

11. Unique Email

12. Longest Substring With At Most Two Distinct

13. Maximum Subarray

14. Find Anagrams Mapping

15. Find All Anagrams

16. SpiralMatrix

17. Group Anagrams

18. Trapping Rain Water

19. Kth Largest Element In An Array

20. Missing Ranges

문제 11) Unique Email Address

Problem

모든 이메일은 @ 기호로 구분된 로컬 이름과 도메인 이름으로 구성됩니다.

예를 들어 abc@codingtest.com에서 abc는 로컬 이름이고 codingtest.com은 도메인 이름입니다. 이러한 이메일에는 소문자 외에 '.'또는 '+'가 포함될 수 있습니다.

이메일 주소의 로컬 이름 부분에서 일부 문자 사이에 마침표 ('.')를 추가하여 전송된 메일은 로컬 이름에 점이없는 동일한 주소로 전달됩니다.

예를 들어

" abc@codingtest.com "및 " abc@coding.test.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.

예를 들어 ab.c+name@email.com은 abc@email.com으로 전달됩니다.

(다시 말하지만,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?

Unique Email Address

Problem

Input: [

"test.email+james@coding.com",
"test.e.mail+toto.jane@coding.com",
"testemail+tom@cod.ing.com"]

Output: 2

"testemail@coding.com"
"testemail@cod.ing.com"

분석

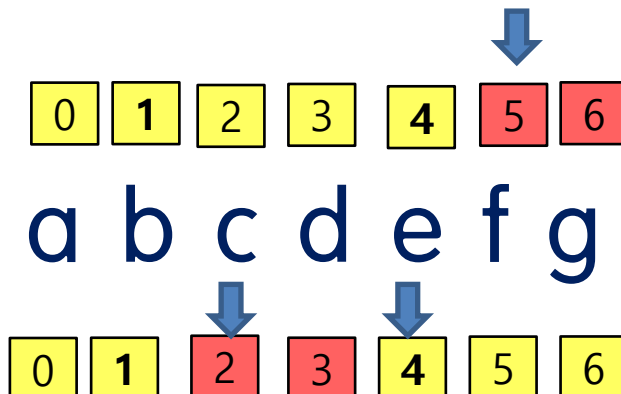
1. 로컬네임 + 도메인네임
2. 로컬네임에서는 . 무시한다
3. 로컬네임에서 + 이후로 나오는 문자열은 무시한다
4. 도메인네임에서 . 이 들어가면 고유하다

String (사전지식)

분석

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

- 1) 고유한값을 저장 => HashSet 을 이용해서 저장
- 2) 2번을 위해서



1. `a.substring(5);` => fg

2. `a.substring(2,4)` => cd

Unique Email Address

Problem

Input: [

"test.email+**james**@coding.com",

"test.e.mail+**toto.jane**@coding.com",

"testemail+**tom**@cod.ing.com"]

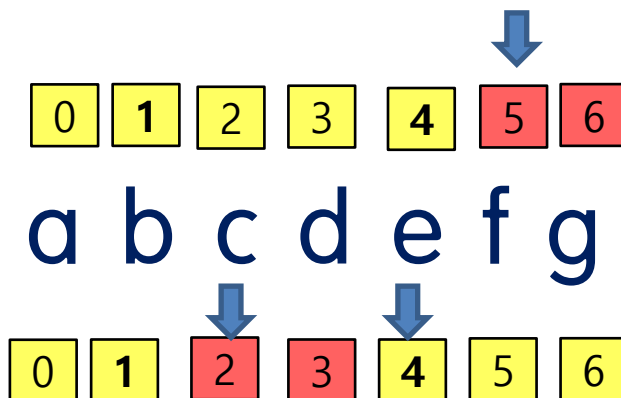
Output: 2

"testemail@coding.com"

"testemail@cod.ing.com"

solution

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



1. a.substring(5); => fg

2. a.substring(2,4) => cd

1. `String stringValueOf = String.valueOf('c'); // most efficient`
2. `String stringValueOfCharArray = String.valueOf(new char[]{x});`
3. `String characterToString = Character.toString('c');`
4. `String characterObjectToString = new Character('c').toString();`
5. `String concatBlankString = 'c' + "";`
6. `String fromCharArray = new String(new char[]{x});`

문제 12) Longest Substring With At Most Two Distinct

Problem

Given a string s , find the length of the longest substring t that contains at most 2 distinct characters.

Example 1:

Input: "eceba"

Output: 3

Explanation: t is "ece" which its length is 3.

Example 2:

Input: "ccaabbb"

Output: 5

Explanation: t is "aabbb" which its length is 5.

Longest Substring With At Most Two Distinct

Problem

Map, two pointer, Math.max

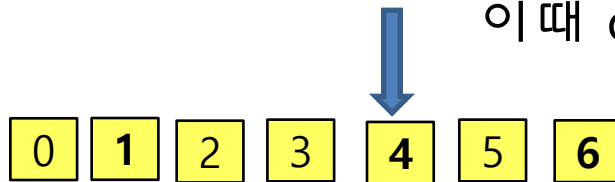
Input : String s = "ccaabbb"

Output: 5

"aabbb" which its length is 5.

solution

1. End-start 로 비교한 값을 length에 담는다.
2. 문자를 2개 인식하기 위해서 Counter로 담는다
이때 counter는 Map



Counter: 문자 갯수

c c a a b b b

length start - end

문제 13) Maximum Subarray

Problem

Given an integer array `nums`, find the contiguous subarray (containing at least one number) which has the largest sum and return its sum.

Example:

Input: `[-2,1,-3,4,-1,2,1,-5,4]`,

Output: 6

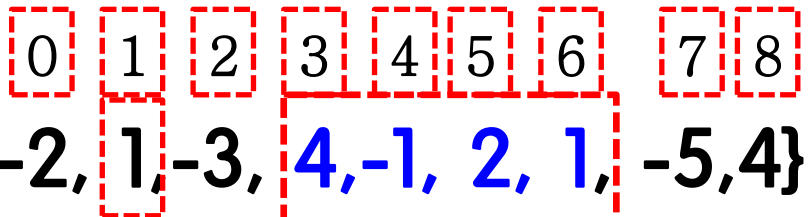
Explanation: `[4,-1,2,1]` has the largest sum = 6.

