2020년도 2학기 컴퓨터공학설계및실험Ⅰ

12주차 미로(Maze) 1주차 예비보고서

20161663 허재성

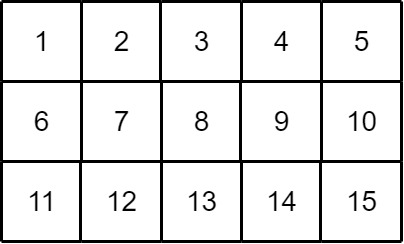
1. 실습 목적

미로는 복잡한 길을 찾아 출발점부터 시작해 도착점까지 도달하는 퍼즐로 직사각형으로 배열된 방과 이러한 방들 사이를 막고 있는 벽으로 구성된다. 벽은 각 방의 상하좌우 사면에 도달할 수 있는데 인접한 두 방 사이에 벽이 없을 경우 두 방 사이를 자유로이 지날 수 있다. 본 프로젝트에서는 다음에 설명하는 완전 미로를 만들 수 있는 여러 가지 방법을 알아보고 이를 직접 구현하기 위한 자료구조 설계 및 프로그래밍을 통해 미로를 생성해보도록 한다.

2. 관련 이론

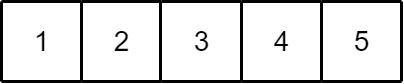
2-1. Eller’s Algorithm을 정리하여 서술하시오.

N\*M 미로는 N개의 열과 M개의 행으로 이루어진 NM개의 방을 가지고 초기에 각 방들은 사면이 벽으로 막혀 있다. 예를 들어 5\*3 미로의 초기 상태는 다음 그림과 같다.

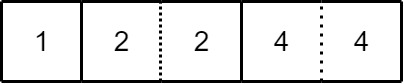


각 방은 벽(실선에 해당)으로 인해 서로 분리된 상태이므로 모두 다른 집합에 해당한다. 서로 다른 집합을 각각 1, 2,…, 15라 하자.

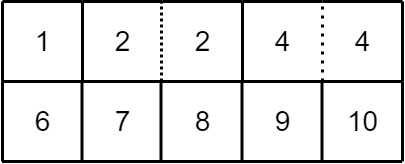
Elller’s Algorithm으로 미로를 생성하기 위해 첫 번째 행부터 N번째 행까지 미로를 차례로 만들어낸다.



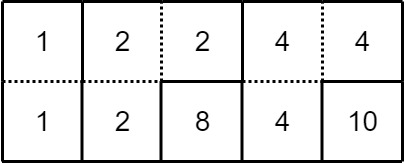
첫 째 행의 방들 사이의 벽을 임의로 제거한다. 제거 후 서로 연결된 방들은 같은 집합에 속하게 된다. 편의상 두 집합이 합쳐질 경우, 더 작은 집합 번호로 합쳐진다고 가정한다. 임의로 벽을 제거하면 다음과 같다. 제거된 벽은 점선으로 나타낸다.



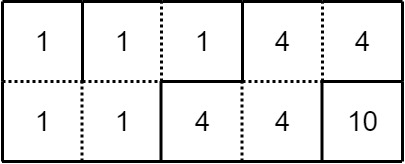
2번 방과 3번 방 사이의 벽을 제거해서 두 방이 연결된 상태이다. 2, 3번 방은 같은 집합 2에 속하게 된다. 마찬가지로 4번, 5번 방도 연결되어 같은 집합 4에 속하게 된 상태이다.



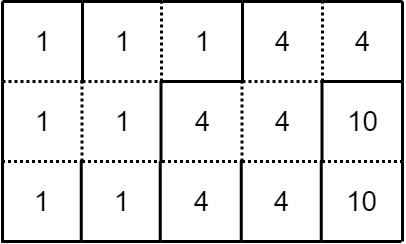
첫 째 행의 방들 사이의 벽을 임의로 제거 후 첫 째 행의 각 방들과 둘 째 행의 같은 열의 방들 사이의 벽을 임의로 제거해야 한다. 하나도 제거하지 않으면 첫 째 행의 방들과 둘 째 행의 방들은 서로 분리되어 각각 폐쇄된 공간이 되어 완전 미로가 될 수 없으므로 적어도 하나의 벽은 제거되어야 한다. 다음과 같이 제거했다.



