풍선 터뜨리기 성공

☆

실버 III

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	4 MB	7723	2975	2303	40.732%

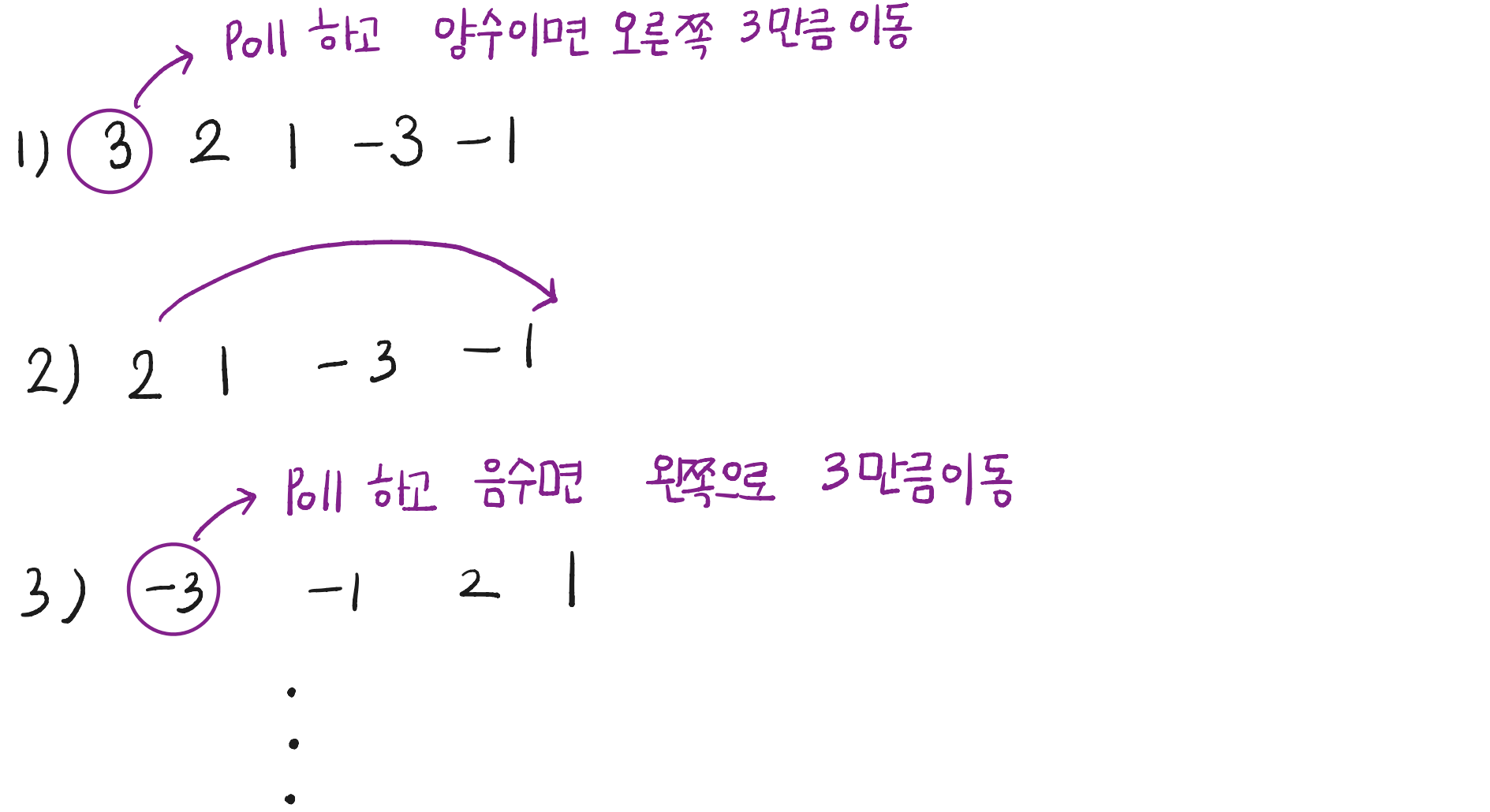
문제

1번부터 N번까지 N개의 풍선이 원형으로 놓여 있고. i번 풍선의 오른쪽에는 i+1번 풍선이 있고, 왼쪽에는 i-1번 풍선이 있다. 단, 1번 풍선의 왼쪽에 N번 풍선이 있고, N번 풍선의 오른쪽에 1번 풍선이 있다. 각 풍선 안에는 종이가 하나 들어있고, 종이에는 -N보다 크거나 같고, N보다 작거나 같은 정수가 하나 적혀있다. 이 풍선들을 다음과 같은 규칙으로 터뜨린다.

