

2020

카카오 인턴십 - 보석쇼핑



20181658 이성진



문제 설명

코딩테스트 연습 > 2020 카카오 인턴십 > 보석 쇼핑

[카카오 인턴] 보석 쇼핑

문제 설명

[본 문제는 정확성과 효율성 테스트 각각 점수가 있는 문제입니다.]

개발자 출신으로 세계 최고의 갑부가 된 **어피치**는 스트레스를 받을 때면 이를 풀기 위해 오프라인 매장에 쇼핑을 하러 가곤 합니다.

어피치는 쇼핑을 할 때면 매장 진열대의 특정 범위의 물건들을 모두 싹쓸이 구매하는 습관이 있습니다.

어느 날 스트레스를 풀기 위해 보석 매장에 쇼핑을 하러 간 어피치는 이전처럼 진열대의 특정 범위의 보석을 모두 구매하되 특별히 아래 목적을 달성하고 싶었습니다.

진열된 모든 종류의 보석을 적어도 1개 이상 포함하는 가장 짧은 구간을 찾아서 구매

예를 들어 아래 진열대는 4종류의 보석(RUBY, DIA, EMERALD, SAPPHIRE) 8개가 진열된 예시입니다.

진열대 번호	1	2	3	4	5	6	7	8
보석 이름	DIA	RUBY	RUBY	DIA	DIA	EMERALD	SAPPHIRE	DIA

진열대의 3번부터 7번까지 5개의 보석을 구매하면 모든 종류의 보석을 적어도 하나 이상씩 포함하게 됩니다.

진열대의 3, 4, 6, 7번의 보석만 구매하는 것은 중간에 특정 구간(5번)이 빠지게 되므로 어피치의 쇼핑 습관에 맞지 않습니다.

진열대 번호 순서대로 보석들의 이름이 저장된 배열 `gems`가 매개변수로 주어집니다. 이때 모든 보석을 하나 이상 포함하는 가장 짧은 구간을 찾아서 `return` 하도록 `solution` 함수를 완성해주세요.

가장 짧은 구간의 **시작 진열대 번호**와 **끝 진열대 번호**를 차례대로 배열에 담아서 `return` 하도록 하며, 만약 가장 짧은 구간이 여러 개라면 **시작 진열대 번호**가 가장 작은 구간을 `return` 합니다.

[제한사항]

- `gems` 배열의 크기는 1 이상 100,000 이하입니다.
 - `gems` 배열의 각 원소는 진열대에 나열된 보석을 나타냅니다.
 - `gems` 배열에는 1번 진열대부터 진열대 번호 순서대로 보석이름이 차례대로 저장되어 있습니다.
 - `gems` 배열의 각 원소는 길이가 1 이상 10 이하인 알파벳 대문자로만 구성된 문자열입니다.

이것이 세



2020 카카오 인턴십 - 보석쇼핑

프로그래머스 레벨 3 사용언어 : python



문제 설명

개발자 출신으로 세계 최고의 갑부가 된 어피치는 스트레스를 받을 때면 이를 풀기 위해 오프라인 매장에 쇼핑을 하러 가곤 합니다.

어피치는 쇼핑을 할 때면 **매장 진열대의 특정 범위의 물건들을 모두 싹쓸이 구매하는 습관**이 있습니다.

어느 날 스트레스를 풀기 위해 보석 매장에 쇼핑을 하러 간 어피치는 이전처럼 진열대의 특정 범위의 보석을 모두 구매하되 특별히 아래 목적을 달성하고 싶었습니다





문제 설명

어피치가 달성하고 싶은 목적

**'진열된 모든 종류의 보석을 적어도 1개 이상 포함하는
가장 짧은 구간을 찾아서 구매'**





문제 설명

진열대 번호 순서대로 보석들의 이름이 저장된 배열 **gems**가 매개변수로 주어집니다. 이때 모든 보석을 하나 이상 포함하는 가장 짧은 구간을 찾아서 return 하도록 solution 함수를 완성해주세요

