세상의 속도를
따라잡고 싶다면

Do it!

알고리즘 코딩 테스트

자바 편

이지스퍼블리싱

01

배열과
배열의 활용 알고리즘

01 알고리즘이란?
02 시간/공간 복잡도
03 배열
04 구간 합
05 투 포인터
06 슬라이딩 윈도우

알고리즘이란?

- 다음 두 코드는 1부터 10000까지의 정수의 합을 구하는 코드입니다.
- 연산 횟수를 비교하고, N이 커질 때 어떤 코드가 효율적인지 생각해봅시다.

반복문 사용

```java
public class SequenceSum {
    public static void main(String[] args) {
        // 1 ~ 10000 정수의 합
        int N = 10000;
        int sum = 0;

        for (int i = 1; i <= N; i++) {
            sum += i;
        }
        System.out.println(sum);
    }
}
```

공식 사용

```java
public class SequenceSum {
    public static void main(String[] args) {
        // 1 ~ 10000 정수의 합
        int N = 10000;
        int sum = N*(N+1)/2;

        System.out.println(sum);
    }
}
```

시간 복잡도

- 주어진 문제를 해결하기 위한 연산 횟수(실행 속도)
- 일반적으로 1억 번의 연산 = 1초
- 시간 복잡도 표기법
  - 빅-오메가 Ω(n) : 최선일 때(best case) 연산 횟수
  - 빅-세타 Θ(n) : 보통일 때(average case) 연산 횟수
  - 빅-오 O(n) : 최악일 때(worst case) 연산 횟수
- 1이 포함된 10개의 정수 배열에서 1의 값을 찾으려고 할 때, 최선, 보통, 최악의 연산 횟수는?
  - 최선 : 1번
  - 보통 : 5.5번
  - 최악 : 10번

시간 복잡도

- 코딩 테스트에서 빅-오 O(n) 표기법을 기준으로 수행시간 계산
  - 모든 테스트 케이스를 통과해야만 합격으로 판단하므로 최악일 때를 염두

시간 복잡도

- 빅-오 O(n) 계산하기
- 크기가 n인 배열 arr에 대해 다음 코드의 연산 횟수를 빅-오 표기법으로 구하세요.

```java
int total = 0;
for (int i = 0; i < n; i++) {
    total += arr[i];
}
```

```java
int total = 0;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        total += arr[j];
    }
}
```

시간 복잡도

- 빅-오 O(n) 계산하기
- 크기가 n인 배열 arr에 대해 다음 코드의 연산 횟수를 빅-오 표기법으로 구하세요.

