

몇주차였더라

1074번 제출 맞힌 사람 스포딩 재채점 결과 채점 현황 내 제출 강의▼ 질문 검색

Z



시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
0.5 초 (추가 시간 없음)	512 MB	52867	19776	14894	38.891%

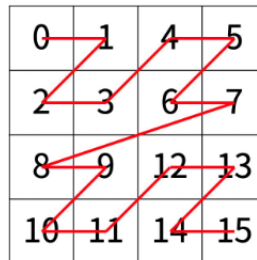
문제

한수는 크기가 $2^N \times 2^N$ 인 2차원 배열을 Z모양으로 탐색하려고 한다. 예를 들어, 2×2 배열을 왼쪽 위칸, 오른쪽 위칸, 왼쪽 아래칸, 오른쪽 아래칸 순서대로 방문하면 Z모양이다.

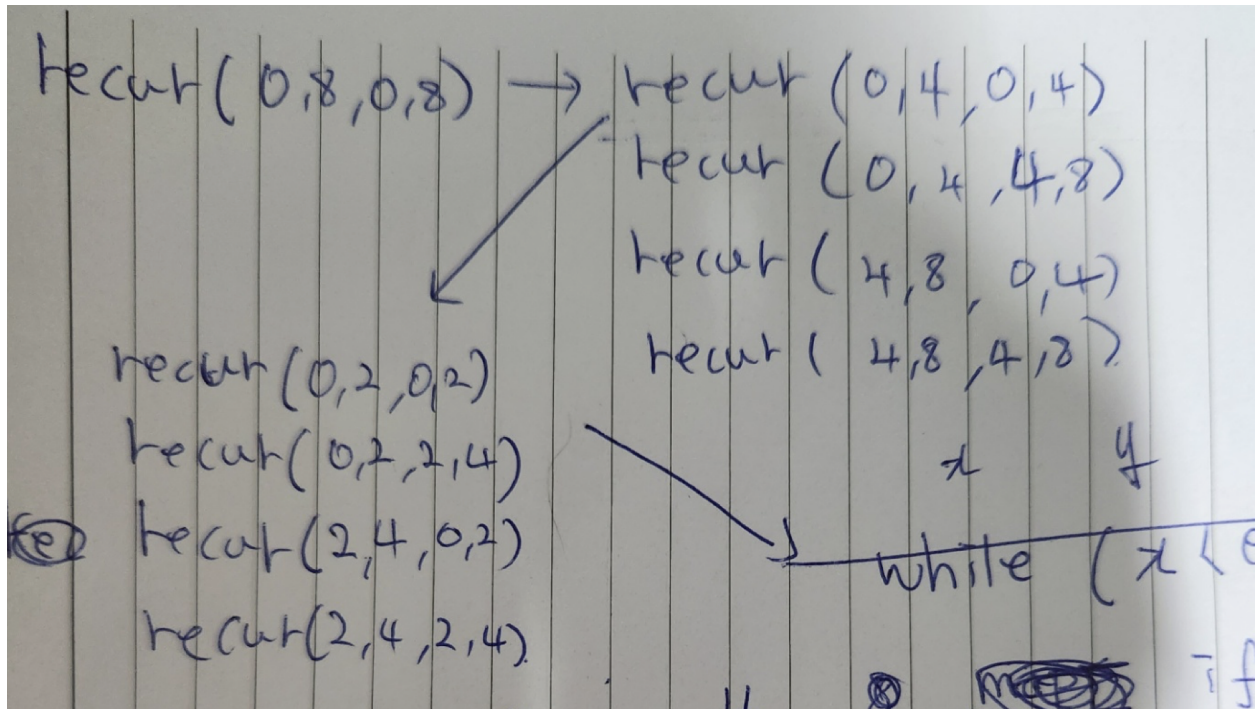


$N > 1$ 인 경우, 배열을 크기가 $2^{N-1} \times 2^{N-1}$ 로 4등분 한 후에 재귀적으로 순서대로 방문한다.

다음 예는 $2^2 \times 2^2$ 크기의 배열을 방문한 순서이다.



N 이 주어졌을 때, r 행 c 열을 몇 번째로 방문하는지 출력하는 프로그램을 작성하시오.



```

public class _1074_Z {

    private static int R;
    private static int C;
    private static int count;
    private static boolean done = false;

    @Test
    public static void main(String[] args) throws Exception {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in));
        String[] input = r.readLine().split(" ");
        int n = Integer.parseInt(input[0]);
        R = Integer.parseInt(input[1]);
        C = Integer.parseInt(input[2]);
        count = 0;

        int N = (int) Math.pow(2, n);
        recursion(0, N, 0, N);
    }

    private static void recursion(int startR, int endR, int startC, int endC) {
        if (done) return;

        if (endR - startR == 2 && endC - startC == 2) {
            check(startR, startC);
            check(startR, startC + 1);
        }
    }
}


```

```

        check(startR + 1, startC);
        check(startR + 1, startC + 1);
    } else {
        int midR = startR + (endR - startR) / 2;
        int midC = startC + (endC - startC) / 2;
        recursion(startR, midR, startC, midC);
        recursion(startR, midR, midC, endC);
        recursion(midR, endR, startC, midC);
        recursion(midR, endR, midC, endC);
    }
}

private static void check(int startR, int startC) {
    if (startR == R && startC == C) {
        System.out.println(count);
        done = true;
    } else {
        count++;
    }
}
}

```

문제	결과	메모리	시간
 1074	시간 초과		

```

public class _1074_Z {

    private static int R;
    private static int C;
    private static int count;
    private static boolean done = false;

    @Test
    public static void main(String[] args) throws Exception {
        BufferedReader r = new BufferedReader(new InputStreamReader(System.in));
        String[] input = r.readLine().split(" ");
        int n = Integer.parseInt(input[0]);
        R = Integer.parseInt(input[1]);
    }
}

```

```

    C = Integer.parseInt(input[2]);
    count = 0;


    int N = (int) Math.pow(2, n);
    recursion(0, N, 0, N);
}

private static void recursion(int startR, int endR, int startC, int endC) {
    if (done) return;
    if (endR < R || endC < C) { <---- 여기 추가
        count += (endR - startR) * (endC - startC);
        return;
    }

    if (endR - startR == 2 && endC - startC == 2) {
        check(startR, startC);
        check(startR, startC + 1);
        check(startR + 1, startC);
        check(startR + 1, startC + 1);
    } else {
        int midR = startR + (endR - startR) / 2;
        int midC = startC + (endC - startC) / 2;
        recursion(startR, midR, startC, midC);
        recursion(startR, midR, midC, endC);
        recursion(midR, endR, startC, midC);
        recursion(midR, endR, midC, endC);
    }
}

private static void check(int startR, int startC) {
    if (startR == R && startC == C) {
        System.out.println(count);
        done = true;
    } else {
        count++;
    }
}
}

```

 1074	맞았습니다!!	14248 KB	124 ms	Java 11 / 수정
--	---------	----------	--------	--------------