**데이터구조 및 프로그래밍실습 3분반**

**설계 프로젝트**

학번 : 202323725

이름 : 공민혁

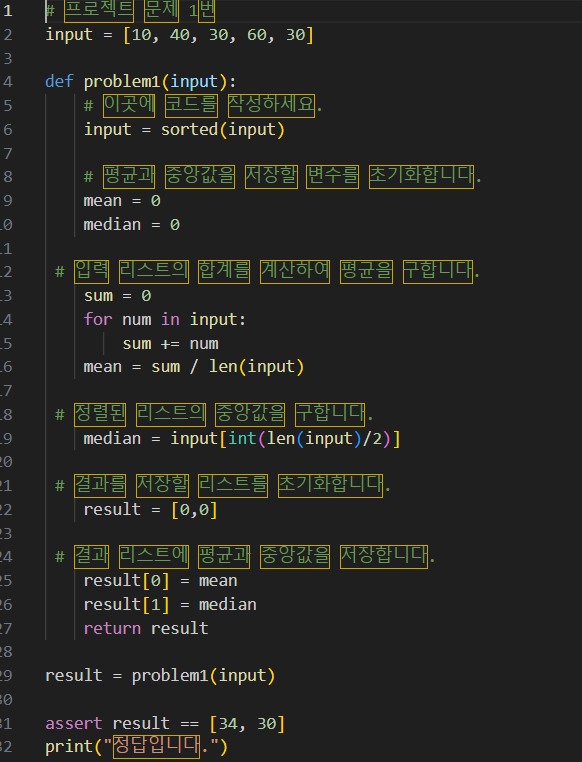
본인의 Github 주소 : https://github.com/Minhyeok1123/Data-structure-lab-project.git

(본 과제의 코드를 본인의 Github에 업로드 하세요)

**문제 1.**

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

입력 값을 input 리스트에 배열합니다. 리스트의 합계를 계산하여 평균을 구하고 리스트의 중앙값을 구합니다. 이후에는 평균과 중앙값을 결과 리스트에 저장하고 반환합니다.

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

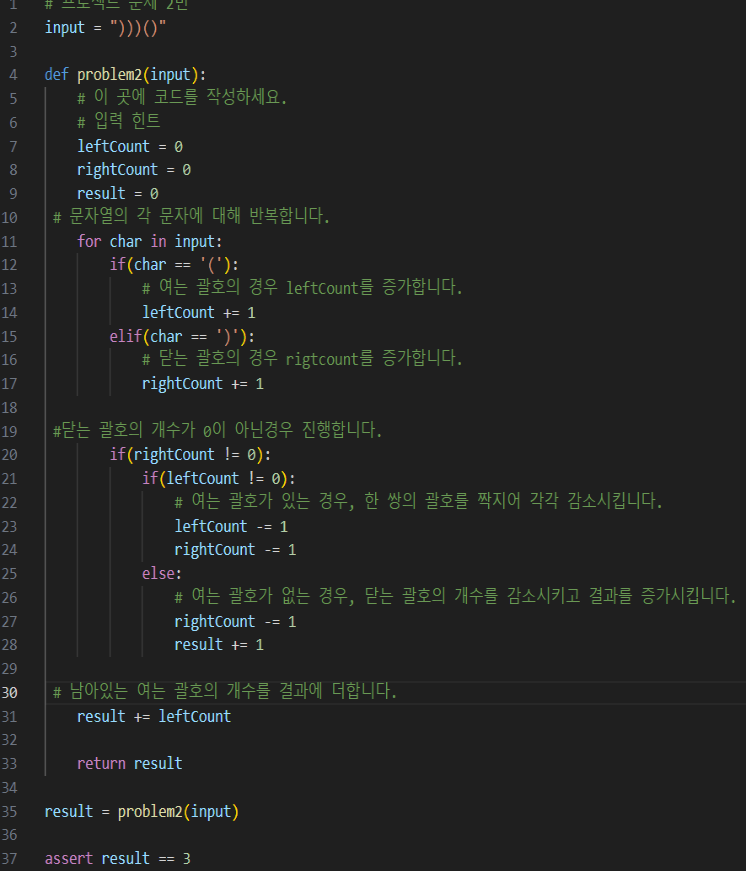
입력으로 받은 배열을 정리하고 평균값을 담은 mean 변수와 중앙값을 median 변수로 선언합니다. 평균값을 구하기 위해서 배열 안에 있는 모든 값을 sum 변수에 더한 후 배열의 길이만큼 나눠서 평균을 구합니다. 중앙값은 정렬된 배열에서 배열의 길이/2의 인덱스 값을 가져옵니다.

Sorted(input)의 경우, O( n log n)의 시간 복잡도를 가지고 합 계산은 (for num in input)은 리스트의 모든 요소를 합하는데 O(n) 시간이 소요됩니다. 나머지 계산에서는 O(1)의 시간이 걸려 코드의 시간 복잡도를 계산하면 O( n log n)이 나옵니다.

