11주차 결과보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20212022 이름: 이예준

**1.**

폰트, 상징, 그래픽, 텍스트이(가) 표시된 사진

자동 생성된 설명

먼저 미로와 미로의 각 집합관계를 표현해줄 2차원 배열을 동적할당하기 위해

이중 포인터를 전역변수로 선언해준다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

그 다음 Init\_Maze() 함수를 통해 Maze 배열은 모든 위치가 사방에 벽으로 둘러싸이게 하고,

Set 배열은 모든 위치가 서로 다른 집합이 되게 초기화 시켜준다.

이때, Maze 배열은 이동하기 위한 공간과 벽이 있는 공간이 모두 필요하기 때문에

입력 받은 크기보다 2배 + 1이 된다.

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

Make\_Maze() 함수를 통해 본격적으로 미로를 생성하게 된다. 먼저 난수가 매 실행마다

다른 난수를 생성해야 하기 때문에 srand(time(NULL)); 를 적는다.

첫 번째 for문은 다른 줄로 이동하기 위한 반복 문이며, 두번째 for문은 해당 줄을 가로로

이동하면서 벽을 제거하기 위한 반복문이다.

벽을 제거할지 말지는 rand()%2 로 임의로 선택해 flag가 1이면 제거, 0이면 유지가 결정된다.

만약 마지막 줄이라면 서로 다른 집합은 모두 결합해야 되기 때문에 flag가 1로 고정한다.

그 다음 if문으로 flag가 1이고, 옆에 있는 곳이 서로 다른 집합이라면 Maze 배열에서 그 사이에

있는 벽을 제거하고, for문으로 해당 줄을 쭉 순회하면서 집합을 업데이트한다.

이렇게 해당 줄의 벽에 대해 모든 작업을 마치면 다음 과정으로 넘어가게 되는데

만약 해당 줄이 마지막 줄이라면 다음 과정으로 넘어가지 않고 그대로 함수가 끝나게 된다.

왜냐하면 다음 과정이 아랫줄에 대해 벽을 제거하는 과정이기 때문이다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

이 과정은 아랫줄에 대해 벽을 제거하고 집합을 업데이트하는 과정이다.

처음에 set변수를 해당 줄의 첫번째 집합 번호로 초기화해주는데, 이 변수를 통해 현재 위치가

같은 집합 내에 있는지 다른 집합으로 넘어갔는지 알 수 있다. 만약 현재 위치가 다른 집합내에

있다면 이전 집합에서 벽을 제거한 횟수가 최소 한번은 넘어가는지 확인하고,

한번도 제거한 적이 없다면 이전 집합내의 위치 중 임의로 하나를 선택해 아랫줄과 연결한다.

그 다음 set 변수는 바뀐 집합 번호로 업데이트하고, 벽을 제거했는지 확인하는 chk 변수는

0으로 초기화한다. 마지막으로 이전 집합의 범위를 지정하기 위해 필요한 idx 변수도 현재 위치로

업데이트한다. flag를 0, 1 중 임의로 선택하고, 만약 flag가 1이라면 현재 위치의 아래 줄과

연결한다. Maze 배열에서 아랫줄과의 벽을 제거하고, Set 배열에서 집합을 업데이트한다.

하지만 이렇게 구현하게 되면 해당 줄의 마지막 집합에서 벽을 제거했는지 안 했는지

판별할 방법이 없기 때문에 for문이 끝나게 되면 chk 변수를 통해 마지막 위치에서 벽을

제거하지 않은 경우 이전 집합내의 위치 중 임의로 하나를 선택해 아랫줄과 연결한다.

텍스트, 폰트, 친필, 스크린샷이(가) 표시된 사진

자동 생성된 설명

미로를 모두 생성하게 되면 Save\_Maze() 함수를 통해 미로를 파일에 저장한다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

마지막으로 동적 할당했던 Maze배열과 Set배열을 메모리 해제한다.

시간 복잡도 : 첫번째 inner loop에서 가로줄 이동하면서 벽을 제거하기 위한 loop와

Set배열을 업데이트하기 위한 loop가 합쳐져서 O(Weight2)이 되며,

outer loop와 합치면 최종적으로 O(Height\*Weight2)이 된다. Maze의 높이와 너비를

미지수 처리한다면 최종적으로 시간 복잡도는 O(n3)이 된다.

공간 복잡도 : Maze 배열과 Set 배열을 동적 할당하기 위한 메모리가 필요하기 때문에

공간 복잡도는 O(n2)이다.

미로를 생성하는 알고리즘을 생각했을 때는 옆에 있는 다른 집합과 합쳤을 때

(*예를 들어 집합 1과 2가 서로 만났을 때*), 그 옆에 있는 부분만 업데이트(*2를 1로 바꾸기*)하면

된다고 생각해서 구현했는데, 집합이 제대로 업데이트되지 않아 Cycle이 발생했었다.

이를 해결하기 위해서 해당 줄에 있는 모든 집합을 업데이트(*위에 있는 예를 갖고 오면*

*해당 줄에 있는 모든 2를 1로 바꾸기*)하는 코드를 추가하여 해결했다.