단어중복없는 가장긴 문자열
(Longest Substring Without Repeating Characters)

최대 2 개의 고유 문자가있는 가장 긴 부분 문자열
(Longest Substring with At Most Two Distinct Characters)

문자열에서 모든 아나그램 찾기
(FindAllAnagramInAString)

TwoPointer (시험에 나오는거 위주로)

1) 같은 방향:
start 포인트가 시작, end 포인트를 저장했다가 이용
문제) MoveZero

2) 반대 방향
while문에서 간격이 좁아지는 형태
문제) Trapping Rain Water

3) 슬라이딩 윈도우(일정 크기 간격을 유지)
문제) Longest Substring Without Repeating Characters

4) Prefix Sum
문제) Subarray Sum

5) Cycle Finding
문제) Linked List Cycle

단어중복없는 가장긴 문자열(Longest Substring Without Repeating Characters)

설명

String s가 주어집니다.
중복이 없는 가장 긴 substring 문자열을 리턴하세요

입출력

Input: s = "abcabcd"
Output: 4
Explanation: "abcd"

Input: s = "pwwkea"
Output: 4
Explanation: "wkea"

Input: s = "aaaaa"
Output: 1
Explanation: "a"

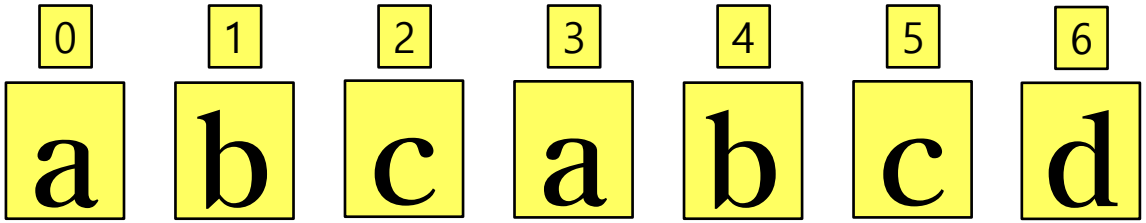
Input: s = ""
Output: 0

단어중복없는 가장긴 문자열(Longest Substring Without Repeating Characters)

제한사항

0 <= s.length <= $5 * 10^4$
s consists of English letters

문제분석



3

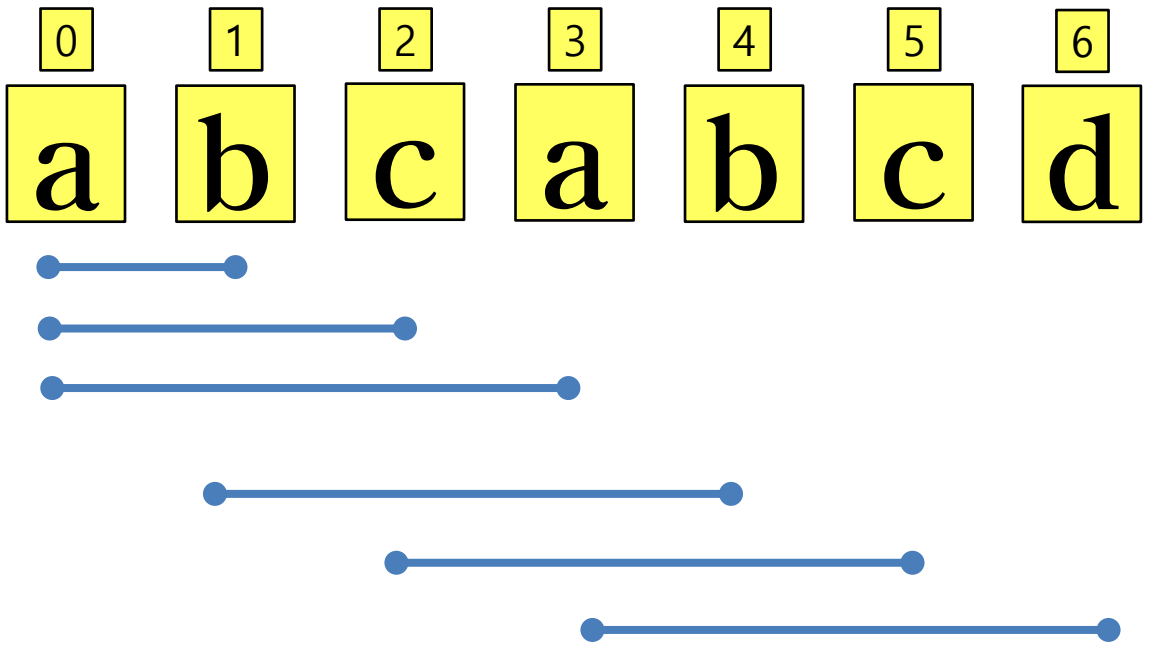
4

right-left

1. 첫번째 포인터(right)가 0번방부터 시작한다.
2. 중복되는 문자 발생시 다시 시작한다.
3. 중복 발생전 max =3 (right-left) 갱신한다
4. 중복 발생 이후 부터 카운트 한다.

