목차

시간복잡도

시간복잡도

시간복잡도

입력 N에 대한 대략적인 소요 시간

시간복잡도

입력 N에 대한 대략적인 소요 시간

```
int sum = 0;
for (int i=1; i<=N; i++) {
    sum += i;
}
```

시간복잡도

입력 N에 대한 대략적인 소요 시간

```
int sum = 0;
for (int i=1; i<=N; i++) {
    sum += i;
}
```

‘반복문’

시간복잡도

입력 N에 대한 대략적인 소요 시간

```
int sum = 0;
for (int i=1; i<=N; i++) {
    sum += i;
}
```

O(3N+2)

시간복잡도

입력 N에 대한 대략적인 소요 시간

```
int sum = 0;
for (int i=1; i<=N; i++) {
    sum += i;
}
```

O(N)

시간복잡도

입력 N에 대한 대략적인 소요 시간

```
int sum = 0;
for (int i=1; i<=N; i++) {
    for (int j=1; j<=N; j++) {
        if (i == j) {
            sum += j;
        }
    }
}
```

$O(N^2)$

시간복잡도

입력 N에 대한 대략적인 소요 시간

```
int sum = 0;
sum = N*(N+1)/2;
```

O(1)

시간복잡도

입력 N에 대한 대략적인 소요 시간

- O(1)
- O(lgN)
- O(N) : 1억
- O(NlgN) : 5백만
- O(N^2) : 1만
- O(N^3) : 500
- O(2^N) : 20
- O(N!) : 10

시간복잡도

입력 N에 대한 대략적인 소요 시간

- O(1)
- O(lgN)
- O(N) : 1억
- O(NlgN) : 5백만
- O(N^2) : 1만
- O(N^3) : 500
- O(2^N) : 20
- O(N!) : 10

참고만 하자!

수 정렬하기 3 분류

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
3 초 (하단 참고)	8 MB (하단 참고)	89780	19877	14865	22.851%

문제

N개의 수가 주어졌을 때, 이를 오름차순으로 정렬하는 프로그램을 작성하시오.

입력

첫째 줄에 수의 개수 N(1 ≤ N ≤ 10,000,000)이 주어진다. 둘째 줄부터 N개의 줄에는 숫자가 주어진다. 이 수는 10,000보다 작거나 같은 자연수이다.

출력

첫째 줄부터 N개의 줄에 오름차순으로 정렬한 결과를 한 줄에 하나씩 출력한다.

```java
public static void main(String[] args) throws Exception {
  StringBuilder sb = new StringBuilder();

  final int N = Input.nextInt();
  int[] numbers = new int[N];

  for (int i = 0; i < N; i++) {
    numbers[i] = Input.nextInt();
  }

  mergeSort(numbers);

  for (int sortedNumber : numbers) {
    sb.append(sortedNumber).append('\n');
  }

  System.out.print(sb);
}
```

```java
public static void main(String[] args) throws Exception {
  StringBuilder sb = new StringBuilder();

  final int N = Input.nextInt();
  int[] numbers = new int[N];

  for (int i = 0; i < N; i++) {
    numbers[i] = Input.nextInt();
  }

  mergeSort(numbers);

  for (int sortedNumber : numbers) {
    sb.append(sortedNumber).append('\n');
  }

  System.out.print(sb);
}
```

O(NlogN)

```java
public static void main(String[] args) throws Exception {
  StringBuilder sb = new StringBuilder();

  final int N = Input.nextInt();
  int[] numbers = new int[N];

  for (int i = 0; i < N; i++) {
    numbers[i] = Input.nextInt();
  }

  mergeSort(numbers);

  for (int sortedNumber : numbers) {
    sb.append(sortedNumber).append('\n');
  }

  System.out.print(sb);
}
```

```java
public static void main(String[] args) throws Exception {
  StringBuilder sb = new StringBuilder();

  final int N = Input.nextInt();
  int[] numbers = new int[N];

  for (int i = 0; i < N; i++) {
    numbers[i] = Input.nextInt();
  }

  mergeSort(numbers);

  for (int sortedNumber : numbers) {
    sb.append(sortedNumber).append('\n');
  }

  System.out.print(sb);
}
```

O(N)

