11주차 예비보고서

전공: 신문방송학과 학년: 3학년 학번: 20191150 이름: 전현길

1. 문제 해결에서 언급한 미로 생성 알고리즘들에서 Eller's algorithm을 제외한 나머지 알고리즘 중 하나를 선택하여 이를 조사하고 이해한 후 그 방법을 기술하시오.

미로는 크게 강의자료에 따르면 크게 **완전 미로(perfect maze)**와 **불완전 미로(imperfect maze)**로 구분된다. 완전 미로의 경우, 임의의 서로 다른 출발점과 도착점을 결정할 경우 두 지점을 연결하는 경로가 오로지 하나 존재하며, 폐쇄된 공간이 존재하지 않는 미로이다. 따라서 완전 미로는 순환 경로(cycle path)가 존재하지 않는다.

완전 미로를 생성하는 미로 생성 알고리즘은 다양하지만, 크게 그래프 이론을 기반으로 한 알고리즘과 그렇지 않은 알고리즘들로 구분된다.

그래프 이론을 기반으로 한 알고리즘들은 대표적으로 Recursive Backtracking, Kruskal, Prim algorithm 등이 존재하고, 흥미롭지만 비효율적인 알고리즘으로 Wilson, Aldous-Broder algorithm이 존재한다. 셀을 선택하는 기준에 따라 Recursive Backtracking, Prim 알고리즘을 오가는 growing tree 알고리즘과 같은 경우도 있다.

이외의 알고리즘의 예로는 recursive division, binary tree, sidewinder, hunt & kill algorithm, 그리고 강의자료에 기록된 eller's algorithm을 들 수 있다.

미로 생성 알고리즘에 대한 유의미한 평가 기준은 크게 두 가지로 나눌 수 있다. **미로의 무작위성의 정도, 시간/공간 복잡도**가 바로 그것이다.

비록 완전 미로를 효율적으로 만들더라도 그 모양이 굉장히 뻔하고 예측 가능하다면 좋은 미로 생성 알고리즘이라고 볼 수 없을 것이다. 나쁜 예로 미로의 각 행의 첫 셀에서부터 오른쪽 벽을 계속 뚫은 뒤, 행의 끝에 도달하면 아래를 뚫으며, 해당 행이 마지막 줄이라면 종료하는 알고리즘을 들 수 있다. 해당 알고리즘은 완전 미로를 생성할 것이 분명하고, 시간 복잡도, 공간 복잡도 측면에서도 효율적일 테지만 좋은 미로 생성 알고리즘이라고 보기는 어렵다.

이처럼 다양한 미로 생성 알고리즘들 중에선 알고리즘의 특성에 따라 특정한 편향성을 갖는 경우가 많다. 대표적으로 recursive backtracking은 갈림길이

적고 길게 이어지는 통로를 갖는 경향이 있으며, prim/kruskal 알고리즘은 이와는 반대로 길이가 짧고 갈림길이 많은 통로들을 갖는 경향이 있다. sidewinder 알고리즘의 경우 미로의 첫 행 또는 마지막 행이 뻥 뚫려 있기 때문에 위에서 아래로 이동하는 경로를 찾기 쉬워지는 경향이 있다. binary tree 알고리즘들은 sidewinder 알고리즘의 특성을 공유하는 한편 대각선 편향이 발생한다. 마지막으로 recursive division 알고리즘은 미로가 네모 반듯한 구획들로 분할된 것처럼 보이게 된다.

이러한 측면에서 봤을 때, 충분히 무작위적이고 난이도 있는 미로를 만들기 위해 kruskal, prim 알고리즘이 적합할 것으로 선택했으며, 그 중 kruskal 알고리즘을 조사하기로 결정했다.

kruskal 알고리즘은, 그래프 내의 모든 정점을 가장 적은 비용으로 연결하기 위해 사용되는 알고리즘이다. 다른 말로 **최소 신장 트리(minimum spanning tree)**를 구하기 위한 알고리즘이라고도 한다. 주로 최소 케이블로 통신 네트워크를 구축할 때, 최소 비용으로 각 도시를 도로로 연결하고자 할 때 사용할 수 있는 알고리즘이다.

각 용어들을 먼저 정의한다. **그래프(graph)**란 공집합이 아닌 정점(vertex)들의 집합 V와 정점을 연결하는 간선(edge)들의 집합 E로 구성되는 자료구조이며, **G = (V, E)**로 표기한다.

