1. Meeting Room
2. MoveZeros
3. Two Sum
4. Daily Temperature
5. Merge Interval
6. Meeting Room2
7. Jewel and Stone
8. LicenseKey Formatting
9. Kclosest
10. PlusOne

전략

1. 문제를 정확히 이해한다.

2. 알고리즘을 생각해 본다.

   단순 Array, List, Map, LinkedList, Stack, Queue, Set

   중간 Comparator, PriorityQueue

   고급 DFS, BFS, DP , Greedy

문제 1) MeetingRoom

Problem

Given an array of meeting time intervals consisting of start and end times [[s1,e1],[s2,e2],...] (si < ei), determine if a person could attend all meetings.

Input: [[0,30],[5,10],[15,20]]
Output: false

Input: [[7,10],[2,4]]
Output: true

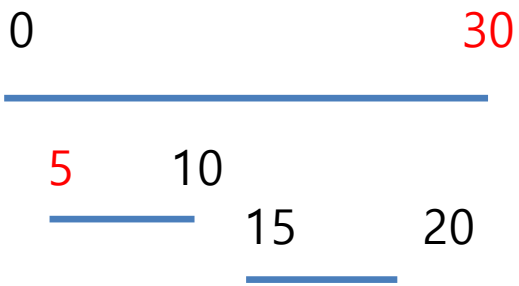
MeetingRoom

Problem

Input: [[0,30],[5,10],[15,20]]
Output: false

Input: [[7,10],[2,4]]
Output: true

Example



1. 소팅
2. hold.end > cur.start

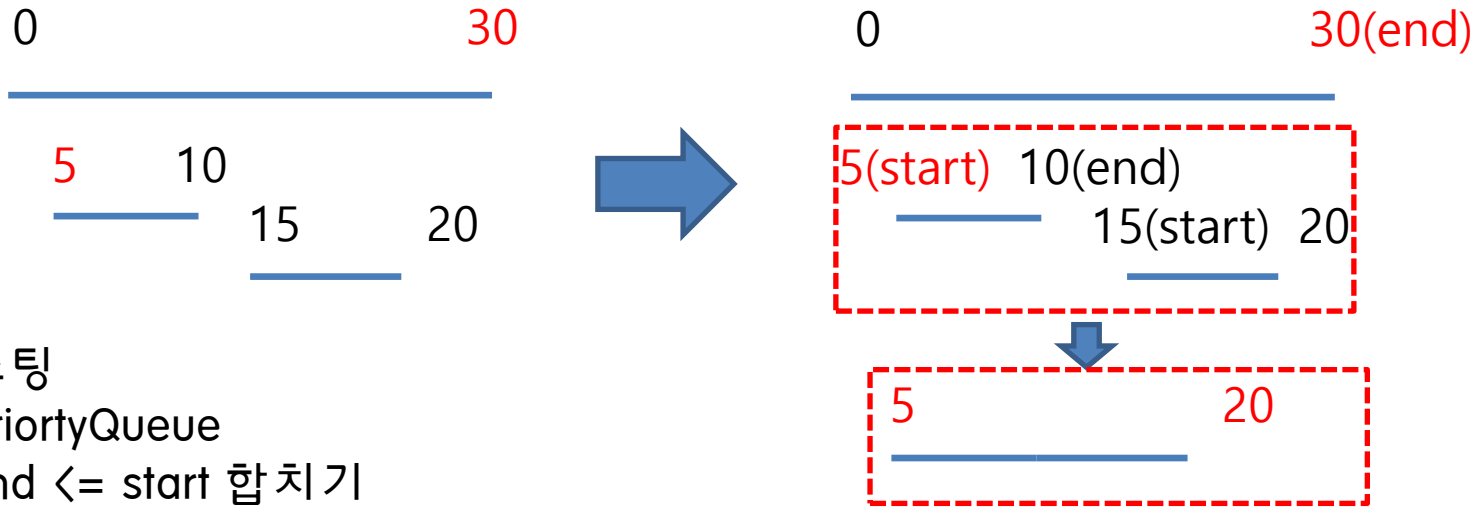
MeetingRoom2

Problem

Input: [[0,30],[5,10],[15,20]]
Output: 2

Input: [[7,10],[2,4]]
Output: 1

Example



1. 소팅
2. PriortyQueue
3. End <= start 합치기

Solution

String

1. 문제를 정확히 이해
2. 알고리즘 정하고 담을 그릇 정한다
3. for 문 돌리기
4. 생각->프로그램(한국말로 생각하고->Java)
   결과를 해석하여 이미지화시킨다

I Can Image

문제 2) MoveZeros

Problem

Given an array nums, write a function to move all 0's to the end of it while maintaining the relative order of the non-zero elements.

Example:

Input: [0,1,0,3,12]
Output: [1,3,12,0,0]
Note:

You must do this in-place without making a copy of the array.
Minimize the total number of operations.

