Computing for Data Science

HW #7

제출 기한: 2021.04.12 오전 10:59

다음 페이지부터 문제가 주어집니다.

주의사항

- 코드를 Jupyter Notebook 에서 작성하였더라도 python 파일(.py)로 제출할 것.
- 함수가 의도한 값을 Return 하는지를 확인. (Print 와 혼동하지 말 것)
- 파일명은 P1.py ~ P5.py 를 유지하고, 해당파일들을 HW7_학번_이름.zip 으로 압축하여 제출할 것. 예를 들면 학번이 2020-12345 이고, 이름이 Keondo Park 이라면 **HW7_2020_12345_KeondoPark.zip** 으로 압축하여 제출.
- 예시로 제시한 입력값 외에 조교가 랜덤으로 생성한 입력값으로 코드가 잘 작성되었는지 테스트할 것이다.
- 채점은 프로그램에 의해 기계적으로 처리되므로 위 사항을 지키지 않은 경우 누락되거나 불이익을 받을 수 있음.
- **늦은 제출은 받지 않음**.
- **표절 검사를 수행하여 발각될 경우 성적 F 부여**.

모든 문제에 해당하는 사항: 뼈대 코드의 함수 이름 및 parameter 는 수정하지 말 것. 기본 모듈만 이용.

P1.

버블 소팅 알고리즘은 인접한 두 원소의 순서가 잘못되었을 때(앞의 원소가 뒤의 원소보다 클 때) 서로 swap 하면서 정렬하는 알고리즘이다.

```
Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the
adjacent elements if they are in wrong order.

Example:
First Pass:
( 5 1 4 2 8 ) -> ( 1 5 4 2 8 ), Here, algorithm compares the first two elements, and
swaps since 5 > 1.
( 1 5 4 2 8 ) ->  ( 1 4 5 2 8 ), Swap since 5 > 4
( 1 4 5 2 8 ) ->  ( 1 4 2 5 8 ), Swap since 5 > 2
( 1 4 2 5 8 ) -> ( 1 4 2 5 8 ), Now, since these elements are already in order (8 > 5),
algorithm does not swap them.

Second Pass:
( 1 4 2 5 8 ) -> ( 1 4 2 5 8 )
( 1 4 2 5 8 ) -> ( 1 2 4 5 8 ), Swap since 4 > 2
( 1 2 4 5 8 ) -> ( 1 2 4 5 8 )
( 1 2 4 5 8 ) ->  ( 1 2 4 5 8 )
Now, the array is already sorted, but our algorithm does not know if it is completed.
The algorithm needs one whole pass without any swap to know it is sorted.

Third Pass:
( 1 2 4 5 8 ) -> ( 1 2 4 5 8 )
( 1 2 4 5 8 ) -> ( 1 2 4 5 8 )
( 1 2 4 5 8 ) -> ( 1 2 4 5 8 )
( 1 2 4 5 8 ) -> ( 1 2 4 5 8 )
```

정수로 이루어진 list 를 입력으로 받아서 버블 소팅할 때, **swap 이 일어나는 횟수**를 return 하는 함수를 구현하시오.

예시.

```
>>> P1([5, 1, 4, 2, 8])
4
>>> P1([-1, -1, -1, -1, -1])
0
>>> P1([6, 5, 4, 3, 2, 1])
15
>>> P1([1,5,2,3,6,6,1,2,3,4,21,1,11,-1])
40
```

P2.

다음 조건을 만족하는 list 가 있다.

1. 모든 원소는 알파벳 소문자로 된 **string** 이다. (공백 없음)
2. 각 **string** 의 길이는 **1** 이상 **20** 이하이다.

이 list 를 입력으로 받아서 다음 조건에 따라 정렬하여 return 하는 함수를 구현하시오.

1. **길이가 짧은 string 이 앞에 있어야 한다.**
2. **길이가 같다면 사전 순서가 빠른 string 이 앞에 있어야 한다.**
3. input list 를 정렬하고 return 해도 되고 새로운 list 를 return 해도 된다.

예시.

