1. String 개념(사전지식)

고유한 이메일(Unique Email Addresses)
보석과 돌(JewelsAndStones)
포맷팅(licenseKeyFormatting)
플러스원(PlusOne)

사전지식

1) charAt(i) => 문자열의 위치

2) toCharArray() => char[]

3) IndexOf(), substring(begin, end) String str = “abcd”

ex) str.indexOf (“a”)// 0

indexOf("찾을 특정 문자" , "시작할 위치")

ex) str.substring(0,3)= abc

4) startsWith(), endWith(), split()

ex)String str = “abc” str. startsWith(“a”); => true

str. endWith(“c”);=true

ex)String str = “010 888 7777” String[] spStr = str. split(“ ”)

String s1 = spStr[0]; String s2 = spStr[1]; String s3 = spStr[0];

5) toLowerCase()

6) replace()

7) Character.isDigit(c), Character.isLetter(c)

8) StringBuilder sb = new StringBuilder(); sb.append(“aa”)

고유한 이메일(Unique Email Addresses)

설명

모든 유효한 이메일은@을 기준으로 **로컬이름과 도메인이름**으로 구성됩니다.
또한 소문자외에 하나 이상의 '.'또는 '+'을 포함합니다.

예) test@coding.com
test는 로컬이름이고 codingtest.com은 도메인 이름입니다.
이메일에는 소문자 외에 '.'또는 '+'가 포함될 수 있습니다.

로컬이름에 일부 문자 사이에 마침표 ('.')를 추가하여
전송된 메일은 로컬 이름에 점이없는 동일한 주소로 전달됩니다.
예)
" test.email@codingtest.com "및 " testemail@codingtest.com "은 동일한 이메일 주소로 전달됩니다.
(이 규칙은 도메인 이름에는 적용되지 않습니다.)

로컬 이름에 더하기 ('+')를 추가하면 첫 번째 더하기 기호 뒤의 모든 항목이 무시됩니다.
이를 통해 특정 이메일을 필터링 할 수 있습니다.
예) test.email+james@codingtest.com은
test.email@codingtest.com으로 전달됩니다.
(이 규칙은 도메인 이름에는 적용되지 않습니다.)
이 두 규칙을 동시에 사용할 수 있습니다.

이메일 목록이 주어지면 목록의 각 주소로 하나의 이메일을 보냅니다.
실제로 메일을받는 주소는 몇 개입니까?

고유한 이메일(Unique Email Addresses)

입출력

Input: emails = [
"test.email+james@coding.com",
"test.e.mail+toto.jane@coding.com",
"testemail+tom@cod.ing.com"]
]
Output: 2
Explanation:
"testemail@coding.com"
"testemail@cod.ing.com"

Input: emails = [
"a@coding.com",
"b@coding.com",
"c@coding.com"]

Output: 3
로컬네임이 다름

문제 Format

```
class Solution {
    public int solve(String[] emails) {
    }
}
```

제한사항

1 <= emails.length <= 100
1 <= emails[i].length <= 100
email[i] consist of lowercase English letters, '+', '.' and '@'.
Each emails[i] contains exactly one '@' character.
All local and domain names are non-empty.
Local names do not start with a '+' character.

Solution

1. 문제분석

- 문제를 정확히 이해
- 분석 내용 정리(Devide & Conquer)

2. 규칙찾기

- 분석 내용으로 규칙을 찾는다

3. 코딩화

- 분석 내용으로 알맞은 구현방법 찾기

4. 알고리즘 적용

- 알고리즘 정하고 담을 그릇 정한다
(사전지식 필요)

문제분석

Input: emails = [
"test.email+james@coding.com",
"test.e.mail+toto.jane@coding.com",
"testemail+tom@cod.ing.com"
]
Output: 2
Explanation:
"testemail@coding.com"
"testemail@cod.ing.com"

1. 로컬네임 + 도메인네임
2. 로컬네임에서는 . 무시한다
3. 로컬네임에서 + 이후로 나오는 문자열은 무시한다
4. 도메인네임에서 . 이 들어가면 고유하다

