

25-1 이니로 알고리즘 멘토링

멘토 - 김수성

멘토링 진행 방식

기본 과제 풀이 C++, Python

알고리즘 설명

예제 문제 풀이 및 코드 설명

심화 과제 풀이(필수 X)

과제

기본 과제 5문제(필수)

심화 과제 3문제(필수 X)


다음 멘토링 시간까지 풀어오기

사용하고 싶은 언어 사용

과제 제출

코드 제출 시 소스 코드 공개 맞았을 때 만 공개로 설정

문제 ▾ 문제집 대회 2 채점 현황 랭킹 게시판 그룹 더 보기 ▾ 🔍

그룹 이름	그룹장	멤버
25-1 이니로 알고리즘 멘토링	 kss418	6

메인 문제집 문제집 만들기 채점 현황 연습 연습 만들기 랭킹 게시판 글쓰기 파일 설정 관리

번호	만든 사람	제목	진행도
77987	 kss418	1주차 심화 과제	<div><div></div>2 / 2</div>
77986	 kss418	1주차 기본 과제	<div><div></div>5 / 5</div>

메인 문제집 문제집 만들기 채점 현황 연습 연습 만들기 랭킹 게시판 글쓰기 파일 설정 관리

연습 이름	시작	종료	상태	채점 현황	수정
1주차 과제	2025년 3월 17일 21:00	2025년 3월 24일 20:00	시작까지 3일 08:24:01	채점 현황	수정

주차 별 내용

1주차	3/17	선형 자료구조	6주차	4/28	DFS
2주차	3/24	이분 탐색	7주차	5/5	BFS
3주차	3/31	비선형 자료구조	8주차	5/12	DP
4주차	4/7	누적 합	9주차	5/19	다익스트라
5주차	4/14	백트래킹	10주차	5/26	Union Find

확정은 아님..

풀다가 막힐 때..

디코, 카톡으로 질문하기(과제가 아니어도 괜찮음)

풀이 검색해보고 문제 이해하기

더 좋은 강의 듣고 복습

kks227 블로그

<https://m.blog.naver.com/kks227/220769859177>

바킹독 유튜브 강의

<https://www.youtube.com/watch?v=Lc0lobH7ues&list=PLtqbFd2VIQv406D6l9HcD732hdrnYb6CY>

질문?

1주차 - 선형 자료구조

스택 / 덱 / 큐

선형 자료구조

자료구조에 대한 구현은 다루지 않을 예정

각 언어에서 제공하는 라이브러리를 사용하면 됨

자료구조 사용법 및 시간 복잡도,

문제 풀이에 어떻게 적용하는지에 대해서만 다룸

자세한 내용이 궁금하면 자료구조 수업에서..

선형 자료구조

자료구조?

데이터를 효율적으로 저장하기 위함

효율적?

같은 로직의 코드여도 자료구조, 알고리즘의 차이로
프로그램의 런타임 시간과 메모리 사용량은 다름

선형 자료구조

“선형” 자료구조?

자료구조의 데이터들의 관계가 1:1

Ex) 배열, 스택, 큐, 덱, 연결 리스트

멘토링에서는 스택, 큐, 덱을 다룰 예정

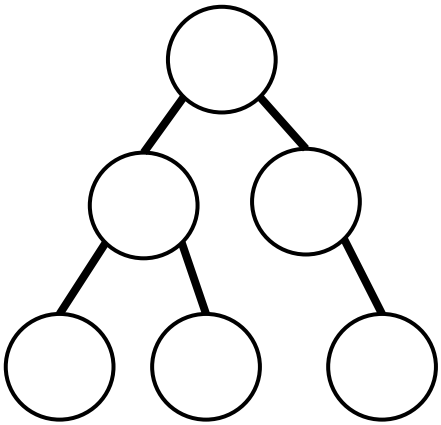
큐, 덱은 잘 안 나오지만 스택은 자주 나옴
스택 과제가 많을 예정

선형 자료구조

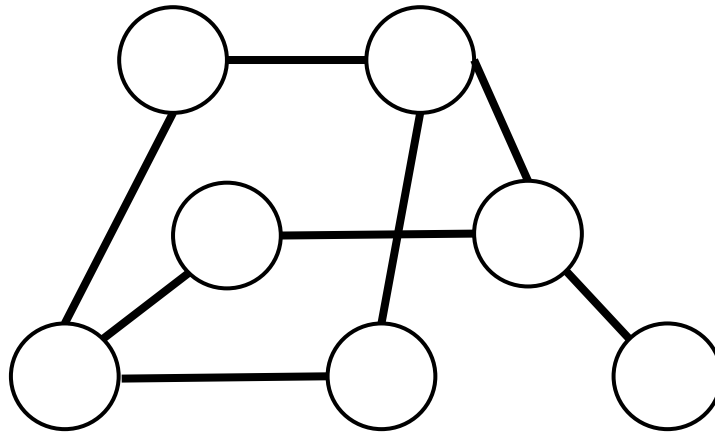
“비선형” 자료구조?

자료구조의 데이터들의 관계가 $n:n$

Ex) 트리, 그래프



트리



그래프

선형 자료구조

“비선형” 자료구조?

자료구조의 데이터들의 관계가 $n:n$

비선형 자료구조는 2주 후에 다룰 예정

스택

스택

마지막에 넣은 데이터가 먼저 나옴

-push(x) : x를 스택에 넣음

-pop() : 스택의 마지막 데이터를 제거함



스택

스택

마지막에 넣은 데이터가 먼저 나옴

-push(x) : x를 스택에 넣음

-pop() : 스택의 마지막 데이터를 제거함

push(2)



2

스택

스택

마지막에 넣은 데이터가 먼저 나옴

-push(x) : x를 스택에 넣음

-pop() : 스택의 마지막 데이터를 제거함

push(5)



스택

스택

마지막에 넣은 데이터가 먼저 나옴

-push(x) : x를 스택에 넣음

-pop() : 스택의 마지막 데이터를 제거함

push(6)



스택

스택

마지막에 넣은 데이터가 먼저 나옴

-push(x) : x를 스택에 넣음

-pop() : 스택의 마지막 데이터를 제거함

pop()



스택

스택

마지막에 넣은 데이터가 먼저 나옴

-push(x) : x를 스택에 넣음

-pop() : 스택의 마지막 데이터를 제거함

pop()



2

스택

스택

마지막에 넣은 데이터가 먼저 나옴

-push(x) : x를 스택에 넣음

-pop() : 스택의 마지막 데이터를 제거함

pop()



스택

C++

`std::stack`

C++ 표준 라이브러리에 있음

헤더 : `<stack>`

함수

`push(x)` : `x`를 스택에 넣음

`pop()` : 스택의 마지막 데이터 제거

`top()` : 스택의 마지막 데이터 반환

`empty()` : 스택이 비어있으면 1 반환 아니면 0 반환

`size()` : 스택의 데이터 개수 반환

스택

C++

함수

push(x) : x를 스택에 넣음

pop() : 스택의 마지막 데이터 제거

top() : 스택의 마지막 데이터 반환

empty() : 스택이 비어있으면 1 아니면 0 반환

size() : 스택의 데이터 개수 반환

시간 복잡도

모두 $O(1)$

스택

C++

선언 할 때는
`stack <T>` 변수명
T는 자료형이 들어가면 됨

Received Output:

6
3
2
1

```
#include <iostream>
#include <stack>
using namespace std;
stack <int> st;

int main(){
    st.push(2); // 2
    st.push(3); // 2 3
    st.push(6); // 2 3 6

    cout << st.top() << "\n"; // 6

    st.pop(); // 2 3

    cout << st.top() << "\n"; // 3

    cout << st.size() << "\n"; // 2

    st.pop(); // 2
    st.pop(); //

    cout << st.empty() << "\n"; // 1
}
```

스택

Python

따로 스택 자료구조가 없음
List를 스택처럼 사용

함수

`append(x)` : `x`를 스택에 넣음
`pop()` : 스택의 마지막 원소 제거 및 반환

`len(stack)` : 스택의 길이 반환
`stack[-1]` : 스택의 마지막 원소 반환

스택

Python

선언 할 때는 빈 리스트로 선언

Received Output:

6
3
2
True

```
stack = []  
stack.append(2) # 2  
stack.append(3) # 2 3  
stack.append(6) # 2 3 6  
  
print(stack[-1]) # 6  
  
stack.pop() # 2 3  
  
print(stack[-1]) # 3  
  
print(len(stack)) # 2  
  
stack.pop() # 2  
stack.pop() #  
  
print(len(stack) == 0) # True
```

스택

백준 10828 / <https://www.acmicpc.net/problem/10828>

문제

정수를 저장하는 스택을 구현한 다음, 입력으로 주어지는 명령을 처리하는 프로그램을 작성하시오.

명령은 총 다섯 가지이다.

- push X: 정수 X를 스택에 넣는 연산이다.
- pop: 스택에서 가장 위에 있는 정수를 빼고, 그 수를 출력한다. 만약 스택에 들어있는 정수가 없는 경우에는 -1을 출력한다.
- size: 스택에 들어있는 정수의 개수를 출력한다.
- empty: 스택이 비어있으면 1, 아니면 0을 출력한다.
- top: 스택의 가장 위에 있는 정수를 출력한다. 만약 스택에 들어있는 정수가 없는 경우에는 -1을 출력한다.

입력

첫째 줄에 주어지는 명령의 수 N ($1 \leq N \leq 10,000$)이 주어진다. 둘째 줄부터 N 개의 줄에는 명령이 하나씩 주어진다. 주어지는 정수는 1보다 크거나 같고, 100,000보다 작거나 같다. 문제에 나와있지 않은 명령이 주어지는 경우는 없다.

출력

출력해야하는 명령이 주어질 때마다, 한 줄에 하나씩 출력한다.