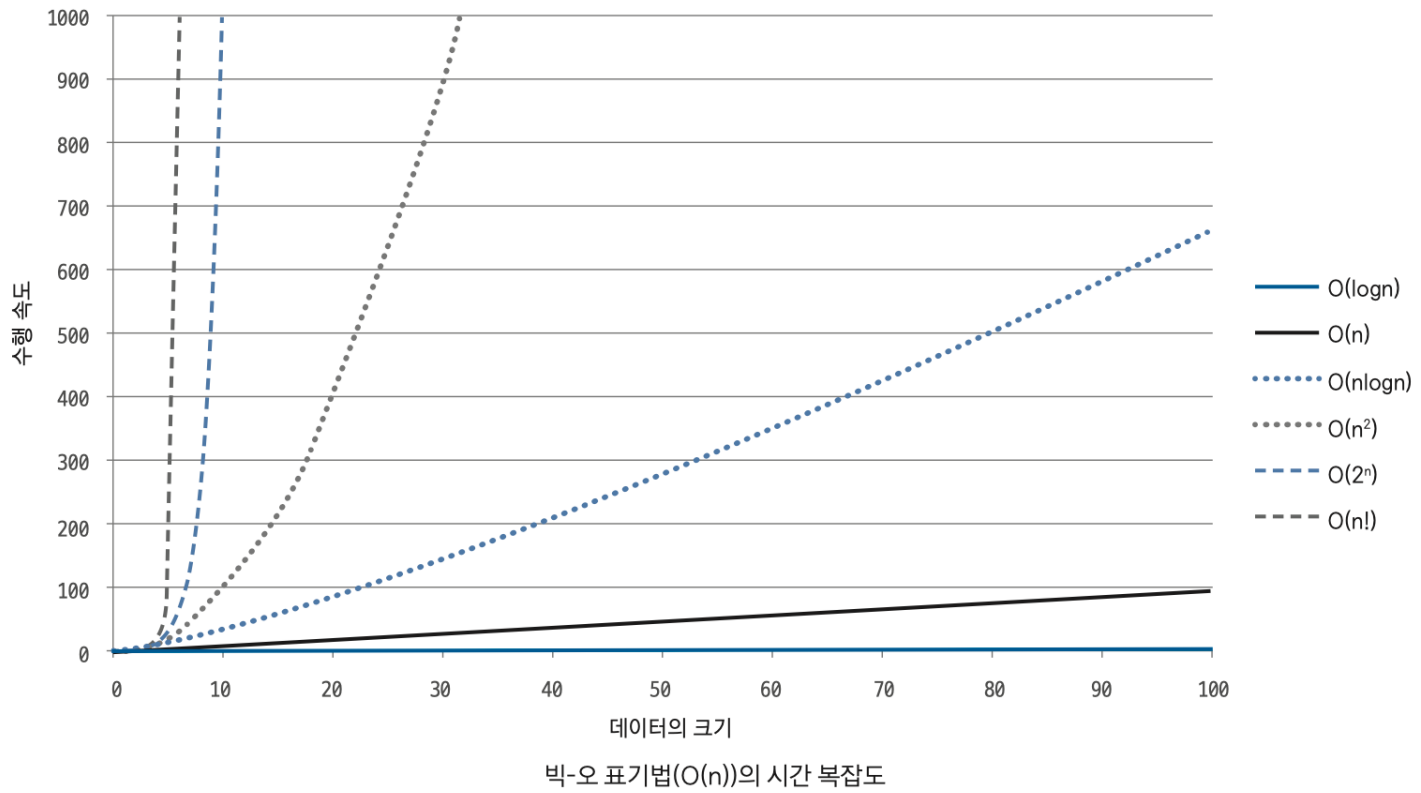
```java
int total = 0;
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < n; j++) {
        total += arr[j];
    }
}
```

```java
int total = 0;
for (int i = 0; i < n; i++) {
    total += arr[i];
}

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        total += arr[j];
    }
}
```

시간 복잡도

- $O(n^2 + n) = O(n^2)$
- $O(5n^3 + 3n + 1) = O(n^3)$
- $O(n + \log n) = O(n)$
- $O(n + n\log n) = O(n\log n)$
- $O(n^2 + 2^n) = O(2^n)$
- $O(n^2 + 2^n + n!) = O(n!)$



빅-오 표기법(O(n))의 시간 복잡도

- $O(n!) > O(2^n) > O(n^2) > O(n\log n) > O(n) > O(\log n)$

공간 복잡도

- 프로그램이 사용하는 메모리 크기
- 일반적으로 알고리즘 문제에서 메모리 크기 제한
  - 128MB, 256MB, 512MB, …

연습문제

- N의 크기에 따라 반복문과 이중 반복문의 실행 시간을 비교하세요.
  - 반복문의 실행 시간 확인하기

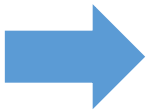
```java
long start, end;
start = System.currentTimeMillis(); // 시작 시간
// 여기에 반복문을 작성하세요.
end = System.currentTimeMillis(); // 종료 시간
System.out.println("반복문 실행시간(ms) : " + (end - start));
```

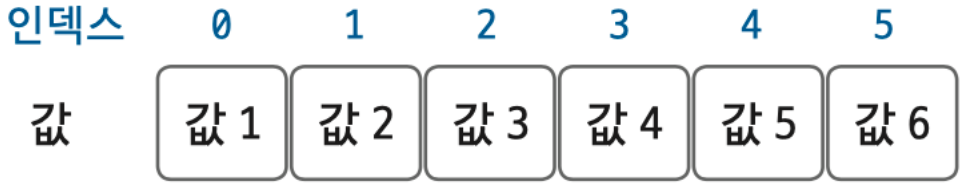
  - System.currentTimeMillis() : 현재 시간

배열

- 메모리의 연속 공간에 값이 채워져 있는 형태의 자료구조
- 배열의 값은 인덱스를 통해 참조할 수 있으며, 선언한 자료형의 값만 저장할 수 있습니다.

```
int [] arr = new int [6];
```