가장 짧은 구간의 시작 진열대 번호와 끝 진열대 번호를 차례대로 배열에 담아서 return 하도록 하며,
만약 가장 짧은 구간이 여러 개라면 시작 진열대 번호가 가장 작은 구간을 return 합니다





문제 설명

진열대 번호	1	2	3	4	5	6	7	8
보석 이름	 DIA	 RUBY	 RUBY	 DIA	 DIA	 EMERALD	 SAPPHIRE	 DIA



문제 설명

어피치가 달성하고 싶은 목적

'진열된 모든 종류의 보석을 적어도 1개 이상 포함하는
가장 짧은 구간을 찾아서 구매'

진열대 번호	1	2	3	4	5	6	7	8
보석 이름	 DIA	 RUBY	 RUBY	 DIA	 DIA	 EMERALD	 SAPPHIRE	 DIA


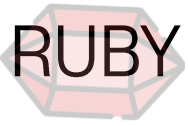






진열대의 3번부터 7번까지 구매하면
모든 종류의 보석을 적어도 하나 이상씩 포함하므로
어피치의 목적 달성!



문제 설명

어피치가 달성하고 싶은 목적

'진열된 모든 종류의 보석을 적어도 1개 이상 포함하는
가장 짧은 구간을 찾아서 구매'

진열대 번호	1	2	3	4	5	6	7	8
보석 이름	 DIA	 RUBY	 RUBY	 DIA	 DIA	 EME RALD	 SAPP HIRE	 DIA

진열대의 3번-4번, 6번-7번을 구매하면
모든 종류의 보석을 적어도 하나 이상씩 포함하지만
5번이 빠지게되므로 특정범위 조건이 설립이 되지
않으므로 어피치의 목적 X



문제 설명 - 제한사항

- gems 배열의 크기는 1이상 100,000이하입니다.
- gems 배열의 각 원소는 진열대에 나열된 보석을 나타냅니다
- gems 배열에는 1번 진열대부터 진열대 번호 순서대로 보석이름이 차례대로 저장되어 있습니다
- gems 배열의 각 원소는 길이가 1이상 10이하인 알파벳 대문자로만 구성된 문자열입니다.



문제 풀이

배열의 처음부터 끝까지 순차적으로 탐색하는 탐색문제

Gems 배열의 크기는 최대 100000 이기 때문에 $O(n^2)$ 이상인 탐색 알고리즘으로는 시간 초과가 발생하여 문제를 풀 수 없다.

효율성 테스트가 존재하는 문제이기 때문에 알고리즘 최적화 필요

=> $O(n)$ 으로 탐색할 수 있는 투 포인터 알고리즘을 사용해야 한다



문제 풀이









이중 포인터 ● 시작 ● 끝

진열대 번호	1	2	3	4	5	6	7	8
보석 이름								

조건에 맞는 구간 찾을때 까지 끝점 증가

진열대 번호	1	2	3	4	5	6	7	8
보석 이름								

해당 구간을 최소화도록 시작점 증가

진열대 번호	1	2	3	4	5	6	7	8
보석 이름								



문제 풀이

```
1 def solution(gems):
2     start = end = 0 시작점, 끝점
3     kinds = len(set(gems)) 보석 종류
4     shortest = float("inf") 가장 짧은 구간 길이
5     container = {} 현재 구간에 포함된 보석
6     answer = [0, len(gems)] 가장 짧은 구간
7
8     while end < len(gems):
9         if gems[end] not in container: 조건에 맞는 구간 찾을 때 까지 끝점 증가
10            container[gems[end]] = 1
11        else:
12            container[gems[end]] += 1
13        end += 1
14
15        if len(container) == kinds: 구간 찾으면
16            while start < end: 구간 최소화를 위한 시작점 증가
17                if container[gems[start]] <= 1:
18                    break
19                container[gems[start]] -= 1
20                start += 1
21
22            if end - start < shortest: 가장 짧은 구간 길이 갱신
23                answer = [start+1, end]
24                shortest = end - start
25    return answer
```



