- 문제: N, M, A[N]이 주어질 때
  A의 부분합이 M이 되는 것의 개수.

- 접근: 선형적인 '투 포인터' 문제. (M과 A의 모든 원소가 자연수!!)

  ① 만약 현재 부분합이 M 이상이거나, 이미 e==N 이면
  `sum -= A[s++];`
  ② 그렇지 않다면
  `sum += A[e++];`
  ③ (sum == M) 이라면
  `ans++;`
  `sum -= A[s++];`

* [s, e)의 부분합은 인덱스 선형으로 '증가'만 하므로
  ↳ s==e이면 길이 0. (e++될 때 길이가 1이되며 e가 원래있던자리 더해줌)
  sum에 적절히 +, - 하며 반복하면 간단.

* sum에 -= A[s++]; 해주면
  ↳ 원래 있던자리것 빼주고 ++되니까 [s, e) 만족.
  sum에 += A[e++]; 해주면
  ↳ 원래 자리 더해주고 ++되니까 [s, e) 만족.
  ([s,e) 만족했을 테니 미포함이었음)

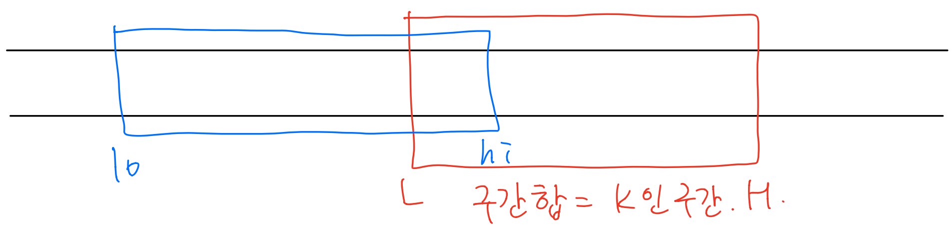
- 시간 복잡도: 각 반복마다 s 또는 e가 1씩 증가하므로
  최대 2N번. ⟶ O(n).
  · 부분합 계산 또한 O(1) 만에 +, - 해주니 영향 X.

- 구현

```
while(1) {
    if(sum >= M) sum -= A[lo++];
    else if(hi==N) break; // 위의 if 만족X → sum < M 이라는 뜻.
                          //  이후 sum은 항상 M보다 작다.
    else sum+=A[hi++];

    if(sum==M) ans++;
}
```

- 증명 : lo, hi가 선형으로 '감소' 없이 '증가'만 하는데 모든 정답을 항상 찾을 수 있을까?



· 현재 투 포인터가 [lo, hi) 구간을 가리킬 때, 앞으로 [L, H)를 만나지 않고 지나치지 못함을 증명하면 됨.

· 선형으로 증가하기 때문에 지나치기 위해 두 가지 조건 존재.

① hi = H 되기 전에 lo > L이 된다.

↳ 선형 증가하기 때문에 lo = L인 시점이 존재. 이때, sum < K 이므로 무조건 hi가 증가할것.

② lo = L 되기 전에 hi > H가 된다.

↳ lo < L && hi = H인 시점에서는 sum > K 이므로 무조건 lo가 증가.

⇒ 항상 인덱스, sum, K가 '자연수' 이기에 가능!