**문제 2.**

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

문자열의 문자를 순회하며 여는 괄호와 닫는 괄호의 개수를 셉니다. 닫는 괄호의 개수가 0이 아닌 경우에는 여는 괄호와 닫는 괄호를 짝지어 각 각 개수를 감소합니다. 여는 괄호가 없는데 닫는 괄호가 나오면 결과값을 증가합니다. 반복이 끝난 이후 남아있는 여는 괄호의 개수를 결과에 더합니다. 이후로는 결과값을 반환합니다.

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

‘(‘문자의 개수를 담는 leftCount 변수와 ‘)’ 문자의 개수를 담는 rightCount 변수를 선언합니다. 입력으로 받은 문자열을 하나씩 꺼내어 leftCount 또는 rightCount 값을 1 더하여 만약 ‘)’ 문자가 나왔을땐 앞의 문자중에 남아있는 ‘)’ 문자 여부를 판단해서 연산하며 마지막에는 남은 ‘(‘문자의 개수를 result 값에 더해줍니다.

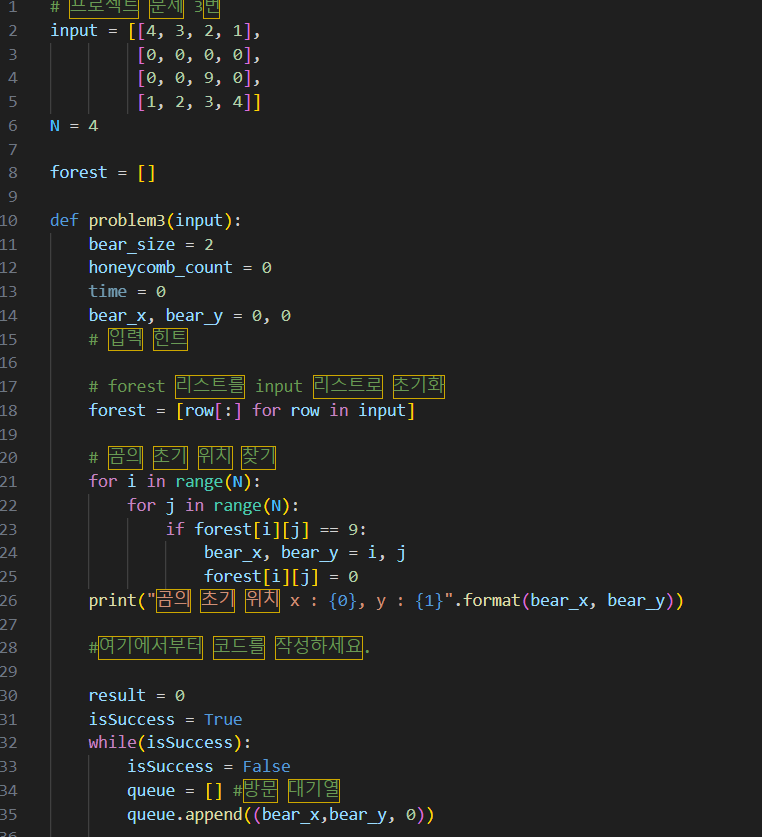
반복문(for)에서 문자열의 길이를 n이라고 하면 반복문은 O(n)의 시간 복잡도를 가집니다. 나머지 부분에서는 모두 O(1)의 시간 복잡도를 가지므로 위 코드는 O(n)의 시간 복잡도를 가집니다.

