

[Python 트랙] 3회차 - 알고리즘 기본



| Background

- ✓ 배열에 대한 이해와 활용
- ✓ 스택에 대한 이해와 활용

| Goal

- ✓ 반복문을 이용하여 배열의 요소에 접근할 수 있다.
- ✓ 문제의 조건을 정확히 이해하고 해결할 수 있다.

| 환경 설정

- 1) Pycharm(Python3.7이상)을 이용해서 코드를 작성하고 결과를 확인한다.
 - 새로운 Pycharm 프로젝트를 생성 후 코드를 작성한다.
- 2) 파일 이름 및 제출 방법
 - 1, 2번 문제에 대한 소스 파일은 Algo문제번호_지역_반_이름.py로 만든다.
 - pypy의 경우 프로젝트와 파일이름에 한글을 사용할 수 없으므로 algo1.py, algo2.py 로 만들고 제출시 아래와 같이 변경한다.
 - 3번은 텍스트 파일로 작성한다.
 - Algo1_서울_1반_이싸피.py
 - Algo2_서울_1반_이싸피.py
 - Algo3_서울_1반_이싸피.txt
 - 위 3개의 파일만 지역_반_이름.zip으로 압축하여 제출한다.
 - 서울_1반_이싸피.zip(탐색기에서 파일 선택 후 오른쪽 클릭 - 보내기 - 압축(zip)폴더 선택)
- 3) 채점
 - 주석이 없는 경우, 주석이 코드 내용과 맞지 않는 경우, 지정된 출력 형식을 만족하지 않는 경우 해당 문제는 0점 처리될 수 있다.
 - import를 사용한 경우 해당 문제는 0점 처리될 수 있다. (import sys도 예외 없음)
- 4) 테스트케이스는 부분적으로 제공되며, 전체가 공개되지는 않는다.
- 5) 각 문제의 배점이 다르므로 표기된 배점을 반드시 확인한다.
 - 1번 50점, 2번 30점, 3번 20점

성실과 신뢰로 테스트에 볼 것 (부정 행위시 강력 조치 및 근거가 남음)

※ 소스코드 유사도 판단 프로그램 기준 부정 행위로 판단될 시,
0점 처리 및 학사 기준에 의거 조치 실시 예정



[Python 트랙] 3회차 - 알고리즘 기본

| 문제 1 : 봉우리의 수 (배점 50점)

싸피산에는 여러 개의 봉우리가 있다. 산을 일정한 구역으로 나눈 지도에서 주변 구역보다 높으면 봉우리가 된다. 싸피산에서 가장 높은 봉우리와 가장 낮은 봉우리의 차이를 알아내는 프로그램을 만드시오.

- $N \times N$ 개의 구역에 대한 높이 정보가 주어진다.
- 한 구역을 중심으로, 주변 8개 구역보다 높으면 봉우리 이다.
- 가장자리 구역은 봉우리인지 판단할 수 없다.
- 만약 봉우리가 하나만 있거나 없는 경우, 높이 차는 -1로 표시한다.

- 다음은 $N=3$ 인 경우의 예이다.

1	5	1
1	6	3
4	1	2

(1) 1개의 봉우리가 있는 경우

1	5	1
1	5	3
4	1	2

(2) 봉우리가 없는 경우

(1) 에서 높이가 6인 영역은 주변 8개의 영역보다 높으므로 봉우리이다.

(2) 의 경우 노란색 영역 주변에 같은 높이의 영역이 있으므로,
노란 영역은 봉우리가 아니다.

- 다음은 $N=5$ 인 경우의 예이다.

1	1	1	1	1
1	6	3	1	1
4	1	2	1	1
1	1	1	3	1
1	1	1	1	1

(1) 높이차가 3인 경우

1	1	1	1	1
1	6	3	1	1
4	1	2	1	1
1	1	1	6	1
1	1	1	1	1

(2) 높이차가 0인 경우

[Python 트랙] 3회차 - 알고리즘 기본



[입력]

첫 줄에 지도의 개수 T 가 주어진다. ($3 \leq T \leq 10$)

다음 줄부터 각 지도 별 첫 줄에 지도의 크기 N , N 줄에 걸쳐 N 개 영역의 높이 h_i 가 주어진다.
($3 \leq N \leq 20$, $0 \leq h_i \leq 10$)

[출력]

#과 지도 번호, 빈칸에 이어 가장 높은 봉우리와 가장 낮은 봉우리의 높이 차이를 출력한다.
만약 봉우리가 한 개이거나 없으면 -1을 출력한다.