스택 / 10828

단순 스택 구현 문제

push, pop, size, empty, top 연산

pop과 top 연산은 스택에 데이터가 없을 때
호출하면 런타임 에러가 남

이 문제에서는 데이터가 없을 때 -1 출력을 하면 됨

스택 / 10828

C++

```
#include <iostream>
#include <stack>
using namespace std;

stack<int> st;
int main() {
    int n; cin >> n;
    for(int i = 1; i <= n; i++){
        string s; cin >> s; // 명령어 입력
        if(s == "push"){ // push
            int x; cin >> x;
            st.push(x); // 스택에 x 삽입
        }
        else if(s == "pop"){ // pop
            if(st.empty()) cout << -1 << "\n"; // 스택이 비어있으면 -1 출력
            else{
                cout << st.top() << "\n"; // 스택의 마지막 데이터 출력
                st.pop(); // 마지막 데이터 삭제
            }
        }
        else if(s == "size") cout << st.size() << "\n"; // 데이터 개수 출력
        else if(s == "empty") cout << st.empty() << "\n"; // 스택이 비어있으면 0 아니면 1
        else{ // top
            if(st.empty()) cout << -1 << "\n"; // 스택이 비어있으면 -1 출력
            else cout << st.top() << "\n"; // 스택의 마지막 데이터 출력
        }
    }

    return 0;
}
```

스택 / 10828

Python

```
import sys

stack = []
n = int(input())

for _ in range(n):
    com = list(map(str, input().split())) # 명령어 입력
    if com[0] == "push":
        stack.append(int(com[1])) # 스택에 데이터 삽입
    elif com[0] == "pop":
        if len(stack): #스택에 데이터가 있으면
            print(stack.pop()) # pop 연산 및 마지막 데이터 출력
        else:
            print(-1) # -1 출력
    elif com[0] == "size":
        print(len(stack)) # size 출력
    elif com[0] == "empty":
        if(len(stack)): # 스택에 데이터가 있으면
            print(0) # 0 출력
        else:
            print(1) # 1 출력
    elif com[0] == "top":
        if (len(stack)): # 스택에 데이터가 있으면
            print(stack[-1]) # top 연산
        else:
            print(-1) # -1 출력
```

오큰수 / 17298

백준 17298 / <https://www.acmicpc.net/problem/17298>

문제

크기가 N 인 수열 $A = A_1, A_2, \dots, A_N$ 이 있다. 수열의 각 원소 A_i 에 대해서 오큰수 $NGE(i)$ 를 구하려고 한다. A_i 의 오큰수는 오른쪽에 있으면서 A_i 보다 큰 수 중에서 가장 왼쪽에 있는 수를 의미한다. 그러한 수가 없는 경우에 오큰수는 -1 이다.

예를 들어, $A = [3, 5, 2, 7]$ 인 경우 $NGE(1) = 5$, $NGE(2) = 7$, $NGE(3) = 7$, $NGE(4) = -1$ 이다. $A = [9, 5, 4, 8]$ 인 경우에는 $NGE(1) = -1$, $NGE(2) = 8$, $NGE(3) = 8$, $NGE(4) = -1$ 이다.

입력

첫째 줄에 수열 A 의 크기 N ($1 \leq N \leq 1,000,000$)이 주어진다. 둘째 줄에 수열 A 의 원소 A_1, A_2, \dots, A_N ($1 \leq A_i \leq 1,000,000$)이 주어진다.

출력

총 N 개의 수 $NGE(1), NGE(2), \dots, NGE(N)$ 을 공백으로 구분해 출력한다.

오른수 / 17298

예제 입력 1에서

3 보다 오른쪽에서 큰 수중 가장 가까운 수는 5

5 보다 오른쪽에서 큰 수중 가장 가까운 수는 7

예제 입력 1 복사

```
4
3 5 2 7
```

예제 출력 1 복사

```
5 7 7 -1
```

예제 입력 2 복사

```
4
9 5 4 8
```

예제 출력 2 복사

```
-1 8 8 -1
```

오큰수 / 17298

각 인덱스에 대해서 자신보다 오른쪽의 큰 수 중에서
자신과 가장 가까운 수를 출력하면 됨
오른쪽에 자신 보다 큰 수가 없으면 -1 출력

오큰수 / 17298

나이브하게 각 인덱스에 대해서 오른쪽을 보면서
큰 수가 있을 때마다 찾는 방식?

6 5 4 3 2 1처럼 감소하는 데이터에 대해서는
각 인덱스마다 반복문이 N까지 돌기 때문에 $O(N^2)$

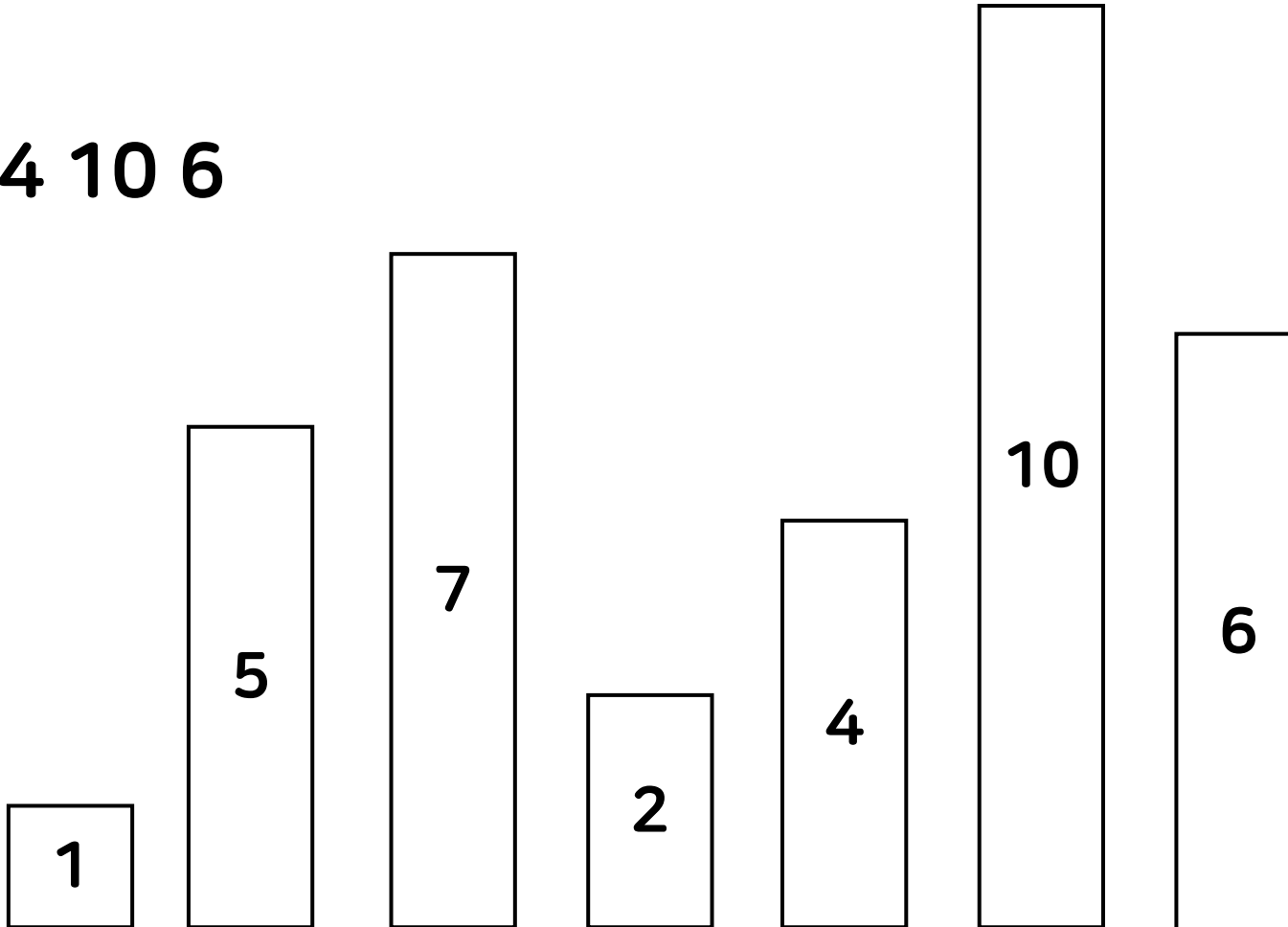
모든 쌍에 대해서 계산 해보는 완전 탐색과 같은 시간 복잡도

그렇다면 정답에 영향을 줄 수 있는 쌍만 보는 법은 없을까?

