1. maxProfit (Math.max())

Problem

- i번째 요소가 i 일에 주어진 주식의 가격 인 배열이 있다고 가정합니다 .

- 최대 한번의 거래만 완료하도록 허용 된 경우
  (즉, 주식을 하나 사고, 한 주식을 매도)
  최대 수익을 찾는 알고리즘을 설계하십시오.

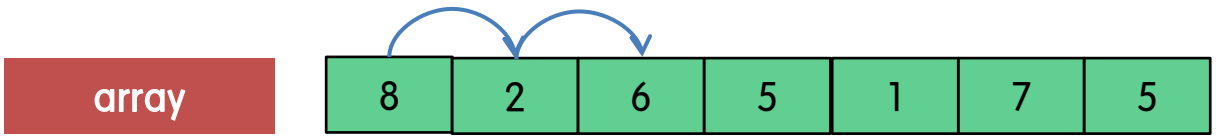
- 주식을 구입하기 전에는 주식을 팔 수 없습니다.

Example

입력 : { 8, 2, 6, 5, 1,7, 5 }
출력 : 6
설명 : 5 일에 구매 (가격 = 1), 6 일에 판매 (가격 = 7), 이익 = 7-1 = 6

Note

1. maxProfit (Math.max())



분석

1. 8에 샀으면 2를 만났을때 버린다
2. 제일 작은값으로 셋팅해야한다.

3. 값6을 만나면 - 값2를 빼서 저장 4
4. 값5를 만나면 - 값2를 빼서 저장 3
5. Math.max를 이용해서 max값 저장

2. subArraySum (sum+=, Map)

Problem

Integer array nums, k 가 있다.
return the total number of continuous subarrays whose sum equals to k.

Example

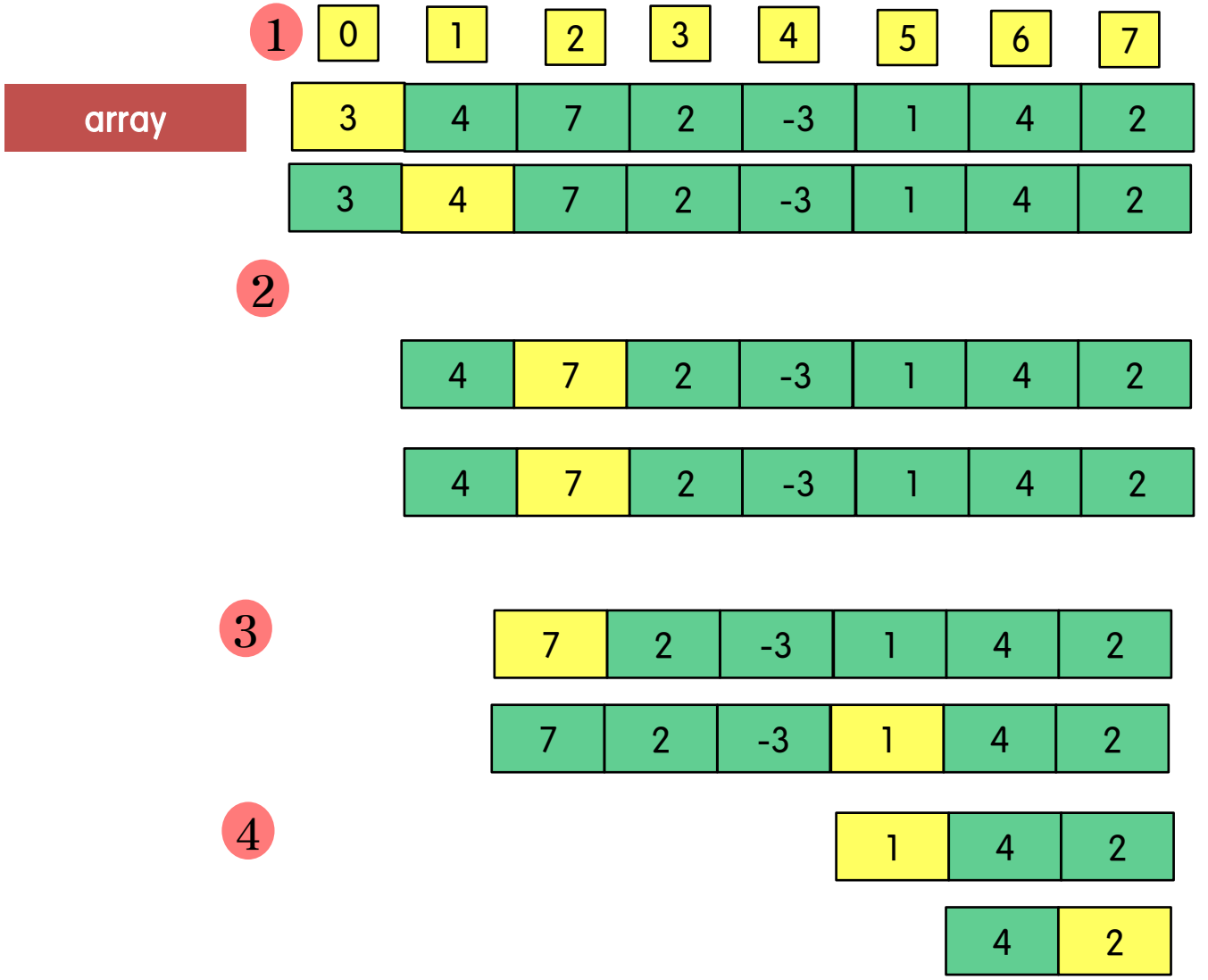
입력 : int[] nums= { 3, 4, 7, 2, -3, 1, 4, 2 };