배열의 구조

배열

- java.util.Arrays 클래스
  - 배열을 다루는 다양한 메소드를 포함
  - 모든 메소드는 클래스 메소드(static method)이므로, 객체 생성 없이 바로 사용

- 자주 사용하는 메소드
  - Arrays.toString(배열) : 배열의 값들을 출력
  - Arrays.deepToString(배열) : 2차원 이상의 배열의 값들을 출력
  - Arrays.fill(배열, 값) : 배열의 모든 값을 특정 값으로 변경
  - Arrays.sort(배열) : 배열을 오름차순으로 정렬

연습문제

- 1차원 배열 [100, 200, 300, 400, 500]을 생성하고 다음을 수행하세요.
  - 300을 선택하여 출력하세요.
  - 400을 999로 변경하고, 변경된 배열을 출력하세요.
  - 배열을 정렬하고 정렬된 배열을 출력하세요.

- 2차원 배열 [[1,2,3], [4,5,6], [7,8,9]]를 생성하고 다음을 수행하세요.
  - 배열의 모든 값을 출력하세요.
  - 5를 999로 변경하고, 변경된 배열을 출력하세요.

연습문제

- 백준 온라인 저지 – 10807 개수 세기

구간 합

구간 합 구하기

시간 제한 **0.5초** | 난이도 **실버III** | 백준 온라인 저지 **11659번**

수 N개가 주어졌을 때 i 번째 수에서 j 번째 수까지의 합을 구하는 프로그램을 작성하시오.

입력

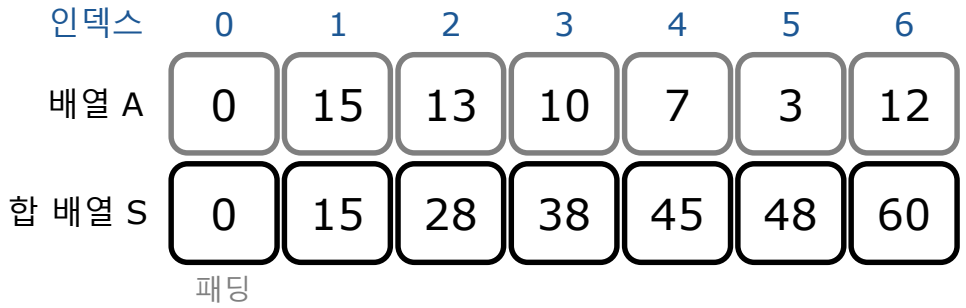
1번째 줄에 수의 개수 N(1 ≤ N ≤ 100,000), 합을 구해야 하는 횟수 M(1 ≤ M ≤ 100,000), 2번째 줄에 N개의 수가 주어진다. 각 수는 1,000보다 작거나 같은 자연수다. 3번째 줄부터는 M개의 줄에 합을 구해야 하는 구간 i와 j가 주어진다.

출력

총 M개의 줄에 입력으로 주어진 i 번째 수에서 j 번째 수까지의 합을 출력한다.

구간 합

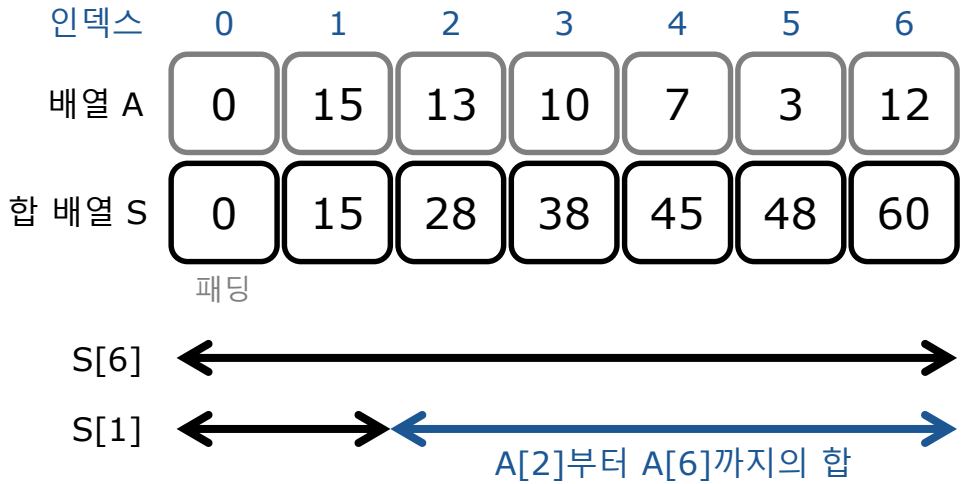
- 구간 합은 합 배열을 이용하여 시간 복잡도를 줄이기 위해 사용하는 알고리즘
- 배열 A가 있을 때 합 배열 S는 다음과 같이 정의
  - S[i] = A[0] + A[1] + A[2] + … + A[i-1] + A[i] // A[0]부터 A[i]까지의 합