공간복잡도

공간복잡도

메모리 사용량

공간복잡도

메모리 사용량

int[10000] = 40000B = 39MB

공간복잡도

공간복잡도

시간 제한을 지키면 대부분 문제X

공간복잡도

시간 제한을 지키면 대부분 문제X

불필요한 배열 선언

스택

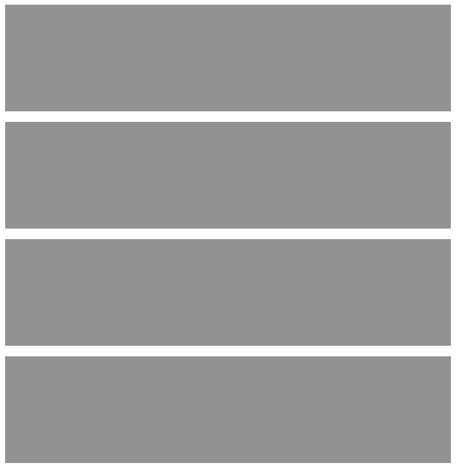
스택

스택

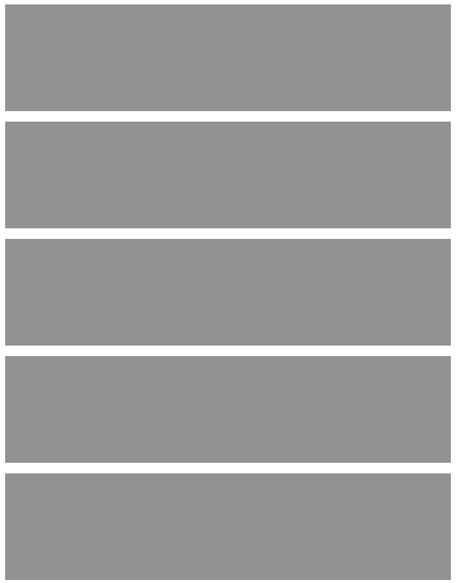
스택

스택

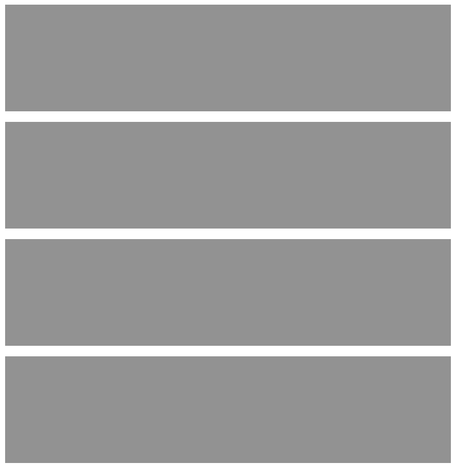