문제 풀이

```
1 def solution(gems):  
2     start = end = 0    시작점, 끝점  
3     kinds = len(set(gems))    보석 종류  
4     shortest = float("inf")    가장 짧은 구간 길이  
5     container = {}    현재 구간에 포함된 보석  
6     answer = [0, len(gems)]    가장 짧은 구간  
7
```



문제 풀이

```
1 def solution(gems):
2     start = end = 0
3     kinds = len(set(gems))
4     shortest = float("inf")
5     container = {}
6     answer = [0, len(gems)]
7     while end < len(gems):
8         if gems[end] not in container:
9             container[gems[end]] = 1
10        else:
11            container[gems[end]] += 1
12        end += 1
13
14        if len(container) == kinds:
15            while start < end:
16                if container[gems[start]] <= 1:
17                    break
18                container[gems[start]] -= 1
19                start += 1
20
21            if end - start < shortest:
22                answer = [start+1, end]
23                shortest = end - start
24
25    return answer
```

조건에 맞는 구간 찾을 때 까지
끝점 증가

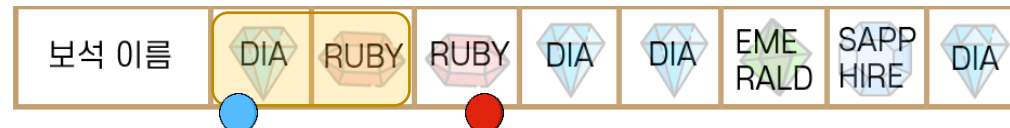
start = 0, end = 0, container = {}



start = 0, end = 1, container = {"DIA":1}



start = 0, end = 2, container = {"DIA":1, "RUBY":1}



start = 0, end = 2, container = {"DIA":1, "RUBY":2}



⋮

start = 0, end = 7, container = {"DIA":3, "RUBY":2, "EMERALD":1, "SHAPPHIRE":1}





문제 풀이

```
1 def solution(gems):
2     start = end = 0
3     kinds = len(set(gems))
4     shortest = float("inf")
5     container = {}
6     answer = [0, len(gems)]
```

```
7
8     while end < len(gems):
9         if gems[end] not in container:
10             container[gems[end]] = 1
11         else:
12             container[gems[end]] += 1
13         end += 1
```

```
14
15         if len(container) == kinds:
16             while start < end:
17                 if container[gems[start]] <= 1:
18                     break
19                 container[gems[start]] -= 1
20                 start += 1 구간을 찾으면,
```

구간 최소화를 위한 시작점 증가

```
21
22             if end - start < shortest:
23                 answer = [start+1, end]
24                 shortest = end - start
25     return answer
```

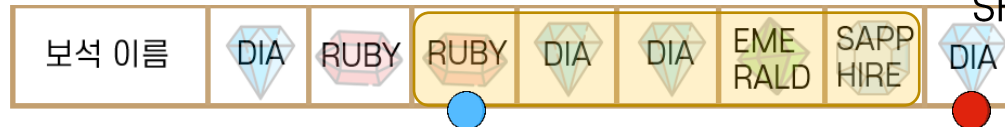
start = 0, end = 7, container = {"DIA":3, "RUBY":2, "EMERALD":1, "SHAPPHIRE":1}



start = 1, end = 7, container = {"DIA":2, "RUBY":2, "EMERALD":1, "SHAPPHIRE":1}



start = 2, end = 7, container = {"DIA":2, "RUBY":1, "EMERALD":1, "SHAPPHIRE":1}





정답 제출

코딩테스트 연습 > 2020 카카오 인턴십 > 보석 쇼핑

[카카오 인턴] 보석 쇼핑

문제 설명

[본 문제는 정확성과 효율성 테스트 각각 점수가 있는 문제입니다.]

