck;
    static final int[] DR = {-1, 0, 1, 0};
    static final int[] DC = {0, 1, 0, -1};

    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());
        N = Integer.parseInt(st.nextToken());
        M = Integer.parseInt(st.nextToken());

        map = new char[N][M];
        for (int i = 0; i < N; i++) {
            char[] charArr = br.readLine().toCharArray();
            for (int j = 0; j < M; j++) {
                map[i][j] = charArr[j];
            }
        }

        // 현재 사이클의 시작점을 알리는 boolean[][]
        isCheck = new boolean[N][M];
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                visited = new boolean[N][M];
                isCheck[i][j] = true;
                DFS(i, j, map[i][j], 1);
                isCheck[i][j] = false;
            }
        }
        System.out.println("No");
    }

    private static void DFS(int r, int c, char comp, int cnt) {
        visited[r][c] = true;

        for (int d = 0; d < 4; d++) {
            int nr = r + DR[d];
            int nc = c + DC[d];

            if (nr < 0 || nc < 0 || nr >= N || nc >= M) continue;
            if (map[nr][nc] != comp) continue;
            // 사이클 조건
            // 최소 4개의 노드로 이루어짐
            // 처음과 끝이 모두 같은 좌표여야 함
            if (visited[nr][nc] && isCheck[nr][nc] && cnt >= 4) {
                System.out.println("Yes");
                System.exit(0);
            }
            if (visited[nr][nc]) continue;

            DFS(nr, nc, comp, cnt + 1);
        }
    }
}
```