문제분석

test.email+james

1. 로컬네임 + 도메인네임
2. 로컬네임에서는 . 무시한다
3. 로컬네임에서 + 이후로 나오는 문자열은 무시한다
4. 도메인네임에서 . 이 들어가면 고유하다

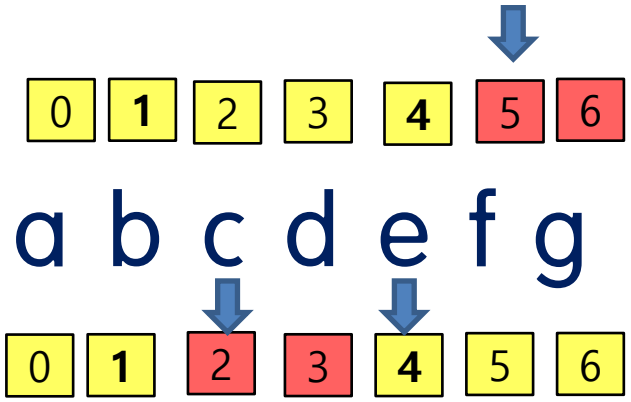
규칙찾기

1. . => Continue로 뺀다
2. + => break로 뺀다
3. Set<String>

문제분석

test.email+james

1. 로컬네임 + 도메인네임
2. 로컬네임에서는 . 무시한다
3. 로컬네임에서 + 이후로 나오는 문자열은 무시한다
4. 도메인네임에서 . 이 들어가면 고유하다
5. 수정된 String을 Set에 넣는다.



1. str.subString(5); => fg
2. str.subString(2,4) =>cd
3. str.indexOf('f'); => 5

문제분석

test.email+james

2. Split()으로 풀기

에러 : Dangling meta character '+' near index 0
split("+")부분에서 컴파일러 에러

해결: .split("₩₩+");
.split("[+]");

시간복잡도/공간복잡도 계산

시간복잡도

1. 대상(Source) : 문제에서 입력받은 파라미터(array 등) (속도)
   Time Complexity : O(N)
   대상 : String[] emails
   이유 : for문 한번 실행 , O(NM) N= Number of emails, M= Length of email

공간복잡도

2. 대상(Source) : 실제 사용되는 저장 공간을 계산(메모리 사용량)
   예) 프로그램을 실행 및 완료하는데 필요한 저장공간
   Space Complexity : O(N)
   대상 : Set<String> set = new HashSet();
   이유 : for문 한번 실행

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)

보석과 돌(Jewels And Stones)

설명

보석(jewels)과 돌(stones) 이 주어집니다.
보석을 이용해서 돌에 얼마나 많은 보석이 포함되어 있는지 알고 싶습니다.

문자는 대소 문자를 구분하므로 "a"와 "A " 은 다른 유형의 스톤으로 간주됩니다 .

입출력

Input: jewels = "aA",
stones = "aAAbbbb"
Output: 3

Input: jewels = "z", stones = "ZZ"
Output: 0

문제 Format

```
class Solution {
 public int solve(String jew, String stones){}
}
```

```
class Solution:
    def solve(self, jewels: str, stones: str) -> int:
```

제한사항

1 <= jewels.length, stones.length <= 50
jewels and stones consist of only English letters.
All the characters of jewels are **unique**.

Solution

1. 문제분석

- 문제를 정확히 이해
- 분석 내용 정리(Devide & Conquer)

2. 규칙찾기

- 분석 내용으로 규칙을 찾는다

3. 코딩화

- 분석 내용으로 알맞은 구현방법 찾기

4. 알고리즘 적용

- 알고리즘 정하고 담을 그릇 정한다
(사전지식 필요)

문제분석

aA

aAAbbbb

1. 보석은 대소문자를 구분해서 갖고 있어야 한다.
2. aA ->2개
3. 보석을 저장한다 set
4. Stone을 for문 루프
5. Set에 있는 값과 비교해서 count