int k=7

출력 : 4

Note

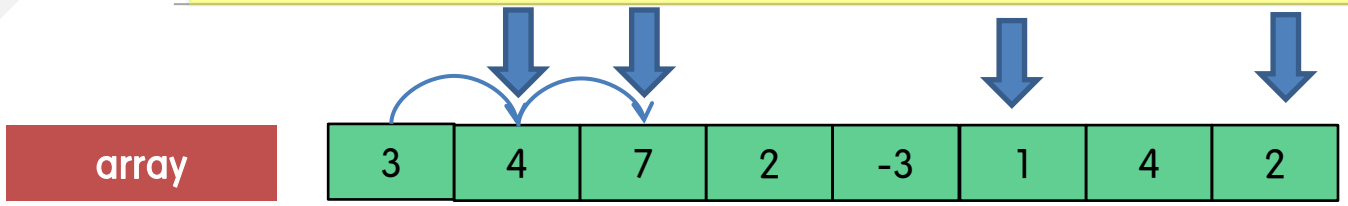
2. subArraySum (sum+=)




```
int sum =0;
sum += nums[end]
sum = sum +nums[end]
```

1. 3+4를 했을때 7이므로
2. 7이 나오므로 count++
3. 모든 단계에 값을 넣은후
   나온값이 7이면 count

2. subArraySum (Map)