다음으로 **신장 트리(spanning tree)**란 그래프의 **최소 연결 부분 그래프(minimum connected subgraph; MCS)**라고도 불리는데, 이는 신장 트리가 간선의 수를 최소한으로 하여 그래프의 정점을 모두 연결하는 그래프를 의미하기 때문이다. 간선의 수를 최소화하므로 신장 트리의 간선들의 집합 E는 항상 원 그래프의 부분집합이며, 정점 n개가 있을 때 간선 n - 1개를 갖게 된다. 정점 n개와 간선 n - 1개를 갖는 그래프는 필연적으로 **순환(cycle)**이 발생할 수 없으므로 트리가 된다.

이 때 **순환(cycle)**은 먼저 **보행(walk)**을 특수하게 한정함으로써 정의할 수 있다. 보행(walk)은 그래프에 속한 정점과 간선의 반복된 sequence로, 정점과 간선의 중복을 허용한다. 이 때, 시작 정점과 끝 정점이 같을 경우 **닫힌 보행(closed walk)**으로, 다를 경우 **열린 보행(open walk)**으로 정의한다. 다음으로 **트레일(trail)**은 같은 간선을 반복하지 않는 보행이며, **경로(path)**는 같은 정점을 반복하지 않는 보행을 의미한다. 이 때 **회로(circuit)**는 시작 정점과 끝 정점이 같은 트레일로 정의할 수 있으며, **순환(cycle)**은 시작 정점과 끝 정점 외

에 같은 정점이 중복되지 않는 회로로 정의할 수 있다.

kruskal 알고리즘이 최소 비용 신장 트리를 찾는 핵심 개념은 간단하다. 간선을 **거리가 짧은 순서대로 그래프에 포함시키는 것**이다. 단 그래프에 순환이 없는지 탐색해 주어야 한다. n - 1개의 간선이 삽입되었다면 정점이 연결되었으므로 종료한다. 구현은 다음과 같다.

1. 간선의 가중치를 기준으로 오름차순 정렬한다.
2. union-find 알고리즘을 사용해 사이클을 탐색한다.
3. 사이클이 없다면 삽입된 간선에 연결된 정점을 MST 집합에 삽입한다.
4. n-1 개의 간선이 삽입되었다면 종료한다.

이 때, union-find 알고리즘이 사이클을 탐색하는 원리는 사이클의 시작 정점과 끝 정점이 같다는 원리에서 비롯된다. 이를 위해서 새로 추가하는 간선의 양쪽 정점의 parent node를 확인할 수 있다. parent node가 같다면 서로 같은 집합이기 때문에 해당 정점들을 삽입하면 순환이 발생함을 알 수 있고, 이 경우 삽입을 수행하지 않으면 된다.

2. 본 실험에서 완전 미로(Perfect maze)를 만들기 위하여 선택한 알고리즘 구현에 필요한 자료구조를 설계하고 기술하시오. 설계한 자료구조를 사용하였을 경우 선택한 알고리즘의 시간 및 공간 복잡도를 보이시오.

미로 생성 문제의 경우 kruskal 알고리즘을 만들기 위해 필요한 자료구조는 다음과 같다.

1. **간선을 저장할 구조체 Edge(시작점, 끝점, 가중치 저장)**
2. **WIDTH * WIDTH - 1개의 Edge를 저장할 1차원 Edge 벡터**
3. **각 정점의 연결 관계를 저장할 1차원 int 배열**
4. **미로들의 모든 벽을 저장하는 2차원 char 배열**

union-find 연산에는 상수 시간 복잡도가 필요하므로, 실질적으로는 간선을 오름차순 정렬하는 작업에 가장 많은 시간 복잡도가 발생한다. 따라서 kruskal 알고리즘의 **시간 복잡도는 O(E · logE)**이며, 간선의 개수가 적은 sparse graph일 경우에 효율적으로 작동한다. 미로들의 모든 배열을 저장하기 위해 요구되는 메모리인 (WIDTH * 2 + 1)^2의 메모리가 가장 큰 메모리이므로 **공간 복잡도는 O(WIDTH^2)**이다. 단, 미로의 기본 필드를 그리기 위해 2 * WIDTH

* (WIDTH – 1)개의 간선이 1차원 Edge 벡터에 저장되기 때문에, 이 점을 고려하면 **공간 복잡도가 O(E)**라고도 볼 수 있다. O(E)나 O(WIDTH^2)이나 공간 복잡도상 상수 수준의 차이만 존재한다.

실제로 미로 생성 알고리즘에서 간선의 가중치는 1이라고 볼 수 있지만, 각 간선별로 가중치를 무작위로 할당함으로써 무작위적인 미로를 만들 수 있다.