[Python트랙] 과목평가2 - 알고리즘 기본

| Background

- ✓ 배열에 대한 이해와 활용
- ✓ 이진 탐색에 대한 이해와 활용

| Goal

- ✓ 반복문과 조건문을 이용하여 배열의 요소에 접근할 수 있다.
- ✓ 문제의 조건을 정확히 이해하고 해결할 수 있다.
- ✓ 이진탐색을 이해하고 활용할 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 폴더, 프로젝트, 파일이름에 한글을 사용할 수 없으므로 algo1.py, algo2.py 로 만들고 제출시 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 40점, 2번 35점, 3번 25점**

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

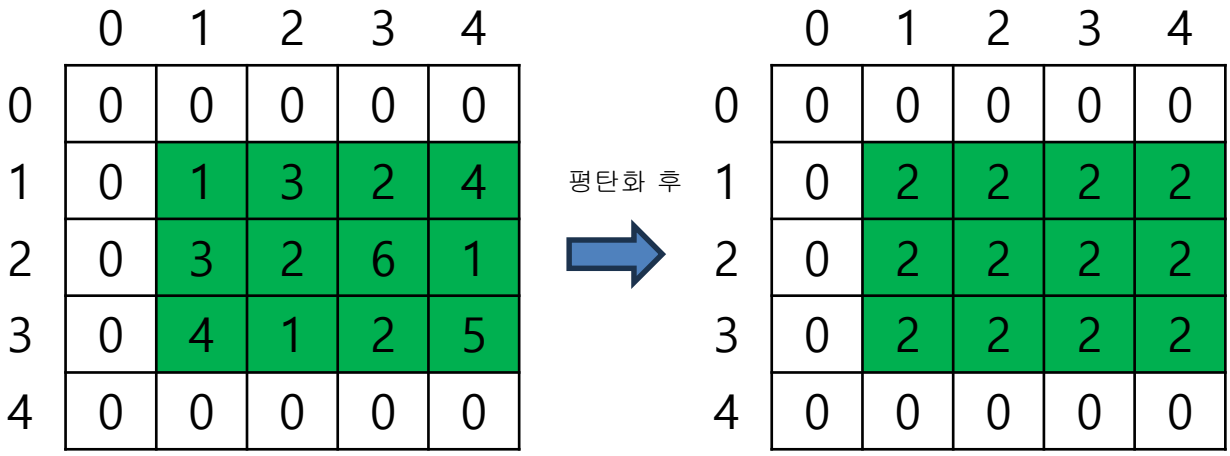
[Python트랙] 과목평가2 – 알고리즘 기본

| 문제 1 : 평지 만들기 (배점 : 40점)

김싸피는 전체 크기가 N*N인 마당 중 일부를 평지로 만들려고 한다. 이 때, 평지 만들기를 완료하기 위해서 몇 번의 작업을 해야 하는지 구하는 프로그램을 작성하라.

[제약사항]

파이썬 내장함수 max, min, sum, abs 및 슬라이싱(slicing)은 사용할 수 없다.



평지로 만들려고 하는 평탄화 영역은 직사각형 형태로 왼쪽 위, 오른쪽 아래 칸의 좌표가 주어진다. (위 예시의 녹색 영역은 1 1 3 4로 주어진다.)

평지를 만들기 위해서 김싸피는 해당 영역 높이의 **평균값**으로 평탄화 작업을 하려고 한다.

***평균값은 (평단화 영역의 높이 값의 합)//(영역의 칸 수)** 이다. (소수점 아래는 버린다.)

한 번의 작업으로 한 칸의 높이를 1 높이거나 1 낮게 할 수 있다. 예를 들어 기존 높이가 4인 칸은 높이 2로 만드는데 2회의 작업이 필요하다.

위 예시에서 녹색인 평탄화 영역의 총합은 34, 영역의 넓이는 12 이므로 영역의 평탄화 높이는 34//12인 2이다. 따라서 해당 영역을 모두 높이 2로 평지를 만들기 위해서 **총 16번**의 작업이 필요하다.