오큰수 / 17298

Ex) 6

1 5 7 2 4 10 6

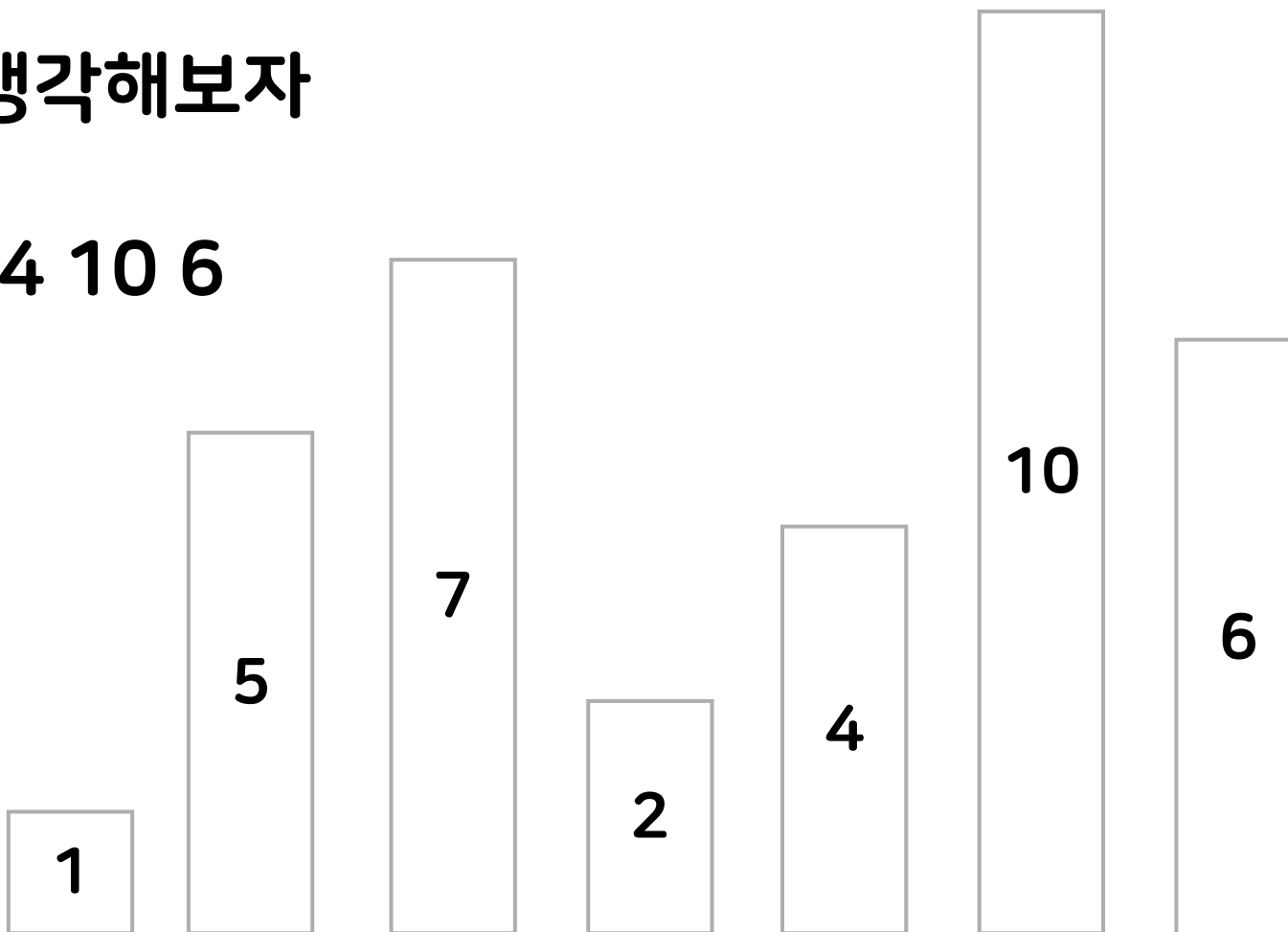


오큰수 / 17298

뒤에서부터 생각해보자

Ex) 6

1 5 7 2 4 10 6

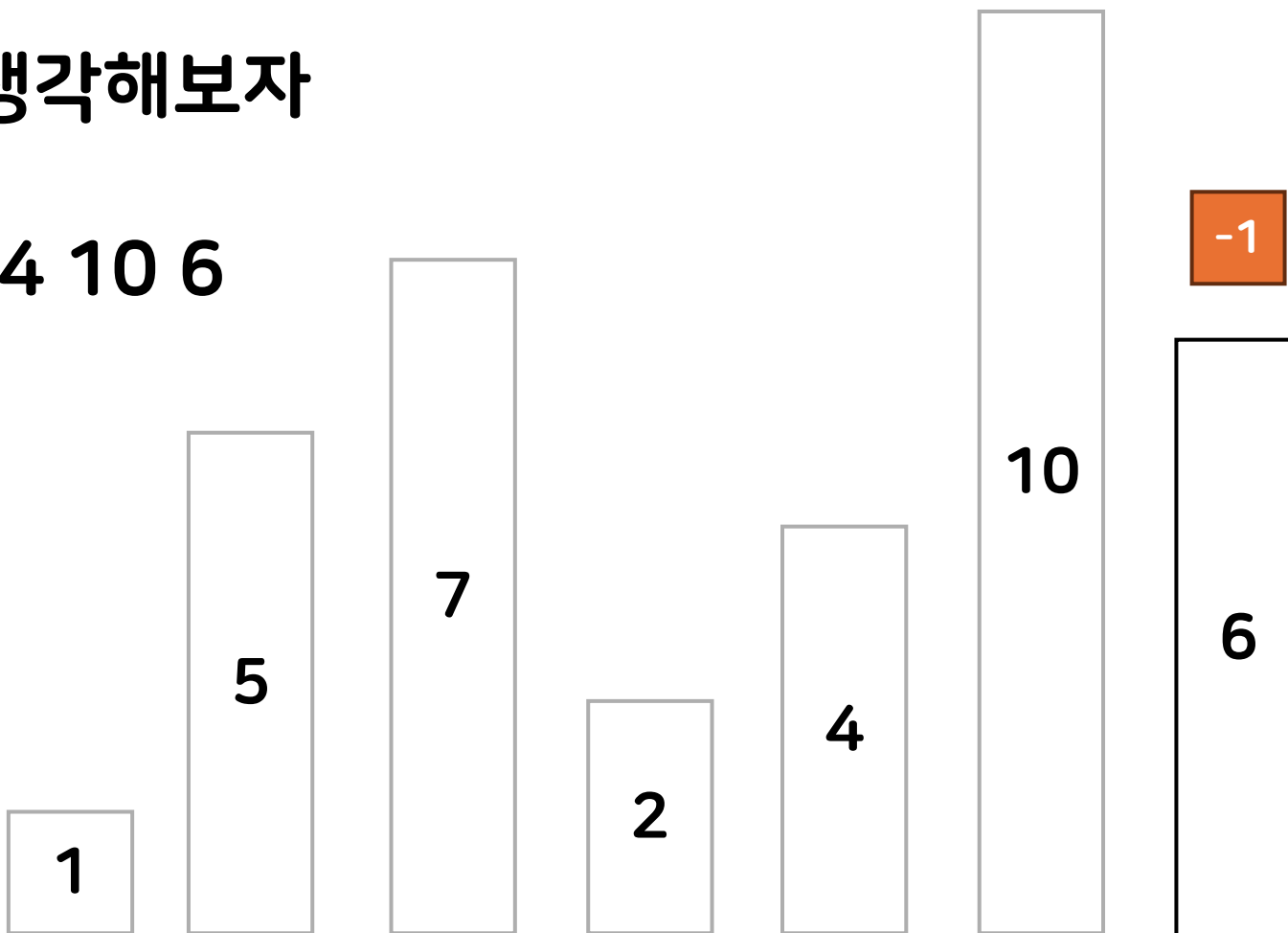


오큰수 / 17298

뒤에서부터 생각해보자

Ex) 6

1 5 7 2 4 10 6

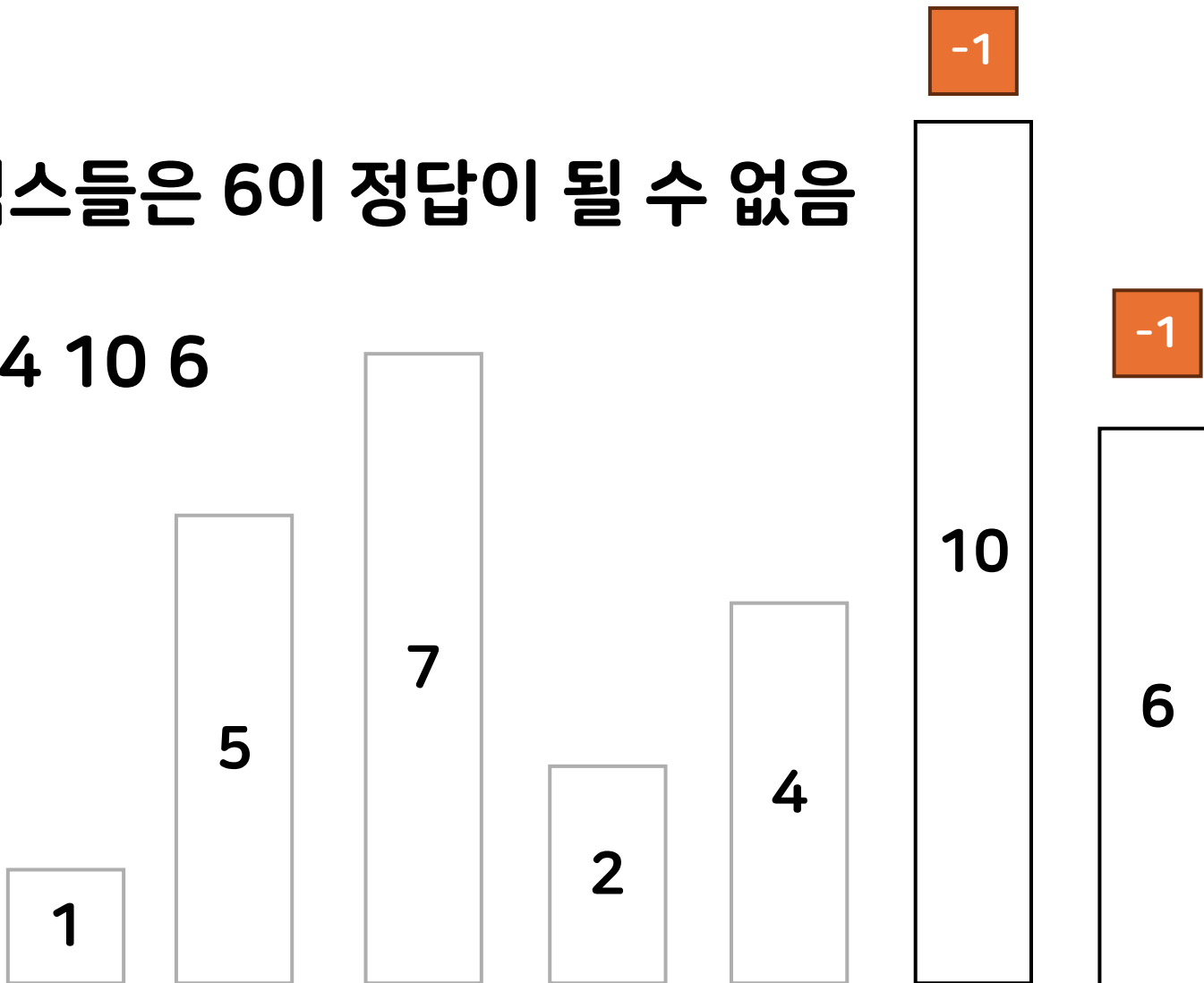


오큰수 / 17298

10 앞의 인덱스들은 6이 정답이 될 수 없음

Ex) 6

1 5 7 2 4 10 6



오큰수 / 17298

10 앞의 인덱스들은 6이 정답이 될 수 없음

Ex) 6

1 5 7 2 4 10 6



오큰수 / 17298

10 앞의 인덱스들은 6이 정답이 될 수 없음