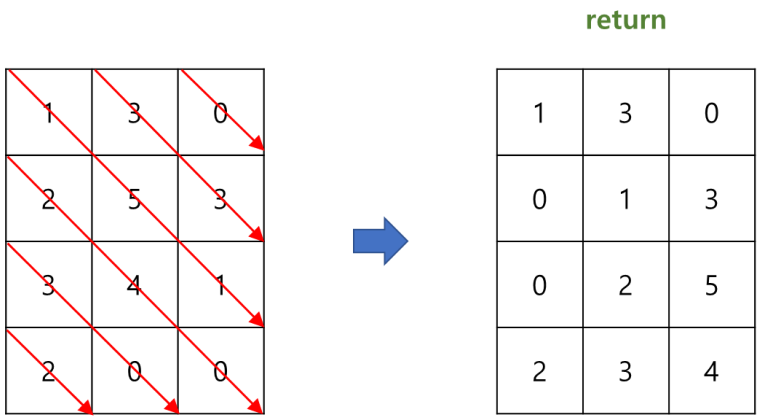
```
>>> P2(['solve','this','problem','or','you','will','get','f'])
['f', 'or', 'get', 'you', 'this', 'will', 'solve', 'problem']
>>> P2(['computing','class','is','so','funny','haha']
['is', 'so', 'haha', 'class', 'funny', 'computing']
>>> P2(['making','homework','is','very','fun','ha','ha'])
['ha', 'ha', 'is', 'fun', 'very', 'making', 'homework']
```

P3.

행렬은 각 행을 list 로 가지는 list 로 표현할 수 있다. 즉, 다음과 같은 행렬은 [[1,3,0],[2,5,3],[3,4,1],[2,0,0]]과 같이 표현한다.

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

행렬을 입력으로 받아서 **오른쪽 아래 대각선 방향으로 정렬**한 행렬을 return 하는 함수를 구현하시오. 즉, 다음과 같이 정렬한 다음 return 해야 한다. (input list 를 정렬하고 return 해도 되고 새로운 list 를 return 해도 된다.)



예시.

```
>>> P3([[1,3,0],[2,5,3],[3,4,1],[2,0,0]])
[[1, 3, 0], [0, 1, 3], [0, 2, 5], [2, 3, 4]]
>>> P3([[1,6,2,4,6,2,4,7,0]])
[[1, 6, 2, 4, 6, 2, 4, 7, 0]]
>>> P3([[1,5,20,-1,3],[-1,3,20,4,-1],[34,3,12,5,-12],[4,64,612,6,10]])
[[1, 5, -12, -1, 3],
 [-1, 3, 5, 4, -1],
 [34, 3, 6, 10, 20],
 [4, 64, 612, 12, 20]]
```

P4.

길이가 L 이고, 음이 아닌 정수로 이루어진 두 list A, B 가 있을 때, A 를 재배열해서 다음 값을 최소로 만들려고 한다.

```
S = A[0] * B[0] + A[1] * B[1] + . . . + A[L-1] * B[L-1]
(같은 index 끼리 곱한 것의 합)
```

최소가 되는 S 를 return 하는 함수를 구현하시오. (list A, B 를 입력으로 받음)

예제.

```
>>> P4([1,1,1,6,0], [2,7,8,3,1])
18
>>> P4([1,4,2,6,1,0], [3,5,1,2,4,1])
21
>>> P4([1,0,1,0,1], [0,1,0,1,0])
0
```

P5.

1 부터 N 까지의 자연수 하나씩으로 이루어진 list 를 입력으로 받는다(길이 N). 당신은 다음 두 가지 행동 중 하나만 할 수 있다.

```
1. 하나의 원소를 골라서 맨 앞에 놓는다.
2. 하나의 원소를 골라서 맨 뒤에 놓는다.
```

최소한의 행동만으로 정렬을 해야 한다. 몇 번 만에 정렬을 할 수 있는지 최소값을 return 하는 함수를 구현하시오.

예제.

```
>>> P5([3,1,2])
1
>>> P5([3,4,1,2])
2
>>> P5([8, 11, 14, 1, 10, 5, 15, 7, 2, 6, 3, 9, 13, 12, 4])
11
```