**데이터구조 및 프로그래밍실습 3분반**

**설계 프로젝트**

학번 : 202111426

이름 : 정동윤

본인의 Github 주소 :

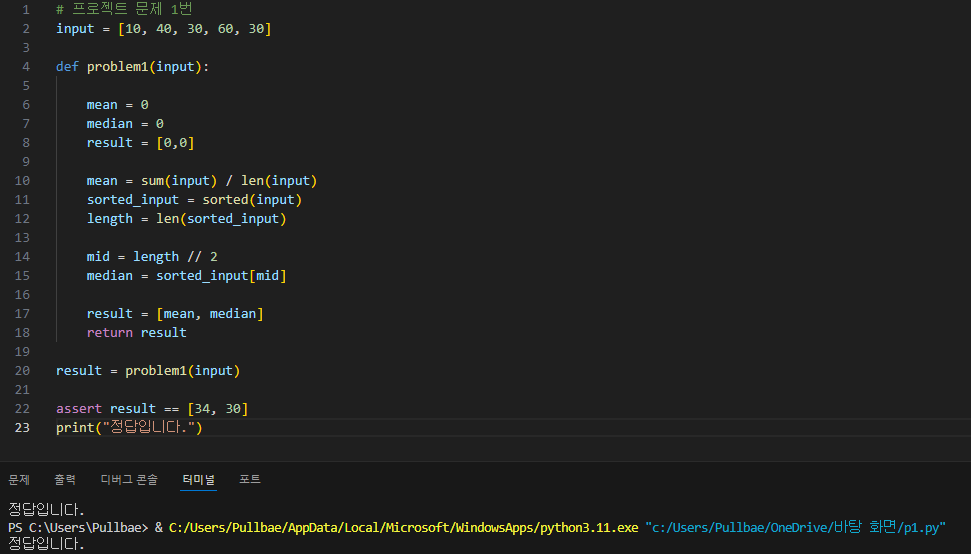
(본 과제의 코드를 본인의 Github에 업로드 하세요)

**문제 1.**

1. 본 문제를 풀기위해 사용한 데이터 구조를 제시하고 간단히 설명하세요.

문제 1번에서는 리스트를 이용했다. input으로 리스트에 입력을 받고 리스트에서 평균과 중앙값을 구했다. mean(평균)은 리스트 값을 더하고 리스트의 길이로 나눠서 구했다. median(중앙값)은 원래 중앙값은 숫자의 갯수가 짝수일 때는 중앙에 있는 값이 2개이므로 중앙값은 2개가 되는데 문제에서는 다섯개의 자연수가 주어졌을 때 중앙값을 구하는 것이므로 그냥 리스트의 중앙을 출력했다.

1. 본인이 작성한 파이썬 Code 캡처 이미지를 첨부하고 Algorithm Analysis를 수행하세요.



input으로 리스트의 값을 입력받는다.

mean(평균)은 input의 값을 전부 합치고 input의 길이로 나눈다.

median(중앙값)은 우선 sorted\_input으로 리스트를 오름차순으로 정렬한다. 그리고 mid(중앙)을 리스트의 길이를 2로 나눠서 정한 후, sorted\_input(정렬된 리스트)의 mid(중앙)을 출력한다.

시간 복잡도는

리스트의 값을 더하는 데 O(n)의 시간이 걸린다.

리스트를 정렬해야 하는데, Bubble sort, Selection sort, Insertion sort, Shell sort, Merge sort, Quick sort 등이 있다. 하지만 파이썬에서는 일반적으로 시간복잡도 O(n log n)을 가진다고 한다. 그래서 정렬하는 데 O(n log n)의 시간이 걸린다.

정렬된 리스트에서 중앙값을 찾는 데 O(1)의 시간이 걸린다.

그래서 전체 시간 복잡도는 O(n log n)이다.

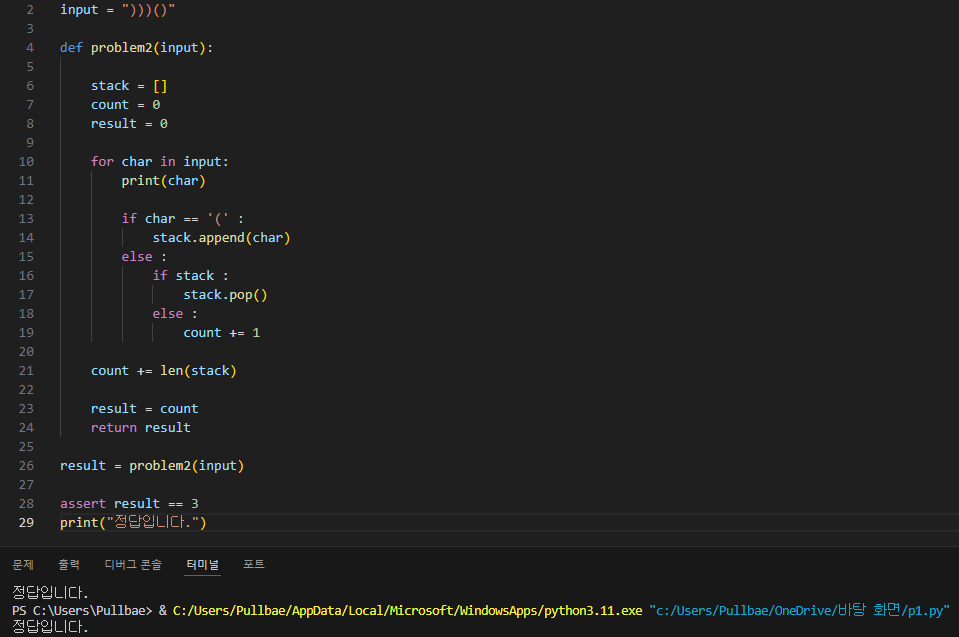
이 문제에서는 O(5 log 5)이다.

