[Algorithm] 14강 : 구현 유형 개요 — 나무늘보의 개발 블로그

노트북: 첫 번째 노트북
만든 날짜: 2020-10-27 오후 3:47 **업데이트:** 2020-10-27 오후 3:48
URL: https://continuous-development.tistory.com/161?category=736684

Algorithm

[Algorithm] 14강 : 구현 유형 개요

2020. 10. 27. 08:23 수정 삭제 공개

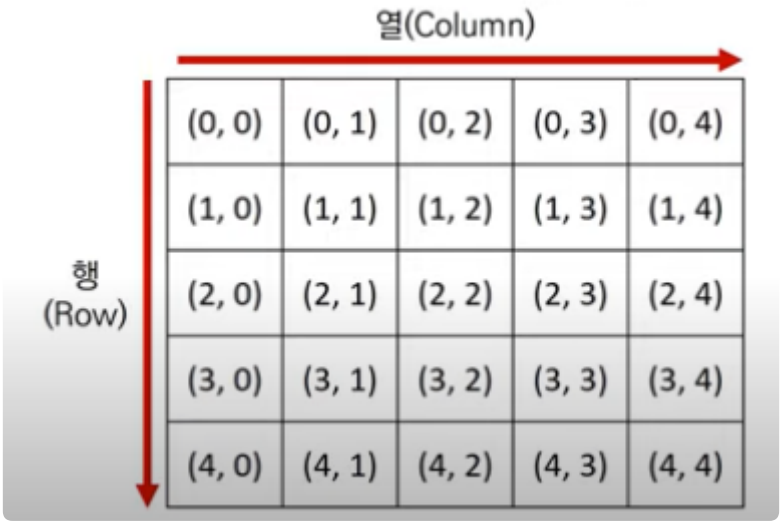
구현 유형의 문제란?

-풀이를 떠올리는 것은 쉽지만 소스코드로 옮기기 어려운 문제

구현 유형의 예시

- 알고리즘은 간단한데 코드가 길어지는 문제
- 실수 연산을 다루고, 특정 소수점 자리까지 출력해야 하는 문제
- 문자열을 특정한 기준에 따라서 끊어 처리해야 하는 문제
- 적절한 라이브러리를 찾아서 사용해야 하는 문제

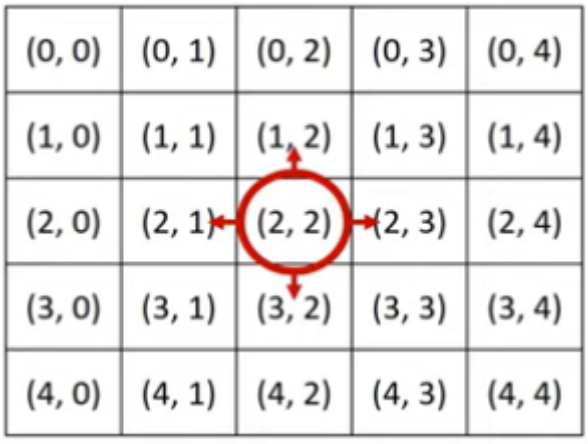
구현예시)



일반적으로 알고리즘 문제에서 2차원 공간은 행렬의 의미로 사용

```
for i in range(5):
    for j in range(5):
    print('(',i,',',j,')',end=' ')
print()
```

시뮬레이션 및 완전 탐색 문제



시뮬레이션 및 완전 탐색 문제에서는 2차원 공간에서의 방향벡터가 자주 활용

```
# 동,북,서,남
dx = [0,-1,0,1]
dy = [1,0,-1,0]

# 현재위치
x,y=2,2

for i in range(4):
    # 다음위치
nx = x + dx[i]
ny = y + dy[i]
print(nx, ny)
```

이런 로직으로 동서남북으로 움직이게 된다.