- 합 배열을 만드는 공식
  - S[i] = S[i-1] + A[i]

구간 합

- A[i]부터 A[j]까지 배열의 합
  - 합 배열 없이 구하는 경우, 최악의 경우는 i가 1, j가 N인 경우로 시간 복잡도는 O(N)
  - 합 배열을 사용하면, S[j] – S[i-1]로 시간 복잡도는 O(1)



연습문제

- 백준 온라인 저지 – 11659 구간 합 구하기4
  - 1차원 배열의 구간 합

연습문제

- 백준 온라인 저지 – 11660 구간 합 구하기5
  - 2차원 배열의 구간 합

연습문제

- SW Expert Academy – 1210 Ladder1
  - 2차원 배열 순회 및 처리

연습문제

- SW Expert Academy – 1954 달팽이 숫자
  - 2차원 배열 순회 및 처리

투 포인터

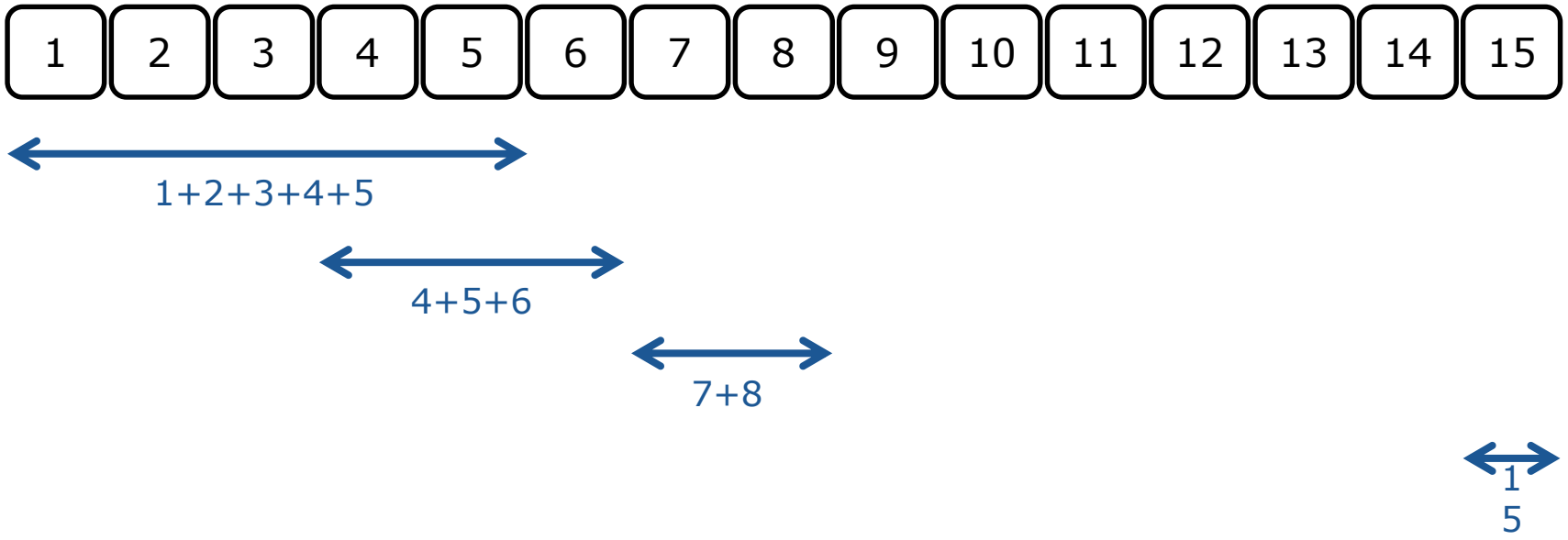
연속된 자연수의 합 구하기

시간 제한 **2초** | 난이도 **실버 V** | 백준 온라인 저지 **2018번**

어떠한 자연수 N은 몇 개의 연속된 자연수의 합으로 나타낼 수 있다. 당신은 어떤 자연수 N($1 \le N \le 10,000,000$)을 몇 개의 연속된 자연수의 합으로 나타내는 가짓수를 알고 싶다. 이때 사용하는 자연수는 N이어야 한다. 예를 들어 15를 나타내는 방법은 15, 7 + 8, 4 + 5 + 6, 1 + 2 + 3 + 4 + 5이다. 반면, 10을 나타내는 방법은 10, 1 + 2 + 3 + 4이다. N을 입력받아 연속된 자연수의 합으로 나타내는 가짓수를 출력하는 프로그램을 작성하시오.