스택



스택



스택



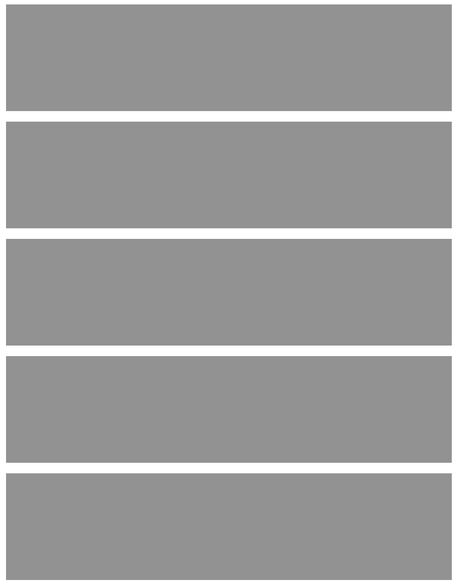
스택

스택

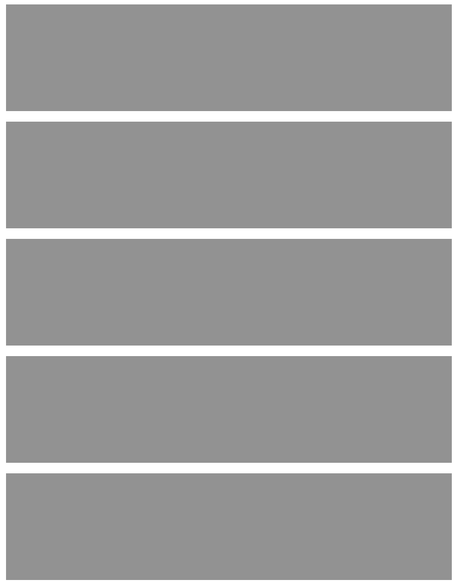
스택

스택

스택

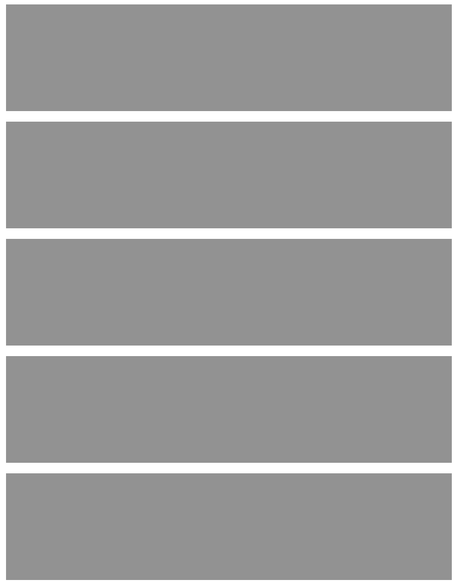


스택



후입선출

스택



후입선출

가장 위의 요소에만 접근할 수 있다

스택

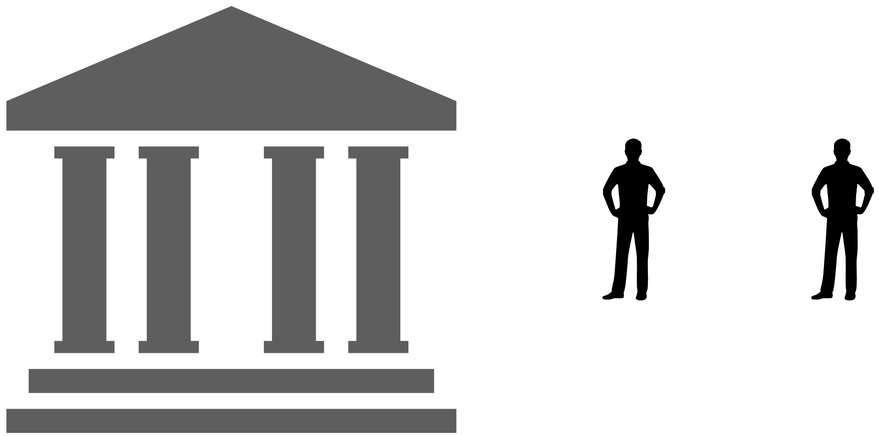
- push
- pop
- peek
