[Python트랙] 과목평가2 – 알고리즘 기본

| Background

- ✓ 배열에 대한 이해와 활용
- ✓ 정렬에 대한 이해와 활용

| Goal

- ✓ 반복문과 조건문을 이용하여 배열의 요소에 접근할 수 있다.
- ✓ 배열에서 특정 조건을 만족하는 연속한 원소를 탐색할 수 있다.
- ✓ 배열을 저장된 자료를 특정 정렬 알고리즘으로 정렬할 수 있다.

| 환경 설정

1) Pycharm과 pypy 또는 python을 이용해서 코드를 작성하고 결과를 확인한다. 새로운 프로젝트를 생성하지 않고 기존 프로젝트 사용시 부정행위로 간주 함.

2) 파일 이름 및 제출 방법

- **1, 2번 문제에 대한 소스 파일 이름은 다음과 같이 영문으로 작성한다.**

  서울 1반 이싸피라면, algo문제번호_반_이름.py 순서로 영문으로 작성

  algo1_01_leessafy.py

  algo2_01_leessafy.py

- **3번 문제에 대한 답안 파일 이름은 .txt 형식으로 다음과 같이 영문으로 작성한다.**

  algo3_01_leessafy.txt

- **위 3개의 파일만 지역_반_이름.zip으로 압축하여 제출한다.**

  서울_1반_이싸피.zip

  (탐색기에서 파일 선택 후 오른쪽 클릭 – 압축대상 – Zip 선택)

3) 채점

- **문제별로 부분 점수가 부여된다.**
- **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 40점, 2번 35점, 3번 25점**

성실과 신뢰로 테스트에 임할 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

[Python트랙] 과목평가2 - 알고리즘 기본

| 문제1 : 김싸피의 고장난 스카우터 (배점 : 40점)

import 사용금지

우주를 탐사하던 김싸피의 스카우터가 그만 고장 나고 말았다! 이제 스카우터는 전체 우주를 한 번에 볼 수 없고, 오직 **M x M 크기의 영역**만 탐색할 수 있게 되었다. 김싸피의 임무는 이 고장 난 스카우터를 이용해 **N x N** 크기의 우주 공간을 탐색하여, **정확히 K개의 별(*)**을 포함하는 M x M 영역을 찾는 것이다. (단, 조건을 만족하는 영역은 **한 개 이하이다**).

조건을 만족하는 영역의 좌표를 출력하는 프로그램을 작성하시오.

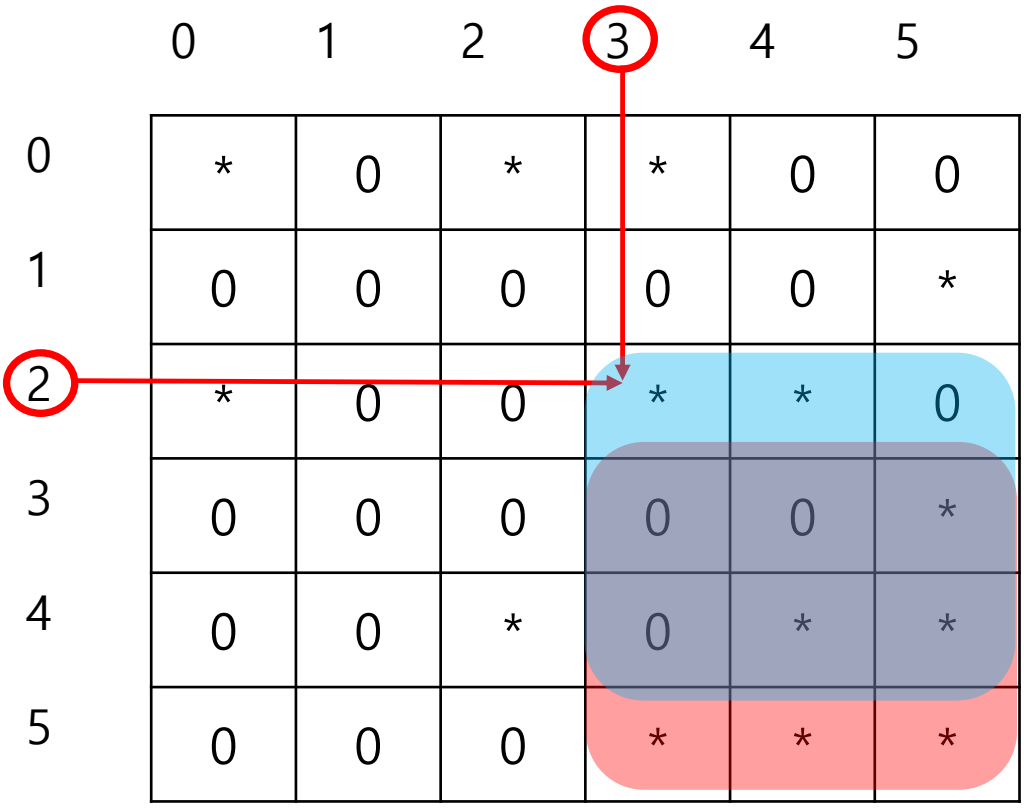
(단, 영역의 좌표는 **왼쪽 상단 모서리의 좌표를 해당 영역의 좌표**로 한다.)