두 번째 행에 대하여 첫 번째 행과 같은 방법으로 두 번째 행의 각 방 사이의 벽을 임의로 제거한다. 다음과 같이 제거했다.



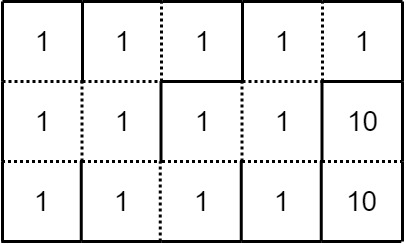
이제 두 번째 행과 세 번째 행의 방들 사이의 벽을 제거해야 한다. 다음과 같이 제거했다.



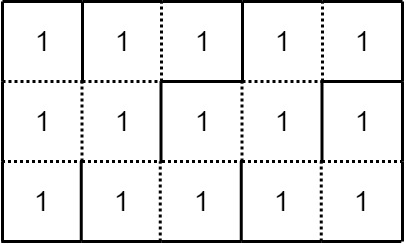
마지막으로 마지막 행의 방들 사이의 벽을 제거해야 한다. 이때 주의사항으로 두 방 사이의 벽을 제거할 때 두 방이 이미 같은 집합에 속할 경우 두 방 사이의 벽을 제거해서는 안된다. 두 방이 같은 집합에 속한다는 것은 두 방 사이의 경로가 이미 존재한다는 의미이다. 그런데 추가적으로 두 방 사이의 벽을 제거하면 경로가 하나 더 생기게 되어 임의의 두 방 사이의 경로가 유일해야 한다는 완전 경로의 조건을 위배하게 된다. 따라서 두 방이 속한 집합이 서로 다를 경우에만 두 방 사이의 벽을 지울 수 있다.

위의 조건은 벽을 지우는 모든 경우에 고려되어야 한다. 이전의 예에서는 지우려는 두 방의 집합이 같은 경우가 나타나지 않아서 언급을 생략했다.

해당 조건을 고려하면 위의 상태에서 세 번째 행에서 우선 지울 수 있는 벽은 두 번째 벽이다. 두 번째 벽을 지우면 다음과 같이 된다.

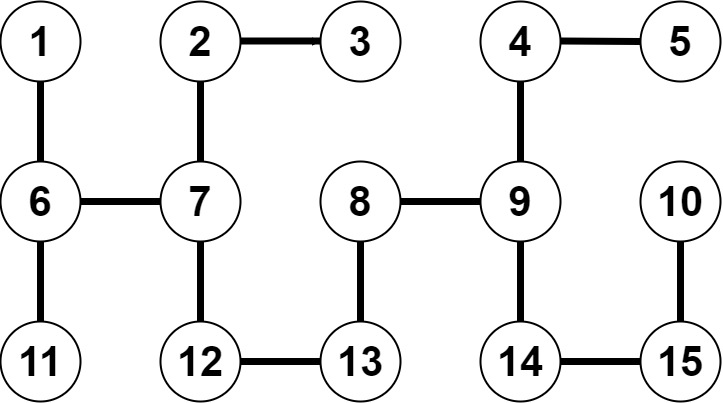


연결된 방의 집합이 1과 10만 남았다. 집합 10의 방들은 폐쇠 공간이므로 위의 미로는 완전 미로가 아니다. 집합 1과 집합 10을 연결해주기 위해 벽을 하나 더 제거한다.



마지막 벽을 제거하면 위와 같은 미로가 완성된다.

