**데이터구조 및 프로그래밍실습 3분반**

**설계 프로젝트**

학번 : 202111286

이름 : 김민주

본인의 Github 주소 : https://github.com/skyminju

(본 과제의 코드를 본인의 Github에 업로드 하세요)

**문제 1.**

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

리스트(List)

입력된 다섯 개의 자연수를 저장하기 위해 리스트를 사용하였습니다. 리스트는 자연수를 저장하고, 이 리스트를 통해 평균과 중앙값을 계산하는 데 사용되었습니다.

평균을 계산하기 위해 리스트의 모든 원소를 더하고, 리스트의 길이로 나누었습니다.

중앙값을 계산하기 위해 리스트를 정렬한 후, 정렬된 리스트에서 가운데 있는 원소를 선택하였습니다.

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

텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

자동 생성된 설명

**시간 복잡도(Time Complexity**) :

sum(input): 리스트의 모든 원소를 더하는 연산으로, 시간 복잡도는 O(n)입니다. 여기서 n은 리스트의 길이입니다.

len(input): 리스트의 길이를 구하는 연산으로, 시간 복잡도는 O(1)입니다.

sorted(input): 리스트를 정렬하는 연산으로, 시간 복잡도는 일반적으로 O(n log n)입니다.

sorted\_input[len(input) // 2]: 정렬된 리스트에서 중앙값을 찾는 연산으로, 시간 복잡도는 O(1)입니다.

전체적으로, 가장 시간이 많이 소요되는 부분은 정렬 부분이므로 이 알고리즘의 총 시간 복잡도는 O(n log n)입니다.

**문제 2.**

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

리스트(List)

입력된 괄호열의 균형을 맞추기 위해 리스트를 스택으로 사용했습니다. 스택은 후입선출(LIFO) 방식으로 작동하므로, 괄호의 짝을 맞추는 데 적합합니다.

문자열을 순회하면서 '('를 만나면 스택에 추가하고, ')'를 만나면 스택에서 제거하여 괄호의 균형을 유지했습니다.

스택이 비어 있을 때 ')'를 만나면, 추가로 필요한 왼쪽 괄호 수를 증가시켰습니다. 문자열 순회가 끝난 후, 스택에 남아 있는 '('의 개수를 세어 추가로 필요한 오른쪽 괄호 수를 계산했습니다.

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

텍스트, 스크린샷, 소프트웨어, 디스플레이이(가) 표시된 사진

자동 생성된 설명

**시간 복잡도(Time Complexity):**

문자열을 한 번 순회하면서 각 문자에 대해 상수 시간 작업을 수행하므로, 전체 시간 복잡도는 O(n)입니다. 여기서 n은 문자열의 길이입니다.

각 문자를 처리할 때 스택 연산(삽입 및 삭제)은 모두 O(1) 시간이 소요되므로, 추가적인 시간 복잡도는 없습니다.

**문제 3.**

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

2차원 리스트 (forest):

숲의 상태를 저장하는 2차원 리스트로, 각 요소는 빈 칸(0), 벌집(1~6), 또는 곰(9)을 나타냅니다.

큐 (deque):

BFS 탐색을 위해 사용되며, 큐에는 현재 곰의 위치와 이동 거리가 저장됩니다.

리스트 (visited):

각 칸의 방문 여부를 기록하는 2차원 리스트로, BFS 탐색 중에 중복 방문을 방지합니다.

리스트 (possible\_honeycombs):

곰이 먹을 수 있는 벌집의 위치와 거리, BFS 탐색 중에 발견된 벌집들을 저장합니다.

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

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

**시간 복잡도(Time Complexity):**

각 BFS 탐색은 모든 칸을 한 번씩 방문하므로 O(N^2)이며, 곰이 벌집을 계속 찾는 동안 이 과정을 반복하므로 최악의 경우 시간 복잡도는 O(N^3) 입니다.