- isEmpty

큐

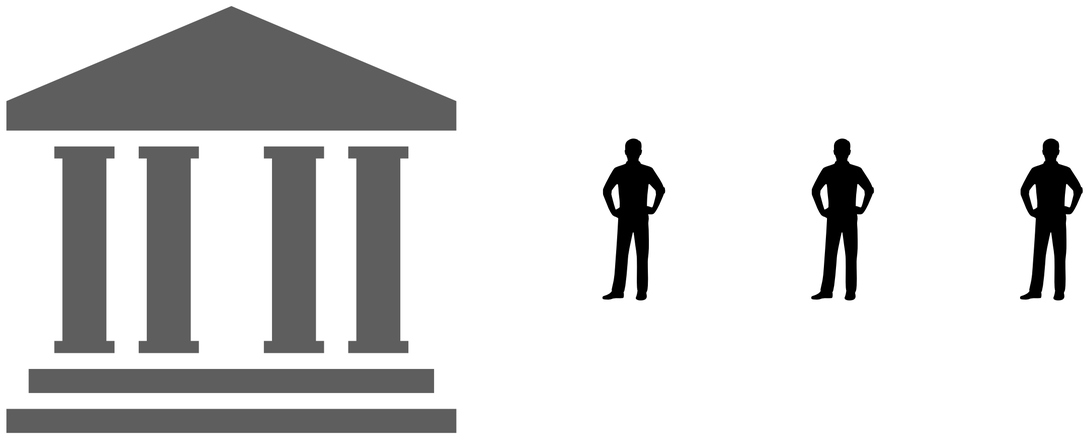
큐

ㅋ

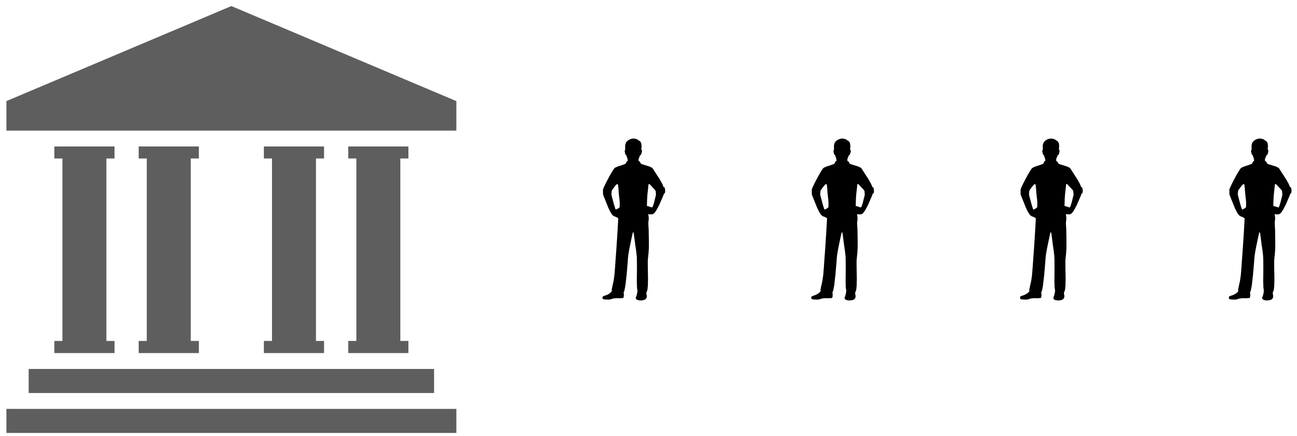
ㅋ



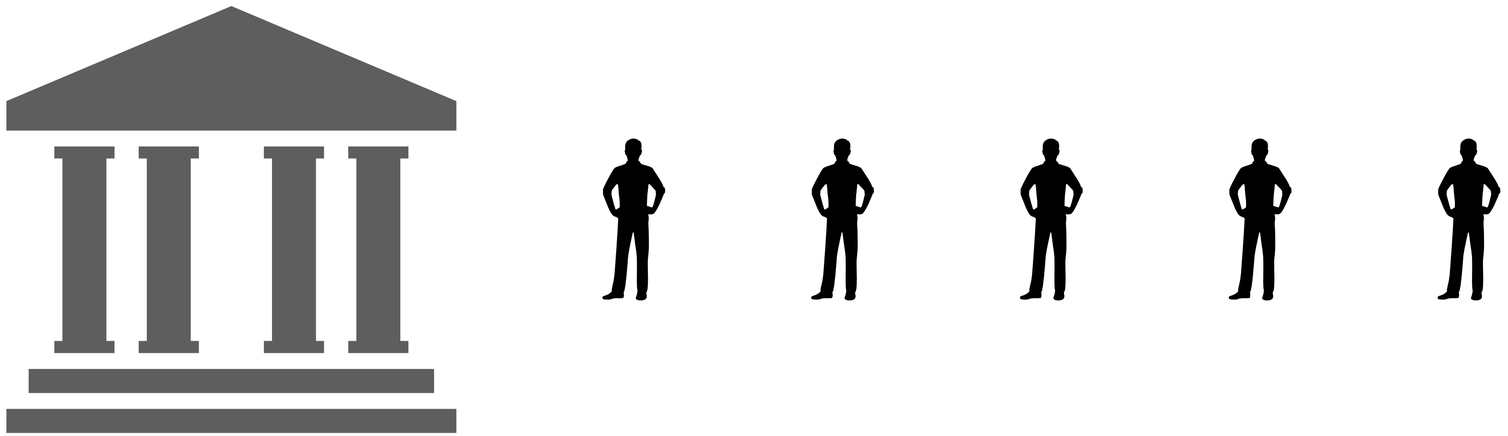
ㅋ



ㅋ



ㅋ



ㅋ

큐

큐

선입선출

큐

선입선출

뒤로 들어오고 앞으로 빠진다

큐

- enqueue
- dequeue
- peek
- isEmpty

덱

덱

앞뒤로 추가/삭제 모두 가능

라이브러리를 사용하자!