Ex) 6

1 5 7 2 4 10 6



오큰수 / 17298

7 앞의 인덱스들은 2, 4가 정답이 될 수 없음

Ex) 6

1 5 7 2 4 10 6

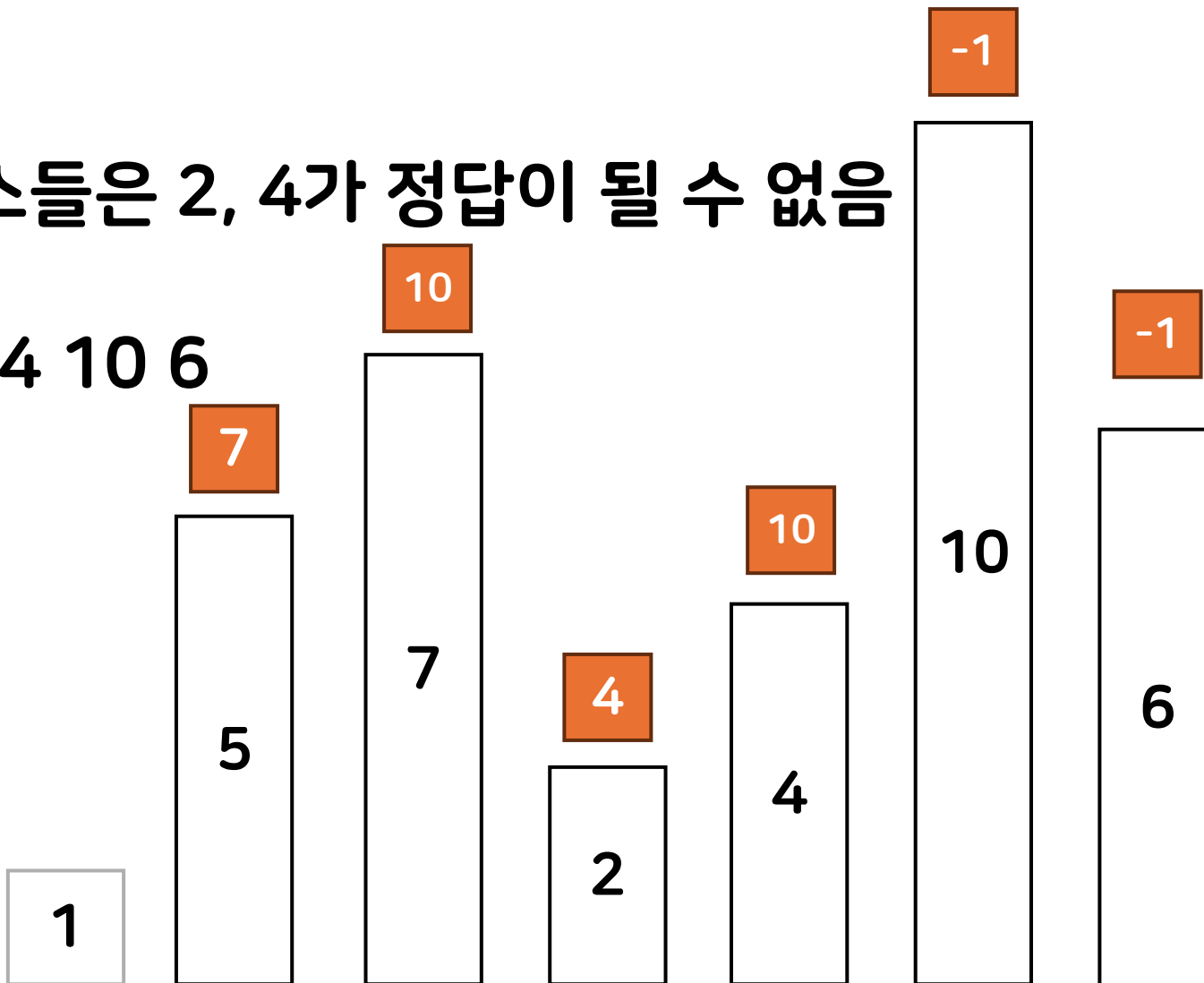


오큰수 / 17298

7 앞의 인덱스들은 2, 4가 정답이 될 수 없음

Ex) 6

1 5 7 2 4 10 6



오큰수 / 17298

7 앞의 인덱스들은 2, 4가 정답이 될 수 없음

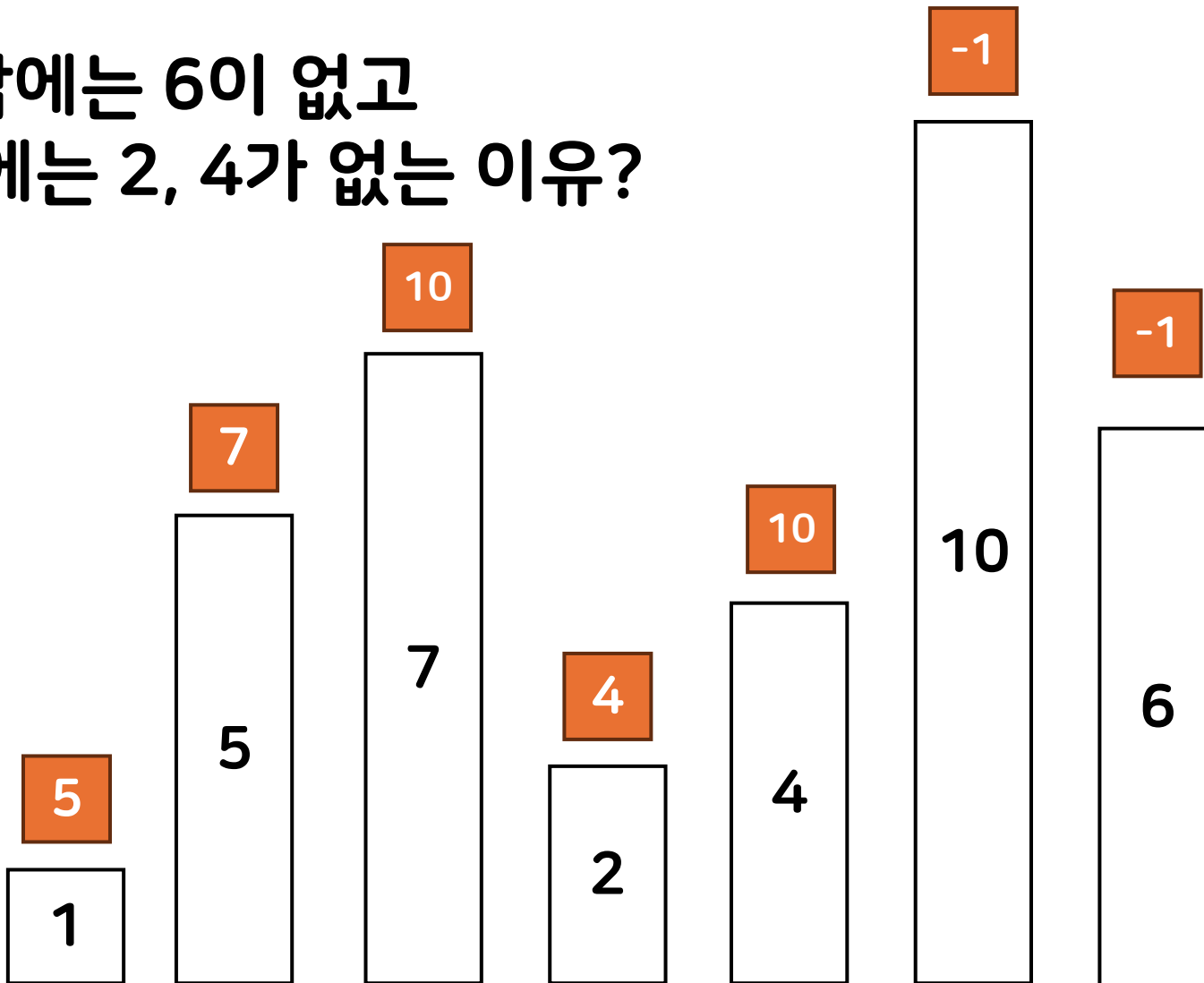
Ex) 6

1 5 7 2 4 10 6



오큰수 / 17298

10 앞의 정답에는 6이 없고
7 앞의 정답에는 2, 4가 없는 이유?



오큰수 / 17298

10 앞의 정답에는 6이 없고
7 앞의 정답에는 2, 4가 없는 이유?



