악당 분석 보고서

-투 포인터-

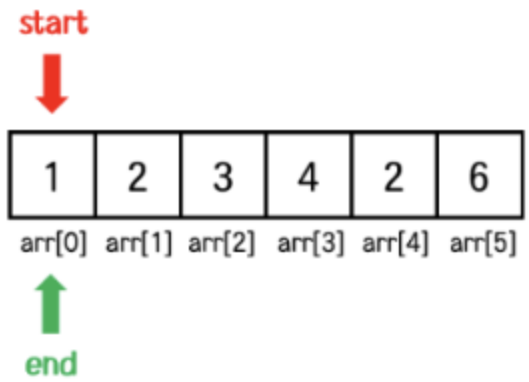
ALGORANGER RED

투포인터

배열에서 두개의 포인터를 이용해 원하는 값을 찾는 알고리즘

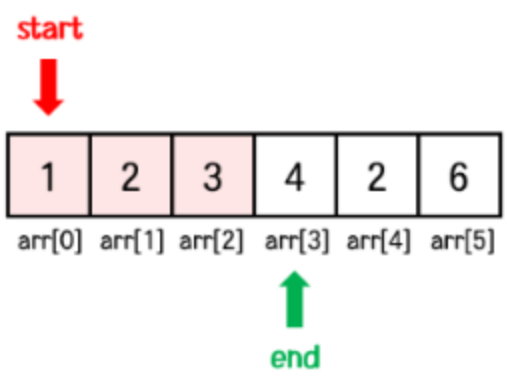
1. 두 포인터가 같은 방향으로 진행하는 방법 - BOJ 2003
2. 두 포인터가 다른 방향으로 진행하는 방법 - BOJ 2470

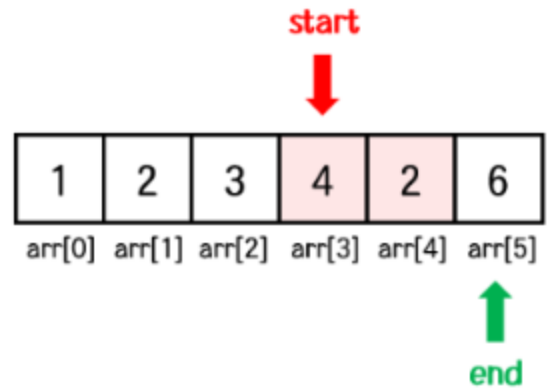
두 포인터가 같은 방향으로 진행

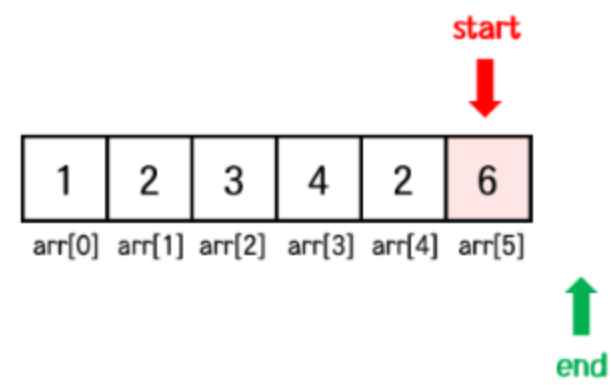


배열의 크기가 6인 배열 arr의 원소가 순서대로 1 2 3 4 2 6 이고,
배열에서 부분합(M)이 6이 되는 경우의 수를 구하여라.

start = 0, end = 0, sum = 0, cnt = 0







두 포인터가 같은 방향으로 진행

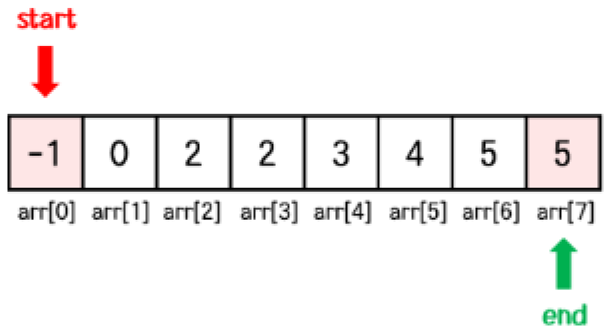
```java
public static int count(int[] arr) {
    int start = 0, end = 0;
    int sum = 0;
    int cnt = 0;

    while (start < arr.length) {
        if (sum == M) {
            cnt++;
            sum -= arr[start++];
        } else if (sum > M || end == N)
            sum -= arr[start++];
        else
            sum += arr[end++];
    }
    return cnt;
}
```

