

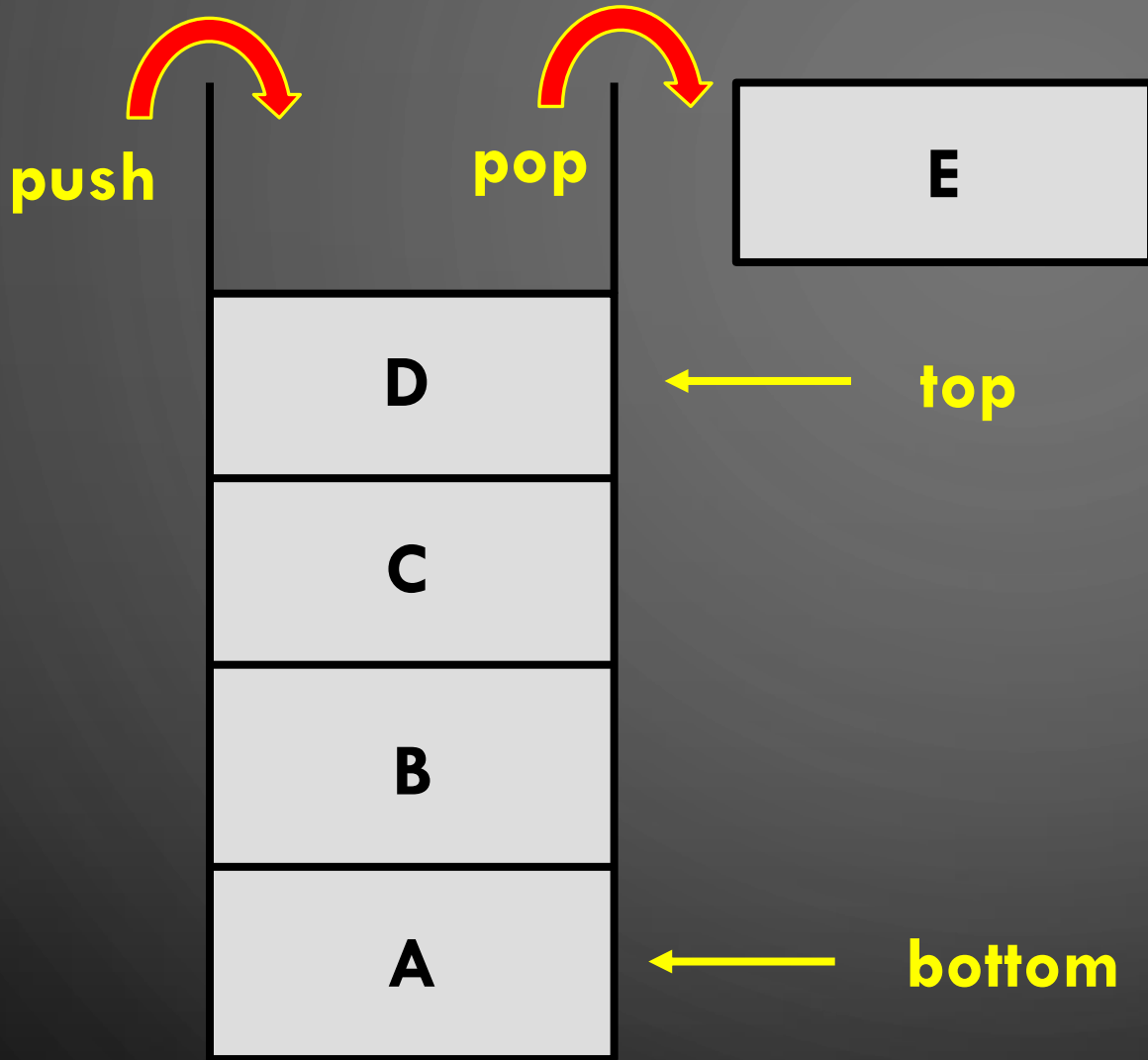
자료구조와 알고리즘

5강 - 스택과 큐

LECTURED BY SOONGU HONG



1. 스택(STACK) 자료구조



- 스택은 더미라는 뜻을 가지는 데, 더미의 예시는 책더미, 신문더미 등이 있습니다.
- 책 더미에서 책을 꺼내는 정상적인 방법은 맨 위에 책을 가져오는 것이며 추가하는 방법은 맨 위에 올려놓는 방법입니다.
- 이처럼 스택은 모든 원소들의 삽입과 제거가 top이라는 자료구조의 한 쪽 끝에서만 수행되는 제한된 리스트 구조를 갖습니다.
- 따라서 스택을 후입 선출 방식인 LIFO(Last-In-First-Out)구조라고 부릅니다.