MoveZeros

Problem

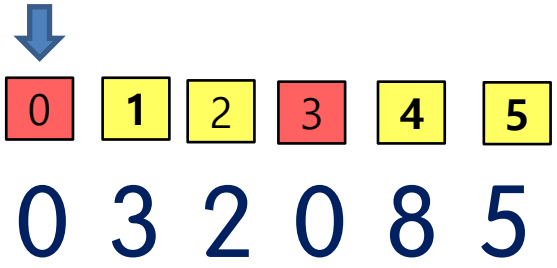
- 배열 num을 감안할 때 0이 아닌 요소의 상대적인 순서를 유지하면서 모든 0을 끝으로 이동시키는 함수를 작성하십시오

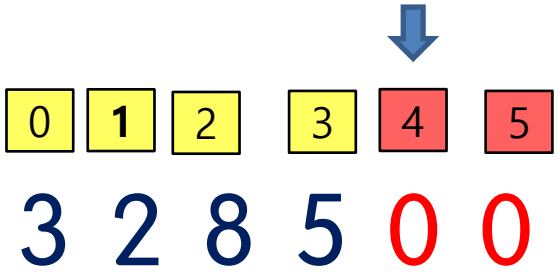
Example

Input : [0,3,2,0,8,5]
Output : [3,2,8,5,0,0]

1. 값이 0이 아닌값을먼저 array에 담는다
2. Index를 기억한다
3. 해당 index 0인 값을 넣는다

Note



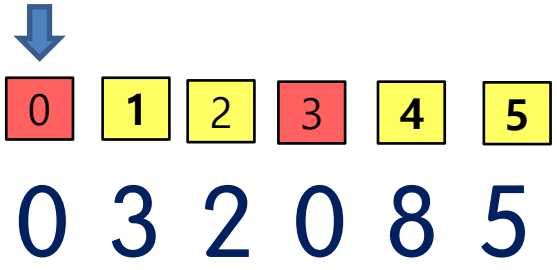


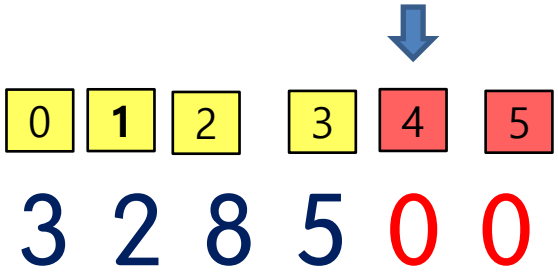
Example

Input : [0,3,2,0,8,5]
Output : [3,2,8,5,0,0]

1. 값이 0이 아닌값을먼저 array에 담는다
2. Index를 기억한다
3. 해당 index 0인 값을 넣는다

Note





2. Solution

Array

1. 문제를 정확히 이해

1. 값이 0이 아닌값을먼저 array에 담는다
2. Index를 기억한다
3. 해당 index 0인 값을 넣는다

2. 담을 그릇 정한다(Data Structure)

3. for , While 문 돌리기(알고리즘 추가)

문제 3) TwoSum

Problem

Given an array of integers, return indices of the two numbers such that they add up to a specific target.

You may assume that each input would have exactly one solution, and you may not use the same element twice.

Example:

Given nums = [2, 8, 11, 14], target = 16,

Because nums[0] + nums[3] = 2 + 14 = 16
return [1, 4].

2. Solution

Map, Array

1. 문제를 정확히 이해

2. 알고리즘 정하고 담을 그릇 정한다

3. for 문 돌리기

Input : int[] nums = {2,8,11,14};
int target =16;
Output : int[] ={1,4}
생각->프로그램(한국->영어, 한국->Java)
결과를 해석을 해서 이미지화시킨다.

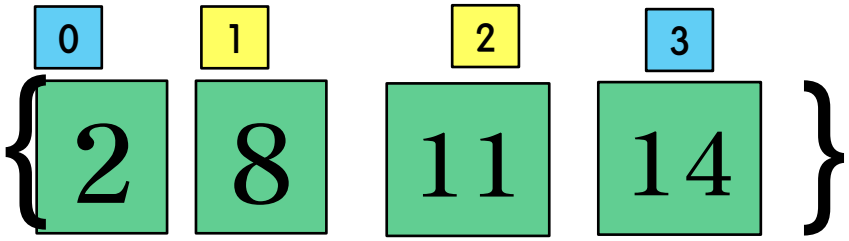
Two sum

Map, Array

1. 문제 flow

1. Array Index 0부터 끝까지 for돌려서 target과 비교
2. 16-2=11
3. Map(숫자, 방번호)
4. 방번호만 리턴한다 int[]

1 array



2 Int target =16

Key	Value
14 (16-2)	방번호 0
8 (16-8)	방번호 1
5 (16-11)	방번호 2

문제 4) Daily Temperature

Problem

Given a list of daily temperatures T, return a list such that, for each day in the input, tells you how many days you would have to wait until a warmer temperature. If there is no future day for which this is possible, put 0 instead.

For example, given the list of temperatures
T = [73, 74, 75, 71, 69, 72, 76, 73],
your output should be [1, 1, 4, 2, 1, 1, 0, 0].

Note: The length of temperatures will be in the range [1, 30000].
Each temperature will be an integer in the range [30, 100].

2. Solution

Array

1. 문제를 정확히 이해

int[] nums = {73, 74, 75, 71, 69, 72, 76, 73};
Output : 1 1 4 2 1 1 0 0

스택을 이용한다.