오큰수 / 17298

각 화살표가 처음으로 닿는 지점이 정답
큰 수 뒤에 있는 작은 수들은 닿을 수 없음



오큰수 / 17298

오른쪽 원소부터 차례대로 스택에 삽입하면서
스택의 원소가 자신보다 크면 그 수가 오큰수가 되고
아니면 스택의 원소가 자신보다 클 때까지 pop 연산을 하면 됨

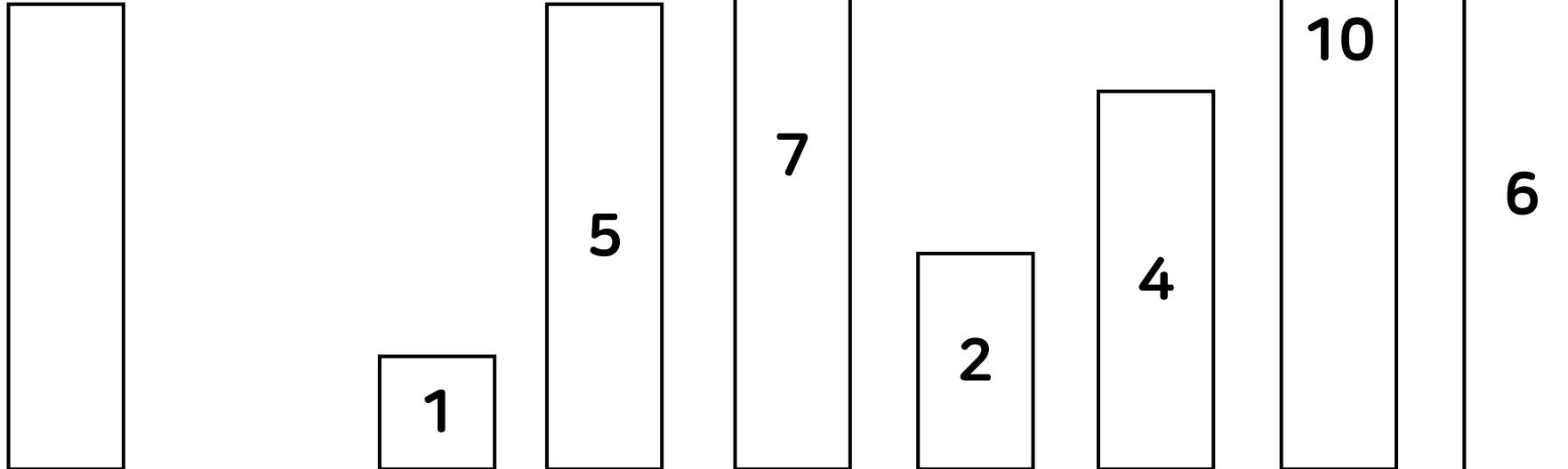
스택의 원소가 없을 때는 -1로 예외처리

오큰수 / 17298

Ex) 6

1 5 7 2 4 10 6

스택

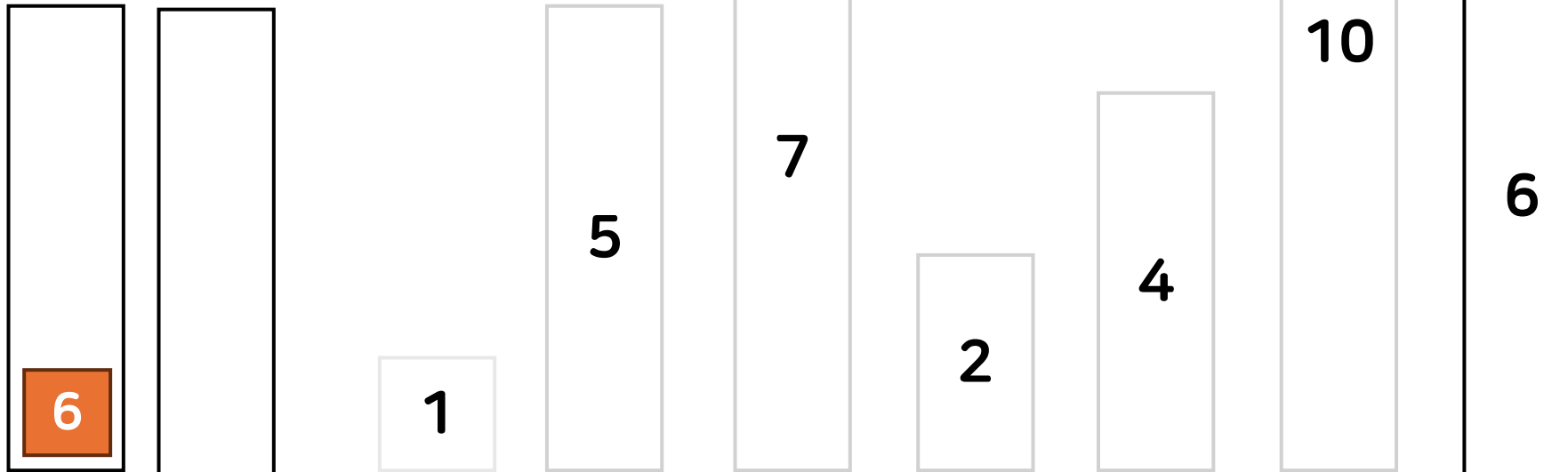


오큰수 / 17298

Ex) 6

1 5 7 2 4 10 6

스택

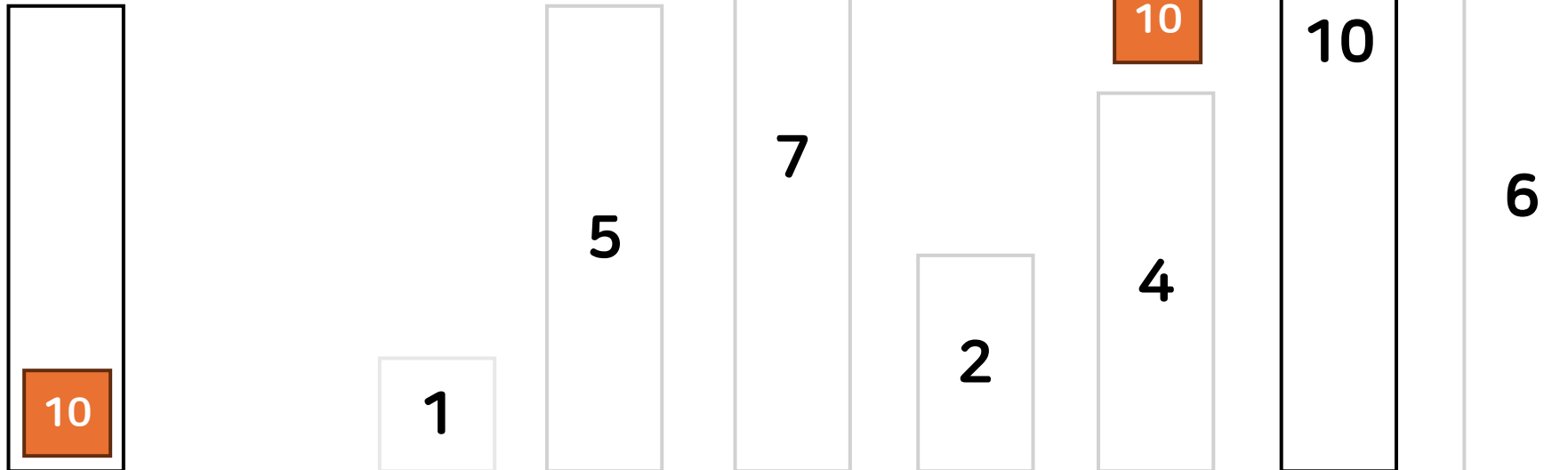


오큰수 / 17298

Ex) 6

1 5 7 2 4 10 6

스택

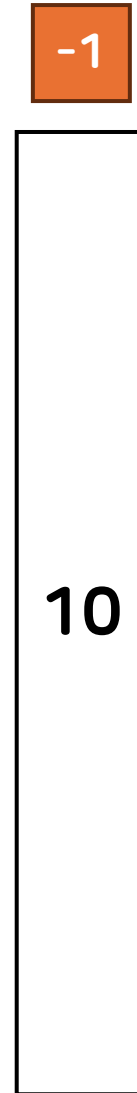
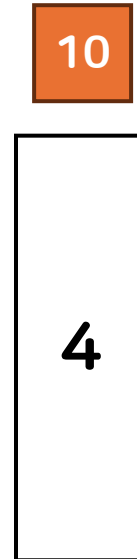
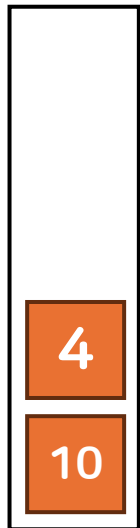