투 포인터

- 자연수 N은 최대 10,000,000이고, 제한 시간은 2초
  - O(nlogn)의 시간 복잡도 알고리즘은 제한 시간 초과
  - O(n)의 시간 복잡도 알고리즘을 사용
- N이 15인 경우 연속된 자연수의 합이 15인 경우는 1+2+3+4+5, 4+5+6, 7+8, 15 총 4가지 경우 존재

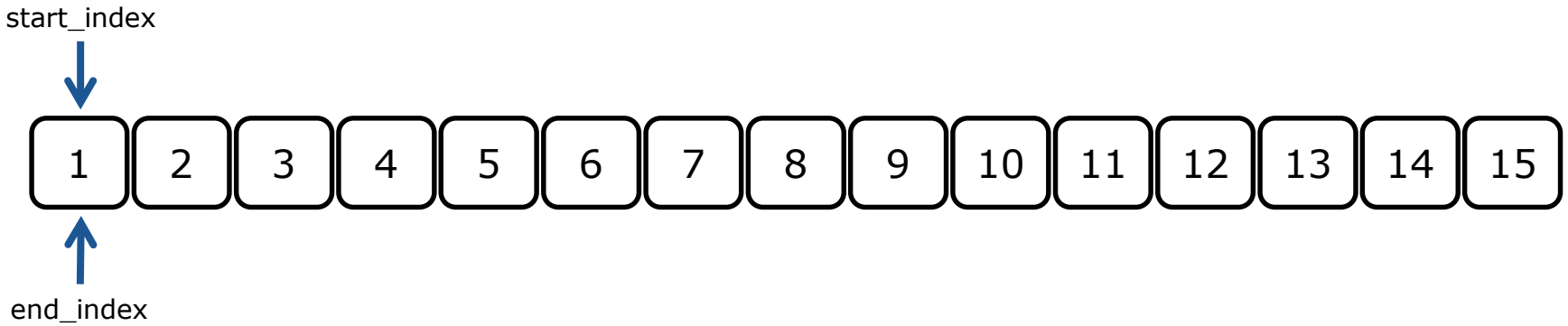


투 포인터

- 알고리즘 설계하기
  - 연속된 자연수의 합이 N이 되어야 하기 때문에 각 연속 합의 시작이 각 자연수 별로 존재하거나 존재하지 않음

투 포인터

- 알고리즘 설계하기
  - 두 개의 포인터를 활용하여 시작 인덱스와 끝 인덱스를 지정하여 합이 N이 되는 경우의 수를 구합니다.



투 포인터

- pseudo-code

```
N // 목표 합
start=1, end=1 // 시작, 끝 자연수(포인터)
count=0, sum=1 // 개수, start부터 end까지 연속된 자연수의 합

while end <= N
    if sum == N then count++, start++, end++, sum 변경
    else if sum < N then end++, sum 변경
    else if sum > N then start++, sum 변경
```

- 시간 복잡도 O(n)

슬라이딩 윈도우

- 백준 온라인 저지 – 12891 DNA 비밀번호