고 급 문 제 해 결

문제 5.18

4월7일 발표 자료

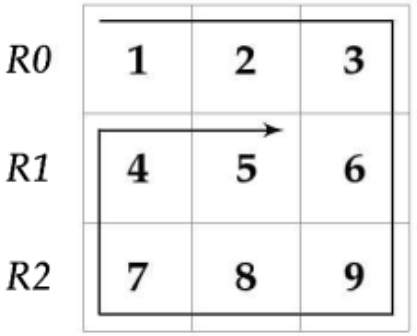
2018117543 한재성

문제 5.18:
2차원 배열 나선형으로 읽기

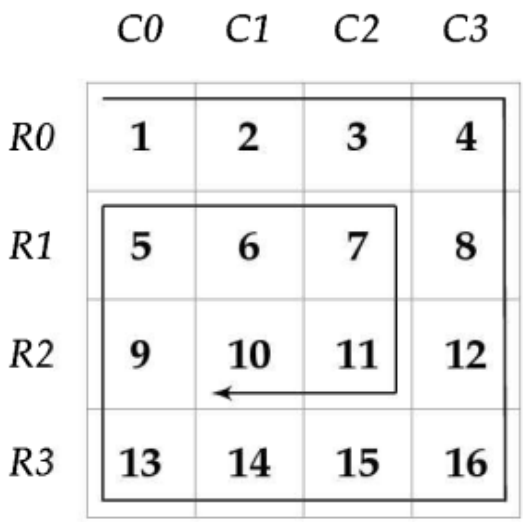
5.18: 2차원 배열 나선형으로 읽기

6.17 COMPUTE THE SPIRAL ORDERING OF A 2D ARRAY

A 2D array can be written as a sequence in several orders—the most natural ones being row-by-row or column-by-column. In this problem we explore the problem of writing the 2D array in spiral order. For example, the spiral ordering for the 2D array in Figure 6.3(a) is $\langle 1,2,3,6,9,8,7,4,5\rangle$. For Figure 6.3(b), the spiral ordering is $\langle 1,2,3,4,8,12,16,15,14,13,9,5,6,7,11,10\rangle$.