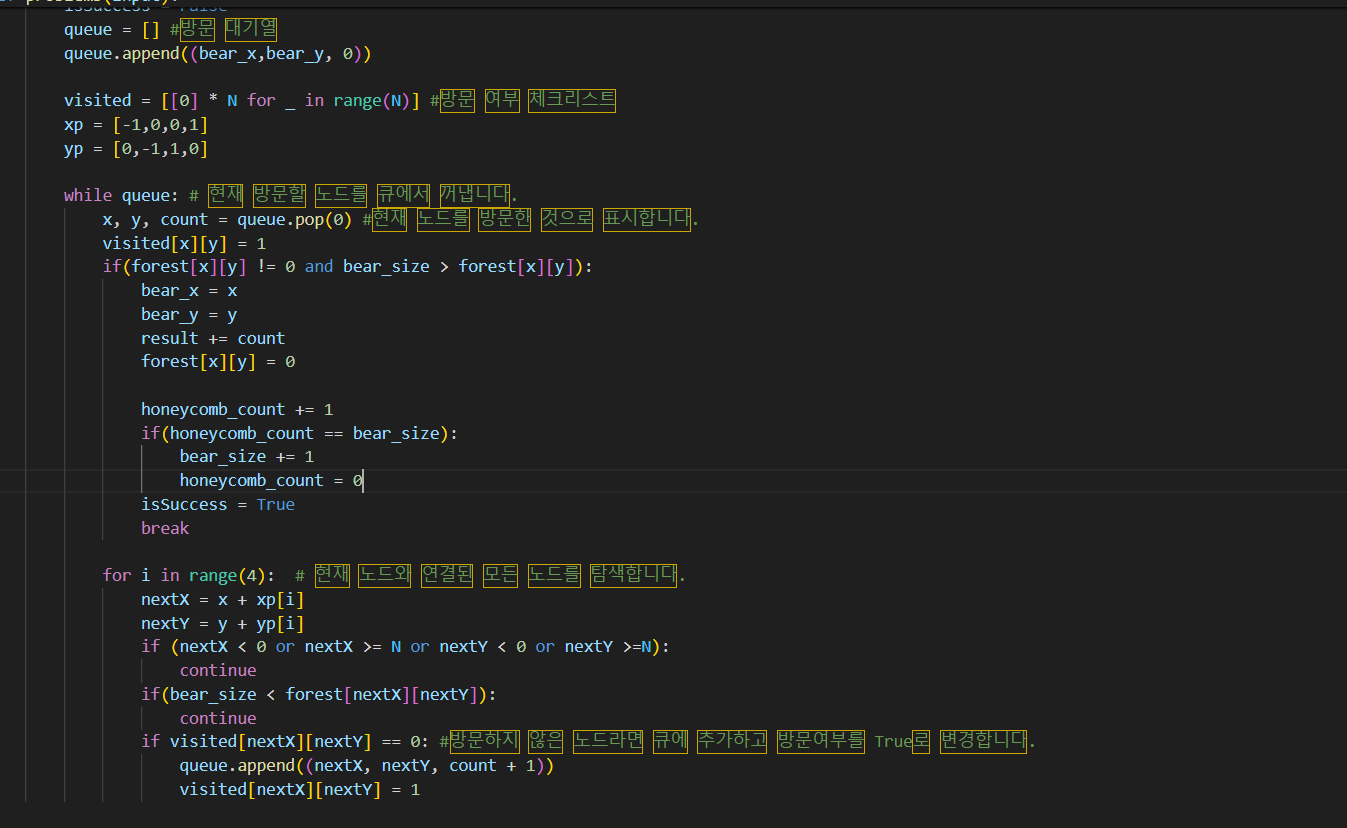
**문제 3.**

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

입력된 2차원 리스트를 ‘forest’로 초기화합니다. 곰의 초기 위치 또한 찾아 초기화합니다. BFS를 사용하여 곰이 이동할 수 있는 경로를 탐색하고 곰이 꿀벌집을 먹을 수 있는 경우, 해당 꿀벌집을 먹고 위치와 시간을 갱신합니다. 곰이 크기를 키우고 꿀벌집을 먹으며 가능한 경로를 탐색합니다. 모든 꿀벌집을 먹거나 더 이상 이동할 수 없을 때 까지 반복합니다. 최종적으로 이동한 총 시간을 반환합니다.

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





곰이 벌꿀을 먹을 수 있는지 여부를 담는 isSuccess 변수가 True 일때까지 반복합니다. 곰의 현재 위치에서 시작 할 queue 큐 구조에 x, y의 좌표와 이동횟수를 담는 0의 값으로 초기화 합니다. 곰의 방문 여부를 표시하는 visited 2차원 배열도 0값으로 초기화합니다. 이제 queue가 비어 있지 않을 때까지 반복하는데 방문하는 x, y좌표마다 방문표시 1을 해주고 위에서 선언한 xp와 yp 변수에서 상, 좌, 우, 하로 이동하여 같은 거리일 때 가장 위, 가장 왼쪽에 있는 벌통을 찾아간다, 조건을 맞춰주고 반복문을 돌려서 좌표를 이동합니다. 이동 시 배열의 범위를 넘어가거나 벌통의 크기가 크거나 이미 방문한 좌표면 무시하며 그렇지 않다면 queue에 이동한 좌표 값을 append합니다. 꿀에 도달했을 때는 곰의 좌표를 바꿔주고 이동 횟수 만큼 result 값에 더하고 곰의 특정조건이 충족되면 크기를 키웁니다.

For 루프를 통하여 초기 위치를 찾는 부분은 O(n^2)의 시간 복잡도를 가집니다. BFS는 각 노드를 한 번씩 방문하므로 O(n^2)의 시간 복잡도를 가집니다. 그 이후 while 루프에서는 곰이 모든 먹이를 찾을 때까지 반복하므로 최악의 경우 곰이 모든 셀을 방문 할 수 있습니다. 따라서 위 코드에서는 O(n^4)의 시간 복잡도를 가집니다.