선형구조 알고리즘 문제해결 탐구_2

학번 : 1403

- 선정 자료구조 : 스택
- 난이도 : 중

문제 : 1874번 스택 수열

문제 작성:

스택 (stack)은 기본적인 자료구조 중 하나로, 컴퓨터 프로그램을 작성할 때 자주 이용되는 개념이다. 스택은 자료를 넣는 (push) 입구와 자료를 뽑는 (pop) 입구가 같아 제일 나중에 들어간 자료가 제일 먼저 나오는 (LIFO, Last in First out) 특성을 가지고 있다.

1부터 n까지의 수를 스택에 넣었다가 뽑아 늘어놓음으로써, 하나의 수열을 만들 수 있다. 이때, 스택에 push하는 순서는 반드시 오름차순을 지키도록 한다고 하자. 임의의 수열이 주어졌을 때 스택을 이용해 그 수열을 만들 수 있는지 없는지, 있다면 어떤 순서로 push와 pop 연산을 수행해야 하는지를 알아낼 수 있다. 이를 계산하는 프로그램을 작성하라.

입력	출력
첫 줄에 n (1 ≤ n ≤ 100,000)이 주어진다. 둘째 줄부터 n개의 줄에는 수열을 이루는 1이상 n이하의 정수가 하나씩 순서대로 주어진다. 물론 같은 정수가 두 번 나오는 일은 없다.	입력된 수열을 만들기 위해 필요한 연산을 한 줄에 한 개씩 출력한다. push연산은 +로, pop 연산은 -로 표현하도록 한다. 불가능한 경우 NO를 출력한다.

입력 예제	출력 예제
8 4 3 6 8 7 5 2 1	+ + + + - - + + - + + - - - -
5 1 2	NO

5 3 4	
4 4 3 2 1	+ + + + - - - -

- 문제 요약
  - 1부터 N까지의 자연수를 오름차순으로 스택에 push하고 pop하는 연산만으로, 주어진 수열을 만들 수 있는지 판단하는 문제이다. 만약 수열을 만들 수 있다면 사용된 +(push)와 -(pop) 연산의 순서를 출력하고, 만들 수 없다면 NO를 출력해야한다.

- 조건 분석
  - N의 범위: N은 1 이상 100,000 이하의 자연수이다.
  - 수열 구성: 입력되는 수열의 모든 수는 1부터 N까지의 자연수이며, 중복 없이 한 번씩만 나타난다.
  - 시간 제한: 2초
  - 메모리 제한: 128MB

- 스택 자료 구조를 선택한 근거
  - 1부터 N까지의 수를 스택에 push하고 pop하는 연산을 통해 수열을 만들어야 하므로, 스택 자료 구조가 문제의 핵심 요구사항과 직접적으로 일치한다. 구현된 ArrayStack을 그대로 활용하기 적합하다.
  - 스택의 후입선출 특성은 오름차순으로 push된 숫자들 중 목표 숫자가 스택의 맨 위에 있는지 확인하고 pop하는 문제의 로직을 효율적으로 시뮬레이션하는데 최적화되어 있다.

- 의사 코드를 이용한 알고리즘 구현(주석을 이용하여 알고리즘을 구체적으로 설명)

```
from ArrayStack import ArrayStack

n = int(input()) # 수열의 길이 N을 입력받음.
stack = ArrayStack(n) # N 크기의 ArrayStack을 생성 (최대 N개의 숫자가 들어갈 수 있음)
result = []       # 연산 결과를 저장할 리스트 (ex: ['+', '-', ...])
current = 1       # 스택에 push할 다음 숫자 (1부터 N까지 순차적으로 증가)
possible = True   # 수열을 만들 수 있는지 여부를 나타내는 플래그

# N개의 목표 숫자들을 하나씩 처리함.
for _ in range(n):
    target = int(input()) # 현재 만들어야 할 수열의 숫자를 입력받음.

    # 1. current 숫자가 target보다 작거나 같으면, target을 만날 때까지 스택에 push
```

```
    while current <= target:
        stack.push(current)     # 스택에 current 숫자를 넣고
        result.append('+')      # 연산 결과에 '+'를 추가
        current += 1            # 다음 push할 숫자를 1 증가

    # 2. 스택의 top에 있는 숫자가 target과 같은지 확인
    #   (스택이 비어있지 않아야 peek() 호출 시 오류가 발생하지 않음.)
    if not stack.isEmpty() and stack.peek() == target:
        stack.pop()             # 스택의 top이 target과 같으면 pop하고
        result.append('-')      # 연산 결과에 '-'를 추가
    else:
        # 스택의 top이 target과 다르면, 주어진 수열을 만들 수 없음
        # (예: 스택에 4가 있는데 목표는 3인 경우, 4를 pop하지 않고는 3을 만들 수 없음)
        possible = False        # 불가능 플래그를 설정하고
        break                   # 반복문을 종료

# 3. 모든 목표 숫자를 처리한 후, 결과 출력
if possible:
    # 수열을 만들 수 있었다면, 저장된 연산들을 하나씩 출력
    for op in result:
        print(op)
else:
    # 수열을 만들 수 없었다면 'NO'를 출력
    print('NO')
```

- 시간복잡도 : **O(N)**
  - 1부터 N까지의 모든 숫자가 스택에 한 번씩 push되고, 각 target 숫자에 대해 스택에서 pop 연산이 한 번씩 일어난다. 즉, 각 숫자는 최대 한 번 push되고 최대 한 번 pop되므로 총 연산 횟수는 N에 비례한다. ArrayStack의 push, pop, peek, isEmpty 연산은 모두 O(1) 시간을 가진다. 따라서 전체 시간 복잡도는 O(N)이 된다.