오큰수 / 17298

Ex) 6

1 5 7 2 4 10 6

스택

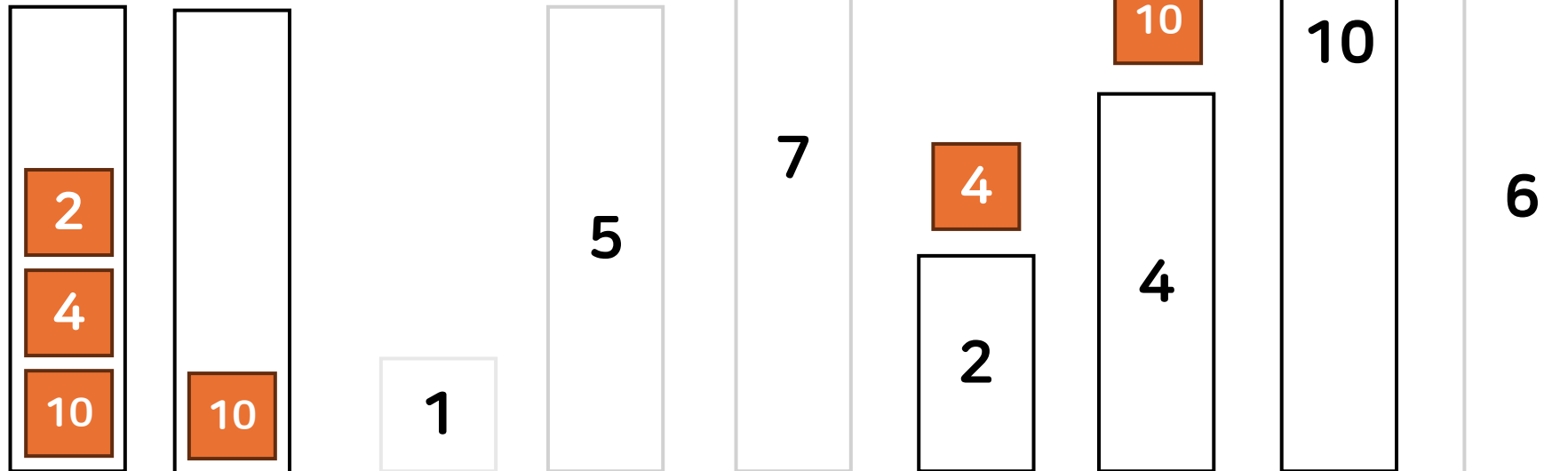


오큰수 / 17298

Ex) 6

1 5 7 2 4 10 6

스택

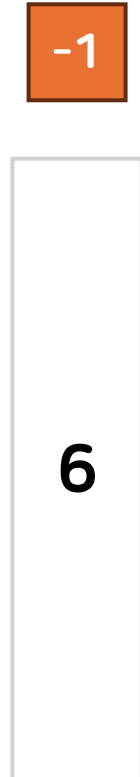
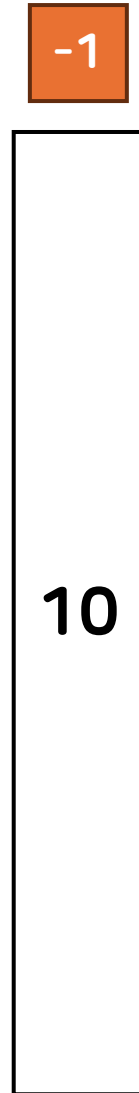
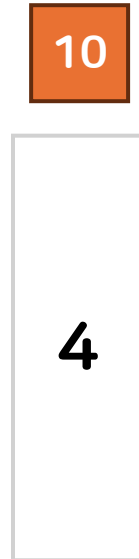
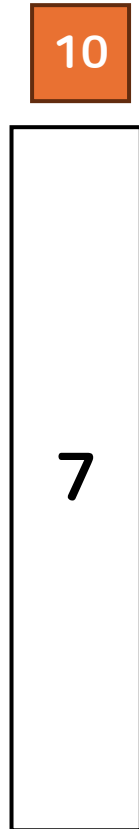
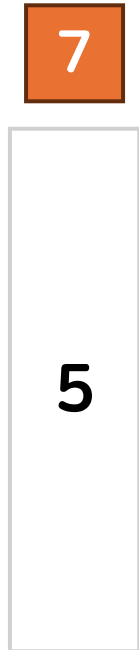
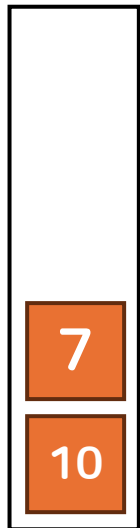