미로는 자료 구조 중 그래프(Graph)로 구현할 수 있다. N\*M 미로의 경우 NM개의 방이 있는데 각 방은 그래프의 정점(Vertext), 두 방 사이에 벽이 있는지는 두 방에 해당하는 정점을 잇는 간선(Edge)가 있는지 없는지를 이용해 구현할 수 있다. 두 정점을 잇는 간선이 있을 경우, 직접 연결되어 있으므로 두 방 사이에는 벽이 없다. 반대로 간선이 없을 경우, 두 방 사이에는 벽이 있다. 위의 미로를 그래프로 그려보면 다음과 같다.



각 정점의 번호는 15개의 방의 구분을 위해 붙인 것으로 방이 속한 집합과는 무관하다. 모든 방은 같은 집합(위의 미로에서 1)에 속한다.

임의의 두 방 사이에 벽이 없을 경우, 그래프에서 두 방을 의미하는 두 정점이 간선으로 연결되어 있다. 예를 들어 미로에서 8번 방과 9번 방의 경우 두 방 사이의 벽이 없으므로 두 정점 8, 9는 간선으로 연결되어 있고, 반대로 7번 방과 8번 방 사이에는 벽이 있으므로 두 정점 7, 8은 정점으로 연결되어 있지 않다.

완전 미로이므로 폐쇄된 방이 없이 임의의 한 방에서 다른 임의의 한 방으로 도달할 수 있어야 한다. 따라서 그래프에서 모든 정점은 연결되어 있어야 한다. 모든 정점이 연결되어 있다는 것은 그래프의 임의의 한 정점에서 간선을 따라 이동하면 그래프 상의 어떤 다른 정점에도 도달할 수 있다는 의미이다. 즉 임의의 두 정점 사이에 경로(Path)가 존재해야 한다. 이러한 그래프를 연결 그래프(connected graph)라고 한다. 완전 미로를 나타내는 그래프는 연결 그래프여야 한다.[1]

또한 완전 미로를 나타내는 그래프는 사이클이 존재해서는 안된다. 이러한 그래프를 acyclic하다고 한다.[1]

어떤 그래프가 연결 그래프이고 사이클이 존재하지 않으면 해당 그래프는 트리(Tree)이다. 이전에 배운 트리 자료구조는 그래프 자료구조의 특수한 형태이다.[1]

즉 다음 관계가 성립한다.

|  |
| --- |
| **Tree ⊂ Graph** |

**Tree ⊂ Graph** 이다.

N 개의 정점을 가진 트리의 간선의 개수는 N-1개이다. 따라서 N\*M 완전 미로를 나타내는 그래프는 NM개의 정점을 가지고 있으므로 간선의 개수는 NM -1 개이다.

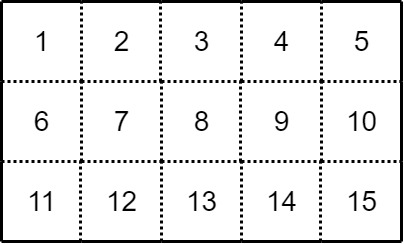
완전 미로가 아닌 경우는 그래프가 연결 그래프가 아니거나, 사이클이 존재하는 경우이다. 이와 같은 경우는 기타에서 알아본다.

2-2. Eller’s Algortihm 외에 미로 생성 알고리즘이 어떤 것이 있는지 조사하고, 그 방법을 서술하시오.

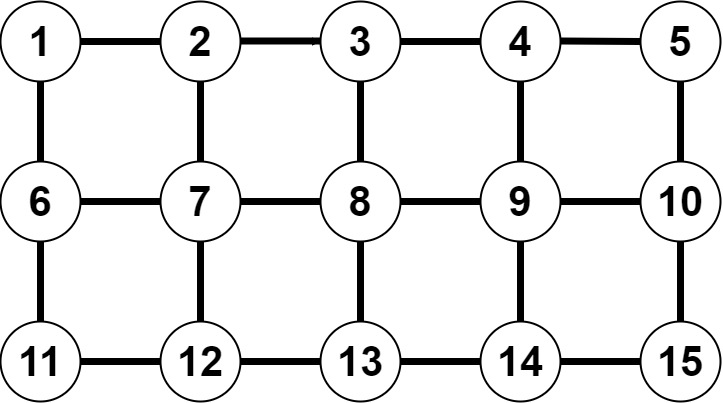
Eller’s Algorithm 외에 미로를 생성할 수 있는 알고리즘은 Kruskal’s Algorithm과 Prim’s Algorithm이 존재한다.