2. 담을 그릇 정한다(Data Structure)

3. for , While 문 돌리기(알고리즘 추가)

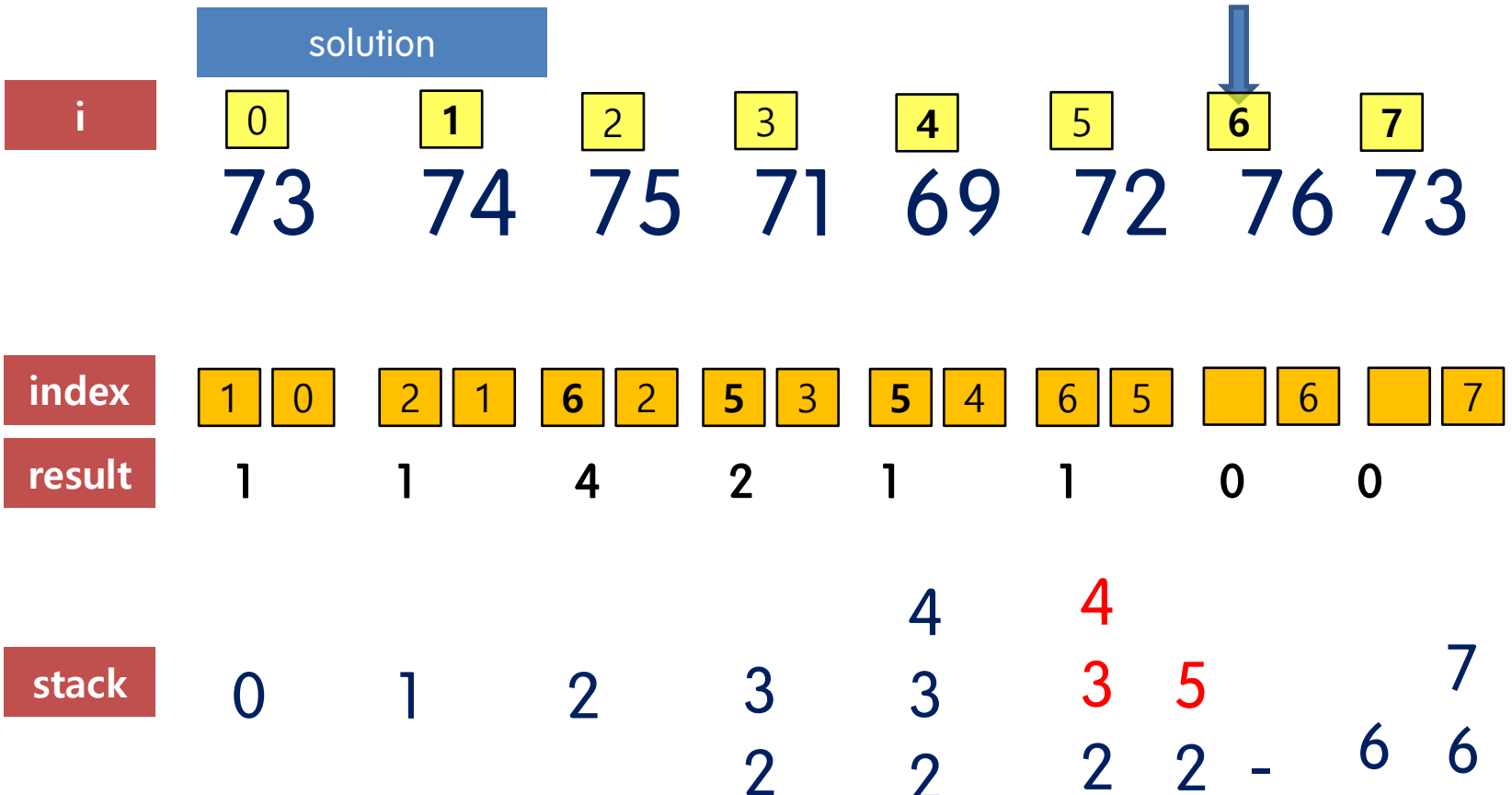
Daily Temperature

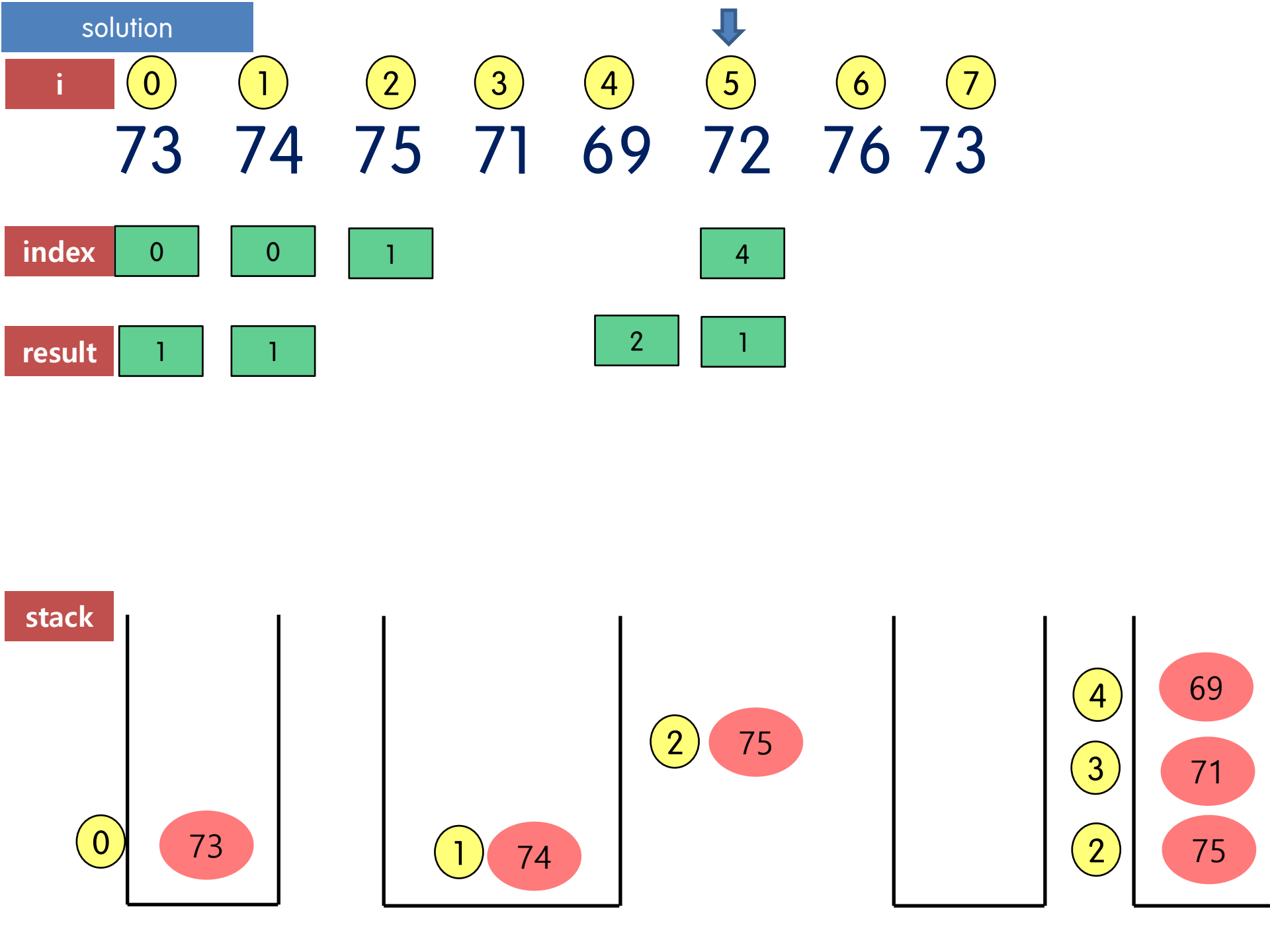
Problem

int[] nums = {73, 74, 75, 71, 69, 72, 76, 73};