* 스택의 연산

* 삽입(PUSH)

- 스택에 새로운 자료를 삽입하는 연산을 push 또는 insertion이라고 합니다.
- 이는 항상 스택의 top에서 일어나며 꽉 차 있으면 오버플로우가 발생하여 더 이상 자료를 삽입할 수가 없게 됩니다.

* 제거(POP)

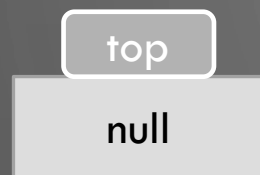
- 스택에서 자료를 제거하는 연산을 pop 또는 deletion이라고 합니다.
- 제거할 자료가 없는 상태를 공백(empty) 상태라고 합니다.

* 읽기(PEEK)

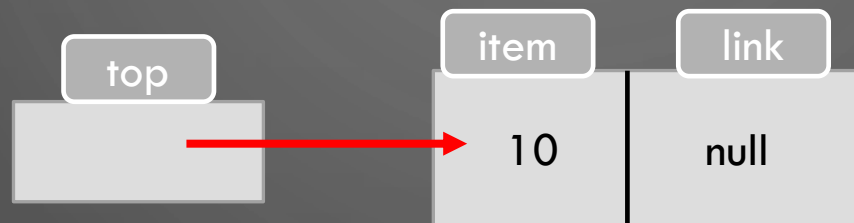
- 스택의 top 위치 자료 값을 단순히 읽어오는 연산을 peek이라고 합니다.
- Pop연산과는 달리 해당 자료를 제거하거나 변경하지 않습니다.

* 삽입 연산 (PUSH)

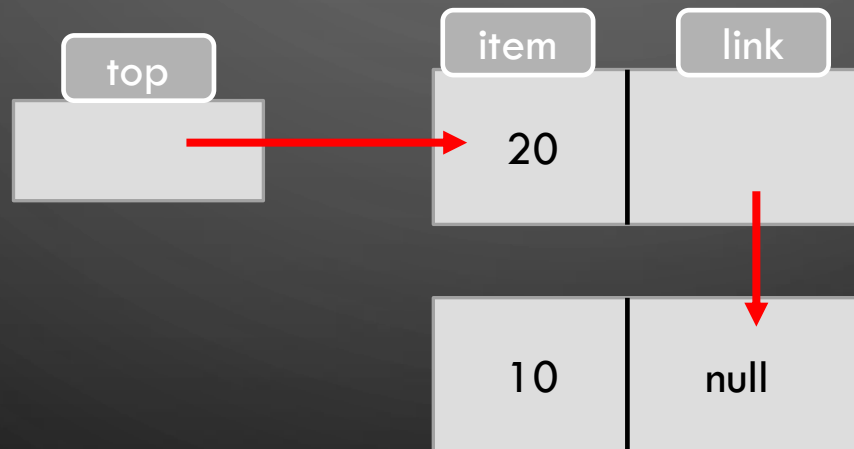
1) 스택 생성



2) 자료 10 삽입

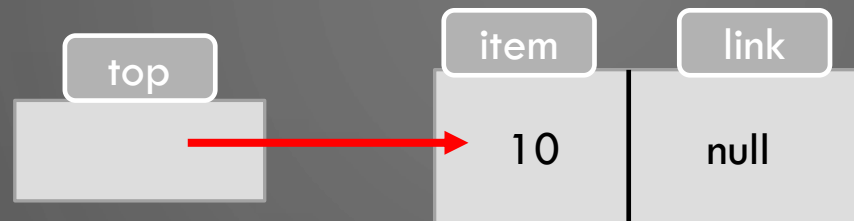


3) 자료 20 삽입



* 삽입 연산 (PUSH)