1. 중복되는 부분 저장 방식?

코딩화



right 1 - light 0 :1
right 2 - light 0 :2
right 3 - light 0 :3
right 4 - light 1 :3
right 5 - light 2 :3
right 6 - light 3 :3
right 7 - light 3 :4

1

Key	Value
a	2
b	1
c	1

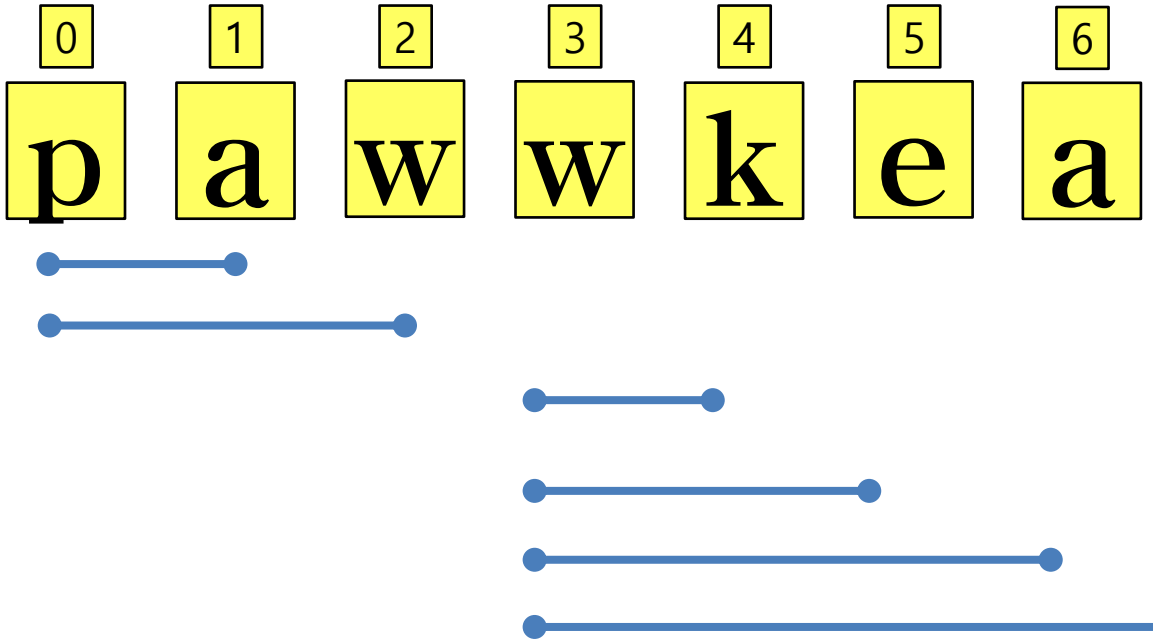
2

right-left

3

Math.max(max, right - left);

코딩화



right 1 left 0 :1
right 2 left 0 :2
right 3 left 0 :3
right 4 left 3 :1
right 5 left 3 :2
right 6 left 3 :3
right 7 left 3 :4