**문제 2.**

1. 본 문제를 풀기위해 사용한 데이터 구조를 제시하고 간단히 설명하세요.

여는 괄호 ‘(’을 저장하고 닫는 괄호 ‘)’을 만나면 여는 괄호 ‘(’를 제거한다. 여는 괄호 ‘(’가 없다면 count를 증가시켜 필요한 괄호를 출력할 수 있게 알고리즘을 구성했다. 어떤 괄호가 먼저 나가도 상관없기 때문에 그냥 가장 먼저 배운 먼저 들어온 것이 먼저 나가는 stack을 사용하였다.

1. 본인이 작성한 파이썬 Code 캡처 이미지를 첨부하고 Algorithm Analysis를 수행하세요.



input으로 받은 리스트에서 여는 괄호 ‘(’를 stack에 추가한다.

닫는 괄호 ‘)’이면 stack에서 여는 괄호 ‘(’를 하나 제거하고 여는 괄호 ‘(’가 없다면 count += 1해준다.

여는 괄호 ‘(’가 만약 count에 남아 있다면 닫는 괄호 ‘)’로 닫아줘야 하므로 남은 여는 괄호의 개수를 count에 더해주는 count += len(stack) 해준다.

시간 복잡도

문자열을 한 번 순회하므로 O(n)의 시간 복잡도를 가진다. 이 문제에서는 O(5)이다.

**문제 3.**

1. 본 문제를 풀기위해 사용한 데이터 구조를 제시하고 간단히 설명하세요.

산에서 곰돌이 푸가 벌집 찾기를 그만두기 전까지 몇 초 동안 벌집을 털 수 있는지 프로그램을 작성해야 한다. 숲이 미로 형태로 볼 수 있고 숲 전체를 탐색하며 벌집을 찾아야 하므로 최단 거리를 찾기에 용이한 너비 우선 탐색 bfs를 사용하였다.

1. 본인이 작성한 파이썬 Code 캡처 이미지를 첨부하고 Algorithm Analysis를 수행하세요.

input으로 산의 형태를 입력받는다.

‘N’은 정사각형 산의 한 변의 길이이다.

forest 리스트를 input 리스트로 초기화 해준다.

bear\_x, bear\_y를 i, j라 하고 그 곳이 9인 곳을 찾아 곰의 초기 위치를 찾는다. 그리고 지났으므로 0이라고 한다.

directions는 곰이 상, 하, 좌, 우로 이동하는 리스트이다.

bfs 함수를 정의한다. bfs로 곰이 현위치에서 이동하는 모든 경로를 탐색한다. visited 배열로 방문 위치를 체크, queue를 사용해서 bfs 탐색을 한다. 곰의 크기는 2로 설정되어 있고, 곰보다 작은 벌집 위치를 탐색하면서 먹을 수 있는 벌집이 있으면 possible\_targets으로 설정해서 리스트에 추가시킨다. possible\_targets가 비어있지 않으면 가장 가까운 벌집을 반환한다.

while문으로 곰이 갈 수 있는 벌집 꿀을 모두 먹을 때까지 반복한다. target에 bfs 함수를 호출해서 최단 경로를 찾는다. 찾으면 이동 시간을 time에 추가한다. 꿀을 먹으면 honey\_count를 증가시킨다. honeycomb\_count가 bear\_size와 같으면 곰의 크기를 1 증가시키고 honeycomb\_count를 0으로 초기화 시킨다.

bfs로 탐색할 수 있는 벌집이 없으면 종료한다.

time을 반환해서 총 걸린 시간을 출력한다.

시간 복잡도

bfs로 숲에서 모든 벌집을 탐색해야 하므로 시간 복잡도는 숲의 크기 N^2, O(N^2)의 시간이 걸린다. 이 문제에서 시간 복잡도는 숲의 크기가 4x4=16이므로 O(16)이다.