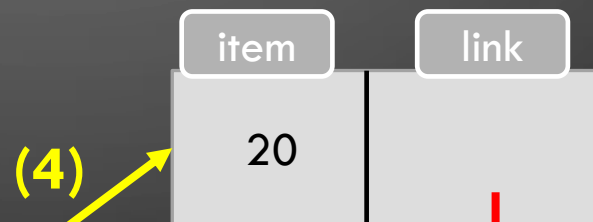
1) 20 삽입 전



2) 새 노드 생성 후 20 삽입



3) 새 노드의 link가
top이 기존에 가리키던
노드를 가리키게 함

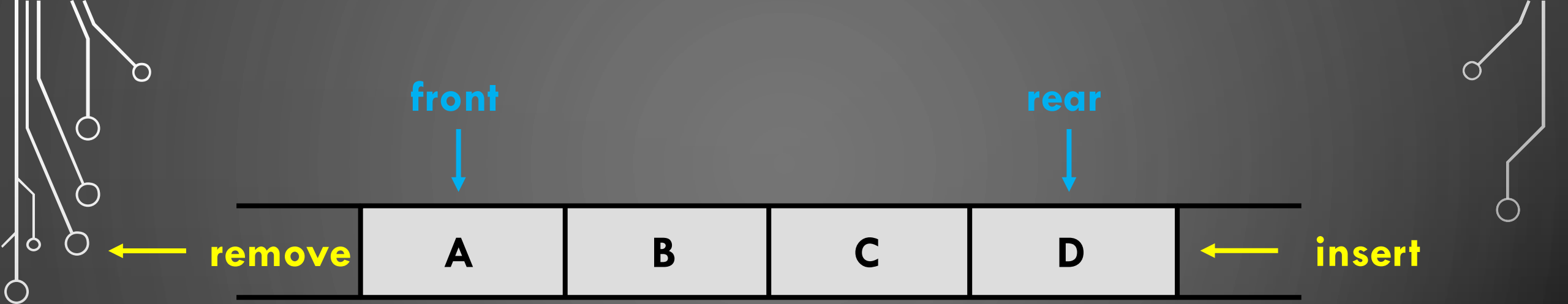


4) Top이 새 노드를
가리키게 함





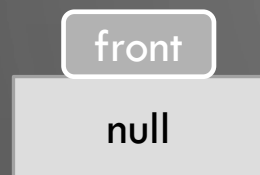
2. 큐(Queue) 자료구조



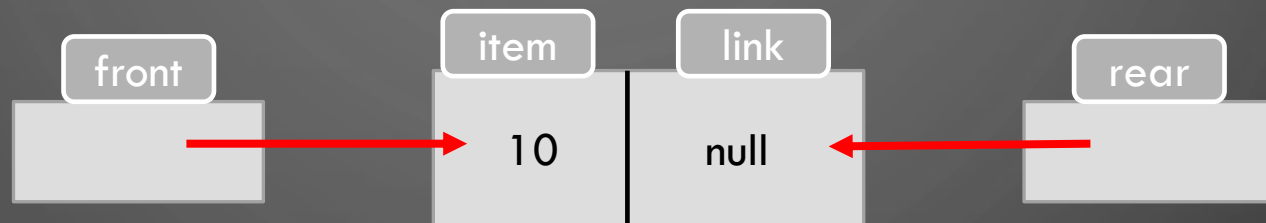
- 큐(Queue)는 줄(line)이라는 의미를 가진 단어로, 쉬운 예시로 택시 승강장을 생각할 수 있습니다.
- 승강장에서 승객은 줄의 가장 뒤에 서게 되며, 택시가 도착하면 가장 앞에 있는 승객부터 택시를 타게 됩니다.
- 이처럼 큐는 원소의 제거는 큐의 가장 앞(front)에서 수행되며, 원소의 삽입은 가장 뒤(rear)에서 수행되는 제한된 리스트의 구조를 갖습니다.
- 따라서 가장 먼저 입력된 자료가 먼저 처리되기 때문에 큐를 FIFO(First-In-First-Out) 구조라고 부릅니다.