오큰수 / 17298

Ex) 6

1 5 7 2 4 10 6

스택

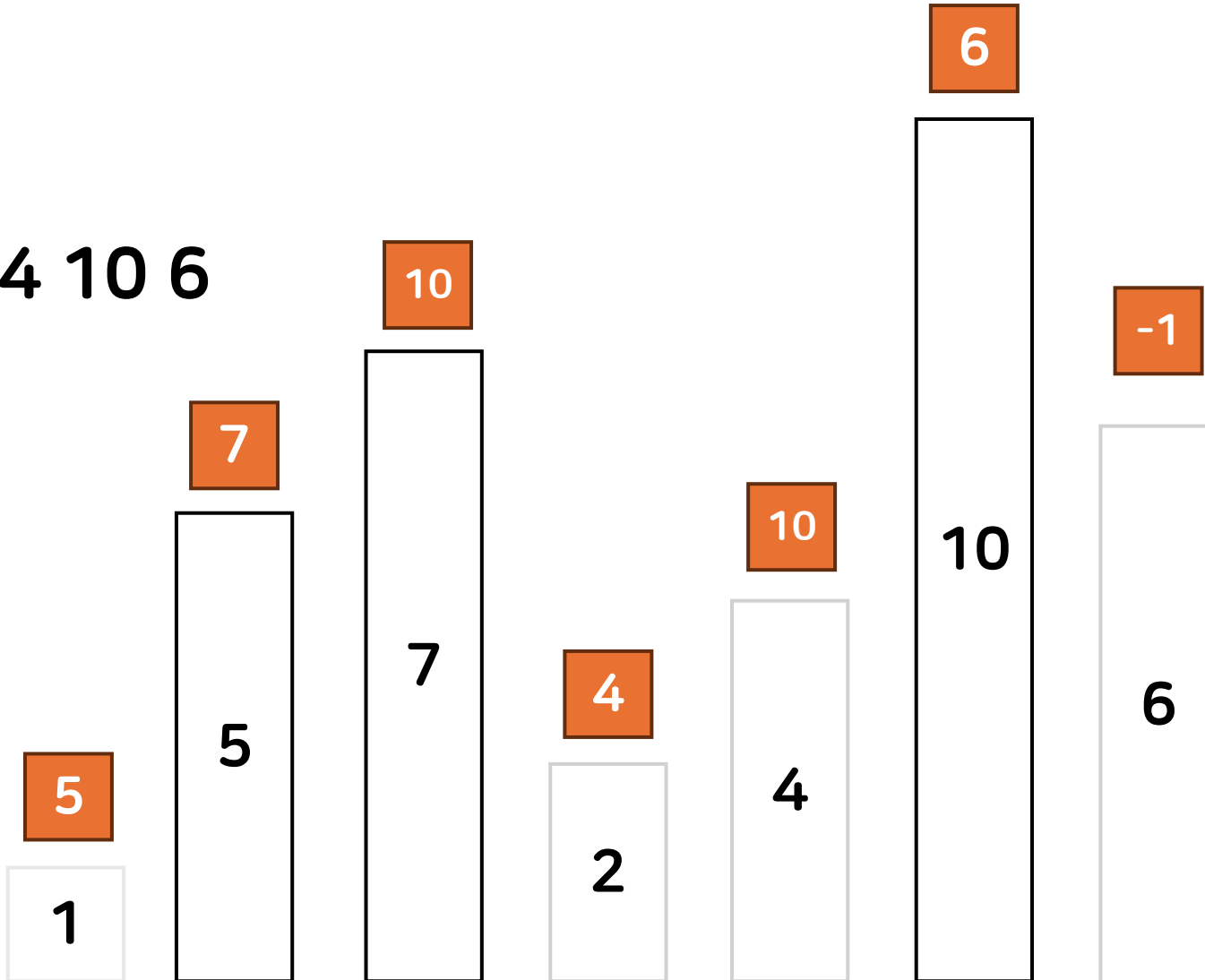


오큰수 / 17298

Ex) 6

1 5 7 2 4 10 6

스택

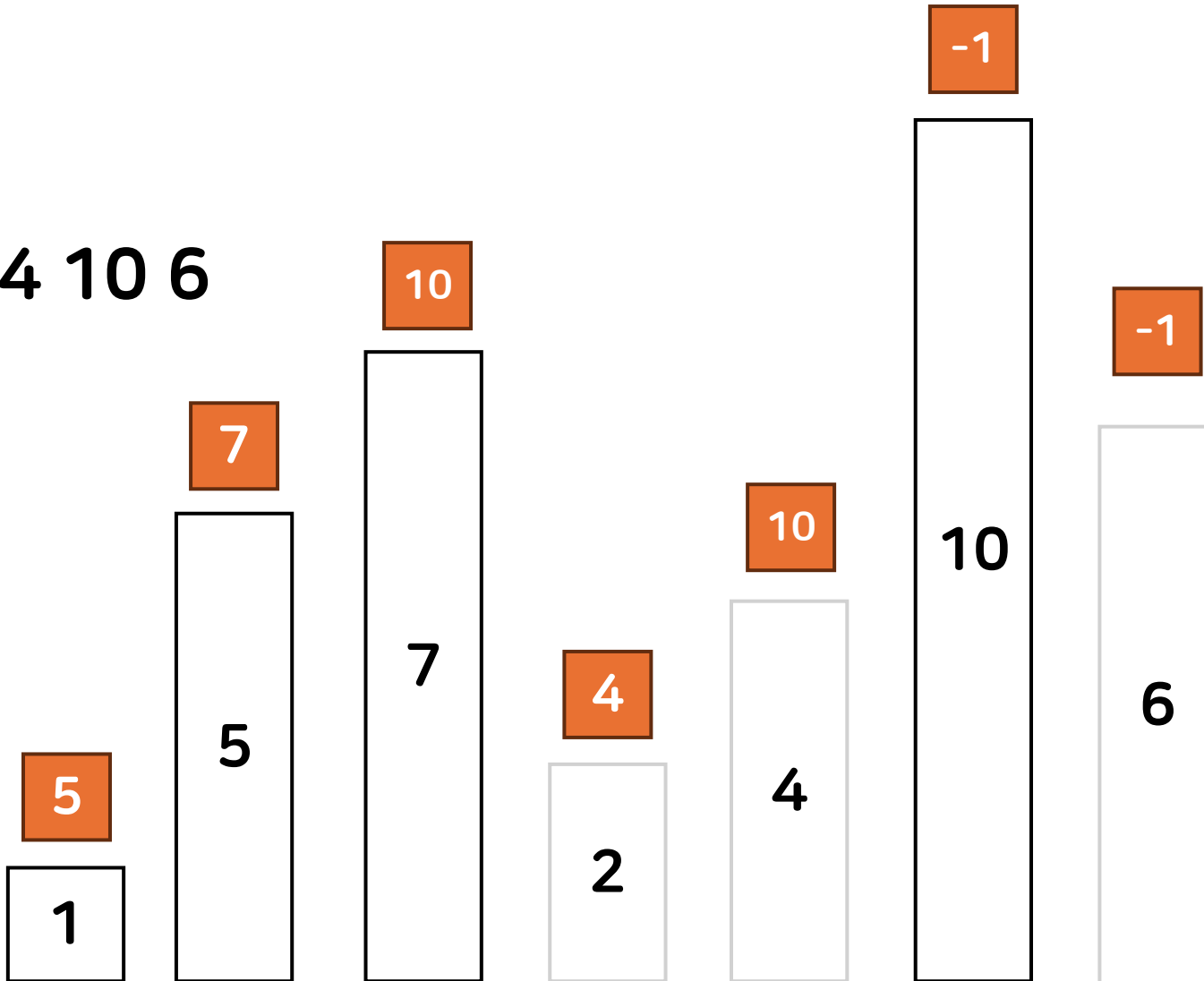


오큰수 / 17298

Ex) 6

1 5 7 2 4 10 6

스택



오큰수 / 17298

스택의 데이터들을 잘 살펴보면 데이터들이 정렬되어 있음

이렇게 스택의 단조성을 유지하는 기법을 모노톤 스택이라고 함

오큰수 / 17298

C++

```
#include <iostream>
#include <stack>
using namespace std;

const int MAX = 1010101;
stack<int> st;
int a[MAX], result[MAX];

int main(){
    int n; cin >> n;
    for(int i = 1; i <= n; i++) cin >> a[i];

    for(int i = n; i >= 1; i--){
        // 스택이 비어있지 않고 스택의 마지막 원소가 a[i]보다 작으면 pop
        while(!st.empty() && st.top() <= a[i]) st.pop();
        if(st.empty()) result[i] = -1; // 스택이 비어있으면 정답은 -1
        else result[i] = st.top(); // 아니면 정답은 스택의 마지막 원소

        st.push(a[i]); // 스택에 a[i] 삽입
    }

    for(int i = 1; i <= n; i++) cout << result[i] << " ";
}
```


오큰수 / 17298

Python

```
n = int(input())
stack = []
a = []
result = [0] * n

a = list(map(int, input().split()))

for i in range(n - 1, -1, -1):
    # 스택이 비어있지 않고 스택의 마지막 원소가 a[i]보다 작으면 pop
    while len(stack) and stack[-1] <= a[i]:
        stack.pop()

    if(len(stack)): # 스택이 비어있지 않으면 정답은 스택의 마지막 원소
        result[i] = stack[-1]
    else: # 아니면 정답은 -1
        result[i] = -1

    stack.append(a[i]) # 스택에 a[i] 삽입

for i in range(n):
    print(result[i], end = ' ')
```

질문?

큐

큐

먼저 넣은 데이터가 먼저 나옴

-enqueue(x) : x를 큐에 넣음

-dequeue() : 큐의 처음 데이터를 제거함



큐

큐

먼저 넣은 데이터가 먼저 나옴

-enqueue(x) : x를 큐에 넣음

-dequeue() : 큐의 처음 데이터를 제거함

enqueue(2)



2

큐

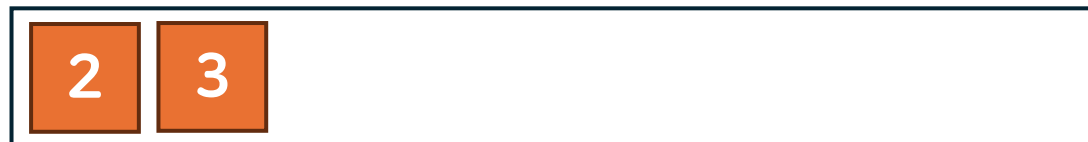
큐

먼저 넣은 데이터가 먼저 나옴

-enqueue(x) : x를 큐에 넣음

-dequeue() : 큐의 처음 데이터를 제거함

enqueue(3)



큐

큐

먼저 넣은 데이터가 먼저 나옴

-enqueue(x) : x를 큐에 넣음

-dequeue() : 큐의 처음 데이터를 제거함

enqueue(6)



큐

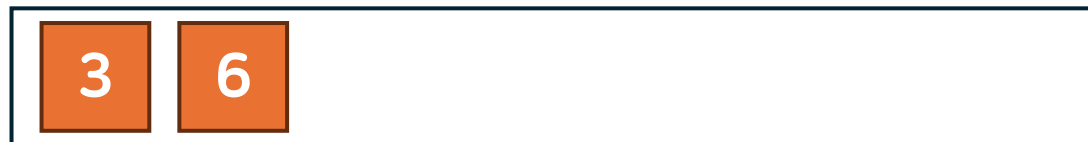
큐

먼저 넣은 데이터가 먼저 나옴

-enqueue(x) : x를 큐에 넣음

-dequeue() : 큐의 처음 데이터를 제거함

dequeue()



큐

큐

먼저 넣은 데이터가 먼저 나옴

-enqueue(x) : x를 큐에 넣음

-dequeue() : 큐의 처음 데이터를 제거함

dequeue()



6

큐

큐

먼저 넣은 데이터가 먼저 나옴

-enqueue(x) : x를 큐에 넣음

-dequeue() : 큐의 처음 데이터를 제거함

dequeue()



큐

C++

`std::queue`

C++ 표준 라이브러리에 있음

헤더 : `<queue>`

함수

`push(x)` : x를 큐에 넣음

`pop()` : 큐의 처음 데이터 제거

`front()` : 큐의 처음 데이터 반환

`empty()` : 큐가 비어있으면 1 반환 아니면 0 반환

`size()` : 큐의 데이터 개수 반환

큐

C++

함수

push(x) : x를 큐에 넣음

pop() : 큐의 처음 데이터 제거

front() : 큐의 처음 데이터 반환

empty() : 큐가 비어있으면 1 반환 아니면 0 반환

size() : 큐의 데이터 개수 반환

마찬가지로 시간 복잡도 모두 $O(1)$



C++

Received Output:

2
3
2
1

```
#include <iostream>
#include <queue>
using namespace std;
queue <int> q;

int main(){
    q.push(2); // 2
    q.push(3); // 2 3
    q.push(6); // 2 3 6

    cout << q.front() << "\n"; // 2

    q.pop(); // 3 6

    cout << q.front() << "\n"; // 3

    cout << q.size() << "\n"; // 2

    q.pop(); // 6
    q.pop(); //

    cout << q.empty() << "\n"; // 1

    return 0;
}
```

큐

Python

파이썬은 따로 큐가 없음

대신 뒤에 나올 큐의 상위 호환인덱스 사용

큐

Python

```
from collections import deque  
q = deque()
```

함수

append(x) : x를 큐에 넣음

popleft() : 큐의 처음 데이터 제거

q[0] : 큐의 처음 데이터 반환

len(q) : 큐의 데이터 개수 반환



Python

Received Output:

```
2
3
2
True
```

```
from collections import deque

q = deque()

q.append(2) # 2
q.append(3) # 2 3
q.append(6) # 2 3 6

print(q[0]) # 2

q.popleft() # 3 6

print(q[0]) # 3

print(len(q)) # 2

q.popleft() # 6
q.popleft() #

print(len(q) == 0) # True
```

덱

덱 / 양방향 큐

큐는 데이터 삽입 시에 뒤로 들어가고
데이터 추출 시에 앞에서 나옴

하지만 덱은 앞과 뒤 모두 삽입 / 추출이 가능함

덱

덱 / 양방향 큐

push_front(x) : 덱의 앞에 원소 삽입

push_back(x) : 덱의 뒤에 원소 삽입

pop_front() : 덱의 앞의 원소 삭제

pop_back() : 덱의 뒤의 원소 삭제

마찬가지로 empty(), size() 함수도 존재함

모두 시간 복잡도 $O(1)$

덱

덱 / 양방향 큐

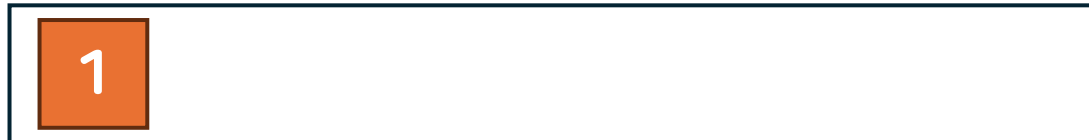
push_front(x) : 덱의 앞에 원소 삽입

push_back(x) : 덱의 뒤에 원소 삽입

pop_front() : 덱의 앞의 원소 삭제

pop_back() : 덱의 뒤의 원소 삭제

push_front(1)



덱

덱 / 양방향 큐

push_front(x) : 덱의 앞에 원소 삽입

push_back(x) : 덱의 뒤에 원소 삽입

pop_front() : 덱의 앞의 원소 삭제

pop_back() : 덱의 뒤의 원소 삭제

push_front(2)



덱

덱 / 양방향 큐

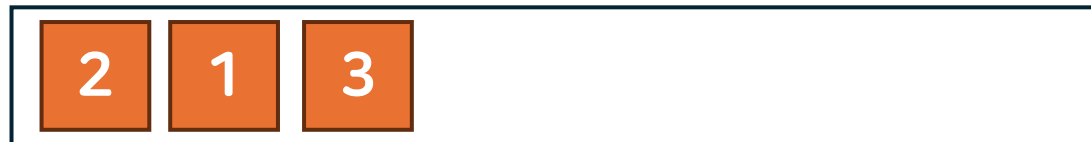
push_front(x) : 덱의 앞에 원소 삽입

push_back(x) : 덱의 뒤에 원소 삽입

pop_front() : 덱의 앞의 원소 삭제

pop_back() : 덱의 뒤의 원소 삭제

push_back(3)