개발자 출신으로 세계 최고의 갑부가 된 **어피치**는 스트레스를 받을 때면 이를 풀기 위해 오프라인 매장에 쇼핑을 하러 가곤 합니다.

어피치는 쇼핑을 할 때면 매장 진열대의 특정 범위의 물건들을 모두 싹쓸이 구매하는 습관이 있습니다.

어느 날 스트레스를 풀기 위해 보석 매장에 쇼핑을 하러 간 어피치는 이전처럼 진열대의 특정 범위의 보석을 모두 구매하되 특별히 아래 목적을 달성하고 싶었습니다.

진열된 모든 종류의 보석을 적어도 1개 이상 포함하는 가장 짧은 구간을 찾아서 구매

예를 들어 아래 진열대는 4종류의 보석(RUBY, DIA, EMERALD, SAPPHIRE) 8개가 진열된 예시입니다.

진열대 번호	1	2	3	4	5	6	7	8
보석 이름	DIA	RUBY	RUBY	DIA	DIA	EMERALD	SAPPHIRE	DIA

진열대의 3번부터 7번까지 5개의 보석을 구매하면 모든 종류의 보석을 적어도 하나 이상씩 포함하게 됩니다.

진열대의 3, 4, 6, 7번의 보석만 구매하는 것은 중간에 특정 구간(5번)이 빠지게 되므로 어피치의 쇼핑 습관에 맞지 않습니다.

진열대 번호 순서대로 보석들의 이름이 저장된 배열 `gems`가 매개변수로 주어집니다. 이때 모든 보석을 하나 이상 포함하는 가장 짧은 구간을 찾아서 `return` 하도록 `solution` 함수를 완성해주세요.

가장 짧은 구간의 **시작 진열대 번호**와 **끝 진열대 번호**를 차례대로 배열에 담아서 `return` 하도록 하며, 만약 가장 짧은 구간이 여러 개라면 **시작 진열대 번호**가 가장 작은 구간을 `return` 합니다.

[제한사항]

- `gems` 배열의 크기는 1 이상 100,000 이하입니다.
 - `gems` 배열의 각 원소는 진열대에 나열된 보석을 나타냅니다.
 - `gems` 배열에는 1번 진열대부터 진열대 번호 순서대로 보석이름이 차례대로 저장되어 있습니다.
 - `gems` 배열의 각 원소는 길이가 1 이상 10 이하인 알파벳 대문자로만 구성된 문자열입니다.

solution.py

실행 결과

테스트 9 >	통과 (0.48ms, 10.2MB)
테스트 10 >	통과 (0.35ms, 10.4MB)
테스트 11 >	통과 (0.87ms, 10.2MB)
테스트 12 >	통과 (1.44ms, 10.1MB)
테스트 13 >	통과 (1.99ms, 10.3MB)
테스트 14 >	통과 (1.05ms, 10.3MB)
테스트 15 >	통과 (2.42ms, 10.5MB)

효율성 테스트

테스트 1 >	통과 (3.07ms, 10.6MB)
테스트 2 >	통과 (4.28ms, 10.5MB)
테스트 3 >	통과 (8.54ms, 11.2MB)
테스트 4 >	통과 (7.65ms, 11.9MB)
테스트 5 >	통과 (13.98ms, 11.9MB)
테스트 6 >	통과 (18.64ms, 12.2MB)
테스트 7 >	통과 (22.81ms, 12.6MB)
테스트 8 >	통과 (25.86ms, 12.9MB)
테스트 9 >	통과 (30.23ms, 13.4MB)
테스트 10 >	통과 (32.92ms, 13.7MB)
테스트 11 >	통과 (38.44ms, 14.7MB)
테스트 12 >	통과 (27.90ms, 15.4MB)
테스트 13 >	통과 (33.94ms, 16.3MB)
테스트 14 >	통과 (53.52ms, 17MB)
테스트 15 >	통과 (60.07ms, 17.8MB)

채점 결과

정확성 : 33.3

효율성 : 66.7

합계 : 100.0 / 100.0

감사합니다