* 삽입 연산 (INSERT)

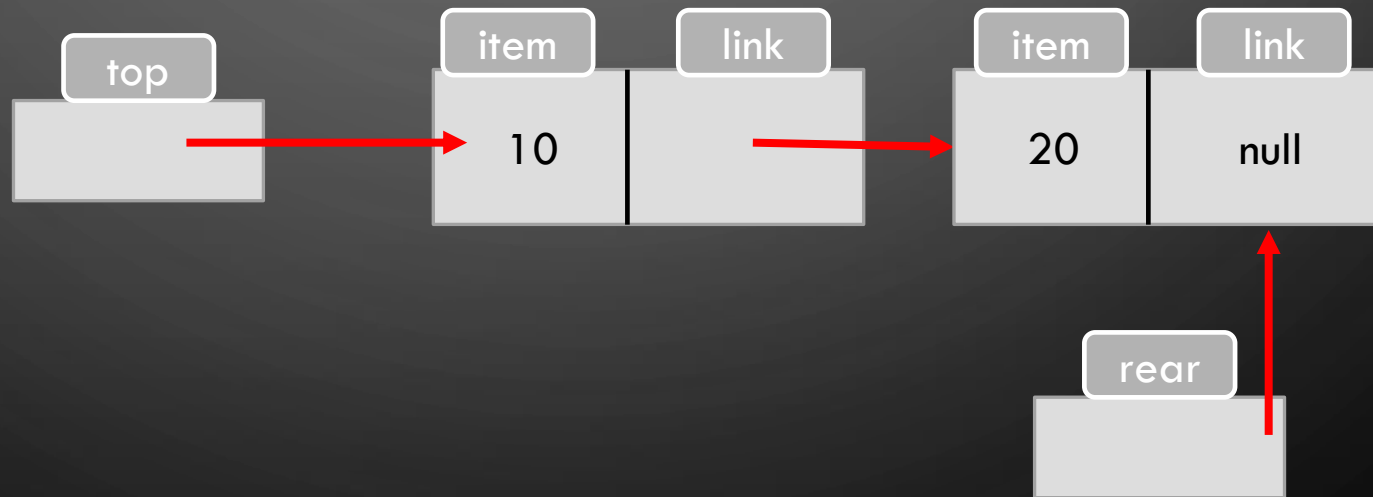
1) 큐 생성



2) 자료 10 삽입



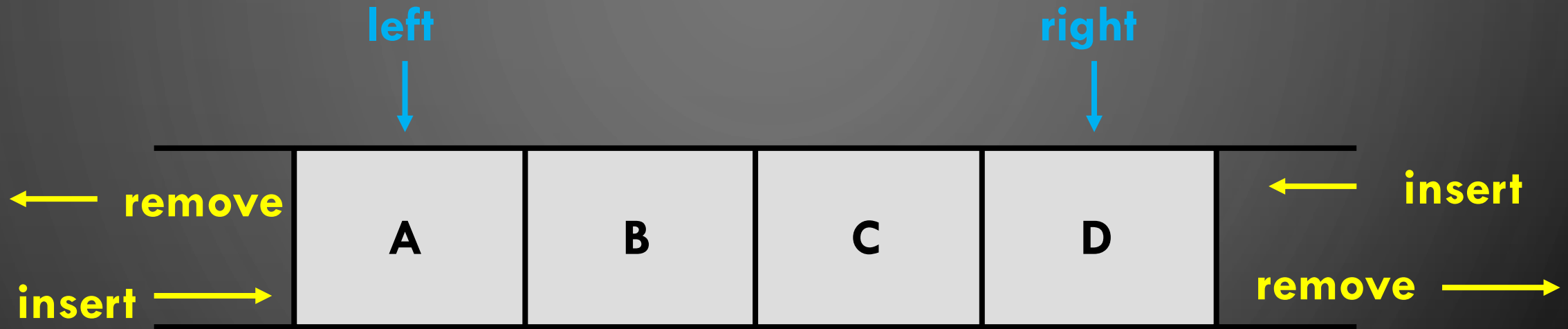
3) 자료 20 삽입



A decorative graphic on the left side of the slide, consisting of white lines and circles on a dark gray background, resembling a circuit board or a stylized tree structure.