[Python트랙] 과목평가2 – 알고리즘 기본

[입력]

첫 줄에 테스트케이스 수가 주어진다.

각 테스트 케이스의 첫 줄에 N 주어진다.

이후 평지로 만들려는 영역의 좌측상단 행과 열 번호와 우측하단 행과 열 번호가 순서대로 띄어쓰기로 구분되어 주어진다. (좌표는 0부터 시작한다.) 앞의 예시의 경우 '1 1 3 4' 가 좌표로 주어진다.

이후 N 줄에 거쳐 N개의 정수 A_i가 주어진다.

(3 ≤ N ≤ 10 , 0 ≤ A_i ≤ 100)

[출력]

각 줄마다 " #T " (T는 테스트 케이스 번호)를 출력한 뒤, 빈칸에 이어 작업의 횟수를 출력한다.

[입력 예시]

```
3
5
1 1 3 4
0 0 0 0 0
0 1 3 2 4
0 3 2 6 1
0 4 1 2 5
0 0 0 0 0
5
0 0 4 4
4 3 3 4 3
3 1 3 2 4
5 3 2 6 1
3 4 1 2 5
3 3 3 3 3
3
0 0 0 2
1 1 1
1 1 1
1 1 1
```

(algo1_sample_in.txt 참고)

[출력 예시]

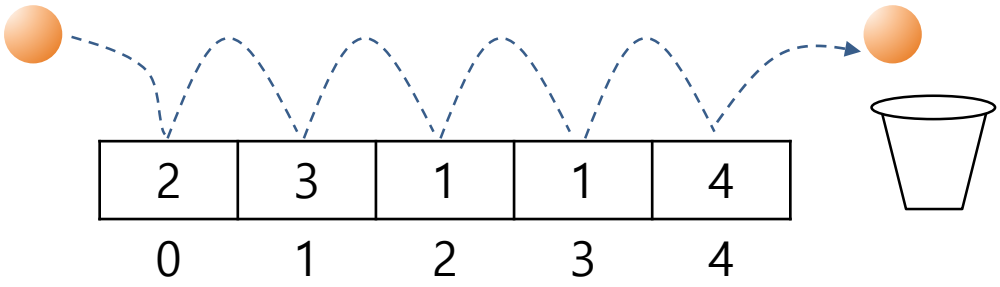
```
#1 16
#2 20
#3 0
```

(algo1_sample_out.txt 참고)

[Python트랙] 과목평가2 – 알고리즘 기본

| 문제 2 : 바운스 챌린지 (배점 : 35점)

김싸피는 탁구공으로 탁자에 공을 튕겨서 점수를 얻는 바운스 챌린지를 하려고 한다.

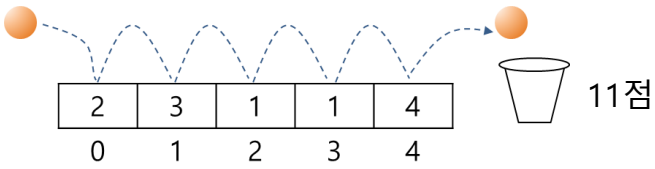


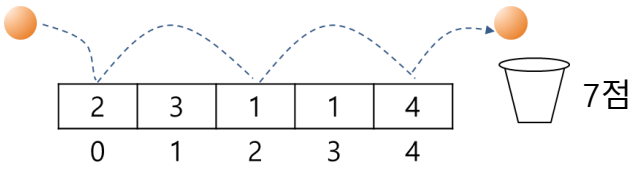
김싸피가 다음과 같은 조건으로 챌린지를 진행했을 때, 김싸피가 얻을 수 있는 총 점수는 몇 점인지 출력하는 프로그램을 작성하시오.