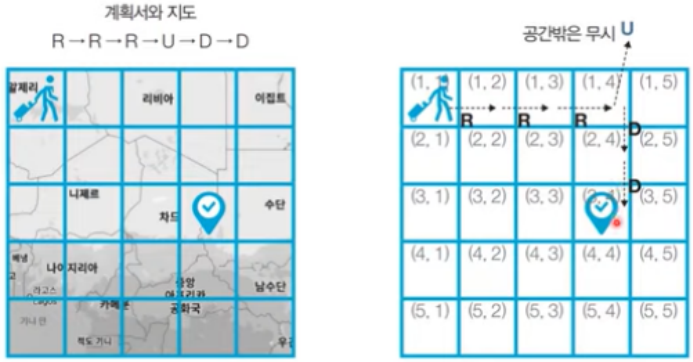
<문제> 상하좌우

〈문제〉 상하좌우: 문제 설명

- 여행가 A는 **N × N** 크기의 정사각형 공간 위에 서 있습니다. 이 공간은 1 × 1 크기의 정사각형으로 나누어져 있습니다. 가장 왼쪽 위 좌표는 (1, 1)이며, 가장 오른쪽 아래 좌표는 (N, N)에 해당합니다. 여행가 A는 **상, 하, 좌, 우 방향으로 이동**할 수 있으며, 시작 좌표는 항상 **(1, 1)**입니다. 우리 앞에는 여행가 A가 이동할 계획이 적힌 계획서가 놓여 있습니다.
- 계획서에는 하나의 줄에 띄어쓰기를 기준으로 하여 L, R, U, D 중 하나의 문자가 반복적으로 적혀 있습니다. 각 문자의 의미는 다음과 같습니다.
  - L: 왼쪽으로 한 칸 이동
  - R: 오른쪽으로 한 칸 이동
  - U: 위로 한 칸 이동
  - D: 아래로 한 칸 이동

〈문제〉 상하좌우: 문제 설명

- 이때 여행가 A가 N × N 크기의 정사각형 공간을 벗어나는 움직임은 무시됩니다. 예를 들어 (1, 1)의 위치에서 L 혹은 U를 만나면 무시됩니다. 다음은 N = 5인 지도와 계획서입니다.



〈문제〉 상하좌우: 문제 조건

난이도 ●○○ | 풀이 시간 15분 | 시간제한 2초 | 메모리 제한 128MB

입력 조건
- 첫째 줄에 공간의 크기를 나타내는 N이 주어집니다. (1 <= N <= 100)
- 둘째 줄에 여행가 A가 이동할 계획서 내용이 주어집니다. (1 <= 이동 횟수 <= 100)

출력 조건
- 첫째 줄에 여행가 A가 최종적으로 도착할 지점의 좌표 (X, Y)를 공백을 기준으로 구분하여 출력합니다.

입력 예시
```
5
R R R U D D
```

출력 예시
```
3 4
```

위와 같은 문제가 있다. 이 문제를 해결하는 방법을 찾아보자
이 문제는 요구사항만 구현하면 된다.

```
# N 입력
n = int(input())
x, y = 1, 1

# L,R,U,D 에 따른 이동 방향
dx = [0,0,-1,1]
dy = [-1,1,0,0]
move_types = ['L','R','U','D']

# 이동 계획을 하나씩 확인
for plan in plans:
    #이동 후 좌표 구하기
for i in range(len(move_types)):
    if plan == move_Types[i]:
```

```
    nx = x + dx[i]
ny = y + dy[i]
#공간을 벗어나는 경우 무시
if nx <1 or ny < 1 or nx > n or ny >n;
    continue
# 이동 수행
x , y = nx, ny
print(x, y)
```

이 자료는 동빈 나 님의 **이코 테** 유튜브 영상을 보고 정리한 자료입니다.
참고 : www.youtube.com/watch?v=m-9pAwq1o3w&list=PLRx0vPvlEmdAghTr5mXQxGpHjWqSz0dgC

'Algorithm' 카테고리의 다른 글□

[Algorithm] 14강 : 구현 유형 개요□

[Algorithm] 13 강 : 그리디 유형 문제풀이 + 백준 알고리즘 11399번 ATM문제□

[Algorithm] 12 강 : 그리디 알고리즘 개요(탐욕법)□

[Algorithm] 11 강 : 자주 사용하는 라이브러리(유용한 라이브러리)□

[Algorithm] 10 강 : 파이썬 문법 - 함수□

[Algorithm] 9 강 : 파이썬 문법 - 반복문□

구현유형 알고리즘 구현문제 알고리즘 완전탐색 완전탐색

나무늘보스

혼자 끄적끄적하는 블로그 입니다.