3. 기타 유사 자료구조

* 덱(DEQUE : DOUBLE-ENDED QUEUE)



- 덱은 위 그림과 같이 리스트의 양 쪽 끝 모두에서 자료에 대한 삽입과 삭제를 할 수 있는 자료구조입니다.
- 이러한 덱은 삽입과 제거 방향에 제한을 두어 그 동작을 변형시킬 수 있습니다.
- 만약 덱을 스택처럼 동작하도록 하려면 오른쪽에서만 삽입과 제거가 일어나도록 제한하면 되고, 큐처럼 동작하게 하려면 삽입은 왼쪽에서만 삭제는 오른쪽에서만 일어나게 제한하면 됩니다.



* 데크 구현 참고 코드 링크

<https://st-lab.tistory.com/187?category=856997>



* 우선순위 큐(PRIORITY QUEUE)

- 우선순위 큐는 각각의 자료에 부여된 우선순위에 따라 우선순위가 높은 자료부터 제거하는 자료구조입니다.
- 자료에 부여되는 우선순위가 무엇인지에 따라서 자료 삽입 시 우선순위가 높은 자료 순으로 제거될 수 있도록 정렬이 필요하기도 합니다.
- 스택과 큐도 우선순위 큐의 일종이라고 볼 수 있는데, 스택은 스택에 머무른 시간이 가장 짧은 자료에 높은 우선순위를 부여한 것이며, 큐는 큐에 머무른 시간이 가장 오래된 자료에 가장 높은 우선순위를 부여한 것입니다.



* 우선순위 큐 구현 참고 코드 링크

<https://st-lab.tistory.com/219?category=856997>





4. 스택 핵심 예제 풀어보기

(백준 알고리즘 1874번)

<https://www.acmicpc.net/problem/1874>

스택 수열



시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	128 MB	89065	32689	23099	36.199%

문제

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 n 까지의 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


첫 줄에 n ($1 \leq n \leq 100,000$)이 주어진다. 둘째 줄부터 n 개의 줄에는 수열을 이루는 1이상 n 이하의 정수가 하나씩 순서대로 주어진다. 물론 같은 정수가 두 번 나오는 일은 없다.

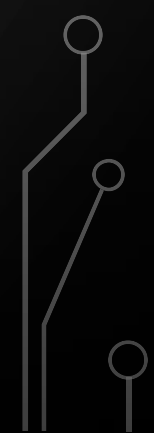
출력

입력된 수열을 만들기 위해 필요한 연산을 한 줄에 한 개씩 출력한다. push연산은 +로, pop 연산은 -로 표현하도록 한다. 불가능한 경우 NO를 출력한다.



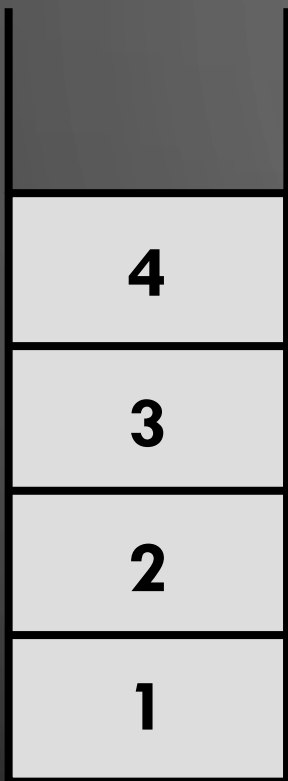
* 문제 분석

- 스택의 원리를 정확하게 알고 있는지를 묻는 문제입니다.
 - 이 문제는 스택의 pop, push 연산과 후입선출 개념을 이해하고 있다면 쉽게 풀 수 있습니다.
- 