임의의 그래프의 부분 그래프 중 해당 그래프의 정점을 모두 가지고 있고 부분 그래프가 트리인 그래프를 신장 트리(Spanning Tree)라고 한다. 간선마다 비용(cost)가 있는 그래프에서 그래프의 신장 트리 중 간선의 비용의 합이 최소인 신장 트리를 최소 비용 신장 트리(Minumum cost spanning tree, MST)라고 한다. Kruskal’s Algorithm과 Prim’s Algorithm은 그래프의 최소 비용 신장 트리를 찾는 알고리즘이다.[1]

미로를 나타내는 그래프에서 간선은 벽이 없어서 두 방 사이를 이동 가능하다는 의미이다. 모든 간선의 비용을 1로 고정할 수 있다. Eller’s Algorithm과 다르게 초기 상태는 다음과 같이 인접한 방끼리 모두 연결되어 있다고 가정한다.



위와 같은 미로의 초기 상태를 그래프로 나타내면 다음과 같다.



위아래 좌우로 인접한 두 방에 해당하는 정점이 간선으로 연결된 것을 알 수 있다.

위의 그래프에서 간선의 비용은 모두 똑같이 1이므로 모든 신장 트리의 비용의 값은 같다. 따라서 모든 신장 트리가 최소 비용 신장 트리라고 생각할 수 있다. 모든 신장 트리가 최소 비용 신장 트리이고 모든 신장 트리는 완전 미로의 그래프이므로 Kruskal’s Algorithm 또는 Prim’s Algorithm을 이용해 최소 비용 신장 트리를 찾아내면 해당 최소 신장 트리가 나타내는 미로는 완전 미로가 된다.

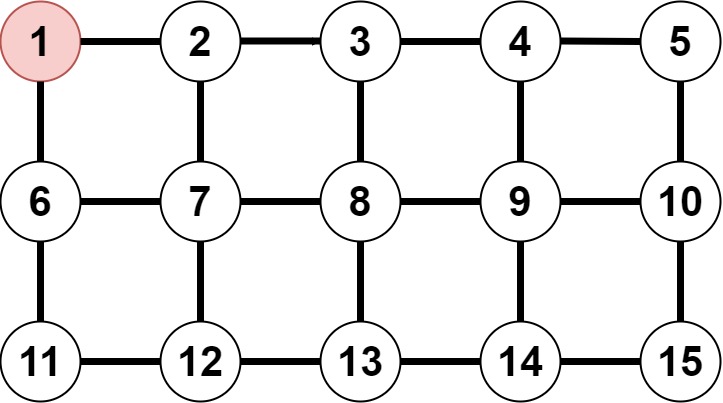
Eller’s Algorithm과의 차이는 Eller’s Algorithm은 모든 방이 서로 막혀 있는 상태에서 벽을 없애 가며 미로를 생성하였다. 그래프의 관점에서 모든 서로 연결되어 있지 않은 정점들을 연결해가면서 신장 트리를 생성하였다. 반대로 Kruskal’s Algorithm과 Prim’s Algorithm은 벽이 없는 미로에서 벽을 만들어가며 미로를 생성한다. 그래프의 관점에서 연결되어 있는 정점들 사이의 간선을 제거하면서 신장 트리를 생성한다. 두 알고리즘 중 Prim’s Algorithm에 대해 더 알아본다.