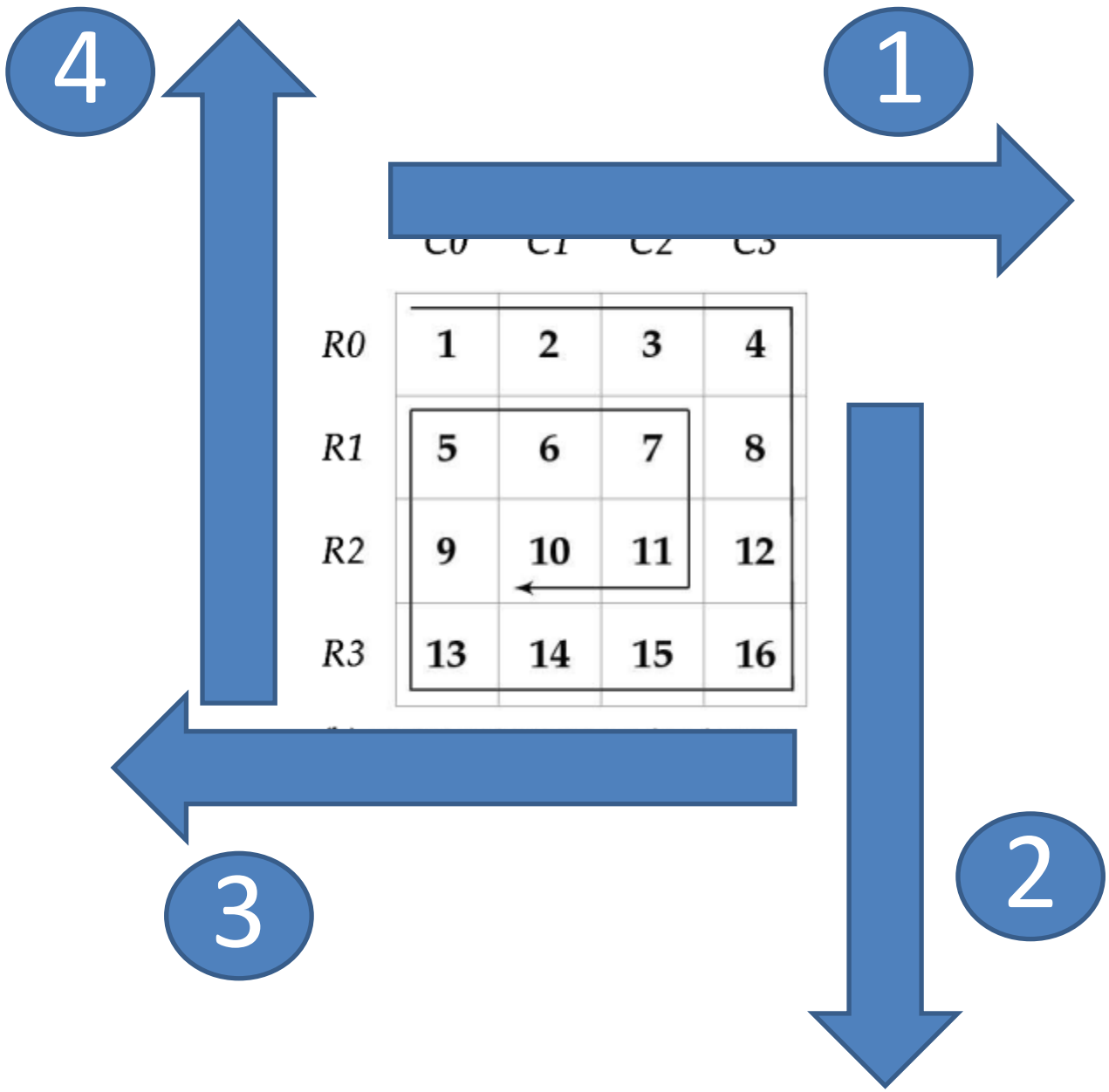


(a) Spiral ordering for a 3 × 3 array.



(b) Spiral ordering for a 4 × 4 array.

Iterative



initialization

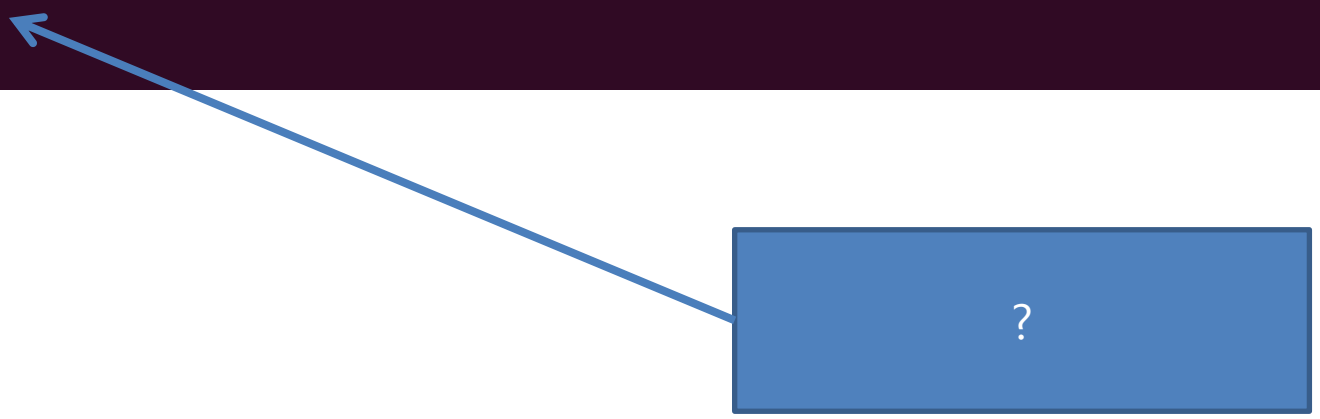
```
# Array initialization
n = 3
A = [[0] * n for _ in range(n)]
k = 1
for i in range(n):
    for j in range(n):
        A[i][j] = k
        k += 1

visit_spiral()
```