기준점 - 7 = Map의 키

k = 7

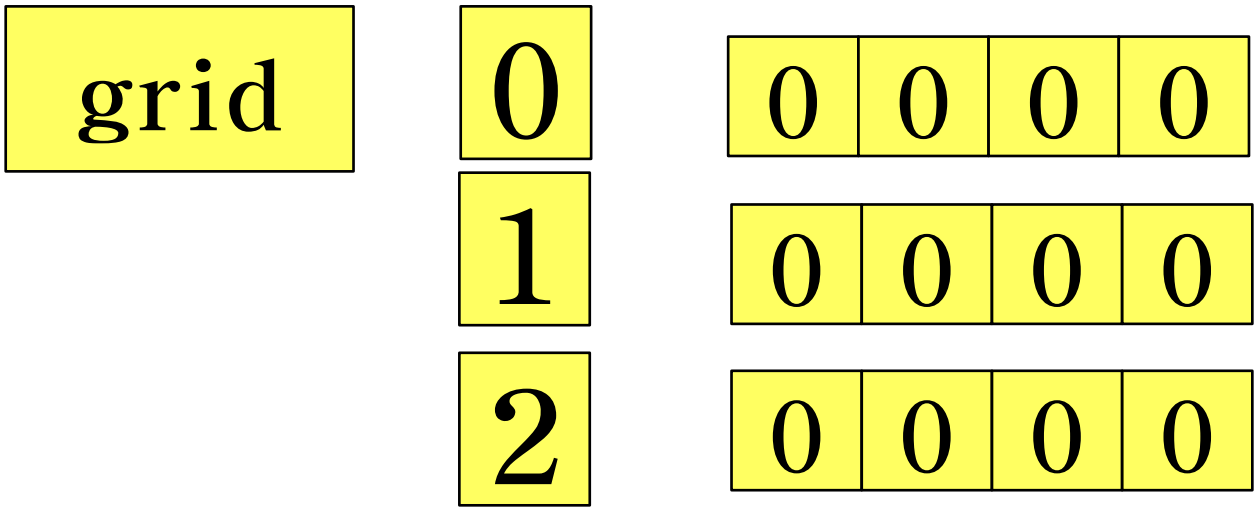
map

	Key	Value	sum	Sum-k	count
	0	1	1	0	0
3	3	1	3	-4	0
4	7	1	7	0	1
7	14	1	14	7	2
2	16	1	16	9	2
-3	13	1	13	6	2
1	14	2	14	7	3
4	18	1	18	11	3
2	20	1	20	13	4

1. 이차원배열

2DArray

1 `int[ ][ ] grid = new int[3][4]`



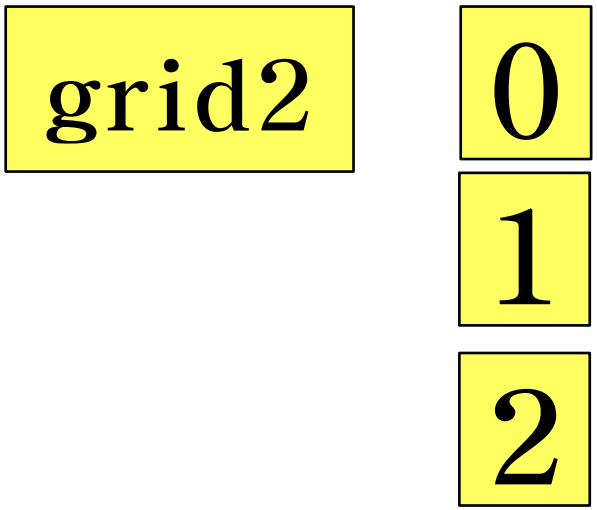
`grid[0] =10;`(X, int배열을 담을 수 있는 참조값)

`grid[0][1] =10;`(0)

1. 이차원배열 Array

2

```
int[ ][ ] grid2 = new int[3][ ]
```



1. 이차원배열 Array

2

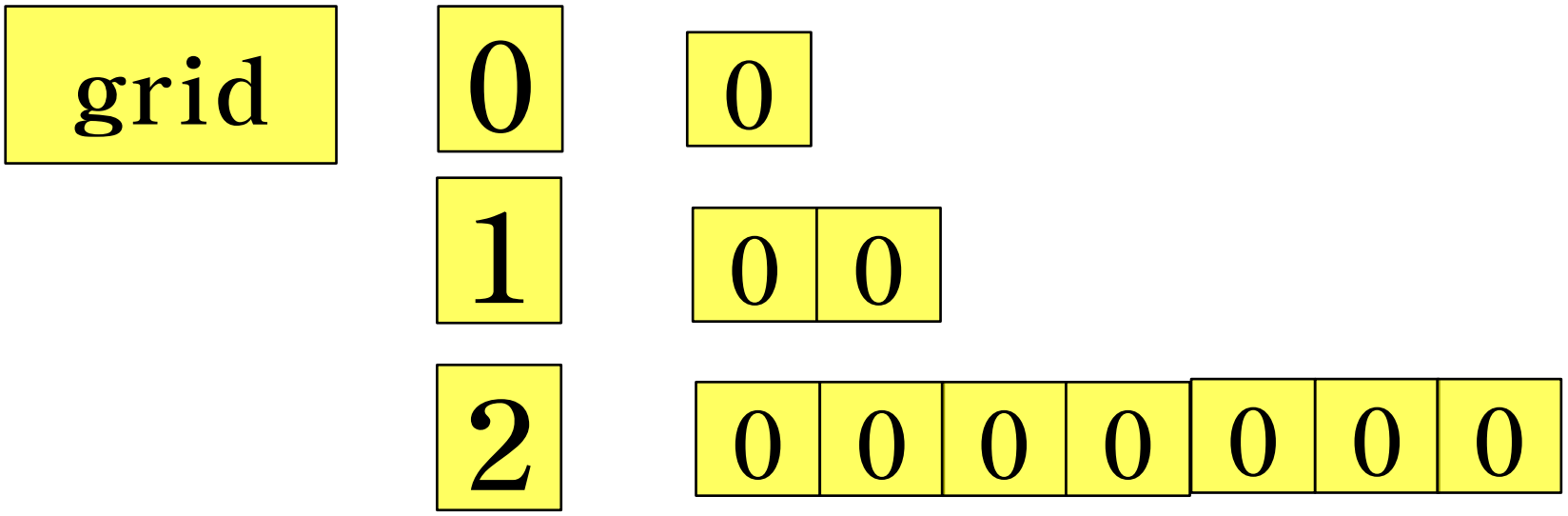
```
int[ ][ ] grid2 = new int[3][ ]
grid2[0] = new int[1];
grid2[0][0] = 100;
```