Maximum Subarray

Problem

Subarray 중에 합이 제일 큰 수 return

Example


`int[] nums = { -2, 1, -3, 4, -1, 2, 1, -5, 4 }`

1. 새로운값=(새로운값(1), 새로운값+SubArray 값(-1)) 비교
2. 1번에서 나온 max값을 따로 저장 = 1

```
newSum = Math.max(새로운값, 기존꺼 합한값)  
Max    = Math.max(newSum, max)
```

Maximum Subarray

Problem

Subarray 중에 합이 제일 큰 수, 1 두개의 합 2 nums[i]

Example

0 1 2 3 4 5 6 7 8
int[] nums = { -2, 1, -3, 4, -1, 2, 1, -5, 4 }

-2									nums[0]: -2	max : -2	
-2+1= -1									newSum+nums[1]: -1	nums[1]: 1	max: 1
1+-3= -2									newSum+nums[2]: -2	nums[2]: -3,	max = 1
-2+4= 2									newSum+nums[3]: 2	nums[3]: 4	max = 4
4+-1= 3									newSum+nums[4]: 3	nums[4]: -1 ,	max = 4
3+-1= 3									newSum+nums[5]: 5	nums[5]: 2 ,	max = 5
5+-1= 3									newSum+nums[6]: 6	nums[6]: 1 ,	max = 6
6+-1= 3									newSum+nums[7]: 1	nums[7]: -5 ,	max = 6
1+4= 5									newSum+nums[8]: 5	nums[8]: 4 ,	max = 6

문제 14) Find Anagrams Mapping

Problem

Given two lists A and B, and B is an anagram of A. B is an anagram of A means B is made by randomizing the order of the elements in A.

We want to find an index mapping P, from A to B. A mapping $P[i] = j$ means the i th element in A appears in B at index j .

These lists A and B may contain duplicates. If there are multiple answers, output any of them.

For example, given

A = [12, 28, 46, 32, 50]

B = [50, 12, 32, 46, 28]

We should return

[1, 4, 3, 2, 0]

as $P[0] = 1$ because the 0th element of A appears at B[1], and $P[1] = 4$ because the 1st element of A appears at B[4], and so on.

Note:

A, B have equal lengths in range [1, 100].

$A[i], B[i]$ are integers in range $[0, 10^5]$.

Find Anagrams Mapping

Problem

```
int[] A = {11, 27, 45, 31, 50};
```

```
int[] B = {50, 11, 31, 45, 27};
```

Output : [1 4 3 2 0]

Example

1. Map을 이용한다. Key, Value를 이용
2. Array B의 순서를 리턴
3. Int[]
4. Map A의 값을 셋팅합니다.
5. 꺼낼 때는 B key 값으로 리턴된 걸 int[]

문제 15) Find All Anagrams

Problem

Given a string *s* and a non-empty string *p*, find all the start indices of *p*'s anagrams in *s*.

Strings consists of lowercase English letters only and the length of both strings *s* and *p* will not be larger than 20,100.

The order of output does not matter.

Example 1:

Input:

s: "cbaebabacd" *p*: "abc"

Output:

[0, 6]

Explanation:

The substring with start index = 0 is "cba", which is an anagram of "abc".

The substring with start index = 6 is "bac", which is an anagram of "abc".

Find All Anagrams

Problem

String txt = "BACDGABCD";

String pat = "ABCD";

Output : [0, 5, 6]

Example

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

B A C D G A B C D A



1. Pattern : ABCD의 아스키값을 Array에 담는다. 65,66,67,68
2. 대상소스(txt)를 비교한다. 이중 for문으로
3. Txt, pat을 같은지 비교한다.

문제 16) SpiralMatrix

설명

Given a matrix of $m \times n$ elements (m rows, n columns),
return all elements of the matrix in spiral order.

입출력

Input:

```
int[][] matrix = {  
    { 1, 2, 3, 4},  
    { 5, 6, 7, 8},  
    { 9,10,11,12}
```

```
};
```

출력

```
[1, 2, 3, 4, 8, 12, 11, 10, 9, 5, 6, 7]
```

제한사항

```
m == matrix.length  
n == matrix[i].length  
1 <= m, n <= 10  
-100 <= matrix[i][j] <= 100
```