우선, 제일 처음에는 1번 풍선을 터뜨린다. 다음에는 풍선 안에 있는 종이를 꺼내어 그 종이에 적혀있는 값만큼 이동하여 다음 풍선을 터뜨린다. 양수가 적혀 있을 경우에는 오른쪽으로, 음수가 적혀 있을 때는 왼쪽으로 이동한다. 이동할 때에는 이미 터진 풍선은 빼고 이동한다.

예를 들어 다섯 개의 풍선 안에 차례로 3, 2, 1, -3, -1이 적혀 있었다고 하자. 이 경우 3이 적혀 있는 1번 풍선, -3이 적혀 있는 4번 풍선, -1이 적혀 있는 5번 풍선, 1이 적혀 있는 3번 풍선, 2가 적혀 있는 2번 풍선의 순서대로 터지게 된다.

〈문제 설명〉



출력 → 1 4 5 3 2 (터지는 순서에 따라 풍선 자리 값 출력)

```java
Scanner sc = new Scanner(System.in);
Deque<Integer> de = new LinkedList<>();
Deque<Integer> de2 = new LinkedList<>();
StringBuilder sb = new StringBuilder();
int N = sc.nextInt();

for (int i = 1; i < N + 1; i++) {
    de.add(sc.nextInt());
    de2.add(i);
}
int num = de.pollFirst();
sb.append(de2.pollFirst());
sb.append(" ");
```

```
while (!de.isEmpty()) {

    if (num > 0) {
        for (int i = 0; i < num - 1; i++) {
            de.addLast(de.pollFirst());
            de2.addLast(de2.pollFirst());
        }
        num = de.pollFirst();
        sb.append(de2.pollFirst());
        sb.append(" ");
        continue;

    } else if (num < 0) {
        for (int i = 0; i < Math.abs(num) - 1; i++) {
            // System.out.println(Math.abs(num) - 1);
            de.addFirst(de.pollLast());
            de2.addFirst(de2.pollLast());
        }
        num = de.pollLast();
        sb.append(de2.pollLast());
        sb.append(" ");
    }
    continue;
}
System.out.println(sb);
```

양수면

앞에서 뽑고 뒤로 보내기

음수면

뒤에서 뽑고 앞으로 넣기

결과	메모리	시간
맞았습니다!!	17304 KB	248 ms
메모리 초과		

첫 페이지

주요 성능 병목 현상 LinkedList은 deque의 끝까지 밀어 넣을 때마다 구현이 본질적으로 JVM / OS와 관련된 새로운 링크 된 목록 노드를 할당하므로 비용이 많이 든다는 것입니다. 또한 끝에서 터질 때마다 내부 노드 LinkedList가 가비지 수집 대상 이 되어 더 많은 작업을 수행합니다. 또한 링크 된 목록 노드가 여기 저기 할당되므로 CPU 캐시를 사용하면 큰 이점이 없습니다.

그것이 흥미로울 수 있다면, 상환 된 일정한 시간에 요소를 추가 (추가) ArrayList하거나 ArrayDeque실행 한다는 증거가 있습니다. 이것을 참조하십시오 .

```
Scanner sc = new Scanner(System.in);
Deque<Integer> de = new LinkedList<>();
Deque<Integer> de2 = new LinkedList<>();
StringBuilder sb = new StringBuilder();
int N = sc.nextInt();
```

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    Deque<Integer> de = new ArrayDeque<>();
    Deque<Integer> de2 = new ArrayDeque<>();
    StringBuilder sb = new StringBuilder();
    int N = sc.nextInt();
```