**难度: 简单**

给你一个目标数组 target 和一个整数 n。每次迭代，需要从  list = {1,2,3..., n} 中依序读取一个数字。

请使用下述操作来构建目标数组 target ：

Push：从 list 中读取一个新元素， 并将其推入数组中。

Pop：删除数组中的最后一个元素。

如果目标数组构建完成，就停止读取更多元素。

题目数据保证目标数组严格递增，并且只包含 1 到 n 之间的数字。

请返回构建目标数组所用的操作序列。

题目数据保证答案是唯一的。

示例 1：

输入：target = [1,3], n = 3

输出：["Push","Push","Pop","Push"]

解释：

读取 1 并自动推入数组 -> [1]

读取 2 并自动推入数组，然后删除它 -> [1]

读取 3 并自动推入数组 -> [1,3]

示例 2：

输入：target = [1,2,3], n = 3

输出：["Push","Push","Push"]

示例 3：

输入：target = [1,2], n = 4

输出：["Push","Push"]

解释：只需要读取前 2 个数字就可以停止。

示例 4：

输入：target = [2,3,4], n = 4

输出：["Push","Pop","Push","Push","Push"]

提示：

1 <= target.length <= 100

1 <= target[i] <= 100

1 <= n <= 100

target 是严格递增的

通过次数10,960提交次数16,933

来源：力扣（LeetCode）

链接：https://leetcode-cn.com/problems/build-an-array-with-stack-operations

著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。

**思路1(使用栈):**

1). 当下标(从0开始)小于数组长度的时候就进行push,并且入栈

2). 当下标(从0开始)小于数组长度 并且 栈顶的值 不等于 数组的值的时候,就出栈

3). 如果2不成立的话就下标+1

4). 最后返回list

|  |
| --- |
| public List<String> buildArray(int[] target, int n) {  Stack<Integer> stack = new Stack<>();  List<String> list = new ArrayList<>();  int index = 0;  for (int i = 1; i <= n; i++) {  if (index < target.length) {  stack.push(i);  list.add("Push");  }  if (index < target.length &&stack.peek() != target[index]) {  stack.pop();  list.add("Pop");  }else {  index++;  }  }  return list;  } |

**思路2(不使用栈,效率相对较高):**

1). 与思路1的步骤基本一样,只是不适用栈,当需要用栈顶进行比较的时候,直接使用i

2). 最后返回list

|  |
| --- |
| public List<String> buildArray2(int[] target, int n) {  List<String> list = new ArrayList<>();  int index = 0;  for (int i = 1; i <= n; i++) {  if (index <= target.length -1) {  list.add("Push");  }  if (index <= target.length -1 && i != target[index]) {  list.add("Pop");  }else {  index++;  }  }  return list;  } |