시간복잡도/공간복잡도 계산

시간복잡도

1. 대상(Source) : 문제에서 입력받은 파라미터(array 등) (속도)
   Time Complexity : O(N)
   대상 : String J, String S
   이유 : for문 한번 실행

공간복잡도

2. 대상(Source) : 실제 사용되는 저장 공간을 계산(메모리 사용량)
   예) 프로그램을 실행 및 완료하는데 필요한 저장공간
   Space Complexity : O(N)
   대상 : Set<Character> set = new HashSet<>();
   이유 : for문 한번 실행

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)

라이센스 키 포맷(License Key Formatting)

설명

숫자, 문자와 대시로만 구성된 문자열로 표시된 라이센스 키가 제공 됩니다.
문자열은 대시(-)로 n 과 n + 1그룹으로 구분됩니다 . 정수값 k도 주어집니다.
첫 번째 그룹을 제외하고 각 그룹이 정확히 k로 문자열을 형식화(Formatting)해야 합니다.
첫 번째 그룹은 더 짧을 수 있지만 여전히 적어도 하나의 문자를 포함해야합니다.
또한 두 그룹 사이에 대시가 삽입되어야하며 모든 소문자를 대문자로 변환해야합니다.
재 포맷 된 라이센스 키를 리턴하세요

입출력

Input: s = "8F3Z-2e-9-w", k = 4
Output: "8F3Z-2E9W"

Input: s = " 8-5g-3-J ", k = 2
Output: "8-5G-3J"

문제 Format

```
class Solution {
    public int solve(String s, int k) {
}
```

제한사항

$1 <= s.length <= 10^5$
s 는 영어문자, 숫자, 대쉬(-)로 이루어져 있습니다.
$1 <= k <= 10^4$

Solution

1. 문제분석

- 문제를 정확히 이해
- 분석 내용 정리(Devide & Conquer)

2. 규칙찾기

- 분석 내용으로 규칙을 찾는다

3. 코딩화

- 분석 내용으로 알맞은 구현방법 찾기

4. 알고리즘 적용

- 알고리즘 정하고 담을 그릇 정한다
(사전지식 필요)

문제분석

8F3Z-2e-9-w

8F3Z-2E9W

규칙찾기

1. - 를 없애기
2. 대문자로 만들기
3. K=4로 문자열 나누기
4. 나눈 문자열 사이 – (대쉬) 넣기

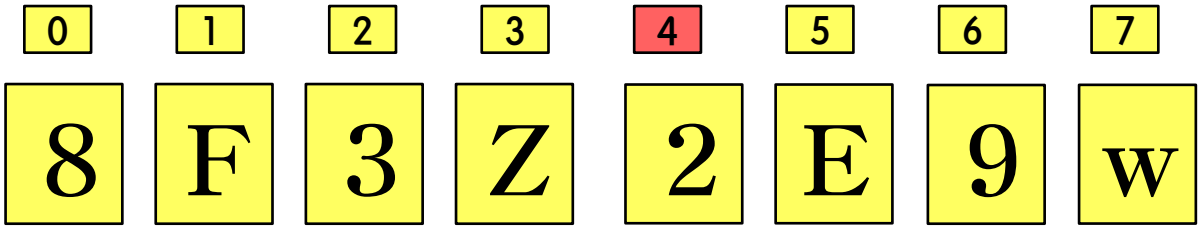
코딩화

1. replace()
2. toUpperCase()
3. K=4로 문자열 나누기
4. 나눈 문자열 사이 – (대쉬) 넣기

문제분석

8F3Z-2e-9-w

8F3Z-2E9W



1. Leng=8
2. 8-k(4)=4

시간복잡도/공간복잡도 계산

시간복잡도

1. **대상**(Source) : **문제에서 입력받은 파라미터**(array 등) (**속도**)
   Time Complexity : O(N)
   대상 : String str
   이유 : for문 한번 실행

공간복잡도

2. **대상**(Source) : **실제 사용되는 저장 공간을 계산**(**메모리 사용량**)
   예) **프로그램을 실행 및 완료하는데 필요한 저장공간**
   Space Complexity : O(N)
   대상 : StringBuilder sb = new StringBuilder();
   이유 : for**문 한번 실행**

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)

플러스 원(Plus One)

설명

음이 아닌 정수를 나타내는 **비어 있지 않은** 십진수 배열이 주어지면 정수 1을 증가시킵니다.
숫자는 최상위 숫자가 목록의 맨 앞에 있고 배열의 마지막에 숫자에 +1을 합니다.
숫자 0 자체를 제외하고 정수에 앞에 0이 포함되지 않습니다.

