2493

접근법

> - 전체 프로그램의 시간을 생각 할 때, 건물의 높이는 크게 상관이 없지만 건물의 갯수는 상관이 있다. 만약 간단하게 생각 할 수 있는 맨 뒤의 건물부터 순차적으로 한 칸씩 앞으로 오면서 만족하는 건물을 찾는다면 n^2 만큼의 시간이 걸리고 백프로 시간 초과가 뜰 것이다!
> - Fair photography때 사용한 알고리즘과 비슷하게 접근을 해보면, 한번에 값들을 훑으면서 그 값에 해당하는 결과를 찾아서 저장하면 더 빠르게 끝낼 수 있다.
> - 문제의 성질 상 1번 건물의 높이가 100이고 2번 건물의 높이가 150이 되면 3번 부터는 어떤 값이 나오더라도 그 결과가 절대 1번이 될 수 없다!
> - 즉 스택을 만들어서 건물의 높이들을 저장하고, 비교하려는 건물의 높이가 스택에 저장된 건물의 높이보다 높다면 그 보다 낮은 건물들을 차례대로 제거 시키면 간단하다! 히득

In [ ]:

```
input()
buildings = list(map(int,input().split()))
stack =[]
result = []
for res, building in enumerate(buildings):
    if not stack:
        stack.append((res+1,building))
        result.append(0)
        continue
    while stack and stack[-1][1] < building:
        stack.pop()
    if not stack:
        result.append(0)
    else:
        result.append(stack[-1][0])
    stack.append((res+1,building))

print(' '.join(str(res) for res in result))
```

9012

접근법

> - 이건 위 문제보다 더 쉽다. 간단하다 (이게 오면 +1을 시키고)이게 오면 -1을 시키면 된다

-1을 시키면 된다.
- 근데 여기서 만약)이게 더 많은 경우 즉 총 값이 -1이 되는 순간 안된다는 것을 알기 때문에 그냥 for 문 빠져나가고 NO 출력한다
- 만약 끝까지 양수인데 (((와 같이 양수일 경우도 안된다 총 값이 0이여야 된다.

In [ ]:

```
for _ in range(int(input())):
    inputs = input()
    result = 0
    for par in inputs:
        if par == '(':
            result+=1
        else:
            result -=1
        if result <0:
            break

    if result==0:
        print("YES")
    else:
        print("NO")
```

1874

접근법

- 특정 값을 찾으려 할 때, 현재 스택에 마지막에 있는 값과 비교하여 +할지 -할지를 정한다는 기본 개념에서 시작한다
- 만약 찾고자하는 값이 현재 스택 마지막 값보다 크면 값을 증가시켜야 한다(이 때 증가를 위해 전역 변수 하나 필요함)
- 증가 시키던 중 찾고자 하는 값이 발견되면 증가되는 것을 중지하고, 찾고자 하는 값을 pop해준다
- 증가 시키던 중 찾고자 하는 값보다 스택 끝의 값이 더 커지면 찾을 수 없기 때문에 "NO" 출력
- 만약 찾고자하는 값이 현재 스택 마지막 값보다 작으면 값을 감소 시킨다.
- 감소 시키던 중 찾고자 하는 값이 발견되면 감소되는 것을 중지하고, 찾고자 하는 값을 pop해준다
- 감소 시키던 중 찾고자 하는 값보다 스택 끝의 값이 더 커지거나 스택이 비어버리면, 값을 찾을 수 없다는 것이니깐 "NO"출력

이 외 조심해야 하는 사항:

- 나는 먼저 감소를 확인하고, 증가를 확인하고, 같은 경우를 확인하는 방

식으로 하였다. 그래서 만약 감소되다가 발견 되면 감소 반복문을 빠져나가 증가 반복문의 조건을 확인하고 같은 경우 조건에 걸리게 되어 거기서 값이 POP 되는 형식이다. 증가도 마찬가지이다.

- 이때, 감소가 되다가 값이 없다는 것을 발견하였는데, 그 경우는 stack이 비거나, stack의 마지막 값이 목표 값보다 작아지는 경운데, 후자의 경우 두번째 증가를 확인하는 부분에 조건이 만족하여 들어간다. stack이 비면 stack[-1]에서 index 에러가 생긴다. 이를 방지하기 위해 yorn라는 카운터 변수를 두었다.

In [ ]:

```
stack = []
result = []
yorn = True
i = 1
for _ in range(int(input())):
    if not yorn:
        break
    target = int(input())

    if not stack:
        stack.append(i)
        i+=1
        result.append("+")

    if stack[-1] > target:
        while stack and stack[-1] != target:
            result.append("-")
            if stack.pop() < target or not stack:
                yorn = False
                break


    if yorn and stack[-1] < target:
        while stack[-1] != target:
            result.append("+")
            stack.append(i)
            i+=1
            if stack[-1] > target:
                yorn = False
                break


    if stack and stack[-1] == target:
        result.append("-")
        stack.pop()


if yorn:
    print('\n'.join(result))
```

```
    print('\n'.join(result))
else:
    print("NO")
```