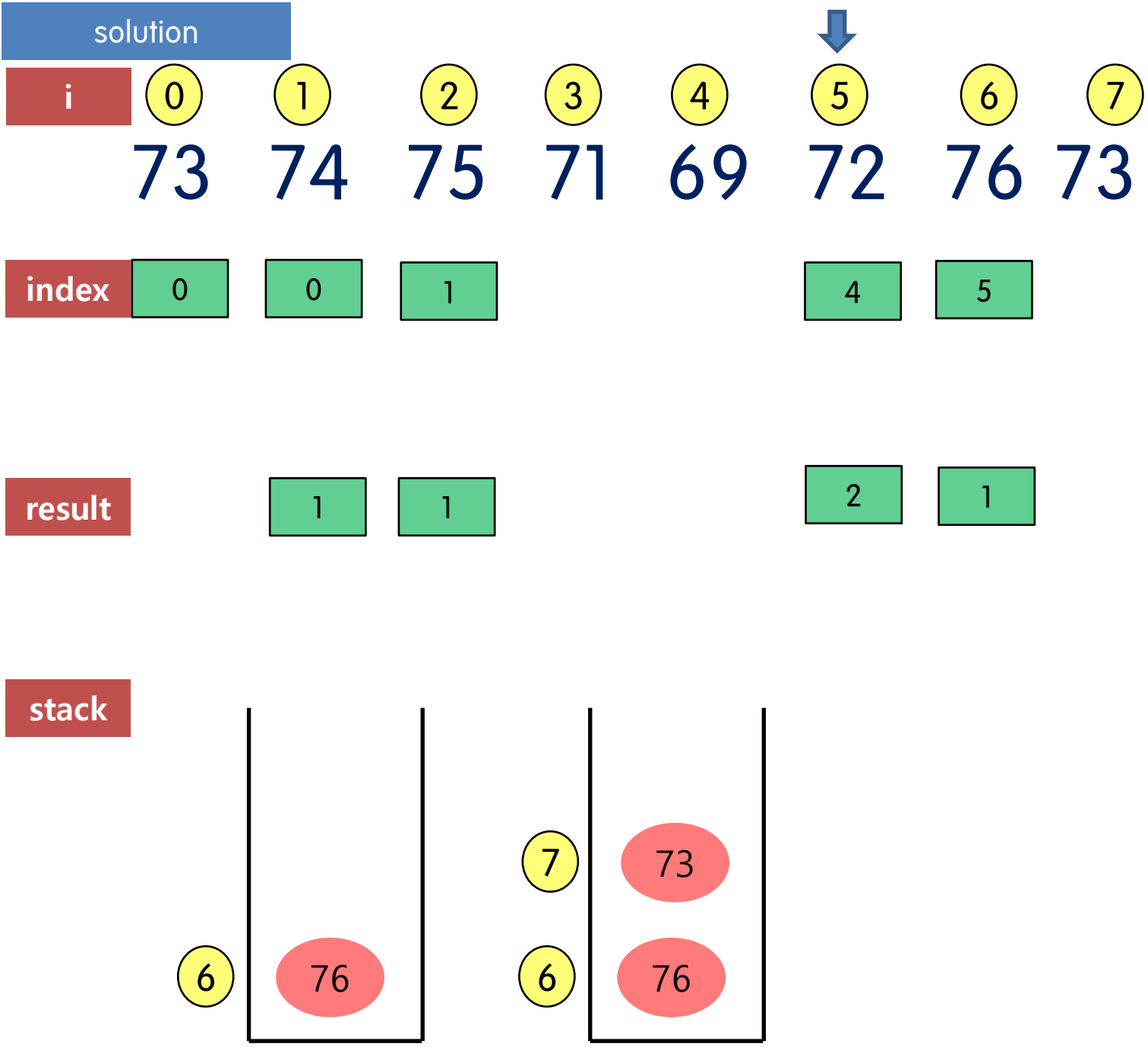
Output : 1 1 4 2 1 1 0 0

solution




solution
i
0
1
2
3
4
5
6
7
73
74
75
71
69
72
76
73
index
0
0
1
4
result
1
1
2
1
stack
0
73
1
74
2
75
4
69
3
71
2
75