문제 Format

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


문제분석

00	1	01	2	02	3	03	4
10	5	11	6	12	7	13	8
20	9	21	10	22	11	23	12

00	1	01	2	02	3	03	4
10	5	11	6	12	7	13	8
20	9	21	10	22	11	23	12

1. 2차원 배열 좌표값 이해
2. 행 (row), 열 (column) 위치 파악
3. 상하좌후 위치 좌표값 변경

문제분석

00	1	01	2	02	3	03	4
10	5	11	6	12	7	13	8
20	9	21	10	22	11	23	12

규칙 찾기

{ **00**, 01, 02, 0**3**}, `int rowStart = 0;`
`int rowEnd = 2 matrix.len`

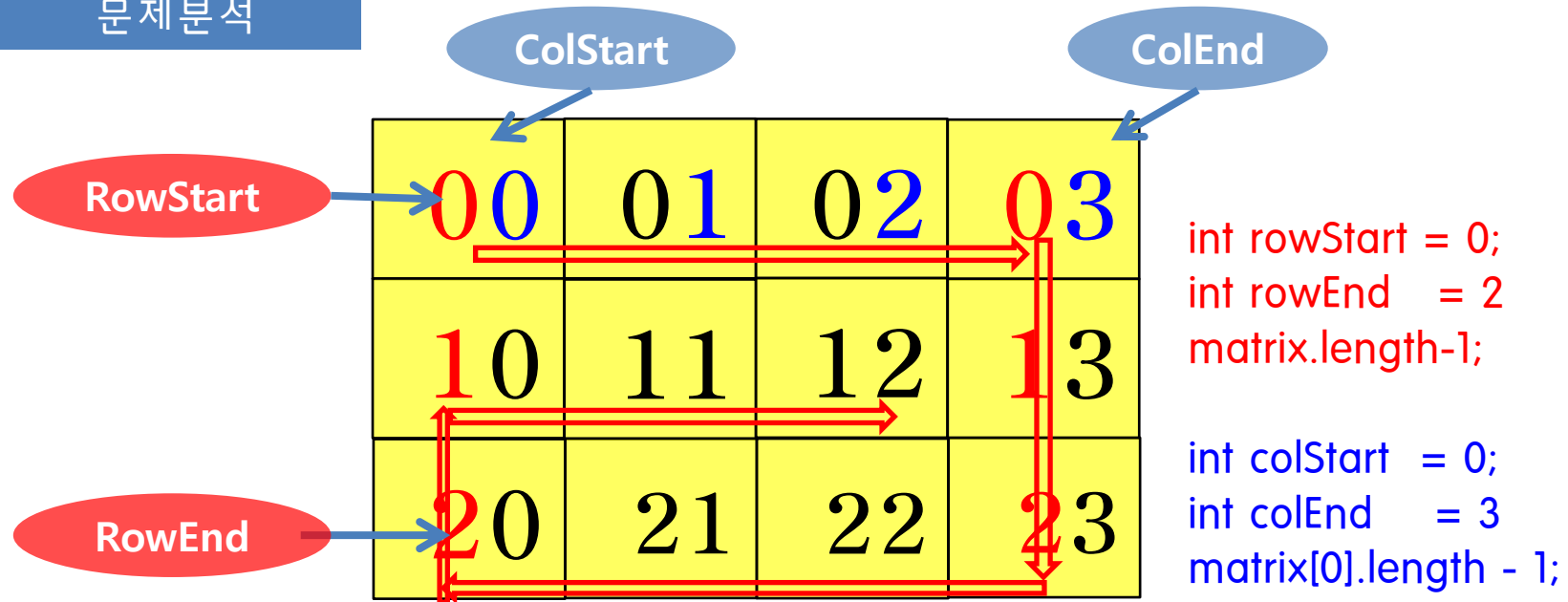
{ 10, 11, 12, 13},

`int colStart = 0;`

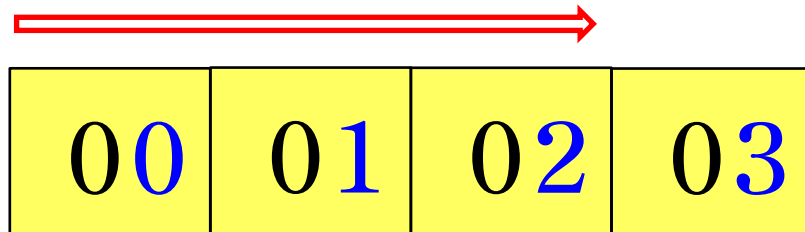
{ **20**, 21, 22, 23 }

`int colEnd = 3 matrix[0].l`

문제분석



right



rowStart = 0는 그대로

int colStart = 0;

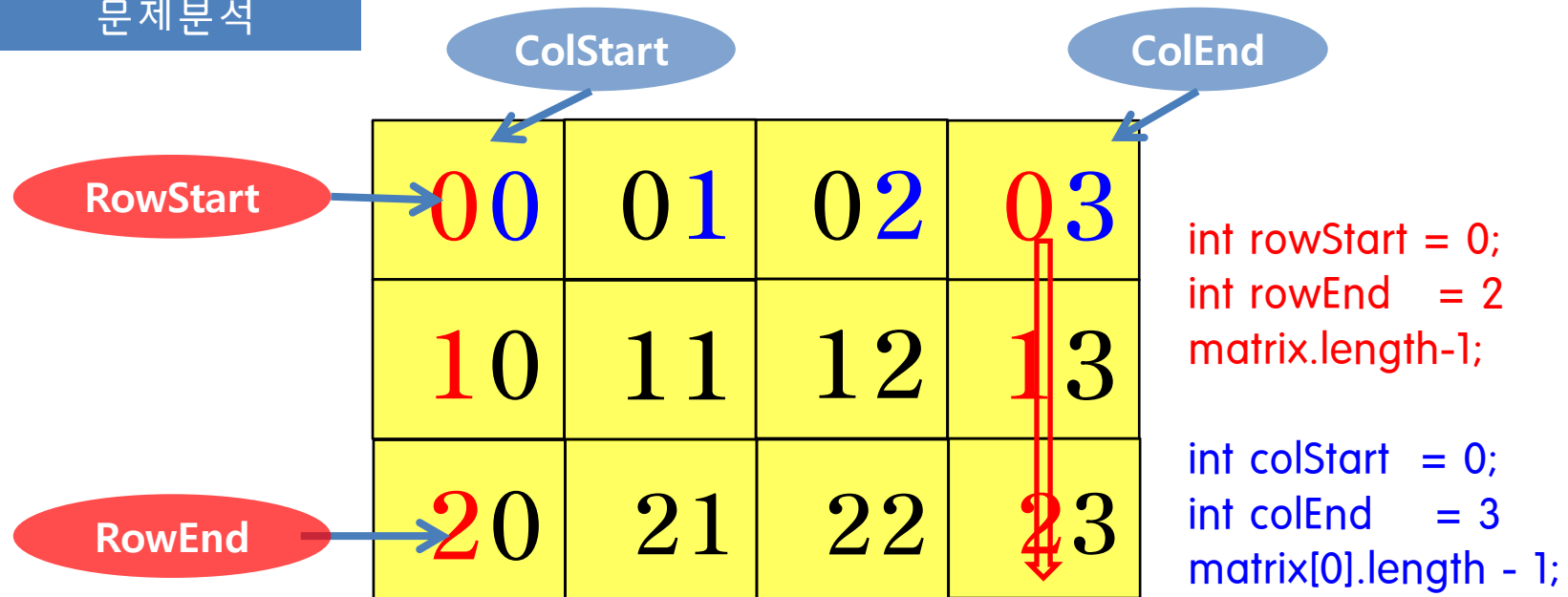
int colEnd = 3

끝나고 rowStart ++

```

//right
for(int i= colStart; i <= colEnd; i++ ) {
    result.add(matrix[rowStart][i]);
}
rowStart++;
    
```

문제분석



down

colEnd = 3는 그대로
 int rowStart = 1
 int rowStart = 2 으로 증가
 끝나고 colEnd --

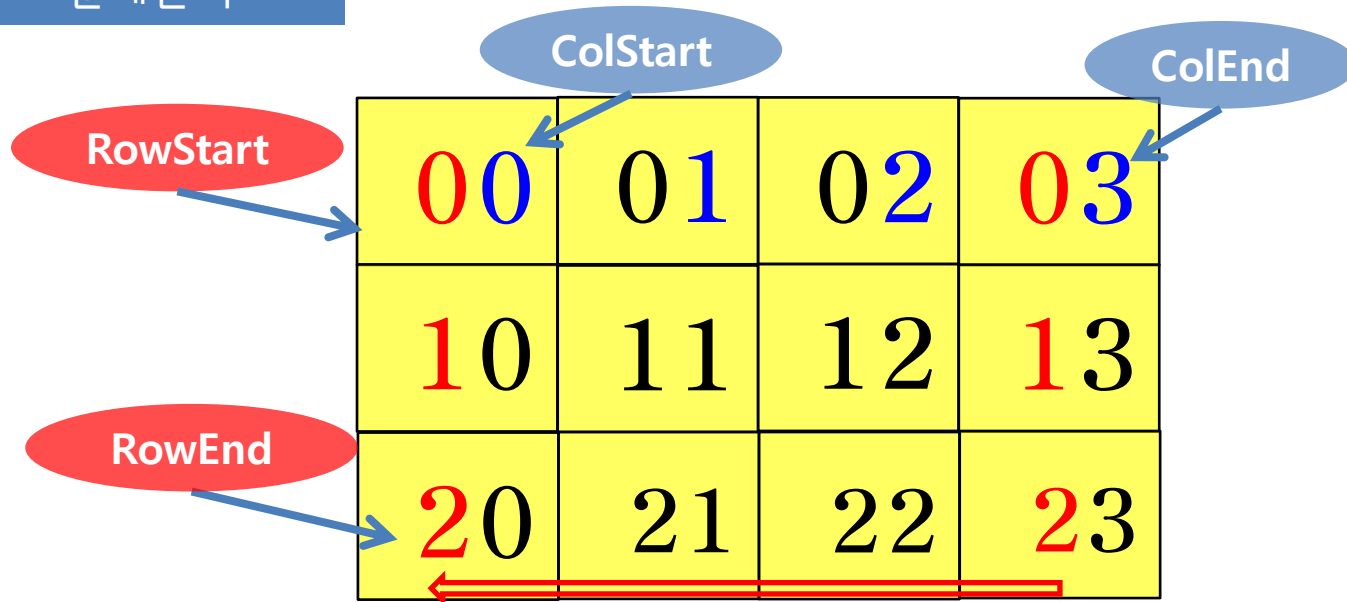
//down

```

for(int i= rowStart; i <= rowEnd; i++ ) {
    result.add(matrix[i][colEnd]);
}
colEnd--;
    
```