code version #1

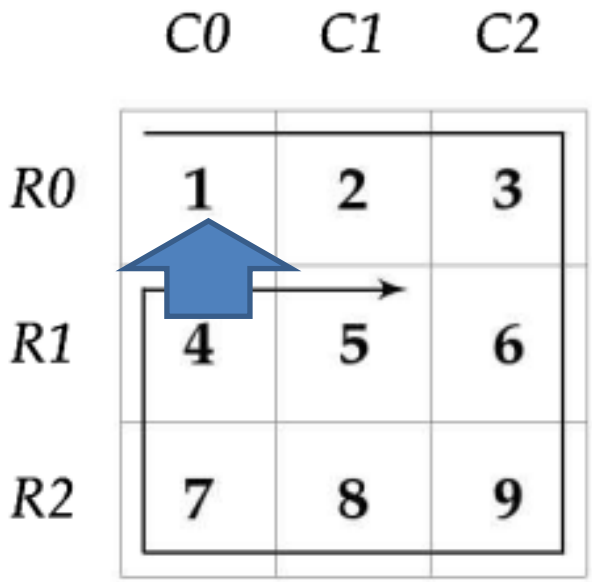
```
def visit_spiral1(n):
    # right, down, left, up
    dirs = [(0, 1), (1, 0), (0, -1), (-1, 0)]
    idx = 0
    cnt = 0
    i = j = 0
    while(True):
        print(i, j, A[i][j])
        cnt += 1
        if cnt >= n * n:
            break
        next_i = i + dirs[idx][0]
        next_j = j + dirs[idx][1]
        while not (0 <= next_i < n and 0 <= next_j < n):
            idx = (idx + 1) % len(dirs)
            next_i = i + dirs[idx][0]
            next_j = j + dirs[idx][1]
        i = next_i
        j = next_j
```

result of code version #1

```
jason@ThinkPad:~/workspace/advalg/ch5$ python3 5.18_1.py
0 0 1
0 1 2
0 2 3
1 2 6
2 2 9
2 1 8
2 0 7
1 0 4
0 0 1
--
```



문제점 파악



(a) Spiral ordering for a 3 × 3 array.

```
next_i = i + dirs[idx][0]
next_j = j + dirs[idx][1]
while not (0 <= next_i < n and 0 <= next_j < n):
    idx = (idx + 1) % len(dirs)
    next_i = i + dirs[idx][0]
    next_j = j + dirs[idx][1]
i = next_i
j = next_j
```

문제점:
방향만 고려하기 때문에 OutOfBound가 아니라면 무조건 기존 방향 으로만 진행함

code version #2 - visit 처리

```
def visit_spiral2(n):
    # right, down, left, up
    dirs = [(0, 1), (1, 0), (0, -1), (-1, 0)]
    idx = 0
    cnt = 0
    i = j = 0
    while(True):
        print(i, j, A[i][j])
        A[i][j] = -1 # Mark as visited
        cnt += 1
        if cnt >= n * n:
            break
        next_i = i + dirs[idx][0]
        next_j = j + dirs[idx][1]
        while not (0 <= next_i < n and 0 <= next_j < n) or A[next_i][next_j] == -1:
            idx = (idx + 1) % len(dirs)
            next_i = i + dirs[idx][0]
            next_j = j + dirs[idx][1]
        i = next_i
        j = next_j
```

code version #3 - visit 처리

```
def visit_spiral3(n):
    # right, down, left, up
    dirs = [(0, 1), (1, 0), (0, -1), (-1, 0)]
    idx = 0
    cnt = 0
    i = j = 0
    visited = set()
    while(True):
        print(i, j, A[i][j])
        visited.add((i,j))
        cnt += 1
        if cnt >= n * n:
            break
        next_i = i + dirs[idx][0]
        next_j = j + dirs[idx][1]
        while not (0 <= next_i < n and 0 <= next_j < n) or (next_i, next_j) in visited:
            idx = (idx + 1) % len(dirs)
            next_i = i + dirs[idx][0]
            next_j = j + dirs[idx][1]
        i = next_i
        j = next_j
```

difference

- 두 코드는 단 한 줄 차이

```
while(True):
    print(i, j, A[i][j])
    A[i][j] = -1 # Mark as visited
    cnt += 1
    if cnt >= n * n:
        break
```

