Algorithm Study

11.18.2024 김홍주

#1522문자열 교환

Q. 문자열 교환

a와 b로만 이루어진 문자열이 주어질 때, **a를 모두 연속**으로 만들기 위해서 필요한 **교환**의 회수를 **최소**로 하는 프로그램을 작성하시오.

이 **문자열은 원형**이기 때문에, 처음과 끝은 서로 인접해 있는 것이다.

예를 들어, aabbaaabaaba이 주어졌을 때, 2번의 교환이면 a를 모두 연속으로 만들 수 있다.

<입력 조건 >

- 첫째 줄에 문자열이 주어진다.
- 문자열의 길이는 최대 1,000이다.

<출력 조건>

- 첫째 줄에 필요한 교환의 회수의 최솟값을 출력한다.

[예제]

abababababababa	*3*
ba	*0*
aaaabbbbba	*0*
abab	*1*
aabbaaabaaba	*2*
aaaa	*0*

Hint

Idea 1. a 연속된 문자열을 만들기 위한 교환횟수 구하기
Idea 2. 원형 문자열

Hint

1. 슬라이싱 윈도우 # Brute force ,# 2 pointer

- 고정된 사이즈의 윈도우를 이동하여 윈도우 내에 있는 데이터가 특정한 조건을 만족하도록 하는 알고리즘
- 구현 :
  i. 두 포인터를 일정한 간격으로 이동 like 2 pointers
  ii. 슬라이싱 이전, 이후를 비교시 교집합 부분은 공유하고 차이가 나는 양쪽 끝 원소만 갱신된다(pop, add)

- 교체후 연속된 문자열 a 의 길이 = 문자열 속 a 총 개수

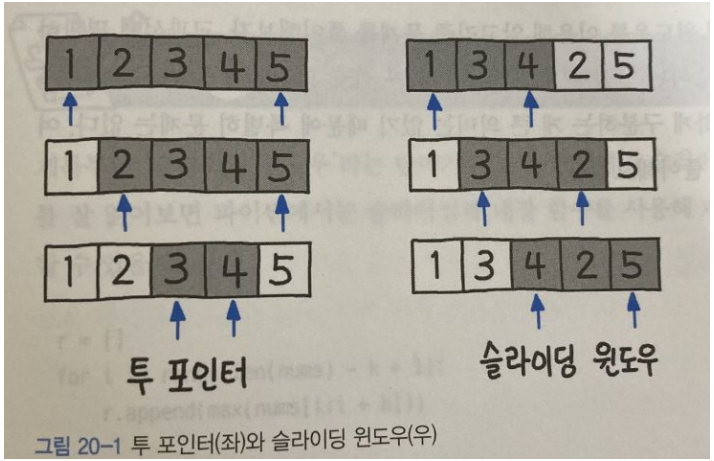

그림 20-1 투 포인터(좌)와 슬라이딩 윈도우(우)

Hint

1. 슬라이싱 윈도우 # Brute force ,# 2 pointer

- 고정된 사이즈의 윈도우를 이동하여 윈도우 내에 있는 데이터가 특정한 조건을 만족하도록 하는 알고리즘
- 구현 :
  i. 두 포인터를 일정한 간격으로 이동 like 2 pointers
  ii. 슬라이싱 이전, 이후를 비교시 교집합 부분은 공유하고 차이가 나는 양쪽 끝 원소만 갱신된다(pop, add)

- 교체후 연속된 문자열 a 의 길이 = 문자열 속 a 총 개수
  -> 슬라이싱 한 문자열 내부 b 와 외부의 a 를 교환
  -> 연속된 a 문자열 생성

  따라서 a 의 개수 = window 사이즈

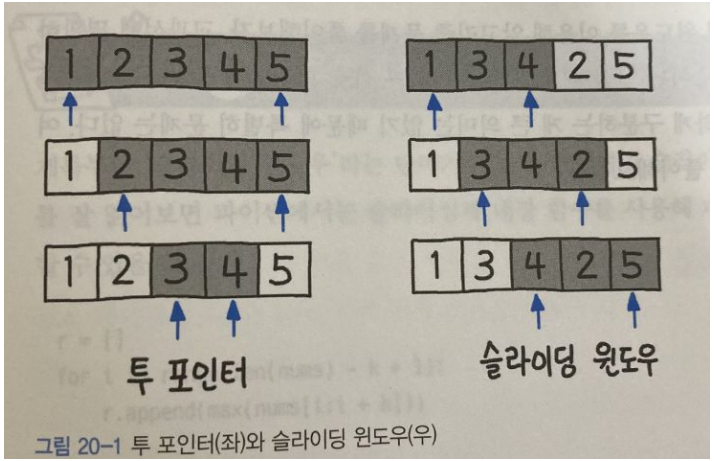

그림 20-1 투 포인터(좌)와 슬라이딩 윈도우(우)