13
23

문제분석



left

20	21	22
----	----	----

rowEnd = 2는 그대로

int colEnd = 2

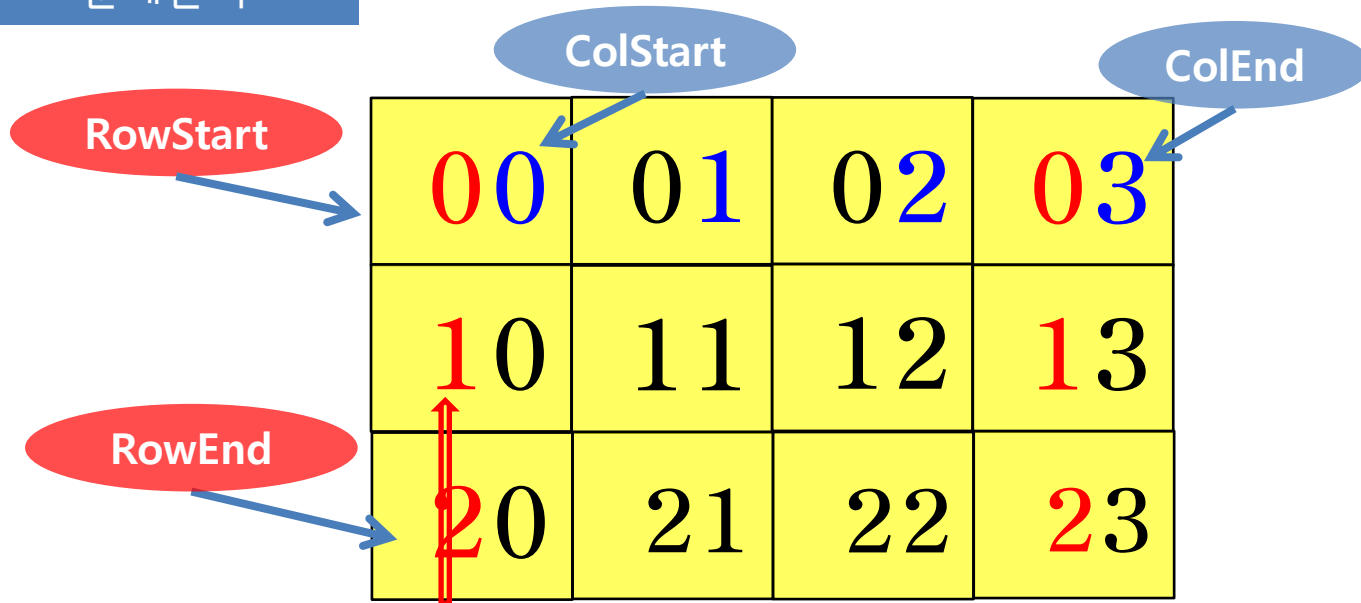
int colStart = 0 감소

While문 안에서 rowStart가 내부적으로 증가

//left

```
if(rowStart <= rowEnd){  
    for(int i= colEnd; i >= colStart; i-- ) {  
        result.add(matrix[rowEnd][i]);  
    }  
}  
rowEnd--;
```

문제분석



While문 안에서 colStart가 내부적으로 증가

up

10

colStart = 0 그대로
int rowEnd = 1
int rowEnd = 0 감소
colStart ++

```
if (colStart <= colEnd) {  
    for (int i = rowEnd; i >= rowStart; i--) {  
        result.add(matrix[i][colStart]);  
    }  
    colStart++;  
}
```

문제 17) Group Anagrams

Problem

Given an array of strings, group anagrams together.

Example:

Input: ["eat", "tea", "tan", "ate", "nat", "bat"],

Output:

```
[  
  ["ate","eat","tea"],  
  ["nat","tan"],  
  ["bat"]  
]
```

Note:

All inputs will be in lowercase.

The order of your output does not matter.

Group Anagrams

Problem

Input: ["eat", "tea", "tan", "ate", "nat", "bat"],

Output:

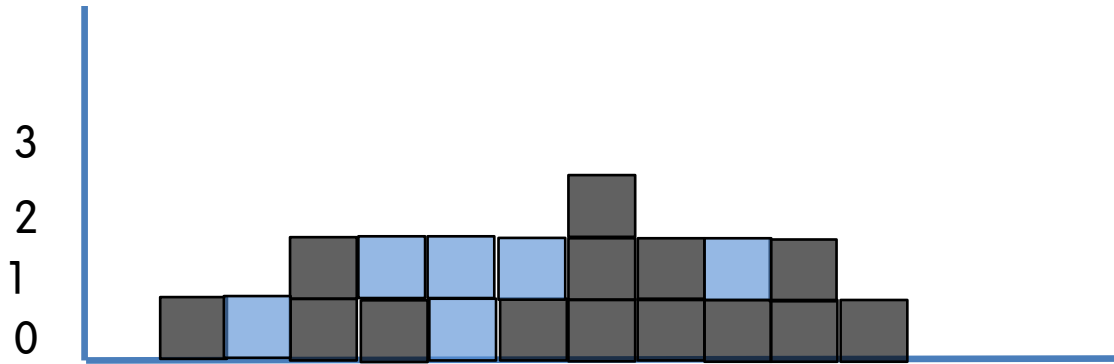
```
[  
  ["ate","eat","tea"],  
  ["nat","tan"],  
  ["bat"]  
]
```