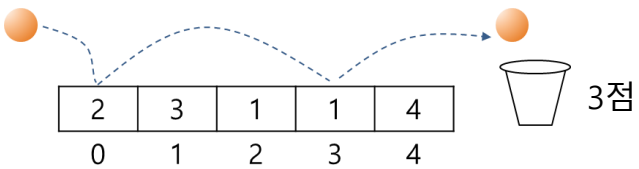
[조건]

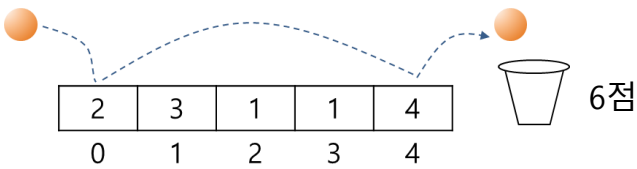
- 탁자에는 N개의 정수가 적혀있다.
- 탁구공이 숫자가 적힌 칸 위에서 튕기면 그 숫자만큼의 점수를 얻는다.
- 김싸피는 다음과 같은 방식으로 챌린지를 진행한다.
  - 김싸피는 항상 0번 칸에서 챌린지를 시작한다.
  - 튕기는 거리는 1, 2, 3, ... , N 으로 증가하여 총 N 번의 챌린지를 수행한다.

다음은 탁자에 5개의 숫자가 적혀 있을 때 수행하는 챌린지의 예시이다.

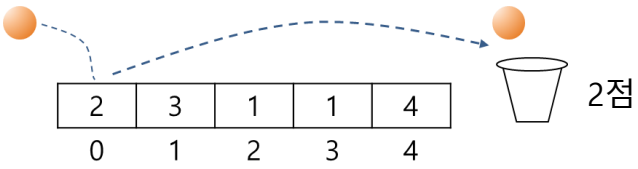








[Python트랙] 과목평가2 – 알고리즘 기본



이 경우 각 5번의 챌린지 점수를 모두 더하면 29점 을 획득할 수 있다.

[입력]

첫 줄에 테스트케이스 수가 주어진다.

각 테스트 케이스의 첫 줄에 N 이 주어진다.

두 번째 줄에서 N개의 정수 A_i (0 ≤ i < N) 가 띄어 쓰기로 구분되어 주어진다.

(3 ≤ N ≤ 20, -100 ≤ A_i ≤ 100)

[출력]

각 줄마다 " #T " (T는 테스트 케이스 번호)를 출력한 뒤, **N번의 챌린지를 모두 성공했을 때 획득 점수**를 출력 하시오. 단, **획득 점수가 0점 이하인 경우는 0점**으로 출력합니다.

[입력 예시]

3
5
2 3 1 1 4
10
-1 1 -1 1 -1 1 -1 1 -1 1
6
-5 -30 3 -39 49 97
(algo2_sample_in.txt 참고)

[출력 예시]

#1 29
#2 0
#3 209

(algo2_sample_out.txt 참고)

[Python트랙] 과목평가2 – 알고리즘 기본

| 문제 3 : 이진 탐색 (배점 : 25점)

1. 이진탐색에 대해 간단히 설명하시오.

2. 다음과 같은 배열과 이 배열에서 key값을 이진탐색으로 찾는 코드 일부가 주어진다. 비어 있는 부분을 파이썬 코드 또는 pseudo-code로 작성해 key를 찾으면 True, 실패하면 False를 반환하는 함수를 완성하라. **(단, 재귀함수로 작성해서는 안된다.)**

	0					N-1
a	2	4	7	9	11	19	23

예를 들어, 이진 탐색으로 배열 a에서 11을 찾을 경우(key = 11) true, 이진 탐색으로 10을 찾는 경우(key = 10) fail을 리턴한다. 함수에서 a는 배열 a[N], N은 총 배열 크기, key는 찾고자 하는 숫자이다.

```
def binarySearch(a, N, key)
        start = 0
        end = N-1
        while (









        )
        return False                    # 검색 실패
```