**데이터구조 및 프로그래밍실습 3분반**

**설계 프로젝트**

학번 : 202219312

이름 : 조은혁

본인의 Github 주소 :

(본 과제의 코드를 본인의 Github에 업로드 하세요)

**문제 1.**

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

배열(Array) 구조를 사용하였다.

단 이후 소팅된 배열은 순서를 가지므로, 리스트라고도 할 수 있다.

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

먼저 평균값(avg)을 구하는 과정은 굳이 소팅을 필요로 하지 않으므로, for 반복문으로 리스트에

있는 모든 원소들은 더한 후, len으로 추출한 칸 수로 나누어 구한다.

중앙값(mid)를 구하기 위해서는 배열이 순차적으로 정렬되어 있어야 하므로, 버블소팅 함수를 정의한 다음 적용하여 새로운 리스트 구조(sorted\_numbers)를 제작하였다.

총 칸수를 n으로 잡고, 이걸 2로 나눈 후의 몫을 중앙값 주소로 하여 중앙값을 구할 수 있었다.

위의 알고리즘을 바탕으로 짠 코드는 다음과 같다.

텍스트, 스크린샷, 소프트웨어, 운영 체제이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**문제 2.**

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

열린 괄호가 있다는 사실을 어떤 구조에 저장해 둘 수 있다면 닫힌 괄호가 나올 때 구조에서 열린 괄호를 제거하는 방식으로 괄호열이 온전한지 검사할 수 있다. 따라서 스택이나 큐 구조를 사용하면 구현할 수 있지만, 괄호열의 특성상 추가되는 순서의 역순으로 반환되어야 서로 섞이지 않으므로, 스택 구조를 사용하는 것이 더욱 적합하다.

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

우선 stack 구조의 클래스를 정의하였다.

괄호갯수를 체크해주는 함수를 정의하였는데, 작동 순서는 다음과 같다.

1)Input을 차례로 분석하며 스택에 괄호를 넣음.

-우선 검사한 input의 괄호가 여는 괄호면 스택에 추가함.

-닫는 괄호면 스택에 저장된 여는 괄호를 제거함.

2)순회 후에도 스택에 괄호가 남아 있다면, 이 괄호의 개수만큼 additions 변수에 추가함.

3)additions 변수의 값은 괄호열 완성을 위해 필요한 개수가 됨.

이 additions가 problem2 함수의 실행결과로 반환되어 정답과 비교하게 된다.

위 알고리즘을 바탕으로 작성한 코드는 다음과 같다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

만약 스택구조를 사용하지 않고 풀이한다면, 괄호를 각각 (=+1, )=-1로 정의한 후 완전한 괄호열의 괄호 총 카운트 결과는 0이 되는 방법을 이용할 수도 있다. 이렇게 판단자를 정의하는 방식은 어떤 괄호가 부족한지도 정확히 알 수 있다.

**문제 3.**

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

곰이 주변을 탐색하기 위해서는 일단 인접한 노드를 검사한 다음 주어진 조건에 따라 이동하는 BFS를 사용하는 것이 좋으므로, 그래프 구조를 사용하였다.

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

곰의 초기 크기는 2이고, 위치는 [2][2]이다. 한 칸 이동 할 때마다 시간이 1씩 추가된다.

이제 input 2차원 리스트를 수정하는 일이 없도록 복제 리스트 forest를 만든다. 그 이후 BFS를 방문 가능한 벌집이 없을 때까지 반복하는데, BFS의 과정은 다음과 같다.

1) 가장 가까운 벌집 탐색

-현재 위치에서 BFS를 시작하여 먹을 수 있는 가장 가까운 벌집 탐색. 먹을 수 있는 조건은 벌집크기 < 곰 크기, 이동 가능한 조건은 벌집크기 <= 곰 크기. 탐색을 하며 큐 구조에는 벌집의 행/열/현재 위치에서의 거리를 저장함.

2) 최종 먹이 결정

-먹을 수 있는 후보 먹이가 없다면 총 이동 시간을 반환하고 종료.

-먹이가 여러 개일 경우 이동할 벌집을 정함. / 거리가 짧은지를 최우선으로, 만약 거리가 같다면 행 번호가 작은 노드를 우선시함.

3) 이동 및 흡수

-곰의 위치를 목표 지점(먹이)로 갱신.

-초기 논리에 따라 누적 벌집 크기 = 곰 크기이면 곰크기+1. 누적 벌집크기 0으로 리셋.

위 과정을 흡수 가능한 벌집이 없을때까지 반복한다. 더 이상 없다면 총 이동시간을 반환한다.

위 내용을 정리해서 코드를 작성한 결과는 아래와 같다.

텍스트, 스크린샷이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.