Example

1. Map 이용
2. Anagrams -> 소팅 (key , List<String>)

문제 18) Trapping Rain Water

Problem

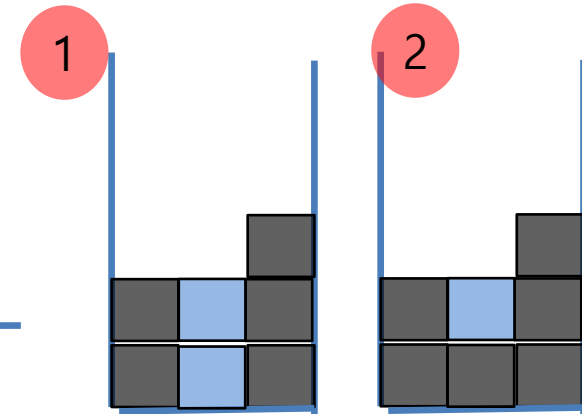
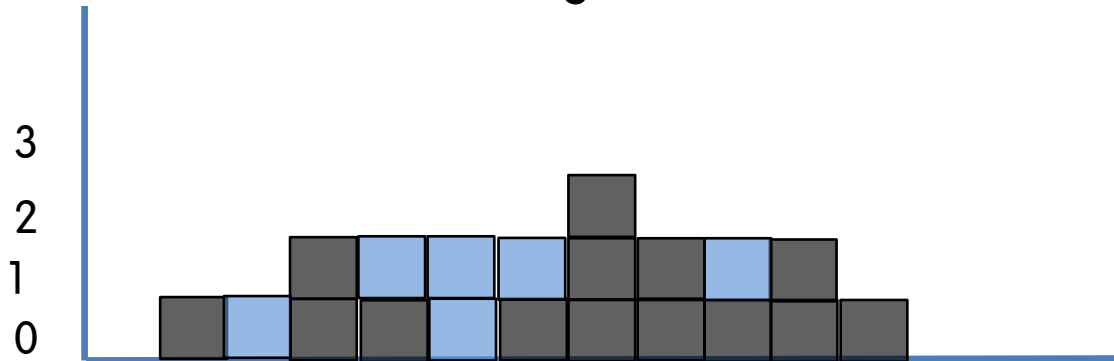


Given n non-negative integers representing an elevation map where the width of each bar is 1, compute how much water it is able to trap after raining.

Trapping Rain Water

Problem

Height : {0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1}



Example

1. int[] left: {0, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3, 3}
2. int[] right: {3, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2, 1}
3. Math.min : {0, 1, 1, 2, 2, 2, 2, 3, 2, 2, 2, 1}
4. Height : {0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1}
5. 0, 0 1 0, 1, 2, 1, 0, 0, 1, 0, 0
1. Math.min(left[i],right[i])-height[i];

설명

1) 물을 부었을때 높이 2
왼쪽벽 2, 오른쪽벽 3
작은값 2

설명

2) 물을 부었을때 높이 1
왼쪽벽 2, 오른쪽벽 3
작은값 2 - 자체높이 1 = 1

문제 19) Kth Largest Element In An Array

Problem

Find the kth largest element in an unsorted array. Note that it is the kth largest element in the sorted order, not the kth distinct element.

Example 1:

Input: [3,2,1,5,6,4] and $k = 2$

Output: 5

Example 2:

Input: [3,2,3,1,2,4,5,5,6] and $k = 4$

Output: 4

Note:

You may assume k is always valid, $1 \leq k \leq \text{array's length}$.

Kth Largest Element In An Array

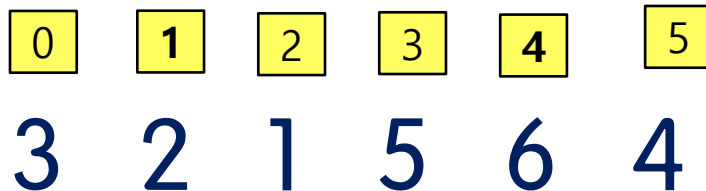
Problem

```
int[] nums = {3,2,1,5,6,4};  
int k =2;
```

Output: 5

solution

1. 배열 오름차순 {1,2,3,4,5,6} $6-2=4$
2. PriorityQueue , k=2



5

6

문제 20) Missing Ranges

Problem

Given a sorted integer array `nums`, where the range of elements are in the inclusive range `[lower, upper]`, return its missing ranges.

Example:

Input: `nums = [0, 1, 3, 50, 75]`, `lower = 0` and `upper = 99`,
Output: `["2", "4->49", "51->74", "76->99"]`

Missing Ranges

Problem

```
int[] nums = {2, 3, 5, 50, 75};  
int lower = 0 ,upper = 99;
```

Output: [0->1, 4, 6->49, 51->74, 76->99]

solution

1. $0 < 2 : \text{lower} < \text{nums}[0] \quad , \quad 0 \rightarrow 1$
2. $2 \ 3 \ , \ 3 \ 5 \quad \text{nums}[i] + 1 < \text{nums}[i+1]$
2~3 인 경우 Pass
3~5 인 경우 4 , $3+1=5-1 \quad \text{nums}[i]+1, \text{nums}[i+1]-1$
3. $75 < 99 \quad \text{nums}[\text{nums.length}-1] < \text{upper}$
 $76 < 99 \quad \text{nums}[\text{nums.length}-1]+1, \text{upper}$

1. 가장 바깥 괄호제거

문제 설명

유효한 괄호(valid parentheses) $S = A+B$ 로 구성되어 있습니다.

A, B는 비어 있지 않은 유효한 괄호 문자열입니다.

S를 분해 할 수 있습니다

$S = P_1 + P_2 + \dots + P_k$, 여기서는 P_i 기본 유효한 괄호 문자열입니다.

예를들어, 문자열 S 는 "", "()", "(()())", "((()()))" 모든 유효한 괄호 문자열입니다.

분해를 해서 모든 기본 문자열의 가장 바깥 쪽 괄호를 제거한 후 return 하세요

입출력

입력 : "(() ()) ()"

출력 : "() () ()"

설명 : 입력 문자열은 "(() ()) ()"이며 primitive 분해하면 "(() ())" + "()"입니다.

각 부분의 바깥 쪽 괄호를 제거한 후 "() ()" + "()" = "() () ()"입니다.

입력 : "() ()"

출력 : ""

설명 :

입력 문자열은 "() ()"이며 기본 분해 "()" + "()"가 있습니다.

각 부분의 바깥 쪽 괄호를 제거한 후 "" + "" = ""입니다.

1. 가장 바깥 괄호 제거

제한사항

S.length <= 10000
S[i]이다 "(" , ")"
S 유효한 괄호 문자열입니다.

