투 포인터 알고리즘

Two Pointers Algorithm

2023-06-21

고 건

목차 Index

01
Two Pointers Algorithm

02
Sliding Window Algorithm

03
Practice Problem Solving

01

투 포인터 알고리즘

Two Pointers Algorithm

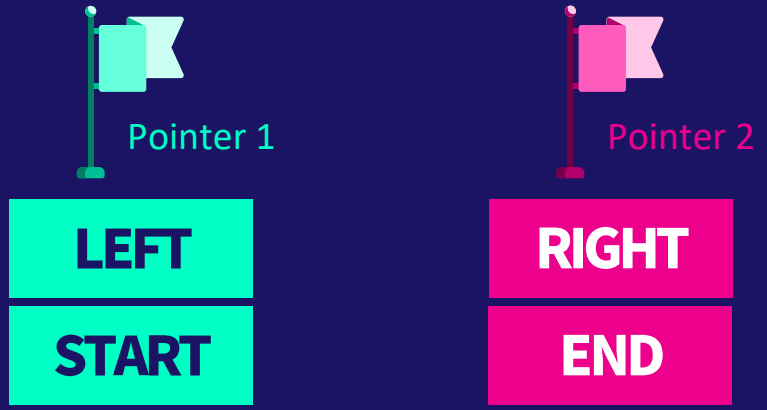
투 포인터 알고리즘

(Two Pointers Algorithm)

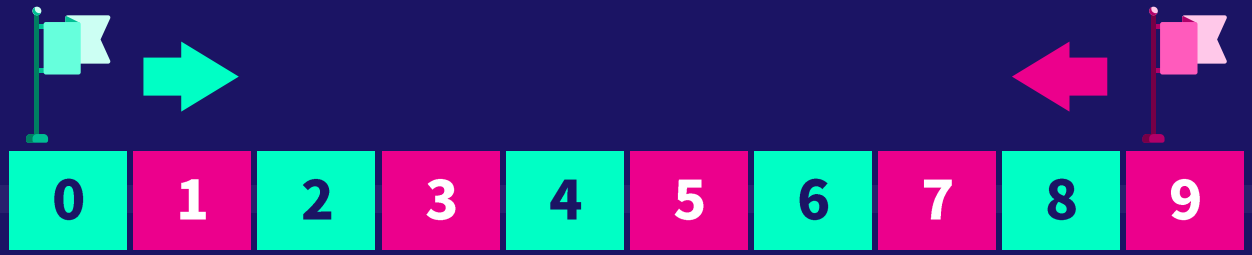
"통상 배열 또는 문자열의 인덱스(Index)를
나타내는 **두 개의 서로 다른 포인터(Pointer)**를
사용하여 문제를 해결하는 알고리즘"

포인터 Pointer

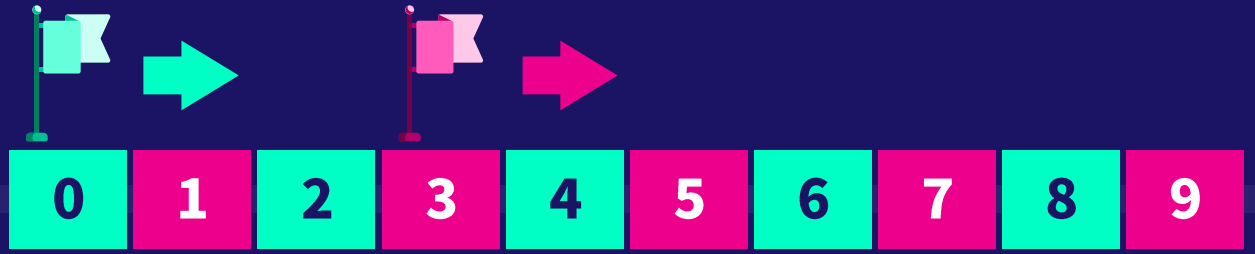
많은 프로그래밍 언어에서 포인터는 다른 객체의 메모리 주소 값을 저장하는 객체를 의미하는 바, 알고리즘 문제에서는 **배열(Array)이나 리스트(List), 문자열(String)의 인덱스를 저장한 객체**를 의미



접근법 Approach



A 각 포인터를 각각 시작 인덱스와 끝 인덱스에 위치해 두고,
서로 반대 방향으로 움직여 두 포인터가 만날 때까지 탐색하는 방법



B 각 포인터를 시작 인덱스에 위치해 두고, 특정한 포인터가 끝
인덱스에 도달할 때까지 탐색하는 방법

시간 복잡도 Time Complexity

$O(n)$

최악의 경우 두 포인터가 각각 인덱스의 처음부터 끝까지 순회한다고 하더라도
선형 시간(Linear Time) 내에 수행 가능

두 수의 합

문자열 뒤집기

02

슬라이딩 윈도우 알고리즘

Sliding Window Algorithm

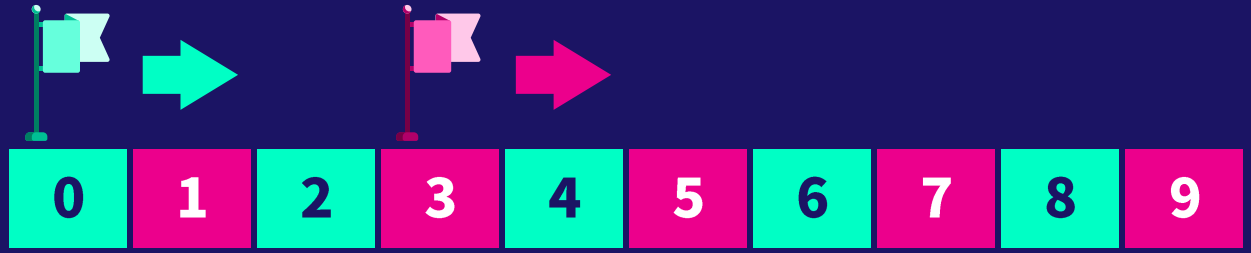
슬라이딩 윈도우 알고리즘

(Sliding Window Algorithm)

"고정된 크기의 윈도우(Window)를 사용하여
문제를 해결하는 알고리즘"

윈도우 Window

슬라이딩 윈도우 알고리즘에서 윈도우는 **고정된 크기의 구간**으로, 보통 **부분 배열, 부분 리스트, 부분 문자열**을 의미하며, 윈도우의 크기는 문제의 요구사항에 따라 결정



일반적으로 **고정된 크기의 윈도우를 왼쪽에서 오른쪽으로 한 칸씩 이동**

시간 복잡도 Time Complexity

$O(n)$

인덱스의 처음부터 끝까지 한 칸씩 이동했을 때
새로운 요소를 추가하고 제거하는 과정은 상수 시간에 불과해 결국
선형 시간(Linear Time) 내에 수행 가능

고정된 길이의 부분 배열의 합 구하기

문자열 패턴 횟수 세기

03

예제 풀이

Practice Problem Solving

1

BOJ 15565
귀여운 라이언

00:30:00

해설 Solution

```
if (lionDolls.size() < k) {
    out.write("-1");

} else {

    int minSize = Integer.MAX_VALUE;

    for (int idx = 0; idx < lionDolls.size() - k + 1; idx++) {
        int start = lionDolls.get(idx);
        int end = lionDolls.get(idx + k - 1);
        int size = end - start + 1;
        minSize = Math.min(size, minSize);
    }
    out.write(String.valueOf(minSize));
}
```

Thank You for Your Attention!

경청해 주셔서 감사합니다!

https://github.com/Crassula1994