* 예제 입출력 분석

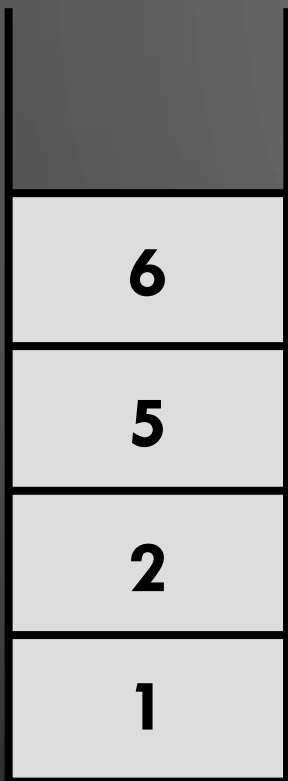
- 이 문제는 1부터 자연수를 증가시키면서 입력으로 주어진 숫자와 비교하여 증가시킨 자연수를 스택에 추가하거나 빼는 방식으로 풀니다.
1. 현재 수열 값 \geq 자연수
 - 현재 수열 값이 자연수보다 크거나 같을 때까지 자연수를 1씩 증가시키며 자연수를 스택에 push한다. 그리고 push가 끝나면 마지막에 pop하여 해당 수를 출력한다.
 2. 현재 수열 값 $<$ 자연수
 - 현재 수열 값보다 자연수가 크다면 pop으로 스택에 있는 값을 꺼낸다. 꺼낸 값이 현재 수열 값이거나 아닐 수 있다. 만약 아니라면 후입선출 원리에 따라 수열을 표현할 수 없으므로 NO를 출력한 후 문제를 종료한다.



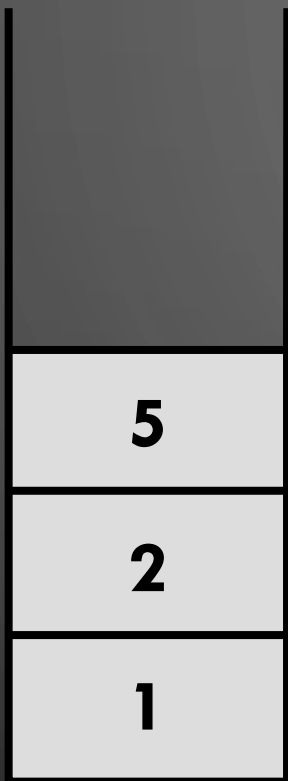
- 첫 수열값이 4이므로 자연수 1부터 4까지 가는 동안 계속 작거나 같으므로 4번 push한다.
- 결과: + + + +




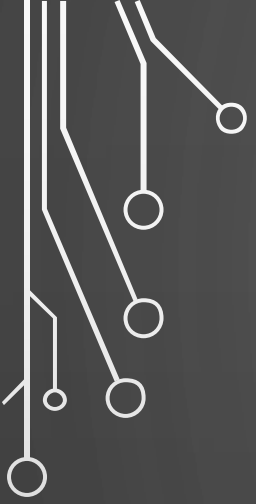
- 다음 자연수는 5이므로 현재 수열값보다 크기 때문에 pop을 수행하여 4를 빼낸다.
- 그 다음 수열값은 3이고 역시 자연수 5보다 작으므로 pop을 수행하여 3도 빼낸다.
- 결과: + + + + - -



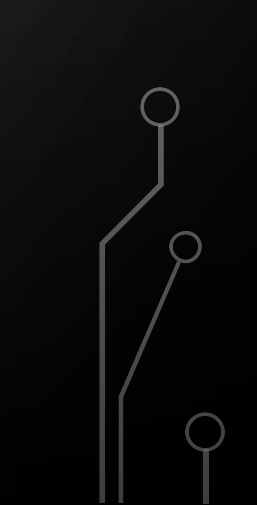

- 다음 수열값은 6이고 현재 자연수는 5이므로 5를 push한다.
- 다음 자연수는 6이고 수열값은 6이므로 같기 때문에 역시 6도 push한다.
- 결과: + + + + - - + +