문제 Format

```
class Solution{  
    public String solve(String S) {  
  
    }  
}
```


Solution

1. 문제를 정확히 이해
2. 알고리즘 정하고 답을 그릇 정한다
3. for ,While 문 돌리고 그 안에 알고리즘 넣기

I Can Image

- . 내가 생각한거 적고->프로그래밍화(한국말로 생각하고->Java)
결과를 해석하여 이미지화시킨다

1. 구현할 내용을 써보자~

0	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

"(()) (())"

(() ()) (())

(a) ()) ()

문제분석

8	1	0
0	7	6
0	9	0

입력 : "() () ()"
출력 : "() () ()"

8	1	0
0	7	6
0	9	0

1. Dfs/bfs
2. 어느 곳에서도 진입이 가능
3. 큰값을 저장 Math.max

Generate Parentheses

Problem

Input: 3

Output:

```
[  
    "((()))",  
    "(()())",  
    "()(())",  
    "()()()",  
    "())()"  
]
```

1. Dfs 문제

7. 금광찾기

설명

grid크기 의 금광 $m \times n$ 에서 각 cell은 금의양을 나타내는 Integer 값입니다.
셀이 0인 경우는 비어있다는 뜻이고, 아래조건에서 수집 할 수있는 최대 금을 구하세요

1. 위치한 셀에서 모든 금을 가질 수 있습니다.
2. 자신의 위치에서 왼쪽, 오른쪽, 위, 아래로 한칸 이동 가능.
3. 같은 셀을 두 번 이상 방문 할 수 없습니다.
4. 0이 있는 셀은 방문할 수 없습니다.
5. 모든 위치에서 금 수집을 시작하고 중지 할 수 있습니다 ..

입출력

Input: grid= $\begin{bmatrix} 8 & 1 & 0 \\ 0 & 7 & 6 \\ 0 & 9 & 0 \end{bmatrix}$

Output: 25

제한사항

$m == \text{grid.length}$
 $n == \text{grid}[i].\text{length}$
 $1 \leq m, n \leq 10$
 $0 \leq \text{grid}[i][j] \leq 100$
최대 25 cells 만 금을 보유함

문제 Format

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

Solution

1. 문제를 정확히 이해
2. 알고리즘 정하고 답을 그릇 정한다
3. for ,While 문 돌리고 그 안에 알고리즘 넣기

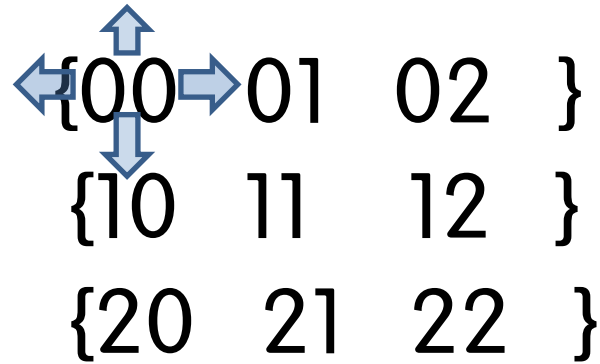
I Can Image

- . 내가 생각한거 적고->프로그래밍화(한국말로 생각하고->Java)
결과를 해석하여 이미지화시킨다

1. 구현할 내용을 써보자~

문제분석

8	1	0
0	7	6
0	9	0



8	1	0
0	7	6
0	9	0

1. Dfs/bfs
2. 어느 곳에서도 진입이 가능
3. 큰값을 저장 Math.max
4. Visited[][]

1 맞는 조건을 찾아내는 부분

```
for(int i = 0; i < m; i++) {  
    for(int j = 0; j < n; j++) {  
        int len = dfs(matrix, i, j, m, n, result);  
        max = Math.max(max, len);  
    }  
}
```

2 방향 설정 & 이차원배열 사이즈

```
int m, n;  
m = grid.length;  
n = grid[0].length;  
int[][] dirs = { { -1, 0 }, { 1, 0 }, { 0, 1 }, { 0, -1 } };
```


3

재귀를 이용한다(stack개념)

4

조건체크해서 재귀적으로 호출

1. 마이너스 좌표체크 2. m*n 범위체크 3. grid[x][y] 값체크(문제제시값)

```
public int dfs(int[][] matrix, int i, int j, int m, int n, int[][] result) {  
    if (result[i][j] != 0) return result[i][j];  
    int max = 1;  
    for (int[] dir : dirs) {  
        int x = i + dir[0] ; int y = j + dir[1];  
        if (x < 0 || x >= m || y < 0 || y >= n || matrix[x][y] <= matrix[i][j])  
            continue;  
        int len = 1 + dfs(matrix, x, y, m, n, result);  
        max = Math.max(max, len);  
    }  
    result[i][j] = max;  
    return max;  
}
```

5

만족하는 시점에서 다시 dfs()호출하여 파고들어 간다

```
public int dfs(int[][] matrix, int i, int j, int m, int n, int[][] result) {  
    if (result[i][j] != 0) return result[i][j];  
    int max = 1;  
    for (int[] dir : dirs) {  
        int x = i + dir[0] ; int y = j + dir[1];  
        if (x < 0 || x >= m || y < 0 || y >= n || matrix[x][y] <= matrix[i][j])  
            continue;  
        int len = 1 + dfs(matrix, x, y, m, n, result);  
        max = Math.max(max, len);  
    }  
    result[i][j] = max;  
    return max;  
}
```

1. DP(House Robber)

Problem

Input: int[] nums = [2,7,9,3,1,8]

Output: 19 (2+9+1+8)

인접한 두 집에 침입하면 경찰에 자동으로 연락된다
각 House의 금액은 자연수로 주어진다
경찰에 걸리지 않고 오늘 밤 도둑질 할 수있는 최대 금액은 ?

문제 Format

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

1. DP(House Robber)

Problem

Input: int[] nums = [2,7,9,3,1,8]

Output: 19 (2+9+1+8)

인접한 두 집에 침입하면 경찰에 자동으로 연락된다
각 House의 금액은 자연수로 주어진다
경찰에 걸리지 않고 오늘 밤 도둑질 할 수있는 최대 금액은 ?