Hint

1. 슬라이싱 윈도우 # Brute force ,# 2 pointer

- 고정된 사이즈의 윈도우를 이동하여 윈도우 내에 있는 데이터가 특정한 조건을 만족하도록 하는 알고리즘
- 구현 :
  i. 두 포인터를 일정한 간격으로 이동 like 2 pointers
  ii. 슬라이싱 이전, 이후를 비교시 교집합 부분은 공유하고 차이가 나는 양쪽 끝 원소만 갱신된다(pop, add)

- 교체후 연속된 문자열 a 의 길이 = 문자열 속 a 총 개수
  -> 슬라이싱 한 문자열 내부 b 와 외부의 a 를 교환
  -> 연속된 a 문자열 생성

  a 의 개수 = window 사이즈 & 교환횟수 = b의 개수

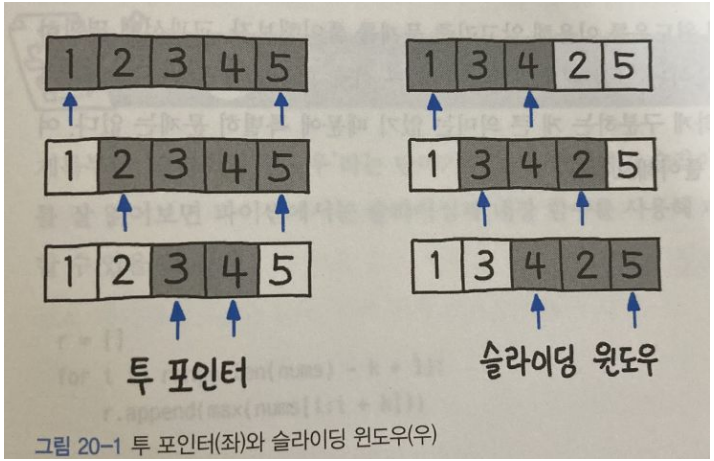

그림 20-1 투 포인터(좌)와 슬라이딩 윈도우(우)

Solution 다른 사람 풀이

a개수 : 8개
aabbaaabaaba => aaaaaaaabbbb

aabbaaabaaba -> 3

aabbaaabaaba -> 3

Solution

1, a의 개수와 동일한 길이의 연속된 문자열 슬라이싱

교체 후 연속된 a 문자열 길이 = 총 a의 개수 = window 사이즈 계산하기

2. 교환 횟수 = 슬라이싱 문자열 속 b의 개수

슬라이싱 한 문자열 속 b 와 외부의 a 교환

3. 원형 문자열

- 기존 문자열에서 “window-1” 길이의 문자열을 왼쪽에 추가
- idex 범위 넘어가면 전체 길이만큼 빼주기
  - `if index>len(arr)-1:index-=len(arr)`

Solution 다른 사람 풀이

(1)

```
# 1. a의 개수 = sliding window 크기
words= input()
window_size = words.count("a")
result = 999999999

# 원형 문자열
words += words[0:window_size-1]
# 슬라이싱 된 문자열 속 b의 개수 최소값
for start in range(len(words) - (window_size-1)):
    result = min(result, words[start: start+ window_size].count("b"))

print(result)
```

(2)

```
#2
arr=input()
result=[]
size=arr.count('a') #윈도우 크기

for i in range(len(arr)):
    b_count=0
    for j in range(size): #윈도우내 b의 개수 = 교환횟수
        index=i+j
        #index가 리스트를 넘어간다면 -=리스트크기 해주기-> 원형 유지
        if index>len(arr)-1:index-=len(arr)
        if arr[index]=="b":b_count+=1
    result.append(b_count)

print(min(result))
```

추가 문제 *#문자열, # 2포인터*

[백준#1806.부분합]

- 문제
https://www.acmicpc.net/problem/1806

-해설
https://aia1235.tistory.com/46

```
"""
문제 : # 1806부분합
https://www.acmicpc.net/problem/1806
해설 :
https://aia1235.tistory.com/46

"""
import sys
N ,S = map(int, sys.stdin.readline().split())
arr = list(map(int, input().split()))

start , end = 0, 0
min_length = 100000
partial_sum = arr[0]
while start <= end :
    if partial_sum >= S :  # S보다 큰 부분합인 경우
        min_length = min(min_length , end - start +1 ) # 최소 길이 업데이트 확인
        partial_sum -= arr[start]  # start +1 이동
        start += 1
    else : #partial_sum < S :
        end += 1
        if end < N :
            partial_sum += arr[end]
        else :  #반복문 끝
            break


if min_length == 100000 : # 수열 끝 -> 조건 충족 x
    print(0)
else :
    print(min_length)
```