solution


i
0
1
2
3
4
5
6
7
73
74
75
71
69
72
76
73
index
0
0
1
4
5
result
1
1
2
1
stack
6
76
7
73
6
76

Daily Temperature(시간복잡도)

Problem

int[] nums = {73, 74, 75, 71, 69, 72, 76, 73};

Output : 1 1 4 2 1 1 0 0

solution

1. Time Complexity: O(N)
   1. 대상 : int[] nums
   2. for문 : n번 돌리기

2. Space Complexity: O(N)
   1. stack 싸이즈 int[] nums

문제 5) MergeInterval

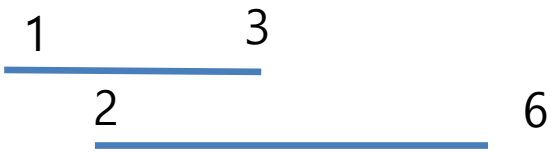
Comparator

Problem

Given a collection of intervals, merge all overlapping intervals.

Example

Input: [[1,3],[2,6],[8,10],[15,18]]
Output: [[1,6],[8,10],[15,18]]



문제 6) MeetingRoom2

Problem

Given an array of meeting time intervals consisting of start and end times [[s1,e1],[s2,e2],...] (si < ei), find the minimum number of conference rooms required.

Input: [[0,30],[5,10],[15,20]]
Output: 2

Input: [[7,10],[2,4]]
Output: 1

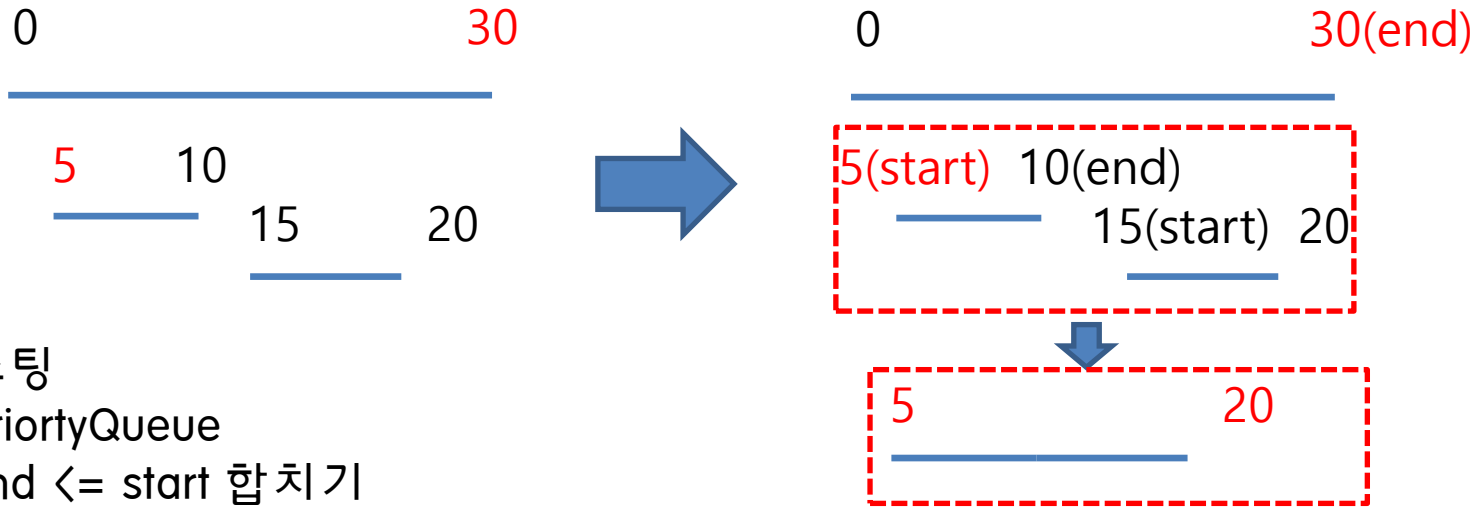
MeetingRoom2

Problem

Input: [[0,30],[5,10],[15,20]]
Output: 2

Input: [[7,10],[2,4]]
Output: 1

Example



1. 소팅
2. PriortyQueue
3. End <= start 합치기

Solution

Map, Array

1. 문제를 정확히 이해

2. 알고리즘 정하고 담을 그릇 정한다

3. for 문 돌리기

생각->프로그램(한국말로 생각하고->Java)
결과를 해석하여 이미지화시킨다

1. 클래스(Interval)
2. 리스트화(자료)
3. Comparator(버젼)
4. 현재Start< 전 end , 새롭게 1,6짜리 클래스를 만든다.

문제 7) Jewels And Stones

Problem

You're given strings J representing the types of stones that are jewels, and S representing the stones you have. Each character in S is a type of stone you have. You want to know how many of the stones you have are also jewels.

