**데이터구조 및 프로그래밍실습 3분반**

**설계 프로젝트**

학번 : 202210801

이름 : 소재혁

본인의 Github 주소 : https://github.com/sojaehyuk/data-structure-lab-2025.git

(본 과제의 코드를 본인의 Github에 업로드 하세요)

**문제 1.**

1. 본 문제를 풀기위해 사용한 데이터 구조를 제시하고 간단히 설명하세요.
2. **핵심 데이터 구조: 리스트 (List)**: 본 문제를 해결하기 위해 파이썬의 **리스트**를 핵심 데이터 구조로 사용했습니다. 리스트는 다음과 같은 역할을 수행했습니다.
3. **데이터 저장**: 입력된 정수 데이터를 순서대로 보관합니다.
4. **평균 계산 지원**: sum()과 len() 내장 함수를 통해 리스트의 총합과 길이를 쉽게 구해 평균을 계산하는 데 사용됩니다.
5. **중앙값 계산 지원**: sorted() 함수를 통해 데이터를 효율적으로 정렬할 수 있어, 중앙값을 찾는 전제 조건을 만족시키는 데 결정적인 역할을 합니다.
6. 본인이 작성한 파이썬 Code 캡처 이미지를 첨부하고 Algorithm Analysis를 수행하세요.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

* 시간 복잡도 (Time Complexity): **O(N log N)**

1. mean 계산: sum(input\_list)는 리스트의 모든 원소를 한 번씩 돌기 때문에 O(N)의 시간이 걸립니다.
2. sorted\_list 생성: sorted() 함수는 리스트를 정렬하고 평균 및 최악의 경우 모두 O(N log N)의 시간 복잡도를 가집니다.
3. median 계산: 인덱스 계산은 O(1)로 처리됩니다.

**문제 2**.

1. 본 문제를 풀기위해 사용한 데이터 구조를 제시하고 간단히 설명하세요.
   1. **핵심 데이터 구조: 정수형 변수를 사용하여 스택** 문제를 별도의 자료구조를 사용하지 않고, **정수형 변수**balance 하나를 사용하여 스택의 핵심 동작을 하도록 구현했습니다.
   2. (를 만나면 balance를 1 증가시켜 스택에 원소를 **push**하는 효과를 냅니다.
   3. )를 만나면 balance를 1 감소시켜 스택에서 원소를 **pop**하는 효과를 냅니다.
   4. 이 방식은 O(1)의 매우 효율적인 공간 복잡도를 달성하게 해줍니다.
2. 본인이 작성한 파이썬 Code 캡처 이미지를 첨부하고 Algorithm Analysis를 수행하세요.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

* 시간 복잡도 (Time Complexity): O(N)
  1. for char in input\_str: 이 루프는 문자열의 첫 번째 문자부터 마지막 문자까지 정확히 한 번씩만 방문합니다. 따라서 루프는 총 N번 반복됩니다.
  2. 루프 내부 연산: 루프의 각 반복에서 수행되는 if/elif 조건문, 덧셈/뺄셈 연산 등은 모두 상수 시간 O(1)에 완료됩니다.
  3. 따라서, 전체 실행 시간은 N(루프 반복 횟수) \*O(1)(내부 연산 시간) 이므로, 입력 문자열의 길이에 정비례하는 \*\*O(N)\*\*의 시간 복잡도를 가집니다. 이는 매우 효율적인 선형 시간 알고리즘입니다.

**문제 3**.

1. 본 문제를 풀기위해 사용한 데이터 구조를 제시하고 간단히 설명하세요.
   1. **핵심 데이터 구조: 2차원 리스트, 데크(deque), 리스트의 복합 사용**  
      이 문제는 여러 데이터 구조가 유기적으로 상호작용하여 해결됩니다.
   2. **2차원 리스트**: N x N 크기의 숲 지도(forest)와 너비 우선 탐색 시 방문 여부를 기록하는 visited 배열을 표현하는 데 사용되었습니다.
   3. **데크 (deque)**: 너비 우선 탐색(BFS)의 **큐(Queue)** 역할을 수행했습니다. O(1) 시간 복잡도의 popleft() 연산을 제공하여 최단 거리 탐색을 효율적으로 구현합니다.
   4. **리스트**: 한 번의 BFS로 찾은 먹이 후보들의 정보 (거리, 행, 열)를 저장하고, sort() 함수를 통해 문제의 우선순위에 맞는 최적의 목표를 쉽게 결정하는 데 사용되었습니다.
2. 본인이 작성한 파이썬 Code 캡처 이미지를 첨부하고 Algorithm Analysis를 수행하세요.

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

* 시간 복잡도 (Time Complexity): O(K \* N²)
  1. 바깥쪽 while 루프: 이 루프는 곰이 벌집을 하나 먹을 때마다 반복됩니다. 따라서 먹을 수 있는 총 벌집의 수인 K번에 비례하여 실행됩니다. 최악의 경우 K번 반복됩니다.
  2. 안쪽 BFS 탐색: 바깥 루프가 한 번 실행될 때마다, 현재 위치에서 가장 가까운 먹이를 찾기 위해 너비 우선 탐색(BFS)이 수행됩니다. BFS는 N x N 크기의 그리드 전체를 한 번씩 방문할 수 있습니다. 그래프 관점에서 정점(Vertex)의 수는 N²개, 간선(Edge)의 수는 약 4N²개이므로, BFS의 시간 복잡도는 O(V+E) = O(N² + 4N²) = O(N²) 입니다.
  3. 전체 시간: 따라서 전체 시간 복잡도는 (바깥 루프 반복 횟수) × (한 번의 BFS 시간)으로 계산할 수 있습니다. 이는 O(K \* N²)이 됩니다.