grid2 | 0 | 100

1. 이차원배열 Array

2

```
int[ ][ ] grid = new int[3][ ]
grid[0] = new int[1];
grid[1] = new int[2];
grid[2] = new int[7];
```



1. Matrix Zeros

Problem

- 주어진 행렬 m*n. 요소가 0 이면 전체 행과 열을 0으로 설정 합니다.

Example

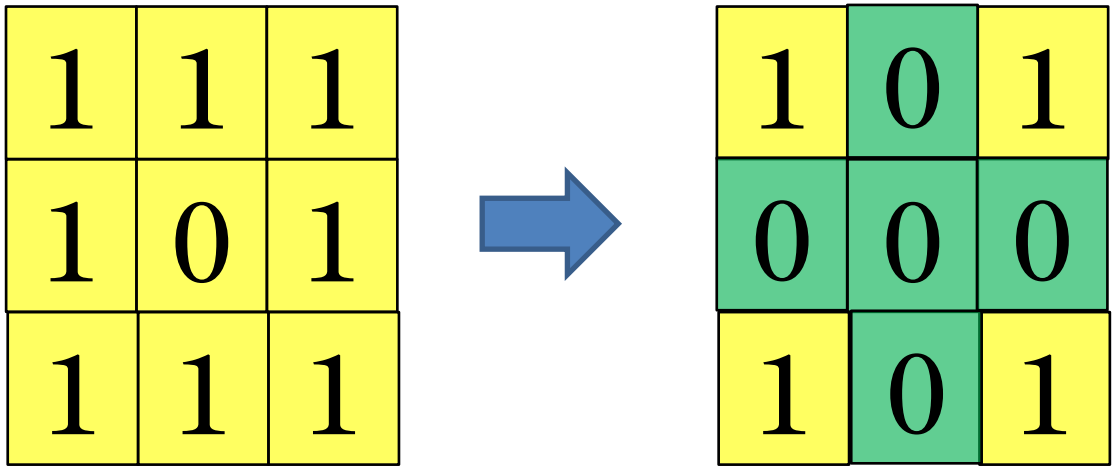
Input: matrix = [[1,1,1],
[1,0,1],
[1,1,1]]

Output: [[1,0,1],
[0,0,0],
[1,0,1]]

Note

2. Solution

Array



생각
1. 이차원배열
2. 0의 위치를 알아낸다(좌표값 1,1)
3. 그 좌표를 알아내서 (row 1, col 1)

Ds 고유한 좌표값 저장= Hashset
For, while

2. Solution

Array

{ 00, 01, 02 },
{ 10, 11, 12},
{ 20, 21, 22 }

{ 00, 01, 02 },
{ 10, 11, 12},
{ 20, 21, 22 }

{ 00, 01, 02 },
{ 10, 11, 12},
{ 20, 21, 22 }