The letters in J are guaranteed distinct, and all characters in J and S are letters. Letters are case sensitive, so "a" is considered a different type of stone from "A".

Input: J = "aA", S = "aAAbbbb"

Output: 3

7. Jewels And Stones

Problem

보석갯수, 스톤갯수, letter 대소문자 구분

Example

Input: J = "aA", S = "aAAbbbb"

Output: 3

Solution

Set (중복X)

1. 문제를 정확히 이해

2. 알고리즘 정하고 담을 그릇 정한다

3. for 문 돌리기

4. 생각->프로그램(한국말로 생각하고->Java)
   결과를 해석하여 이미지화시킨다

I Can Image

1. 보석은 대소문자를 구분해서 갖고 있어야 한다.
2. aA ->2개
3. 스톤에가서 aA가 개별적으로 몇 개이었는지 체크

알아야 할 개념 설명

1) Set (중복을 허용하지 않은 값만 저장)

2) For문 돌리기,

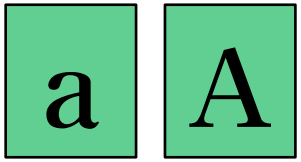
   char[] 리턴 toCharArray()

Jewels And Stones

1. 문제 flow

1. Set을 만들고 a, A를 저장한다
2. Stone값을 for문을 돌려서 빼낸다
3. Set에 있는지 확인한다.

1 Set 에 저장



2 for

문제 8) LicenseKeyFormatting

Problem

You are given a license key represented as a string S which consists only alphanumeric character and dashes. The string is separated into N+1 groups by N dashes.

Given a number K, we would want to reformat the strings such that each group contains exactly K characters, except for the first group which could be shorter than K, but still must contain at least one character. Furthermore, there must be a dash inserted between two groups and all lowercase letters should be converted to uppercase.

Given a non-empty string S and a number K, format the string according to the rules described above.

input
String str = "8F3Z-2e-9-w";
String str = "8-5g-3-J";
int k =4;
Output : **8F3Z-2E9W 8F3Z-2E9W, 8-5G3J**

LicenseKeyFormatting

Problem

input
String str = "8F3Z-2e-9-w";
String str = "8-5g-3-J";
int k =4;

Output : **8F3Z-2E9W 8F3Z-2E9W, 8-5G3J**

Example

1. 하이픈제거, 대문자
2. 끝에서 4자리 끊기

String, StringBuffer, StringBuilder 차이점과 장단점.

1. String + 연산이나 concat을 -> 새로운 String객체를 new로 만듦
2. StringBuffer, 멀티쓰레드환경에서는 synchronized
3. StringBuilder asynchronized 싱글쓰레드 환경

문제 9) K Closest Points to Origin

PriorityQueue

Problem

We have a list of points on the plane. Find the K closest points to the origin (0, 0).
(Here, the distance between two points on a plane is the Euclidean distance.)
You may return the answer in any order. The answer is guaranteed to be unique (except for the order that it is in.)

Example 1:
Input: points = [[1,3],[-2,2]], K = 1
Output: [[-2,2]]
Explanation:
The distance between (1, 3) and the origin is sqrt(10).
The distance between (-2, 2) and the origin is sqrt(8).
Since sqrt(8) < sqrt(10), (-2, 2) is closer to the origin.
We only want the closest K = 1 points from the origin, so the answer is just [[-2,2]].

K Closest Points to Origin

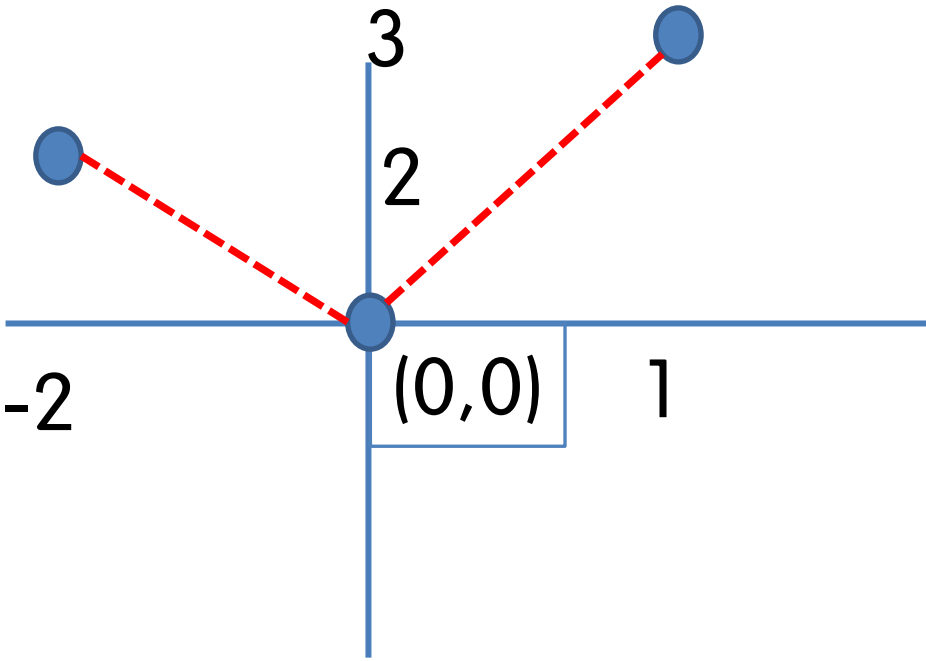
PriorityQueue

1. 문제해석 flow

1. 원점으로 부터의 거리를 구한다.

$$(X2-X1)^2+(Y2-Y1)^2$$

2. 원점에서 제일 작은 거리에 있는 값을 구한다
4. PriorityQueue를 이용한다.