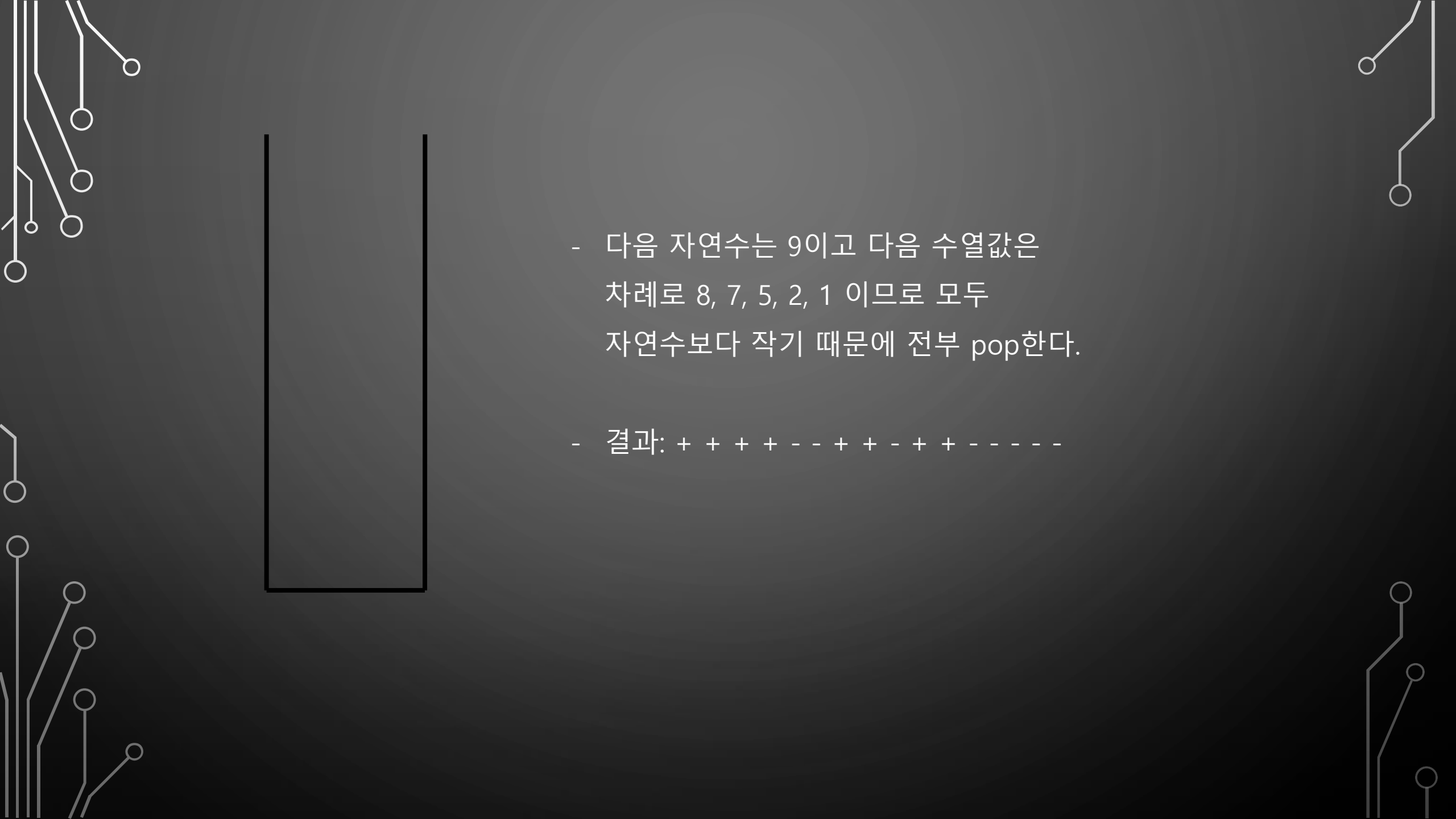

- 다음 자연수는 7이고 수열값은 6이어서
현재 수열값보다 크므로 6을 pop한다.
- 결과: + + + + - - + + -



8
7
5
2
1



- 다음 자연수는 7이고 다음 수열값은 8이므로 7을 push한다.
- 다음 자연수는 8이고 수열값은 8이므로 같기 때문에 역시 8도 push한다.
- 결과: + + + + - - + + - + +

- 
- 
- 다음 자연수는 9이고 다음 수열값은 차례로 8, 7, 5, 2, 1 이므로 모두 자연수보다 작기 때문에 전부 pop한다.
 - 결과: + + + + - - + + - + + - - - -