문제 Format

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

Dp

Dynamic Programming

	0	1	2	3	4	5
nums	2	7	9	3	1	8

Dp 문제를 알아내는 방법

1. 이미 셋팅 된 값에서 규칙을 찾는다.(새로운 어떤값을 찾는게 아님)
2. 2번방을 기준으로 잡는다
3. 2번방은 11이 나와야 함 $11 = \max(7, 11(1+7))$

	0	1	2	3	4	5	6
DP 배열	0	2	7	11	11	12	19

```
dp[i + 1] = Math.max(dp[i], dp[i - 1] + val);
```

Dp

Dynamic Programming



0	1	2	3	4	5
2	7	9	3	1	8

nums

*공식 Dp 문제를 알아내는 방법

1. 2번방을 기준으로 잡는다 (0번방, 1번방은 셋팅값)
2. 2번방은 11이 나와야 함

$$11 = \max(7, 11(2+9))$$

3번방 입장에서는

$$11 = \max(11, 10(7+3))$$

4번방 입장에서는

$$12 = \max(11, 12(11+1))$$

5번방 입장에서는

$$19 = \max(12, 19(12+8))$$

식물심기(

Problem

- 당신은 n 에서 라벨 정원 1에를 n , 및 배열이 $paths$ 곳 의 정원 사이에 양 방향 경로를 설명 정원 . 각 정원에 4 가지 종류의 꽃 중 하나를 심고 싶습니다. $paths[i] = [x_i, y_i]$

모든 정원에는 들어 오거나 나가는 경로가 최대 3 개 있습니다.

당신의 임무는 경로로 연결된 두 개의 정원에 대해 다른 유형의 꽃을 갖도록 각 정원에 대한 꽃 유형을 선택하는 것입니다.

배열과 같은 선택 항목을 반환 합니다 $answer$. 여기서는 정원에 $answer[i]$ 심은 꽃의 유형 입니다. 꽃 유형이 표시되어 , , , 또는 . 대답이 존재한다는 것이 보장됩니다. $(i+1)th$ 1234

Example

Input: $n = 3$, $\text{paths} = [[1,2],[2,3],[3,1]]$

Output: $[1,2,3]$

Explanation:

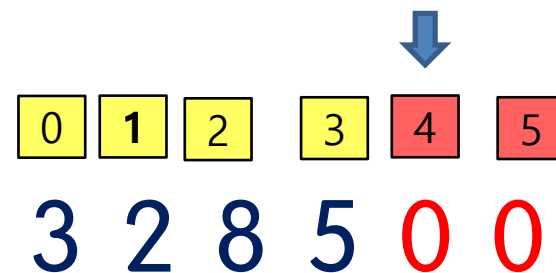
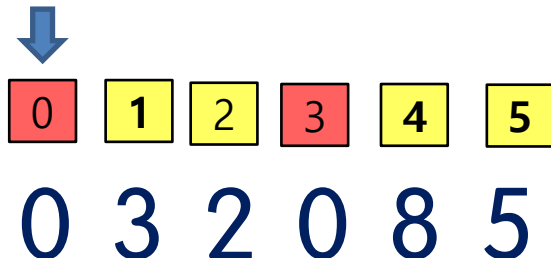
Gardens 1 and 2 have different types.

Gardens 2 and 3 have different types.

Gardens 3 and 1 have different types.

Hence, $[1,2,3]$ is a valid answer. Other valid answers include $[1,2,4]$, $[1,4,2]$, and $[3,2,1]$.

Note



2차원배열 재구성

Problem

음이 아닌 정수의 두 배열 rowSum 및 colSum이 제공됩니다.
여기서 rowSum [i]는 i 번째 행에있는 요소의 합계이고 colSum [j]는 2D 행렬의 j 번째 열에있는 요소의 합계입니다.

즉, 행렬의 요소는 모르지만 각 행과 열의 합계는 알고 있습니다.
rowSum 및 colSum 요구 사항을 충족하는 rowSum.length x colSum.length 크기의 음이 아닌 정수로 구성된 행렬을 찾습니다.

요구 사항을 충족하는 모든 행렬을 나타내는 2D 배열을 반환합니다.
요구 사항을 충족하는 매트릭스가 하나 이상 존재한다는 것이 보장됩니다

2) 2차원배열 재구성

Array

Problem

다음과 같은 n 열과 2행 이있는 행렬의 세부 정보가 제공됩니다 .

- 행렬은 binary matrix입니다. 즉, 행렬의 각 요소는 0또 1.
- 0 번째 (위) 행의 요소 합계는 upper.
- 첫 번째 (하위) 행의 요소 합계는 lower.
- i 번째 열 (0 인덱스)에있는 요소의 합은 colsum[i].
여기서는 colsum길이가 정수 배열로 제공됩니다

행렬을 재구성하는 것입니다 upper, lower하고 colsum.

2 차원 정수 배열로 반환합니다.

유효한 솔루션이 둘 이상있는 경우 둘 중 하나가 허용됩니다.

유효한 솔루션이 없으면 빈 2 차원 배열을 반환합니다.

예제

예 1 :

입력 : 위쪽 = 2, 아래쪽 = 1, 열 합계 = [1,1,1]

출력 : [[1,1,0], [0,0,1]]

설명 : [[1,0,1], [0,1,0]], [[0,1,1], [1,0,0]]도 정답

예 2 :

입력 : 상위 = 2, 하위 = 3, colsum = [2,2,1,1]

출력 : []

Note

$1 \leq \text{colsum.length} \leq 10^5$
 $0 \leq \text{upper}, \text{lower} \leq \text{colsum.length}$
 $0 \leq \text{colsum}[i] \leq 2$

문제포맷

```
class Solution {  
    public List<List<Integer>> solve(int upper, int lower, int[] colsum) {  
  
    }  
}
```

Dynamic Programming 개념 설명

1, 1, 2, 3, 5, 8, 13, 21, 34, 55

$$f[2] = f[1] + f[0]$$

$$f[i] = f[i-1] + f[i-2]$$

1. 특정 규칙을 찾아내서 적용한다. 재사용성 이다.
2. 구간의 값을 배열을 이용하여 저장한다 (memoization) => 동일 구간 계산시 이용
3. Top Down 방식과 Bottom Up 방식 (memoization)이있다.

Dp 문제를 알아내는 방법

1. 각 구간에서 규칙이 동일한 값을 찾아낸다 (예) $f[i] = f[i-1] + f[i-2]$
2. dp배열을 만들어서 Bottom Up 방식으로 해당 공식을 접근한다.