배열에서 연속하는 부분합이 특정값 M과 일치하는 경우의 수를 구할 때

1. arr[start] ~ arr[end-1]까지의 구간합이 M과 같은 경우
   카운트를 하나 증가
2. arr[start] ~ arr[end-1]까지의 구간합이 M보다 크거나, end = arr.length 인 경우
   구간합에서 arr[start] 값을 빼고, start 포인터를 하나 증가
   -> end = arr.length 인 경우에는 start만 하나씩 증가
3. arr[start] ~ arr[end-1]까지의 합이 구간합 M보다 작은 경우
   구간합에서 arr[end] 값을 더해주고, end 포인터를 하나 증가시킴

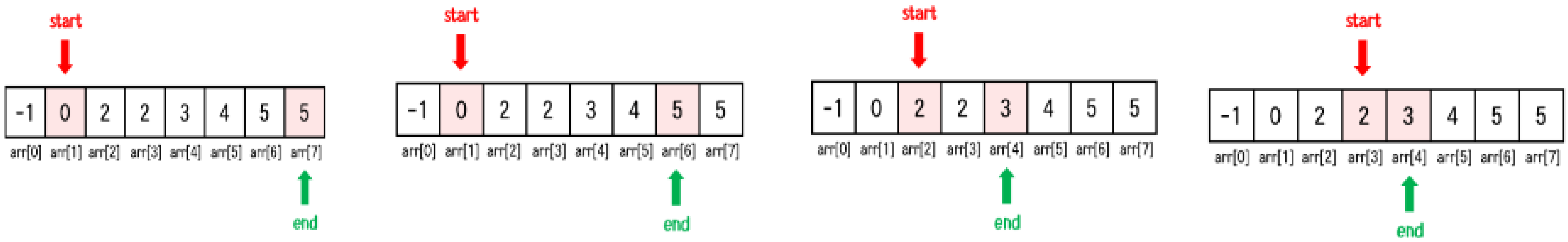
두 포인터가 다른 방향으로 진행



X = 5이고, 배열은 차례대로 -1 0 2 2 3 4 5 5.

(배열은 정렬된 상태.)

start = 0, end = 7, sum = 4, cnt = 0



```java
public static int count(int X) {

    int count = 0;
    int s = 0, e = arr.length - 1;

    while (s < e) {

        int sum = arr[s] + arr[e];

        if (sum == X) {
            int a = arr[s];
            int b = arr[e];
            int acnt = 0, bcnt = 0;

            while (s < arr.length && arr[s] == a) {
                s++;
                acnt++;
            }
            while (e >= 0 && arr[e] == b) {
                e--;
                bcnt++;
            }
            count += acnt * bcnt;
        } else if (sum > X) {
            e--;
        } else if (sum < X) {
            s++;
        }
    }

    return count;

}
```

두 포인터가 다른 방향으로 진행

배열에서 두 원소의 합이 어떠한 값 X와 일치하는 경우의 수를 구할 때

*배열이 정렬된 상태에서 가능

1. sum = X 일 경우

arr[start]와 동일한 연속하는 값의 개수를 카운트(acnt),

arr[end]와 동일한 연속하는 값의 개수를 카운트(bcnt)

그리고 이 둘을 곱하여 최종 cnt에 더함.

cnt += acnt*bcnt

2. sum > X 인 경우

end 포인터를 감소시킴

3. sum < X 인 경우

start 포인터를 증가시킴

4. start<end 을 만족할 때까지 위 과정 반복