덱

덱 / 양방향 큐

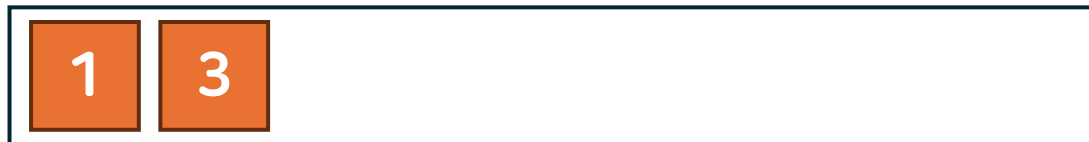
push_front(x) : 덱의 앞에 원소 삽입

push_back(x) : 덱의 뒤에 원소 삽입

pop_front() : 덱의 앞의 원소 삭제

pop_back() : 덱의 뒤의 원소 삭제

pop_front()



덱

덱 / 양방향 큐

push_front(x) : 덱의 앞에 원소 삽입

push_back(x) : 덱의 뒤에 원소 삽입

pop_front() : 덱의 앞의 원소 삭제

pop_back() : 덱의 뒤의 원소 삭제

pop_back()



덱

덱 / 양방향 큐

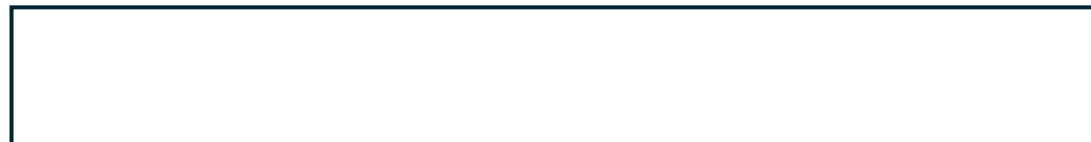
push_front(x) : 덱의 앞에 원소 삽입

push_back(x) : 덱의 뒤에 원소 삽입

pop_front() : 덱의 앞의 원소 삭제

pop_back() : 덱의 뒤의 원소 삭제

pop_back()



덱

C++

`std::deque`

C++ 표준 라이브러리에 있음

헤더 : `<deque>`

함수

`push_front(x)`: 데이터를 덱의 앞에 넣음

`pop_front()` : 덱의 처음 데이터 제거

`push_back(x)` : 데이터를 덱의 뒤에 넣음

`pop_back()` : 덱의 마지막 데이터 제거

`empty(), size()`: 스택, 큐와 동일

덱

C++

`std::deque`

C++ 표준 라이브러리에 있음

헤더 : `<deque>`

함수

`back()` : 뒤의 데이터 반환

`front()` : 앞의 데이터 반환

덱

C++

Received Output:

3

2

```
#include <iostream>
#include <deque>
using namespace std;
deque <int> dq;

int main(){
    dq.push_front(1); // 1
    dq.push_front(2); // 2 1
    dq.push_back(3); // 2 1 3

    cout << dq.back() << "\n"; // 3

    cout << dq.front() << "\n"; // 2

    dq.pop_front(); // 1 3
    dq.pop_back(); // 1
    dq.pop_back(); //

}
```

덱

Python

```
from collections import deque  
q = deque()
```

함수

<code>append_left(x)</code>	: 데이터를 덱의 앞에 넣음
<code>pop_left()</code>	: 덱의 처음 데이터 제거 및 반환
<code>append(x)</code>	: 데이터를 덱의 뒤에 넣음
<code>pop()</code>	: 덱의 마지막 데이터 제거 및 반환

덱

Python

```
from collections import deque  
q = deque()
```

함수

데이터 반환

`dq[0]` : 처음 데이터 반환

`dq[-1]`: 마지막 데이터 반환

덱

Python

Received Output:

3
2

```
from collections import deque

# 덱 초기화
dq = deque()

dq.appendleft(1) # 1
dq.appendleft(2) # 2 1

dq.append(3) # 2 1 3

# back 출력 (오른쪽 끝 원소)
print(dq[-1]) # 3

# front 출력 (왼쪽 끝 원소)
print(dq[0]) # 2

dq.popleft() # 1, 3

dq.pop() # 1
dq.pop() #
```

덱

백준 10866 / <https://www.acmicpc.net/problem/10866>

문제

정수를 저장하는 덱(Deque)를 구현한 다음, 입력으로 주어지는 명령을 처리하는 프로그램을 작성하시오.

명령은 총 여덟 가지이다.

- push_front X: 정수 X를 덱의 앞에 넣는다.
- push_back X: 정수 X를 덱의 뒤에 넣는다.
- pop_front: 덱의 가장 앞에 있는 수를 빼고, 그 수를 출력한다. 만약, 덱에 들어있는 정수가 없는 경우에는 -1을 출력한다.
- pop_back: 덱의 가장 뒤에 있는 수를 빼고, 그 수를 출력한다. 만약, 덱에 들어있는 정수가 없는 경우에는 -1을 출력한다.
- size: 덱에 들어있는 정수의 개수를 출력한다.
- empty: 덱이 비어있으면 1을, 아니면 0을 출력한다.
- front: 덱의 가장 앞에 있는 정수를 출력한다. 만약 덱에 들어있는 정수가 없는 경우에는 -1을 출력한다.
- back: 덱의 가장 뒤에 있는 정수를 출력한다. 만약 덱에 들어있는 정수가 없는 경우에는 -1을 출력한다.

입력

첫째 줄에 주어지는 명령의 수 N ($1 \leq N \leq 10,000$)이 주어진다. 둘째 줄부터 N 개의 줄에는 명령이 하나씩 주어진다. 주어지는 정수는 1보다 크거나 같고, 100,000보다 작거나 같다. 문제에 나와있지 않은 명령이 주어지는 경우는 없다.

출력

출력해야하는 명령이 주어질 때마다, 한 줄에 하나씩 출력한다.

덱 / 10866

단순 덱 구현 문제

스택처럼 덱에 데이터가 없을 때

pop_back(), pop_front(), back(), front()

연산을 호출하면 런타임 에러가 남

마찬가지로 데이터가 없을 때 -1 출력을 하면 됨

덱 / 10866

C++

```
#include <iostream>
#include <deque>
using namespace std;

deque<int> dq;

int main() {
    int n; cin >> n;
    for(int i = 1; i <= n; i++){
        string s; cin >> s; // 명령어 입력
        if(s == "push_back"){
            int x; cin >> x;
            dq.push_back(x); // 덱 뒤에 x 삽입
        }
        else if(s == "push_front"){
            int x; cin >> x;
            dq.push_front(x); // 덱 앞에 x 삽입
        }
        else if(s == "front"){
            if(dq.empty()) cout << -1 << "\n"; // 비어있으면 -1 출력
            else cout << dq.front() << "\n"; // 아니면 처음 데이터 출력
        }
        else if(s == "back"){
            if(dq.empty()) cout << -1 << "\n"; // 비어있으면 -1 출력
            else cout << dq.back() << "\n"; // 아니면 마지막 데이터 출력
        }
        else if(s == "empty") cout << dq.empty() << "\n"; // 비어있으면 1 아니면 0 출력
        else if(s == "pop_back"){
            if(dq.empty()) cout << -1 << "\n"; // 비어있으면 -1 출력
            else{
                cout << dq.back() << "\n"; // 마지막 데이터 출력
                dq.pop_back(); // 마지막 데이터 삭제
            }
        }
        else if(s == "pop_front"){
            if(dq.empty()) cout << -1 << "\n"; // 비어있으면 -1 출력
            else{
                cout << dq.front() << "\n"; // 처음 데이터 출력
                dq.pop_front(); // 처음 데이터 삭제
            }
        }
        else cout << dq.size() << "\n"; // 데이터 개수 출력
    }

    return 0;
}
```


덱 / 10866

Python

```
from collections import deque

# 덱 생성
dq = deque()
n = int(input())

for _ in range(n):
    command = list(map(str, input().split())) # 명령어 입력

    if command[0] == "push_back":
        dq.append(int(command[1])) # 덱 뒤에 삽입
    elif command[0] == "push_front":
        dq.appendleft(int(command[1])) # 덱 앞에 삽입
    elif command[0] == "front":
        if len(dq):
            print(dq[0]) # 비어있지 않으면 처음 데이터 출력
        else:
            print(-1) # 아니면 -1 출력
    elif command[0] == "back":
        if len(dq):
            print(dq[-1]) # 비어있지 않으면 마지막 데이터 출력
        else:
            print(-1) # 아니면 -1 출력
    elif command[0] == "empty":
        if len(dq):
            print(0) # 비어있지 않으면 0 출력
        else:
            print(1) # 아니면 1 출력
    elif command[0] == "pop_back":
        if len(dq):
            print(dq.pop()) # 비어있지 않으면 마지막 데이터 출력 후 제거
        else:
            print(-1) # 아니면 -1 출력
    elif command[0] == "pop_front":
        if len(dq):
            print(dq.popleft()) # 비어있지 않으면 처음 데이터 출력 후 제거
        else:
            print(-1) # 아니면 -1 출력
    else:
        print(len(dq)) # 덱 크기 출력
```

질문?

기본 과제

스택 - <https://www.acmicpc.net/problem/10828>

오큰수 - <https://www.acmicpc.net/problem/17298>

괄호의 값 - <https://www.acmicpc.net/problem/2504>

덱 - <https://www.acmicpc.net/problem/10866>

스택 수열 - <https://www.acmicpc.net/problem/1874>

심화 과제

ABB to BA - <https://www.acmicpc.net/problem/32293>

오아시스 재결합 - <https://www.acmicpc.net/problem/3015>

고생하셨습니다