1

Key	Value
w	2
p	1
a	1

counter >1

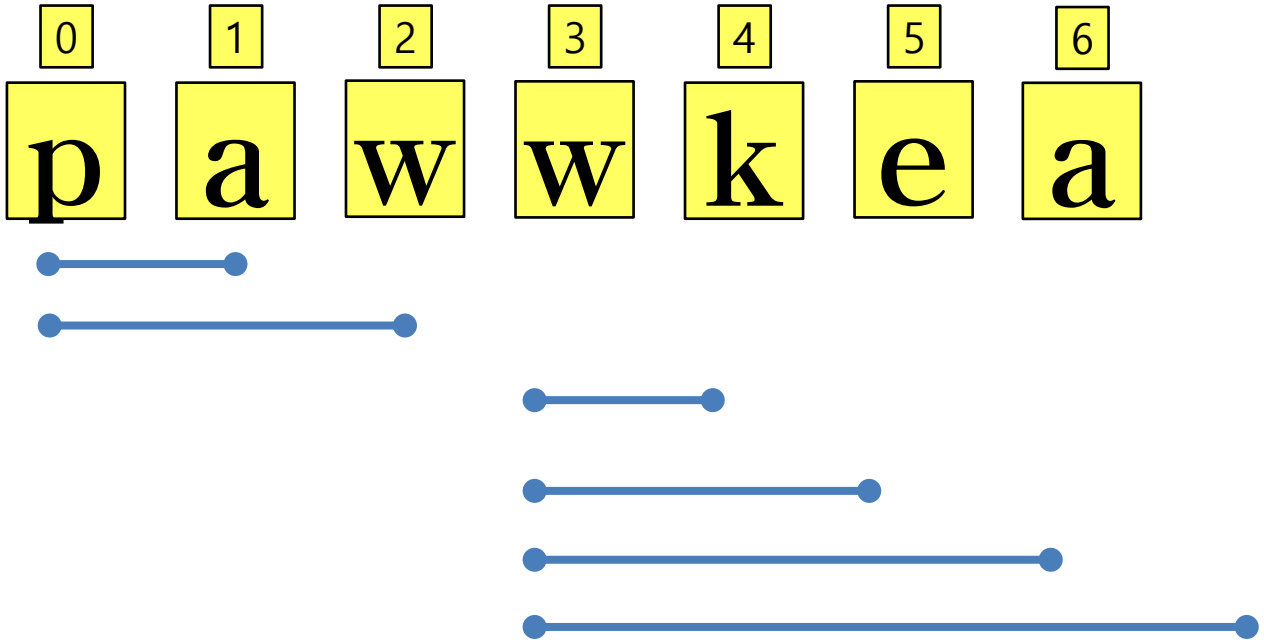
2

right-left

3

Math.max(max, right - left);

코딩화



right 1 left 0 :1
right 2 left 0 :2
right 3 left 0 :3
right 4 left 3 :1
right 5 left 3 :2
right 6 left 3 :3
right 7 left 3 :4

1

{ 1,0,0,0,1,0,0,0....1...1.}

2

right-left

3

Math.max(max, right - left);

시간복잡도/공간복잡도 계산

시간복잡도

1. 대상(Source) : 문제에서 입력받은 파라미터(array 등) (속도)
   Time Complexity : O(N)
   대상 : String s
   이유 : s의 길이만큼 실행

공간복잡도

2. 대상(Source) : 실제 사용되는 저장 공간을 계산(메모리 사용량)
   예) 프로그램을 실행 및 완료하는데 필요한 저장공간
   Space Complexity : O(n)
   대상 : Map<Character, Integer> map = new HashMap<>();
   이유 : s의 단어를 한개씩 키로 저장

참고

O(1) : 스택, 큐,Map
O(n) : for문 => 데이터를 한번씩 다 호출하니까 (제일 많음)
O(log N) : sort, prirotiyQueue, binary Search Tree, Tree
O(Klog N) : k번만큼 소팅하는 경우
O(n^2) : 이중for문
O(m*n) : 이중for문인데, n이 다른경우bfs,dfs 류 (예 n=100 인데 m=5인경우)

최대 2 개의 고유 문자가있는 가장 긴 부분 문자열
(Longest Substring with At Most Two Distinct Characters)

설명

String s가 주어집니다.
최대 두 개의 고유 문자 를 포함하는 가장 긴 부분 문자열의 길이를 리턴하세요

입출력

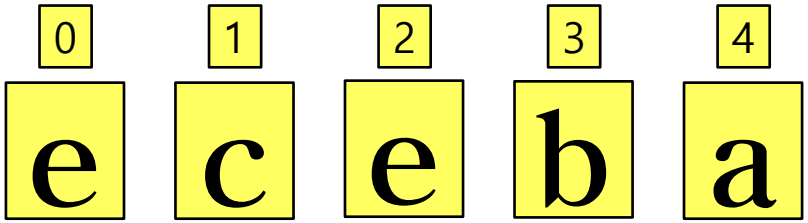
Input: s = "eceba"
Output: 3
Explanation: "ece"

Input: s = "ccaabbb"
Output: 5
Explanation: "aabbb"

제한사항

1 <= s.length <= 10^4
s consists of English letters.

문제분석



3

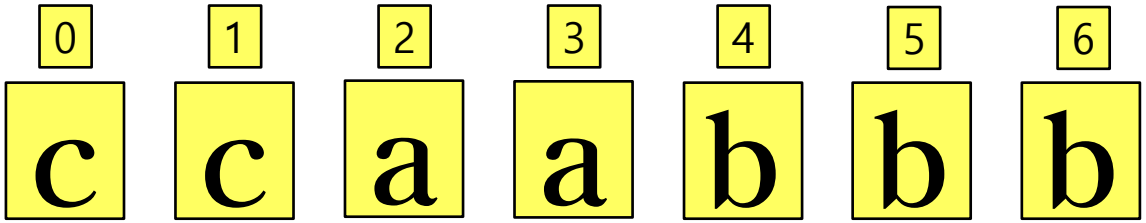
2

right-left

1. 첫번째 포인터(right)가 0번방부터 시작한다.
2. 단어2개 까지 허용
3. 단어가 3개가 되는 시점 중지
4. max =3 (right-left) 갱신한다
5. 이후 부터 카운트 한다.

1. 중복되는 부분 저장 방식?