입출력

Input: digits = [1,2,3]
Output: [1,2,4]

Input: [9,9,9]
Output: [1,0,0,0]

문제 Format

```
class Solution {
    public int[] solve( int[] k) {
}
```

제한사항

1 <= digits.length <= 100
0 <= digits[i] <= 9

Solution

1. 문제분석

- 문제를 정확히 이해
- 분석 내용 정리(Devide & Conquer)

2. 규칙찾기

- 분석 내용으로 규칙을 찾는다

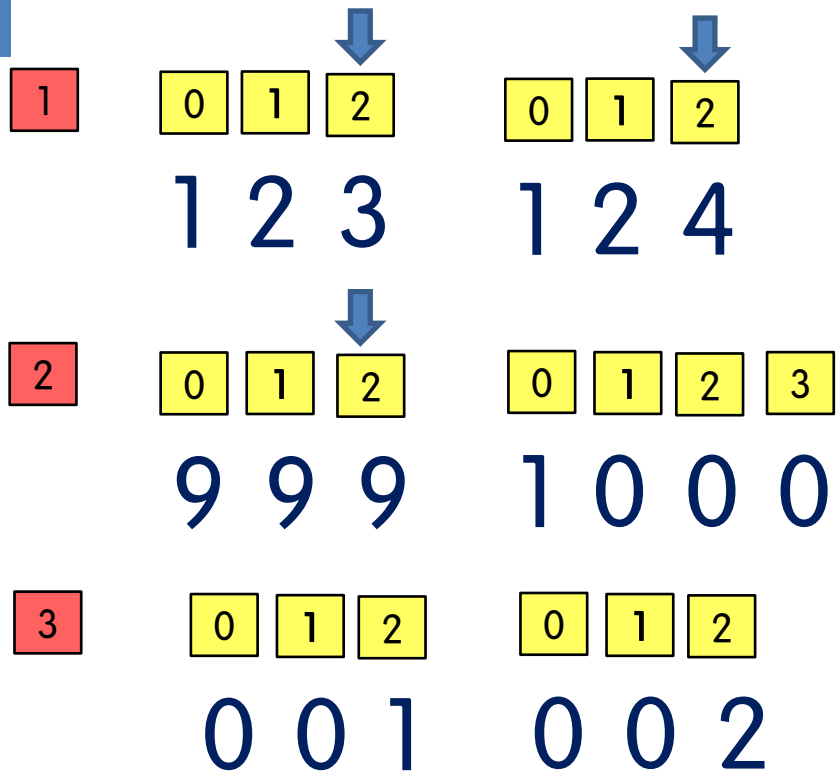
3. 코딩화

- 분석 내용으로 알맞은 구현방법 찾기

4. 알고리즘 적용

- 알고리즘 정하고 담을 그릇 정한다
(사전지식 필요)

문제분석



1. 뒷자리부터 체크 digits[2]+1,
2. 값이 10이 되면 carry=1로 1을 맨앞자리에 추가

시간복잡도/공간복잡도 계산

시간복잡도

1. 대상(Source) : 문제에서 입력받은 파라미터(array 등) (속도)
   Time Complexity : O(N)
   대상 : int[] digits
   이유 : for문 한번 실행

공간복잡도

2. 대상(Source) : 실제 사용되는 저장 공간을 계산(메모리 사용량)
   예) 프로그램을 실행 및 완료하는데 필요한 저장공간
   Space Complexity : O(N)
   대상 : int[] digits
   이유 : digits안에서 수행

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)