[예시]

다음 예시는 N = 6 , M = 3, K = 5 일 때의 예시이다.

예시에서 K개의 별이 존재하는 영역의 좌표는 (2, 3) 이다.

(3, 3) 영역의 경우 별이 6개 이므로 정답이 될 수 없다.



[Python트랙] 과목평가2 - 알고리즘 기본

[입력]

첫 번째 줄에 테스트케이스의 개수 T가 주어진다.

각 테스트케이스의 첫 줄에는 N, M, K가 공백으로 구분되어 주어진다.

다음 N개의 줄에 걸쳐, N x N 크기의 우주 영역 정보가 각 줄에 N개의 문자로 공백 없이 주어진다.

우주의 빈 공간은 '0', 별은 '*' 으로 주어진다.

(5 <= N <= 20, 1 <= M <= N , 0 <= K <= M*M)

[출력]

각 테스트케이스마다 '#tc'(tc는 테스트케이스 번호)를 출력하고, 한 칸을 띄운 후 정답 좌표의 행과 열을 공백으로 구분하여 출력한다. 단, 조건에 맞는 영역이 없을 경우 (-1 , -1) 을 출력한다.

[입력 예시]

```
3
6 3 5
*0**00
00000*
*00**0
00000*
00*0**
000***
5 2 3
00000
0**00
00000
00000
00000
... 생략 ...
(algo1_sample_in.txt 참고)
```

[출력 예시]

```
#1 2 3
#2 -1 -1
... 생략 ...

(algo1_sample_out.txt 참고)
```

[Python트랙] 과목평가2 - 알고리즘 기본

| 문제2 : 강우 안전구역 (배점 : 35점)

import 사용금지

싸피시는 폭우에 대비해 다른 곳보다 지대가 높은 강우 안전구역을 정하기로 했다. 각 구역 높이는 격자형태로 구분된 디지털 지도에 저장되어 있으며, **모든 인접구역보다도 높은 지역은 안전구역**이 된다. 디지털 지도가 주어지면 각 지도의 안전구역 수를 알아내라. (**인접구역은 어떤 구역을 중심으로 상하좌우에 있는 구역**을 말한다.)

[제약 조건]

- T개의 지도가 차례로 주어진다. (3<=T<=10)
- 디지털 지도는 NxM개의 격자 형태로 주어진다. (3<=N, M<=20)
- 각 칸에는 각 구역의 높이 Hij가 0이상 20이하의 정수로 표시되어 있다. (0<=Hij<=20)
- 지도의 가장 자리 구역은 인접구역 정보가 부족하므로, 안전구역에서 제외한다.

다음은 N=3, M=4인 디지털 지도의 예로, 노란 색으로 표시된 곳이 상하좌우 인접 구역보다 높은 안전구역이다.

1	0	1	0
0	1	0	1
1	0	1	0

[Python트랙] 과목평가2 - 알고리즘 기본

[입력]

첫 줄에 지도의 수 T, 다음 줄부터 지도 별로 첫 줄에 N과 M이 공백으로 구분되어 주어지고, 다음 줄부터 N개의 줄에 걸쳐 M개씩의 높이 정보 Hij가 주어진다.

[출력]

#과 1번부터인 지도번호, 빈칸에 이어 안전구역의 수를 출력한다.

[입력 예시]

```
3
3 4
1 0 1 0
0 1 0 1
1 0 1 0
4 3
11 5 0
15 9 3
4 1 12
8 11 3
5 5
3 18 0 19 0
1 19 10 1 6
10 1 9 16 10
10 2 7 0 16
18 14 6 9 14
```

(algo2_sample_in.txt 참고)

[출력 예시]

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

(algo2_sample_out.txt 참고)

[Python트랙] 과목평가2 - 알고리즘 기본

| 문제3 : 버블 소트 (배점 : 25점)

첫 페이지 '파일 이름 및 제출 방법'을 확인할 것

다음은 이싸피가 버블 소트로 오름차순 정렬을 구현하려고 작성한 파이썬 코드 일부이다. **이싸피는 j의 범위는 정확하게 기억하고 있으나** 나머지 부분은 잘 기억이 나지 않았다. 코드의 **문법적 또는 논리적 오류**를 모두 찾아서 올바른 버블 소트(**오름차순 정렬**)가 되도록 고치고, 고친 이유를 간단히 설명하시오. (수정해야 하는 부분은 2개 이상이므로, 수정한 전체 코드를 적고, 고친 이유는 코드내에 주석으로 표시하거나 또는 코드와 별도로 적어도 된다. 코드만 수정하고 **설명이 없는 경우 감점됨**.)

```
arr = [5, 1, 4, 2, 8]

n = len(arr)
for i in range(n):
    for j in range(i):    # j의 범위는 수정할 필요 없음
        if arr[j] < arr[j+1]:
            arr[j], arr[j+1] == arr[j+1], arr[j]
```