문제분석



4

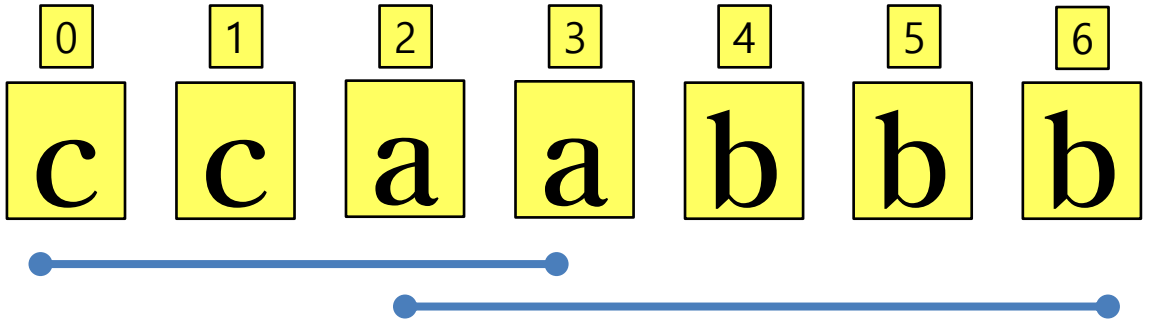
5

right-left

1. 첫번째 포인터(right)가 0번방부터 시작한다.
2. 단어2개 까지 허용
3. 단어가 3개가 되는 시점 중지
4. max =3 (right-left) 갱신한다
5. 이후 부터 카운트 한다.

1. 중복되는 부분 저장 방식?

코딩화



right 1 - left 0 :1
right 2 - left 0 :2
right 3 - left 0 :3
right 4 - left 0 :4
right 5 - left 2 :3
right 6 - left 2 :4
right 7 - left 2 :5

1

Key	Value
a	2
b	1
c	2

counter=3

2

right-left

3

Math.max(max, right - left);

시간복잡도/공간복잡도 계산

시간복잡도

1. 대상(Source) : 문제에서 입력받은 파라미터(array 등) (속도)
   Time Complexity : O(N)
   대상 : String s
   이유 : s의 길이만큼 실행

공간복잡도

2. 대상(Source) : 실제 사용되는 저장 공간을 계산(메모리 사용량)
   예) 프로그램을 실행 및 완료하는데 필요한 저장공간
   Space Complexity : O(1)
   대상 : Map<Character, Integer> map = new HashMap<>();
   이유 : 키를3개만 저장하므로

참고

O(1) : 스택, 큐,Map
O(n) : for문 => 데이터를 한번씩 다 호출하니까 (제일 많음)
O(log N) : sort, prirotiyQueue, binary Search Tree, Tree
O(Klog N) : k번만큼 소팅하는 경우
O(n^2) : 이중for문
O(m*n) : 이중for문인데, n이 다른경우bfs,dfs 류 (예 n=100 인데 m=5인경우)

문자열에서 모든 아나그램 찾기 (FindAllAnagramInAString)

설명

두 개의 문자열이 주어집니다. String s, p
p의 문자열이 s의 문자열에 특정인덱스에서부터 모든 아나그램 찾아서 인덱스를 리턴하세요

입출력

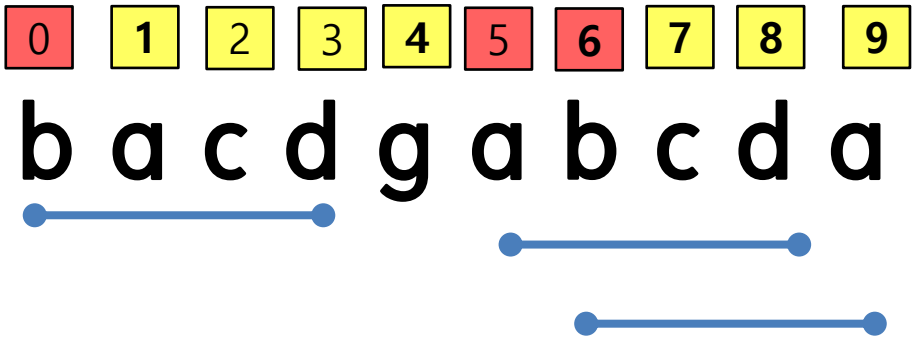
Input: s = "bacdgabcda", p = "abcd"
Output: [0,5,6]
Explanation:
0번 인덱스 bacd
5번 인덱스 abcd
6번 인덱스 bcda

p=abcd는 s에 특정 인덱스에 대입했을때,
아나그램(순서는 상관없이 단어일치) 되는 시점을 반환합니다.