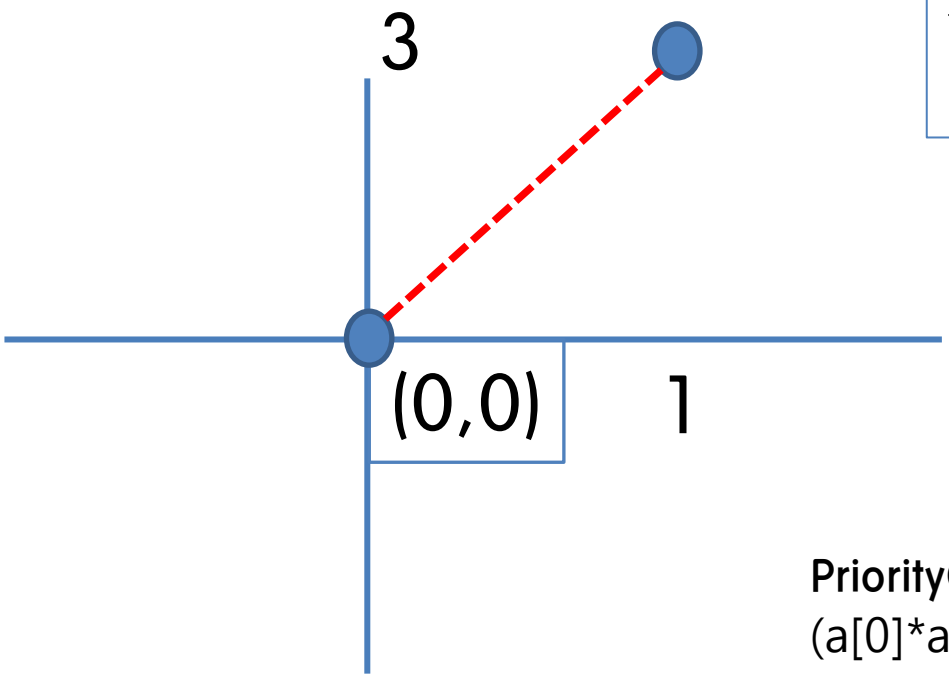
K Closest Points to Origin

PriorityQueue

Problem

Input: points = [[1,3],[-2,2]], K = 1
Output: [[-2,2]]

(0,0)(1, 3) and the origin is sqrt(10) 1+9
(0,0)(-2, 2) and the origin is sqrt(8) 4+4



원점으로 부터 거리구하는 공식

$$(X2-X1)^2+(Y2-Y1)^2$$

거리구하는 공식

$$(1-0)^2+(3-0)^2=10$$

$$(-2-0)^2+(2-0)^2=8$$

PriorityQueue를 만들때, a-b 오름차순으로 만든다
(a[0]*a[0]+a[1]*a[1])-(b[0]*b[0]+b[1]*b[1])

개념) 이차원배열

2DArray

```
int[][] points = {{1,3},{-2,2}};
```

1

```
int[ ][ ] p= new int[2][2]
```

	0 열	1 열
0 행	1	3
1 행	-2	2

개념) Priority Queue

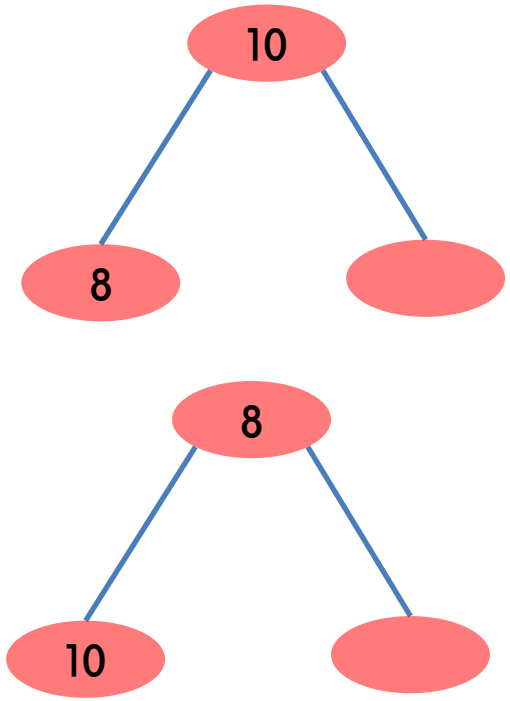
Queue(Min Heap)

1. 높은 우선순위의 요소를 먼저 꺼내서 처리하는 구조
2. 큐에 들어가는 원소 비교 기준 필요
3. 이진트리 구조, 시간 복잡도 O(NLogN) 예) 8개는 3번만에 찾을 수 있음

1. 기준타입 : int[]

2. 비교식 : (a,b)

3. Heapy : (a,b)



1. minHeap 구조로 pq를 만듭니다

```
Queue<int[]> queue=new PriorityQueue<>((a,b)->(a[0]*a[0]+a[1]+a[1])-b[0]*b[0]+b[1]*b[1]))
```

문제10) Plus One

Problem

Given a non-empty array of digits representing a non-negative integer, plus one to the integer.

The digits are stored such that the most significant digit is at the head of the list, and each element in the array contain a single digit.

You may assume the integer does not contain any leading zero, except the number 0 itself.

