Baekjoon 16935 행렬 돌리기 3

날짜	@2021년 8월 11일
링크	
열	
태그	Baekjoon Silver2 matrix

@2021년 8월 11일 오늘은 더 이상 머리가 안 굴러간다... 정말 최대한 쥐어 짬.... 간만에 12전에 자네...

```
import java.io.BufferedReader;
import java.io.FileInputStream;
```

```
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Stack;
import java.util.StringTokenizer;

//public class Main {
public class Baek16935_1 {
  static int N, M, R;
  static int[][] map;

  public static void main(String[] args) throws IOException {
//    System.setIn(new FileInputStream("C:/CodingStudy/Baekjoon/Silver2/16935_input.tx
t"));
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    StringTokenizer st = new StringTokenizer(br.readLine());
    N = Integer.parseInt(st.nextToken());
    M = Integer.parseInt(st.nextToken());
    R = Integer.parseInt(st.nextToken());

    map = new int[N][M];
    for (int n = 0; n < N; n++) {
      st = new StringTokenizer(br.readLine());
      for (int m = 0; m < M; m++) {
        map[n][m] = Integer.parseInt(st.nextToken());
      }
    }
    st = new StringTokenizer(br.readLine());
    for (int r = 0; r < R; r++) {
//      System.out.println("전 => N : " + map.length + ", M : " + map[0].length);
      switch (st.nextToken()) {
      case "1":
        filpVertical();
        break;
      case "2":
        filpHorizontal();
        break;
      case "3":
        rotationClock();
        break;
      case "4":
        rotationCounterClock();
        break;
      case "5":
        switchClock();
        break;
      case "6":
        switchCounterClock();
        break;
      }
//      System.out.println("후 => N : " + map.length + ", M : " + map[0].length);
//      System.out.println();
    }
    showMap();
    br.close();
  }

  static void showMap() { // 행렬 보기
```

```
    int NN = map.length;
    int MM = map[0].length;
    for (int n = 0; n < NN; n++) {
      for (int m = 0; m < MM; m++) {
        System.out.print(map[n][m] + " ");
      }
      System.out.println();
    }
  }

  static void filpVertical() { // 1번 Stack을 활용하니까 쉽다.
    int NN = map.length;
    int MM = map[0].length;
    Stack<Integer> stack = new Stack<>();
    for (int m = 0; m < MM; m++) {
      for (int n = 0; n < NN; n++) {
        stack.push(map[n][m]);
      }
    }
    for (int m = MM - 1; m >= 0; m--) {
      for (int n = 0; n < NN; n++) {
        map[n][m] = stack.pop();
      }
    }

  }

  static void filpHorizontal() { // 2번 Stack을 활용하니까 쉽다.
    int NN = map.length;
    int MM = map[0].length;
    Stack<Integer> stack = new Stack<>();
    for (int n = 0; n < NN; n++) {
      for (int m = 0; m < MM; m++) {
        stack.push(map[n][m]);
      }
    }
    for (int n = NN - 1; n >= 0; n--) {
      for (int m = 0; m < MM; m++) {
        map[n][m] = stack.pop();
      }
    }

  }

  static void rotationClock() { // 3번
    int NN = map.length;
    int MM = map[0].length;
    int[][] tmp = new int[MM][NN];
    for (int n = 0; n < NN; n++) { // 주대각선 뒤집기
      for (int m = 0; m < MM; m++) {
        tmp[m][n] = map[n][m];
      }
    }
    map = tmp;

    filpHorizontal();

  }
```

```
static void rotationCounterClock() { // 4번