제한사항

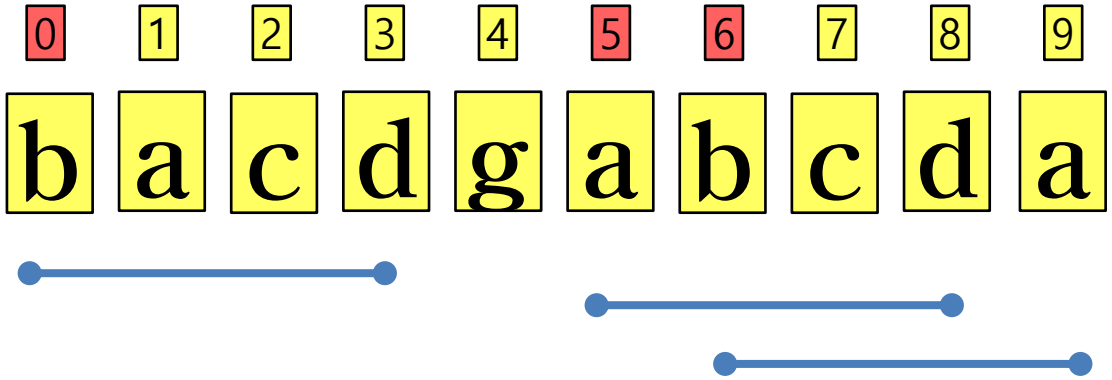
1 <= s.length, p.length <= $3 * 10^4$
s and p consist of lowercase English letters.

문제분석



1. abcd가 소스문자열에서 나오는 부분을 검색한다.
2. 나오는 시점부터 모든문자가 있는지 체크한다.