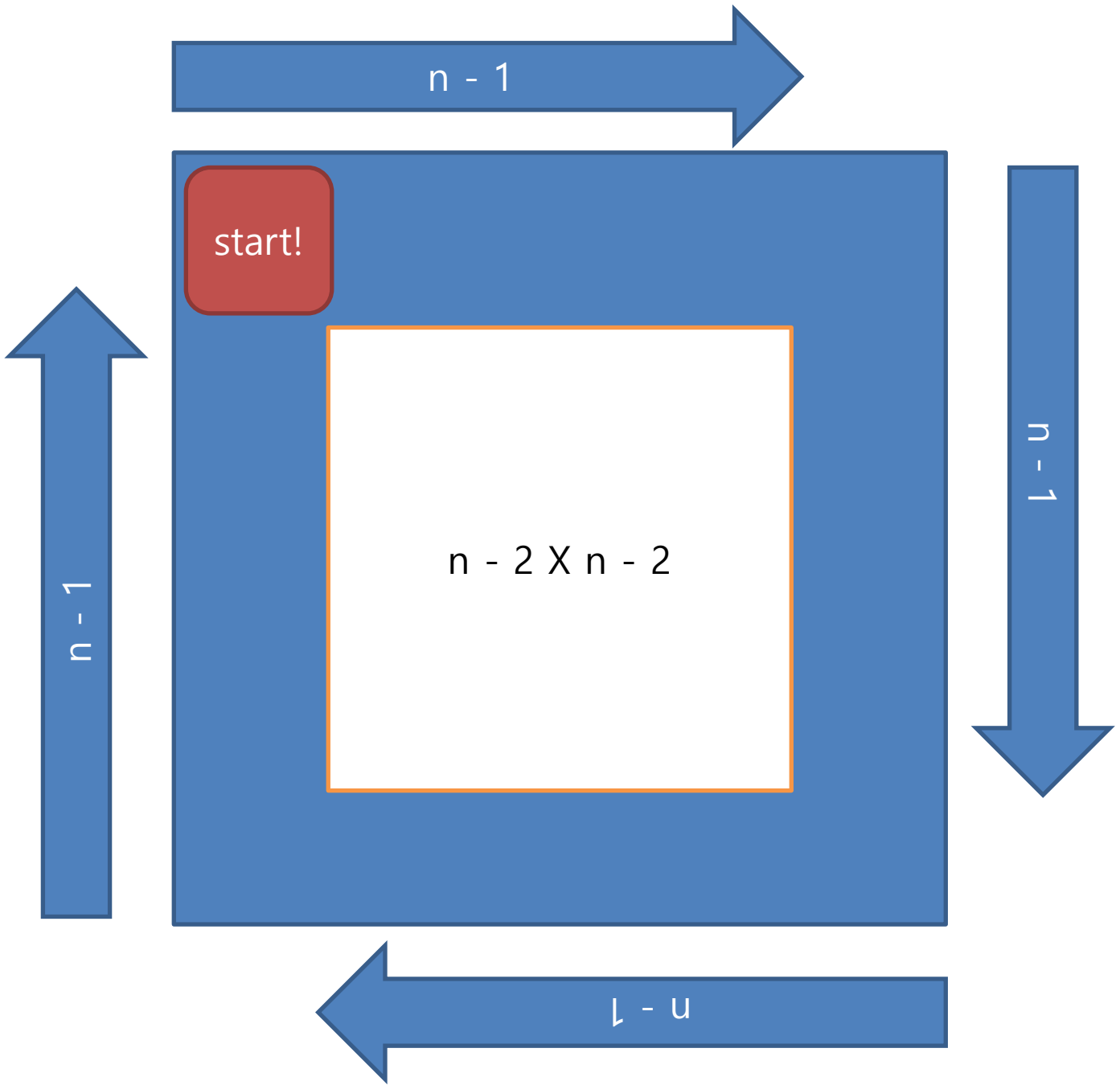
- 추가 Memory 없음 - 기존 배열 바뀜

```
while(True):
    print(i, j, A[i][j])
    visited.add((i,j))
    cnt += 1
    if cnt >= n * n:
        break
```

