2023.01.18 스터디 발표

❄️ 내가 푼 문제 1

- 백준 27210. 신을 모시는 사당

❄️ 방법

```
public class Main {
  public static void main(String[] args) throws IOException {

    /* 시간 제한이 있기 때문에 BufferedReader 사용 */
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    int n = Integer.parseInt(br.readLine());
    int sum = 0;
    int sub = 0;

    int max = 0;
    int min = 0;

    /* 2를 -1로 변환시켜 계산 -> 어차피 숫자는 1과 2가 전부이기 때문 */
    String[] arr = br.readLine().replaceAll("2", "-1").split(" ");

    for (int i = 0; i < n; i++) {
      sum = Math.max(sum, 0) + Integer.parseInt(arr[i]);
      sub = Math.min(sub, 0) + Integer.parseInt(arr[i]);

      max = Math.max(sum, max);
      min = Math.min(sub, min);
    }

    /* 마이너스 값이 더 클 수도 있기 때문에 절댓값으로 확인하여 출력 */
    max = Math.max(max, Math.abs(min));

    System.out.println(max);
  }
}
```

❄️ 내가 푼 문제 2

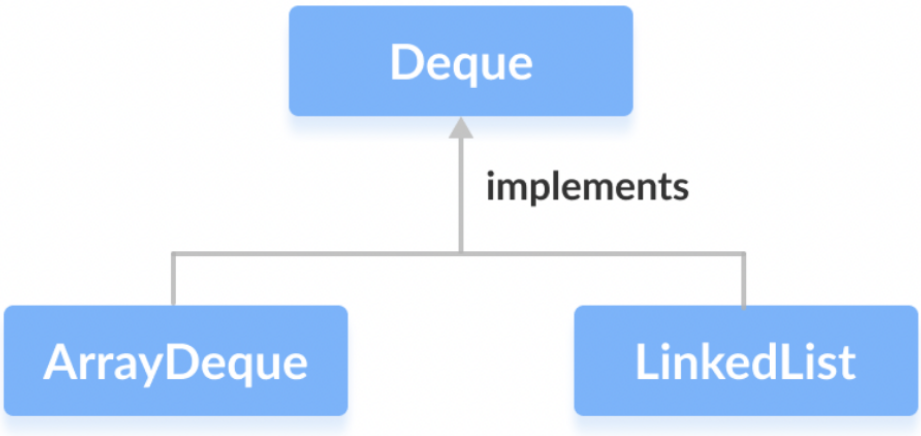
- 백준 27211. 도넛 행성

❄️ 방법

```
public class Main {
  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

  private static final int[] DR = {0,0,-1,1};
  private static final int[] DC = {-1,1,0,0};

  public static void main(String[] args) throws Exception {
    StringTokenizer st = new StringTokenizer(br.readLine());
    int row = Integer.parseInt(st.nextToken());
    int column = Integer.parseInt(st.nextToken());

    boolean[][] arr = new boolean[row][column];

    for (int i = 0; i < row; i++) {
      st = new StringTokenizer(br.readLine());
      for (int j = 0; j < column; j++) {
        if (Integer.parseInt(st.nextToken()) == 0) {
          arr[i][j] = true;
        }
      }
    }

    int answer = 0;

    for (int i = 0; i < row; i++) {
      for (int j = 0; j < column; j++) {
        if (!arr[i][j]) continue;
        answer++;
        Queue<int[]> q = new ArrayDeque<>();
        q.add(new int[]{i,j});
        arr[i][j] = false;

        while (!q.isEmpty()) {
          int cr = q.peek()[0];
          int cc = q.poll()[1];
          for (int d = 0; d < 4; d++) {
            int nr = cr + DR[d];
            int nc = cc + DC[d];
            if (nr < 0) nr += row;
            if (nc < 0) nc += column;
            nr %= row;
            nc %= column;

            if (!arr[nr][nc]) continue;
            arr[nr][nc] = false;
            q.add(new int[]{nr,nc});
          }
        }
      }
    }

    System.out.println(answer);
  }
}
```

❄️ 공부한 것

🔍 `ArrayDeque`, `LinkedList` 의 차이



🔍 `ArrayDeque` VS `LinkedList`

- `ArrayDeque`는 `Array`에 의해 지원, `Array`는 `LinkList`보다 cache-locality 친화적
  - `LinkedList`는 다음 노드가 있는 곳으로 가려고 다른 간접적인 경로를 거쳐감
- `ArrayDeque`는 다음 `Node`에 대한 추가 참조를 유지할 필요가 없기에 `LinkedList`보다 메모리 효율적
- 따라서 `ArrayDeque`를 `Queue`로 사용할 때 `LinkedList` 대신에 사용

❄️ 어떤 것을 써야하나?

- 인덱스로 데이터에 접근하고 끝에 삽입, 삭제만 할 경우에는 `ArrayList`를 사용
- `Stack`, `Queue`, 혹은 `Deque`로 `ArrayDeque`를 사용하라
- 리스트를 순회할때 삽입, 삭제하거나 O(1)인 최악의 경우에 마지막에 삽입 시 `LinkedList`를 사용