Example 1:

Input: [1,2,3]
Output: [1,2,4]
Explanation: The array represents the integer 123.

Plus One

Problem

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

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

Example

1. 뒷자리부터 9가 있는지 체크
2. input[0], input[1], input[2]
3. input[2] =0;
4. input[0]==1 , 새로운 int[] 의 크기를 1증가
5. 새로운 배열 res[0]=1

1. 9가 아닌경우는 해당 자리수의 값을 +1하면 끝

Plus One

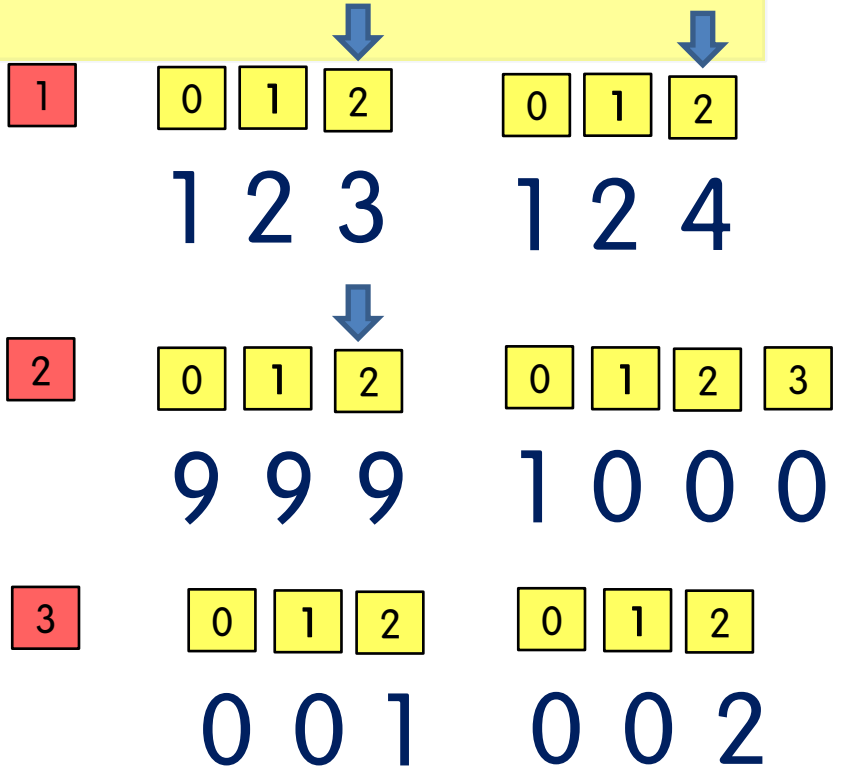
Problem

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

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

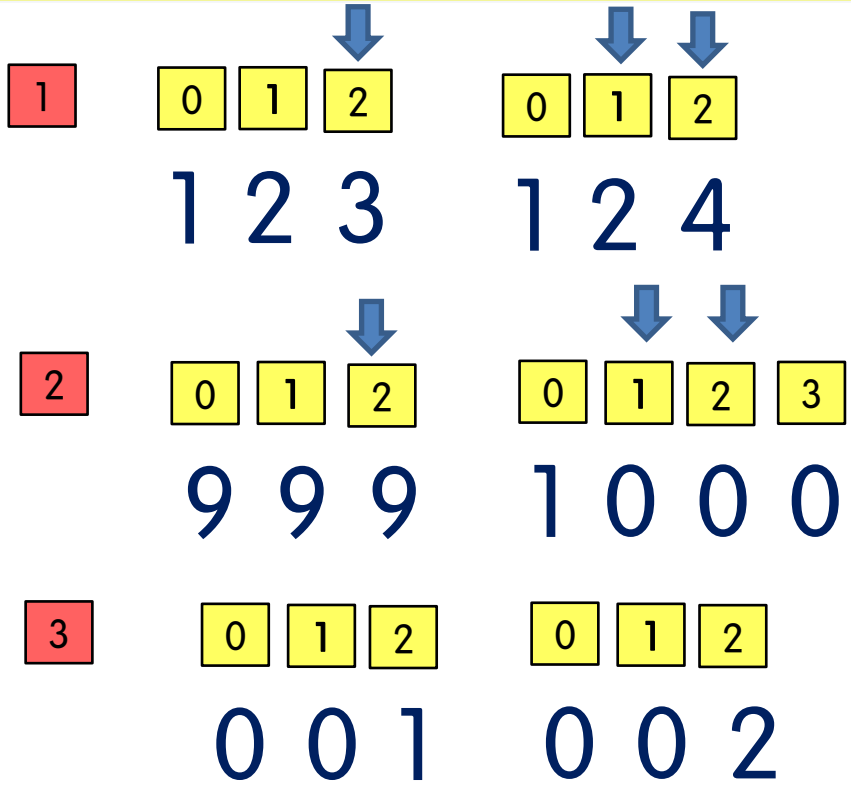
Input: [0,0,1]
Output: [0,0,2]

Example



1. 뒷자리부터 체크
   digits[2]+1, 값이 그냥 증가하면 간단
   값이 10이 되면 carry=1로 1을 전달

Plus One



1. 뒷자리부터 index부터 체크
   digits[index] = (digits[index]+carry)%10;
   0이 나오는 경우, 단순히 +1 늘어난경우

1. 9가 아닌경우는 해당 자리수의 값을 +1하면 끝
2. 9인 경우에는 배열 index값이 한 개 증가한 상태가 된다