sample) Fibonacci Number

Problem

$$F(0) = 0,$$

$$F(1) = 1$$

$$F(n) = F(n - 1) + F(n - 2), \text{ for } n > 1.$$

Example

Input: $n = 3$

Output: 2

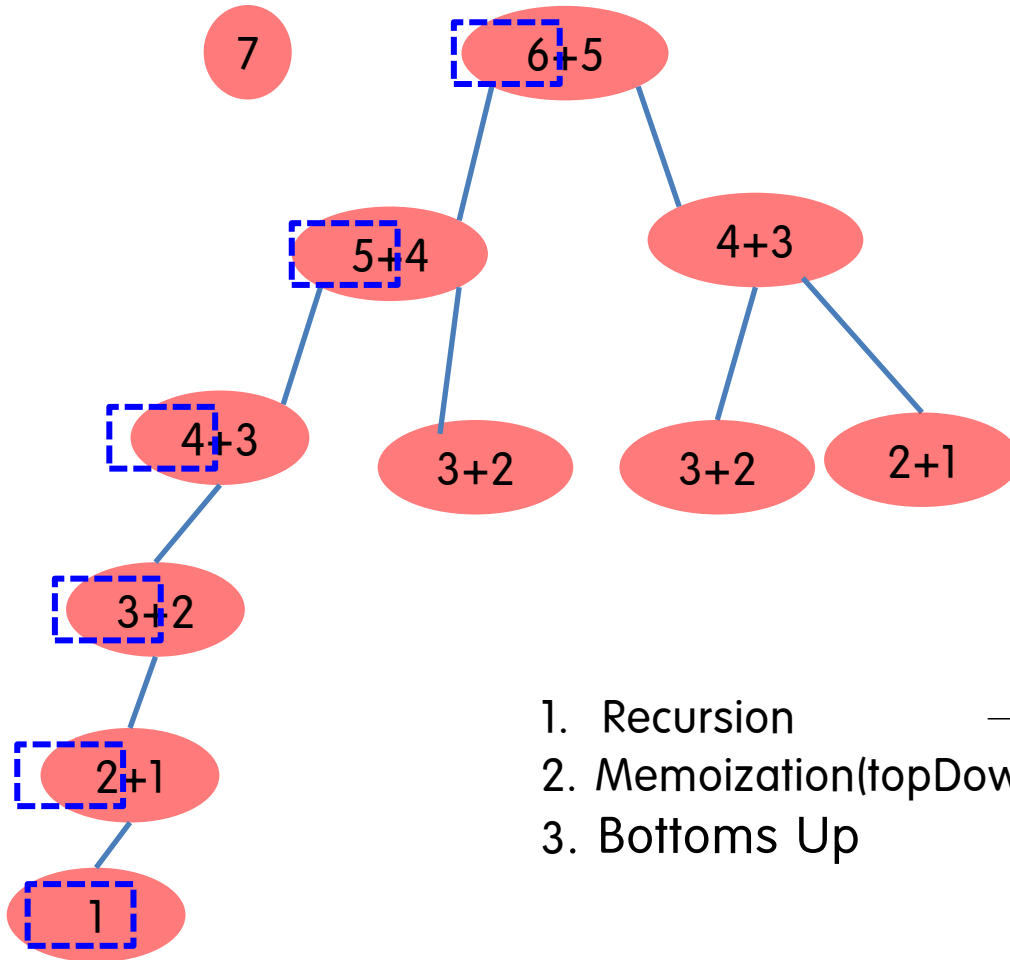
Explanation: $F(3) = F(2) + F(1) = 1 + 1 = 2$

0 1 1 2 3 5 8 13 21

Fibonacci Number

Solution

```
fib[i] = fib[i - 1] + fib[i - 2];
```



1. Recursion — Big $O(2^N)$
2. Memoization(topDown) — $O(N)$ — time, $O(N)$ — space
3. Bottoms Up — $O(N)$ — time, $O(1)$ — space

2. DP

Problem

'A' -> "1"
'B' -> "2"
... 'Z' -> "26"

문자 -> 숫자 (인코딩)
숫자 -> 문자 (디코딩)

BBTE -> (2 2 20 5)
UJE -> (22 20 5)

문제 Format

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

Input : String s = "12"

Output : 2

"12"는 "A,B" => (1, 2) 또는 "L" => (12)로 디코딩

Input : String s = "121"

Output : 3

1,2,1 / 2,12 / 21, 1

A,B,A / B,L / U,A

Input String s = "3621"

Output: 2

3,6,2,1 / 1,21

Input String s = "08"

Output: 0

Solution

String

1. 문제를 정확히 이해
2. 알고리즘 정하고 답을 그릇 정한다
3. for ,While 문 돌리고 그 안에 알고리즘 넣기

I Can Image

- . 내가 생각한거 적고->프로그램화(한국말로 생각하고->Java)
결과를 해석하여 이미지화시킨다

1. 구현할 내용을 써보자~

Dp

Dynamic Programming

A	B	C	D	E	F	G	H	I	J	K	L	M
1	2	3	4	5	6	7	8	9	10	11	12	13
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
14	15	16	17	18	19	20	21	22	23	24	25	26

1 2

답 2개, 1,2/ 2 12

3 0

답 0개

1 2 1

답 3개

1 2 1 1

답 5개

DP Array로 특정한 규칙을 담는다

Dp

Dynamic Programming

$$dp[i] = dp[i-1] + dp[i-2]$$

ex1

0	1
1	1



2

DP 초기화,
값 1

ex3

0	1	2
1	0	0

0 8

DP

ex2

0	1	2
1	1	2

DP 초기화,
값 2
1, 2 / 12

DP

1 2

Dp

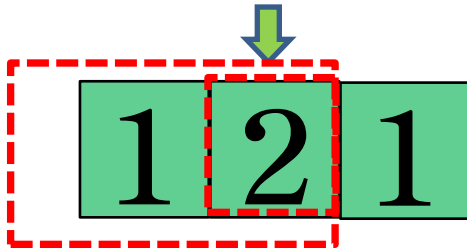
Dynamic Programming

$$dp[i] = dp[i-1] + dp[i-2]$$

ex4

DP

0	1	2	3
1	1	0	0



Input : String s = "121"

Output : 3

1,2,1 / 2,12 / 21, 1

A,B,A / B,L / U,A

Input : String s = "129"

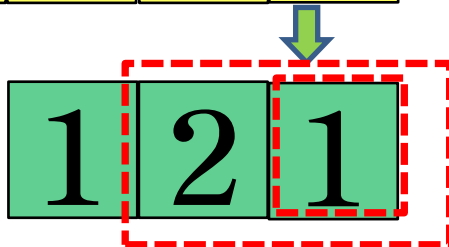
Output : 2

1,2,9 / 2,12 / **29**, 9

A,B,I / B,L / **X**,I

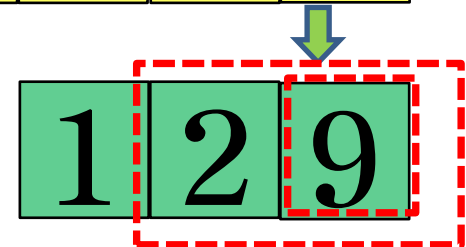
DP

1	1	2	3
---	---	---	---



DP

1	1	2	2
---	---	---	---



Dp

Dynamic Programming

$$dp[i] = dp[i-1] + dp[i-2]$$

DP

0	1	2	3
1	1	0	0

First 2
second 12

1	2	2
---	---	---

DP

1	1	2	3
---	---	---	---

1	2	1
---	---	---

DP

1	1	2	3	5
---	---	---	---	---

1	2	1	1
---	---	---	---