코딩화



1

Key	Value
a	1
b	1
c	1
d	1

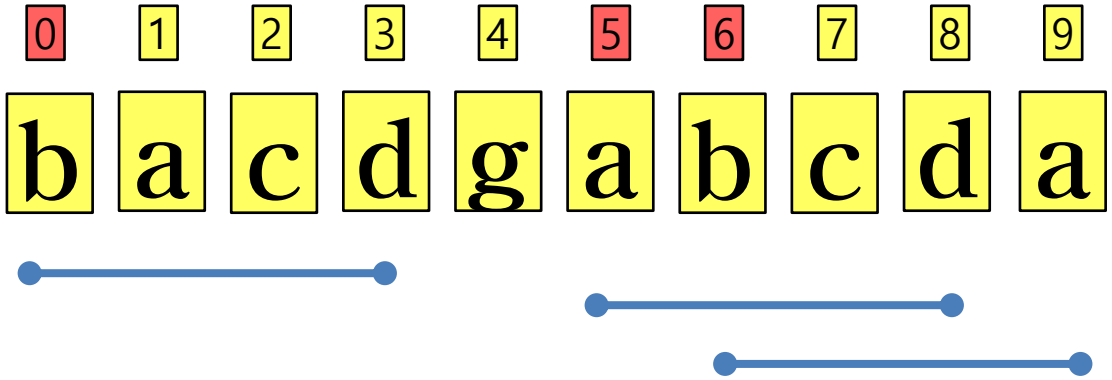
counter=4

2

Key	Value
a	0
b	0
c	0
d	0

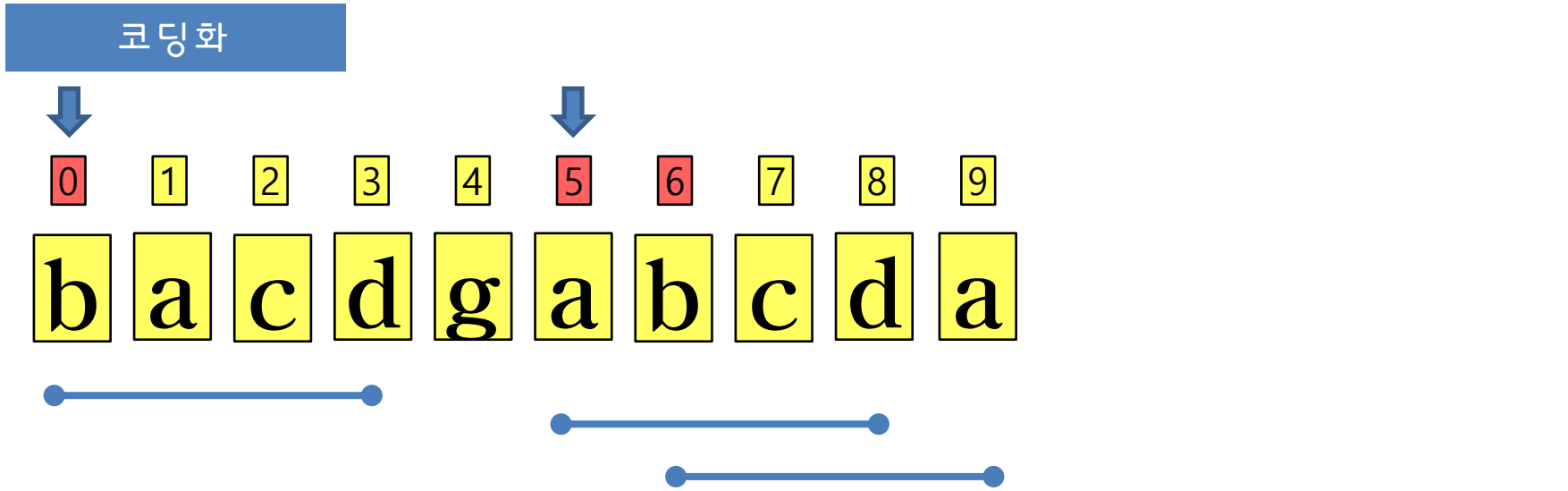
counter=0

코딩화



{ 1,1,1,1,0,0,0,0,0,0....}

배열을 이용한 키값,
알파벳 26개 자리표시

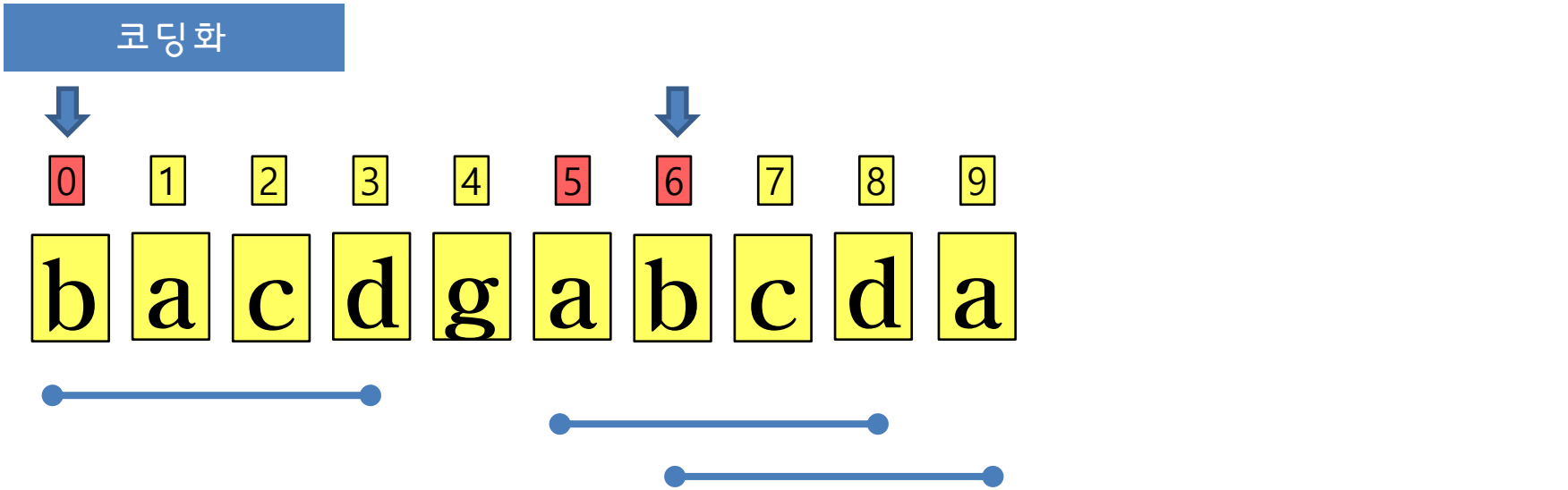


Key	Value
a	0
b	1
c	0
d	0

counter=1

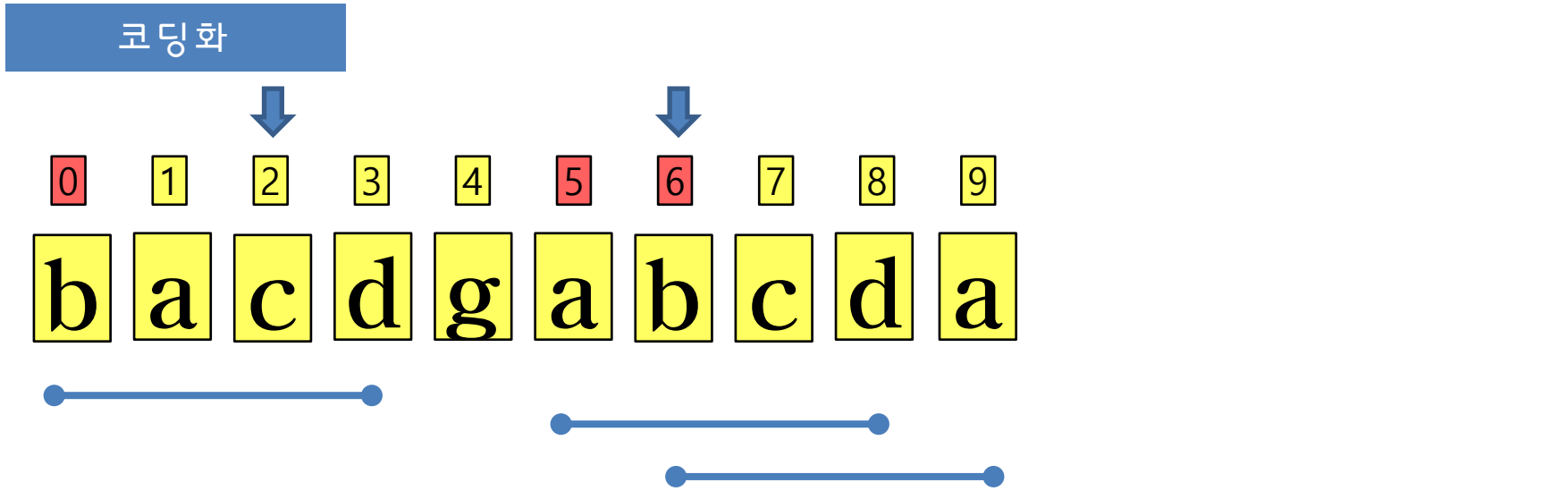
Key	Value
a	-1
b	1
c	0
d	0

counter=1



Key	Value
a	-1
b	0
c	0
d	0

counter=0



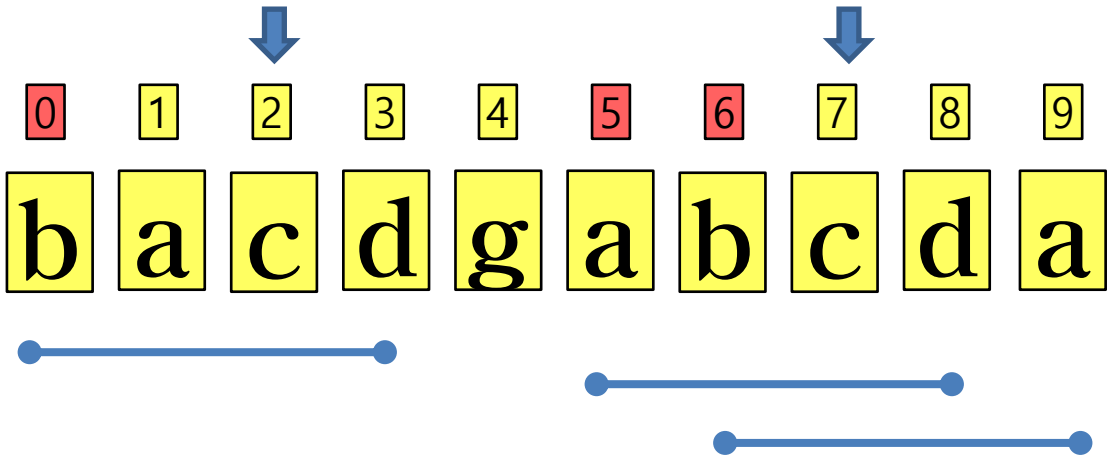