[입력 예시]

```
3
3
1 5 1
1 6 3
4 1 2
3
1 5 1
1 5 3
4 1 2
5
1 1 1 1 1
1 6 3 1 1
4 1 2 1 1
1 1 1 3 1
1 1 1 1 1
```

(algo2_sample_in.txt 참고)

[출력 예시]

```
#1 -1
#2 -1
#3 3
```

(algo2_sample_out.txt 참고)

[Python 트랙] 3회차 - 알고리즘 기본



| 문제 2 : 탐사 로봇 (배점 30점)

싸피 로봇이 새로운 행성을 탐사 중이다. 싸피 우주국은 탐색 구역을 $N \times N$ 개의 격자 형태로 나누고, 지형을 미리 조사해 각 구역에서 로봇이 움직일 수 있는 방향을 정해 두었다. 로봇의 이동 방향은 숫자로 표시하며, 표시된 방향으로만 다른 구역으로 이동할 수 있다. 다음은 숫자가 의미하는 이동 방향이다.

0	1	2	3
---	---	---	---

로봇은 항상 $N \times N$ 구역의 왼쪽 맨 윗 칸에서 출발하며, 가능한 방향으로 이동하게 된다. 다음은 $N=3$ 인 구역의 예로, 로봇은 화살표 방향으로 움직이게 된다.

0	1	2
2	1	3
3	0	3

0	1	2
2	1	3
3	0	3

- 로봇은 $N \times N$ 구역을 벗어날 수 없다.
- 로봇이 구역의 모든 칸을 지나야 하는 것은 아니다.
- 로봇은 지나간 구역은 다시 지나지 않는다.

위의 예에서 로봇은 노란색 칸에서 방향 전환 후 멈추게 된다.

로봇의 최초 이동부터 멈출 때까지 이동 방향을 표시하면 0 1 1 0 3 3 2가 된다. 로봇은 처음 이동할 때와 새로운 방향으로 바꿀 때 많은 에너지가 들기 때문에, 중복된 방향을 제거한 0 1 0 3 2로 에너지 소비를 나타낸다고 한다.

[Python 트랙] 3회차 - 알고리즘 기본



다음은 $N = 4$ 인 구역의 예로, 로봇의 에너지 소비는 1 0 3 2 1 0 3으로 나타낸다.

1	1	2	2
1	1	3	3
1	0	3	3
0	0	0	3

[입력]

첫 줄에 구역의 개수 T 가 주어진다. ($3 \leq T \leq 10$)

다음 줄부터 각 구역 별 첫 줄에 구역의 크기 N , 다음 줄 부터 N 줄에 걸쳐 N 개씩, 이동 방향 d_i 가 빈칸으로 구분되어 주어진다. ($3 \leq N \leq 100$, $0 \leq d_i \leq 3$)

[출력]

#과 1번부터인 구역 번호, 빈칸에 이어 빈칸으로 구분된 에너지 소비를 출력한다.

[입력 예시]

```
3
3
0 1 2
2 1 3
3 0 3
4
1 1 2 2
1 1 3 3
1 0 3 3
0 0 0 3
5
0 0 0 0 1
0 1 1 2 1
3 1 1 3 1
3 0 0 3 1
3 2 2 2 2
```

(algo2_sample_in.txt 참고)

[출력 예시]

```
#1 0 1 0 3 2
#2 1 0 3 2 1 0 3
#3 0 1 2 3 0 1 0 3 2 1
```

(algo2_sample_out.txt 참고)

[Python 트랙] 3회차 - 알고리즘 기본



| 문제 3 : 스택 (배점 : 20점)

다음과 같이 소괄호를 포함한 문자열이 있다.

```
if ( ( i == 0 ) && ( j == 0 ) )
```

- (1) 스택을 이용해 괄호가 정상적으로 표시되어 있는지 검사하는 알고리즘에 대해 간단히 설명하라.
- (2) 위의 문자열을 스택을 이용해 검사하는 과정에 대해, 나머지 단계의 스택 내부 상태를 표시하고 간단히 설명하라. 마지막에는 괄호가 정상인지 오류인지와 그 이유를 설명해야 한다.

스택은 []로 표시하고, 저장 원소의 구분은 쉼표나 빈 칸으로 표시한다.

```
[ ( ] # 여는 괄호를 만나 push
[ ( , ( ] # 여는 괄호를 만나 push
[ ( ] # 닫는 괄호를 만나 pop
...
```