

# [Python트랙] 과목평가2 – 알고리즘 기본



## | Background

- ✓ 배열에 대한 이해와 활용
- ✓ 정렬에 대한 이해와 활용
- ✓ 문자열에 대한 이해와 활용

## | Goal

- ✓ 반복문과 조건문을 이용하여 배열의 요소에 접근할 수 있다.
- ✓ 배열원소를 정렬할 수 있다.
- ✓ 문자열을 검색하고 구성할 수 있다.

## | 환경 설정

1) PyCharm과 pypy 또는 python을 이용해서 코드를 작성하고 결과를 확인한다.

- 새로운 프로젝트를 생성하지 않고 기존 프로젝트 사용시 부정행위로 간주 함.
- Pypy에서만 동작하는 비표준 문법 사용시 0점 처리됨.

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



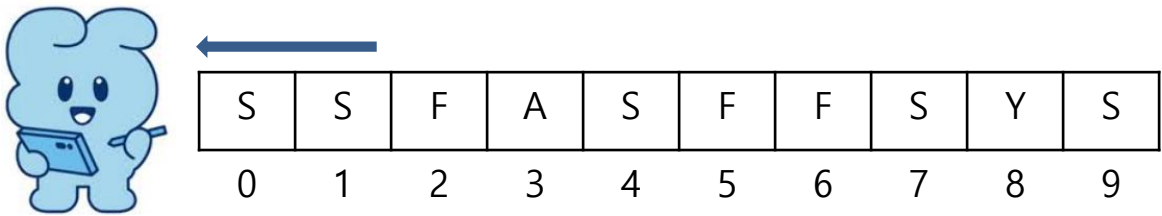
## | 문제1 : 단어 만들기 (배점 : 40점)

### import 사용금지

스타티의 앞에는 글자 카드가 한 장씩 지나가는 컨베이어 벨트가 있다.

스타티는 벨트 위를 지나가는 카드를 적절히 집어서 원하는 단어를 완성해야 한다. 스타티는 단어를 완성 할 수 있을까?

[예시] 다음은 길이  $N = 9$  인 컨베이어 벨트에서 단어를 만드는 예시이다. 컨베이어 벨트에 다음과 같은 글자 카드가 지나간다.



스타티가 'SSAFY'라는 단어를 만들려고 할 때 다음과 같은 과정을 통해서 단어를 완성할 수 있다.

1. 0 번 칸에서 'S' 를 획득
2. 1 번 칸에서 'S'를 획득
3. 3 번 칸에서 'A'를 획득
4. 5 번 칸에서 'F'를 획득
5. 8 번 칸에서 'Y'를 획득

### [유의사항]

- 컨베이어 벨트는 한 방향으로만 움직이며 시작점은 0번이다.
- 한 번 지나간 글자 카드는 재사용할 수 없다.

# [Python트랙] 과목평가2 - 알고리즘 기본



## [입력]

첫 번째 줄에 테스트케이스의 개수 T가 주어진다.

각 테스트 케이스의 첫 번째 줄에 컨베이어 벨트의 길이 N 과 스타티가 만들려는 단어의 길이 M 이 띄어쓰기로 구분되어 주어진다. 이어서 두 번째 줄에 N개의 글자 카드가 주어지고, 세 번째 줄에 스타티가 만들려는 단어가 주어진다.

( $2 \leq N \leq 20$ ,  $3 \leq M \leq 20$ )

## [출력]

각 테스트케이스마다 '#tc'(tc는 테스트케이스 번호)를 출력하고, 단어를 완성할 수 있는 가장 빠른 칸의 번호를 출력한다. 단, 단어를 완성할 수 없다면 -1을 출력한다.

### [입력 예시]

```
3
10 5
SSFASFFSYS
SSAFY
8 4
ABCDEFGH
ACEH
5 3
XYCAB
ABC
```

(algo1\_sample\_in.txt 참고)

### [출력 예시]

```
#1 8
#2 7
#3 -1
```

(algo1\_sample\_out.txt 참고)

# [Python트랙] 과목평가2 - 알고리즘 기본



## | 문제2 : 파리 트랩 (배점 : 35점)

**import 사용금지**

$N \times M$  격자에 각 칸마다 파리의 수가 적혀 있다. 값이 0인 칸은 비어 있는 칸으로, 이 빈 칸(0인 칸)에만 파리 트랩을 설치할 수 있다. 파리 트랩을 0인 칸에 1개 설치하면, 트랩은 그 칸의 상·하·좌·우 칸에 있는 파리를 모두 끌어들이어 잡는다. 인접 칸은 격자 범위 안에 있는 칸만 포함한다.

- 모서리(코너)의 0인 칸은 인접 칸이 2개
- 가장자리(코너 제외)의 0칸은 인접 칸이 3개
- 내부의 0칸은 인접 칸이 4개

격자에는 0인 칸이 최소 2개 이상 존재한다. 트랩을 0인 칸 중 딱 한 곳에 설치할 때, 최대로 잡을 수 있는 파리 수는 몇 마리인가?

### 예시 1: 3x3 격자

파리 수

1	2	3
4		6
7	8	0

0 = “파리 트랩” 설치 가능 칸

- 트랩 위치 후보

① (1,1)에 설치

1	2	3
4		6
7	8	0

→  $2 + 4 + 6 + 8 = 20$ 마리

② (2,2)에 설치

1	2	3
4		6
7	8	

→  $6 + 8 = 14$ 마리

**최대 파리 수: 20마리**

# [Python트랙] 과목평가2 - 알고리즘 기본



## [입력]

첫 줄에 테스트케이스 개수  $T$ , 각 테스트케이스 별로 첫 줄에 격자의 높이와 폭  $N, M$  ( $1 \leq N, M \leq 10$ ), 다음  $N$ 줄에 걸쳐 빈칸으로 구분된  $M$ 개의 정수가 주어진다. (각 칸의 파리 수, 0 이상 9이하)

## [출력]

각 테스트케이스마다 #과 케이스번호에 이어 최대 파리수를 출력한다.

### [입력 예시]

```
3
3 3
1 2 3
4 0 6
7 8 0
4 5
0 1 2 3 4
5 6 0 8 9
1 0 2 0 3
4 5 6 7 0
5 5
1 1 1 1 1
1 0 9 0 1
1 9 0 9 1
1 0 9 0 1
1 1 1 1 1
```

(algo2\_sample\_in.txt 참고)

### [출력 예시]

```
#1 20
#2 20
#3 36
```

(algo2\_sample\_out.txt 참고)

# [Python트랙] 과목평가2 - 알고리즘 기본



## | 문제3 : 오름차순 정렬 (배점 : 25점)

첫 페이지 '파일 이름 및 제출 방법'을 확인할 것

다음은 어떤 오름차순 정렬에 대한 코드이다.

```
def my_sort(a, N) :          # 정렬할 리스트 a, N 원소 수
    for i in range(N-1, 0, -1) :
        for j in range(i) :
            if a[j] > a[j+1] :
                a[j], a[j+1] = a[j+1], a[j]      # 원소 교환
```

(1) 적용된 정렬 알고리즘의 이름은 무엇인가?

(2) 정렬할 배열이 다음과 같을 때 7이 자리를 찾는 과정을 표현한 것이다.

arr = [5, 7, 2, 4, 3]

[5, 7, 2, 4, 3]

[5, 2, 7, 4, 3]

[5, 2, 4, 7, 3]

[5, 2, 4, 3, 7]

이후 5가 자리를 찾는 과정을 같은 방식으로 표현하라.

(3) (2)에서 설명한 과정에서 5는 몇 번의 원소 교환이 일어나는가?