* 의사코드 작성하기

```
N(수열 개수) 입력받기
A[] (수열 배열 생성)
입력 받아 수열 배열 채우기
Stack 생성하기
결과 문자열 변수 S 선언
for ( N만큼 반복) {
    if (현재 수열 값 >= 현재 자연수) {
        while (현재 수열 값 >= 현재 자연수) {
            Stack에 push
            S에 + 기호 저장
        }
        마지막 저장값 pop
        S에 - 기호 저장
    } else (현재 수열값 < 현재 자연수) {
        스택에서 pop후에 S에 - 기호저장
        if (pop한 숫자 > 현재 수열값) { NO 출력 후 for문 강제종료 }
        else { S에 - 기호 저장 }
    }
    if ( NO를 출력하지 않았다면) { S 출력하기 }
}
```



5. 큐 핵심 예제 풀어보기

(백준 알고리즘 2164번)

<https://www.acmicpc.net/problem/2164>

카드2



시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초 (추가 시간 없음)	128 MB	50279	26025	21498	52.267%

문제

N장의 카드가 있다. 각각의 카드는 차례로 1부터 N까지의 번호가 붙어 있으며, 1번 카드가 제일 위에, N번 카드가 제일 아래인 상태로 순서대로 카드가 놓여 있다.

이제 다음과 같은 동작을 카드가 한 장 남을 때까지 반복하게 된다. 우선, 제일 위에 있는 카드를 바닥에 버린다. 그 다음, 제일 위에 있는 카드를 제일 아래에 있는 카드 밑으로 옮긴다.

예를 들어 N=4인 경우를 생각해 보자. 카드는 제일 위에서부터 1234 의 순서로 놓여있다. 1을 버리면 234가 남는다. 여기서 2를 제일 아래로 옮기면 342가 된다. 3을 버리면 42가 되고, 4를 밑으로 옮기면 24가 된다. 마지막으로 2를 버리고 나면, 남는 카드는 4가 된다.

N이 주어졌을 때, 제일 마지막에 남게 되는 카드를 구하는 프로그램을 작성하시오.

입력



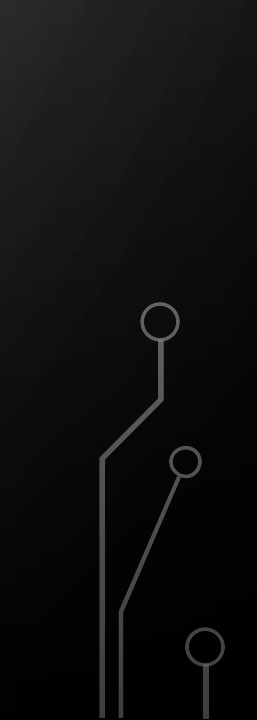
첫째 줄에 정수 N($1 \leq N \leq 500,000$)이 주어진다.

출력

첫째 줄에 남게 되는 카드의 번호를 출력한다.



* 문제 분석

- 큐를 잘 이해하고 있는지를 묻는 문제입니다.
 - 가장 위의 카드를 가장 아래에 있는 카드 밑으로 옮기는 동작은 큐의 선입선출 성질을 이용하면 쉽게 구현할 수 있습니다.
- 
- 
- 

front



rear



	1	2	3	4	
--	---	---	---	---	--

1. 가장 위의 카드 1 버리기 : remove()

	2	3	4	
--	---	---	---	--

2. 가장 위의 카드 2를 맨 뒤에 추가 : remove() -> insert(2)

	3	4	2	
--	---	---	---	--

front



rear



	3	4	2	
--	---	---	---	--

3. 가장 위의 카드 3 버리기 : remove()

	4	2		
--	---	---	--	--

4. 가장 위의 카드 4를 맨 뒤에 추가 : remove() -> insert(4)

	2	4		
--	---	---	--	--

front



rear



2

4

5. 가장 위의 카드 2 버리기 : remove()

4

6. 마지막으로 남은 카드 제거 후 출력: remove()

* 의사코드 작성하기

N(카드의 개수)

Queue 생성

for (카드의 개수만큼 반복) {

 큐에 카드 저장하기

}

while (카드가 1장 남을 때까지) {

 맨 위 카드를 버림 : remove

 맨 위 카드를 가장 아래로 이동 : remove -> insert

}

마지막으로 남은 카드 출력