  int NN = map.length;
  int MM = map[0].length;
  int[][] tmp = new int[MM][NN];
  for (int n = 0; n < NN; n++) { // 주대각선 뒤집기
    for (int m = 0; m < MM; m++) {
      tmp[m][n] = map[n][m];
    }
  }
  map = tmp;

  filpVertical();
}

static void switchClock() { // 5번
  int NN = map.length;
  int MM = map[0].length;
  int[][] tmp1 = new int[NN / 2][MM / 2];
  int[][] tmp2 = new int[NN / 2][MM / 2];
  int[][] tmp3 = new int[NN / 2][MM / 2];
  int[][] tmp4 = new int[NN / 2][MM / 2];
  // 분리하기 시작
  for (int n = 0; n < NN / 2; n++) { // 1번 분리
    for (int m = 0; m < MM / 2; m++) {
      tmp1[n][m] = map[n][m];
    }
  }
  for (int n = 0; n < NN / 2; n++) { // 2번 분리
    for (int m = 0; m < MM / 2; m++) {
      tmp2[n][m] = map[n][m + MM / 2];
    }
  }
  for (int n = 0; n < NN / 2; n++) { // 3번 분리
    for (int m = 0; m < MM / 2; m++) {
      tmp3[n][m] = map[n + NN / 2][m + MM / 2];
    }
  }
  for (int n = 0; n < NN / 2; n++) { // 4번 분리
    for (int m = 0; m < MM / 2; m++) {
      tmp4[n][m] = map[n + NN / 2][m];
    }
  }
  // 집어 넣기 시작
  for (int n = 0; n < NN / 2; n++) { // 4 -> 1
    for (int m = 0; m < MM / 2; m++) {
      map[n][m] = tmp4[n][m];
    }
  }
  for (int n = 0; n < NN / 2; n++) { // 1 -> 2
    for (int m = 0; m < MM / 2; m++) {
      map[n][m + MM / 2] = tmp1[n][m];
    }
  }
  for (int n = 0; n < NN / 2; n++) { // 2 -> 3
    for (int m = 0; m < MM / 2; m++) {
      map[n + NN / 2][m + MM / 2] = tmp2[n][m];
    }
```

```
    }
    for (int n = 0; n < NN / 2; n++) { // 3 -> 4
      for (int m = 0; m < MM / 2; m++) {
        map[n + NN / 2][m] = tmp3[n][m];
      }
    }
  }

  static void switchCounterClock() { // 6번
    int NN = map.length;
    int MM = map[0].length;
    int[][] tmp1 = new int[NN / 2][MM / 2];
    int[][] tmp2 = new int[NN / 2][MM / 2];
    int[][] tmp3 = new int[NN / 2][MM / 2];
    int[][] tmp4 = new int[NN / 2][MM / 2];
    // 분리하기 시작
    for (int n = 0; n < NN / 2; n++) { // 1번 분리
      for (int m = 0; m < MM / 2; m++) {
        tmp1[n][m] = map[n][m];
      }
    }
    for (int n = 0; n < NN / 2; n++) { // 2번 분리
      for (int m = 0; m < MM / 2; m++) {
        tmp2[n][m] = map[n][m + MM / 2];
      }
    }
    for (int n = 0; n < NN / 2; n++) { // 3번 분리
      for (int m = 0; m < MM / 2; m++) {
        tmp3[n][m] = map[n + NN / 2][m + MM / 2];
      }
    }
    for (int n = 0; n < NN / 2; n++) { // 4번 분리
      for (int m = 0; m < MM / 2; m++) {
        tmp4[n][m] = map[n + NN / 2][m];
      }
    }
    // 집어 넣기 시작
    for (int n = 0; n < NN / 2; n++) { // 2 -> 1
      for (int m = 0; m < MM / 2; m++) {
        map[n][m] = tmp2[n][m];
      }
    }
    for (int n = 0; n < NN / 2; n++) { // 3 -> 2
      for (int m = 0; m < MM / 2; m++) {
        map[n][m + MM / 2] = tmp3[n][m];
      }
    }
    for (int n = 0; n < NN / 2; n++) { // 4 -> 3
      for (int m = 0; m < MM / 2; m++) {
        map[n + NN / 2][m + MM / 2] = tmp4[n][m];
      }
    }
    for (int n = 0; n < NN / 2; n++) { // 1 -> 4
      for (int m = 0; m < MM / 2; m++) {
        map[n + NN / 2][m] = tmp1[n][m];
      }
    }
```

```
    }
  }
```