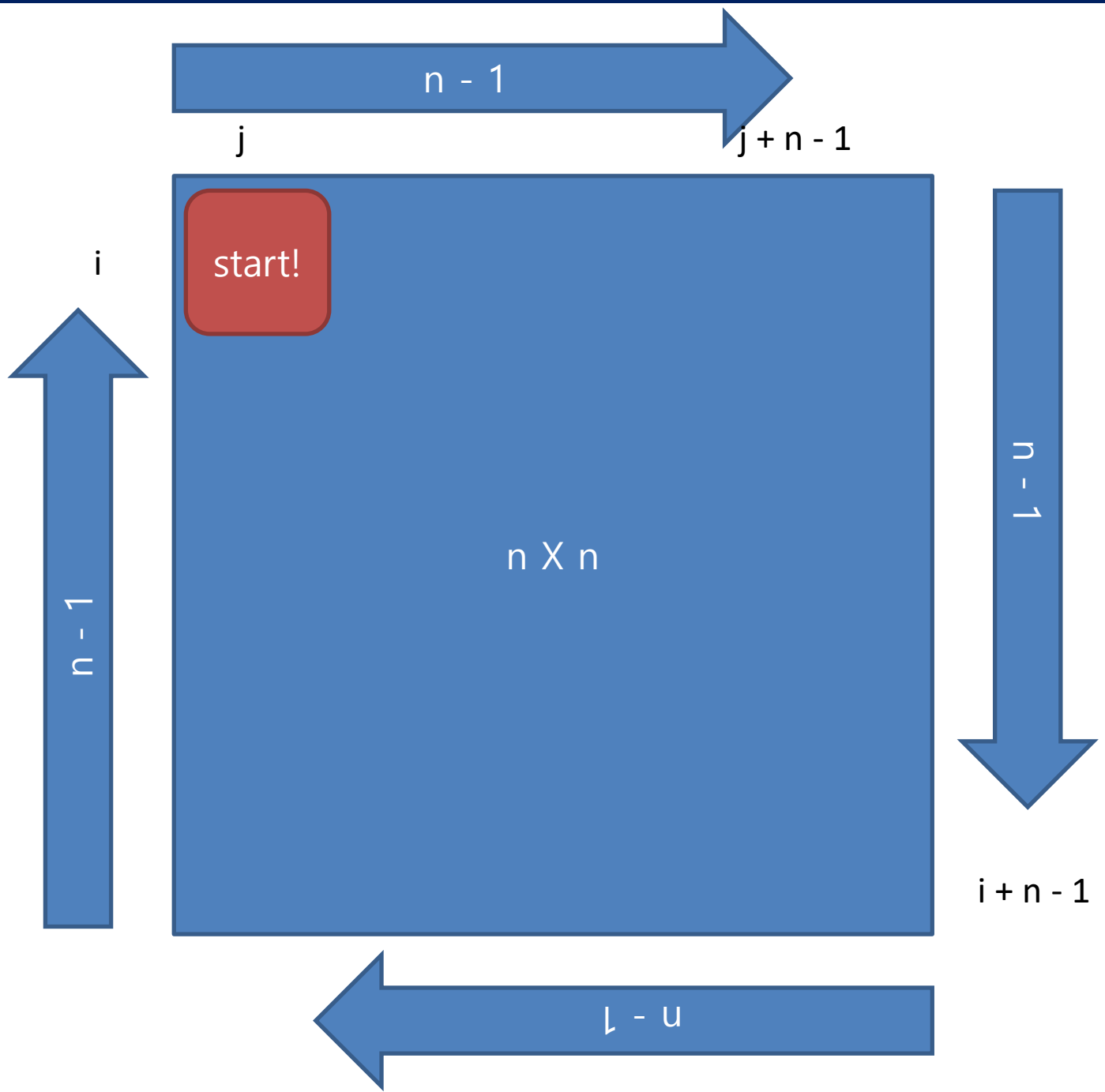
- 추가 Memory 있음 - 기존 배열 안바뀜

- 코딩 테스트라면?

recursive



recursive



spiral - recursion

```
def recursive_spiral(i, j, n):
    if n == 0:
        print('--')
        return
    if n == 1:
        print(i, j, A[i][j])
        print('--')
        return
    for k in range(j, j + n - 1):
        print(i, k, A[i][k])
    for k in range(i, i + n - 1):
        print(k, j + n - 1, A[k][j + n - 1])
    for k in range(j + n - 1, j, -1):
        print(n - 1 - i, k, A[n - 1 - i][k])
    for k in range(n - 1 - i, i, -1):
        print(k, j, A[k][j])
    recursive_spiral(i + 1, j + 1, n - 2)
```

result

```
jason@ThinkPad:~/workspace/advalg/ch5$ python3 5.18_2.py
0 0 1
0 1 2
0 2 3
1 2 6
2 2 9
2 1 8
2 0 7
1 0 4
1 1 5
--
```

Summary

- **Problem 5.18**
- **Iterative version #1, #2, #3**
- **Recursive version**

들어 주셔서 감사합니다