Key	Value
a	0
b	0
c	0
d	0

counter=0

Key	Value
a	0
b	0
c	1
d	0

counter=1

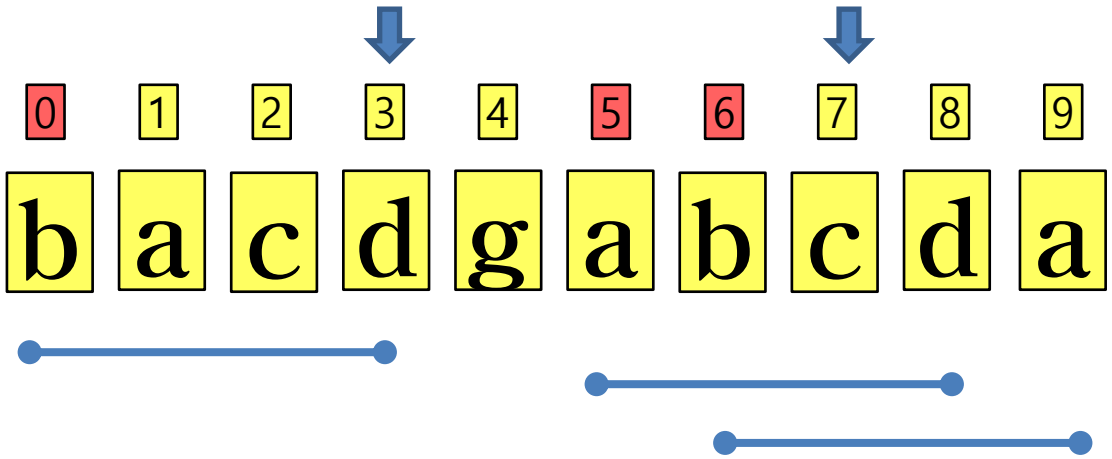
코딩화



Key	Value
a	0
b	0
c	0
d	0

counter=0

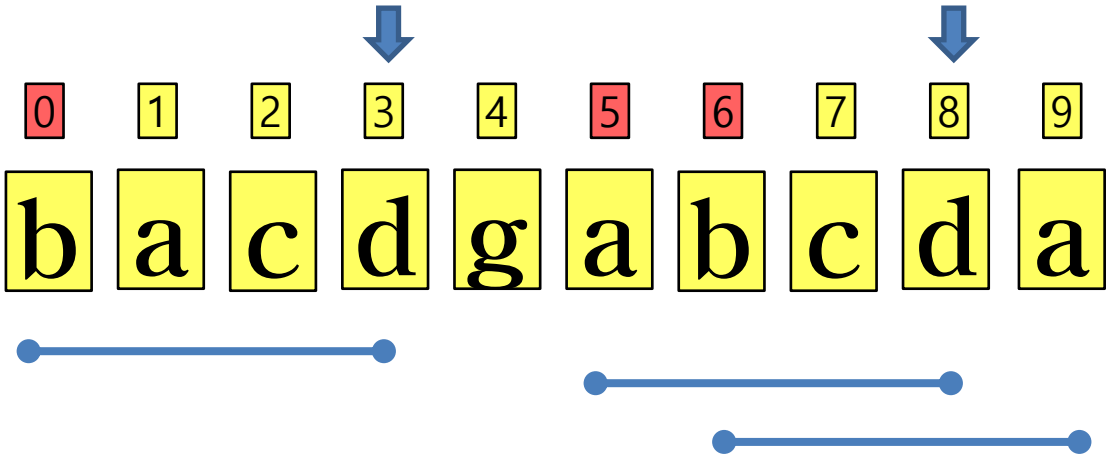
코딩화



Key	Value
a	0
b	0
c	0
d	1

counter=0

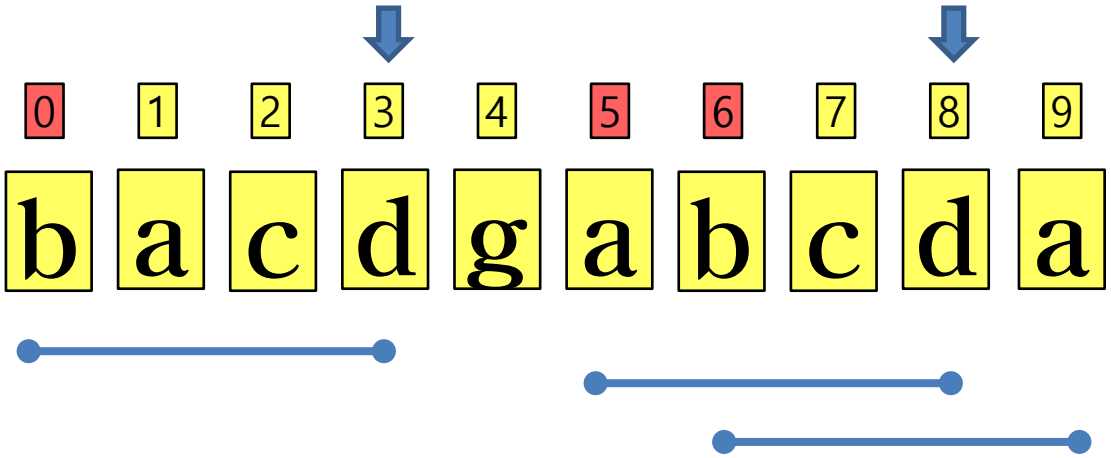
코딩화



Key	Value
a	0
b	0
c	0
d	0

counter=0

코딩화



Key	Value
a	0
b	0
c	0
d	0

counter=0

시간복잡도/공간복잡도 계산

시간복잡도

1. 대상(Source) : 문제에서 입력받은 파라미터(array 등) (속도)
   Time Complexity : O(N*M)
   대상 : String s
   이유 : s의 길이만큼 실행 , p의 길이만큼 실행

공간복잡도

2. 대상(Source) : 실제 사용되는 저장 공간을 계산(메모리 사용량)
   예) 프로그램을 실행 및 완료하는데 필요한 저장공간
   Space Complexity : O(1)
   대상 : int[] pArr = new int[26]
   이유 : 알파벳 26개 한정

참고

O(1) : 스택, 큐,Map
O(n) : for문 => 데이터를 한번씩 다 호출하니까 (제일 많음)
O(log N) : sort, prirotiyQueue, binary Search Tree, Tree
O(Klog N) : k번만큼 소팅하는 경우
O(n^2) : 이중for문
O(m*n) : 이중for문인데, n이 다른경우bfs,dfs 류 (예 n=100 인데 m=5인경우)