Prim’s Algorithm의 고수준 알고리즘은 다음과 같다.[2]

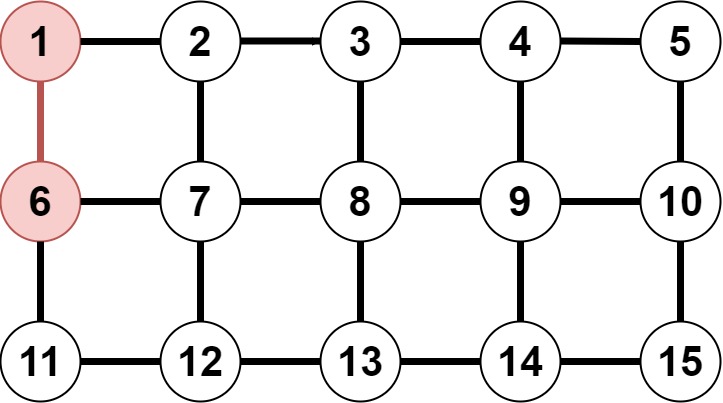
|  |
| --- |
| // V : 그래프의 모든 정점의 집합  F = ф; // 최소 비용 신장 트리(MST)에 포함되는 간선의 집합, 처음에는 공집합  Y = {v1}; // MST에 속하는 정점의 집합, 처음에는 첫 번째 정점 v1만 포함  while(MST 미완성) {  Y에서 가장 가까운 V-Y에 속한 정점 선택;  해당 정점’을 Y에 추가;  그 정점과 Y를 잇는 간선 추가;  if(Y ==V)  MST 완성;  } |

Prim’s Algorithm은 처음 정점을 집합 Y에 포함해서 Y에 속하지 않는(V-Y에 속하는 정점 중) Y와 가장 가까운 정점을 찾아 정점은 Y에, Y와 연결하는 간선을 F에 추가한다. 이와 같은 과정을 반복해 Y가 V와 같아지면 신장 트리가 완성되므로 종료된다.

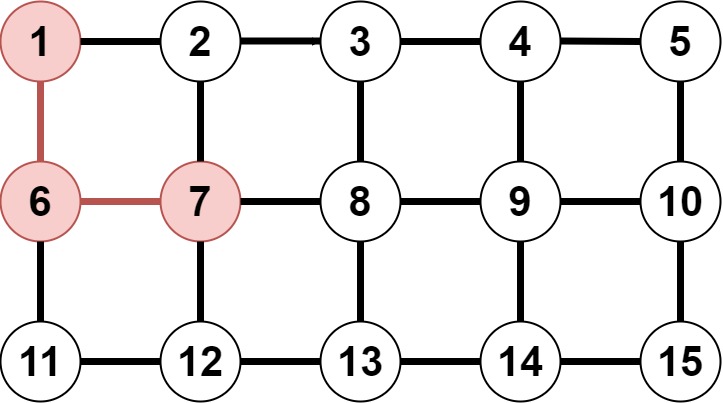
Prim’s Algorithm을 이용해 미로를 나타내는 그래프의 신장 트리를 구하는 과정을 같다. Y에 포함되는 정점과 F에 포함되는 간선을 붉은색으로 나타낸다.

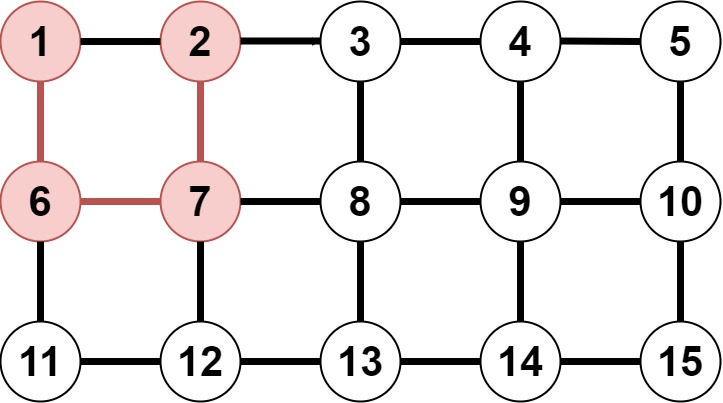


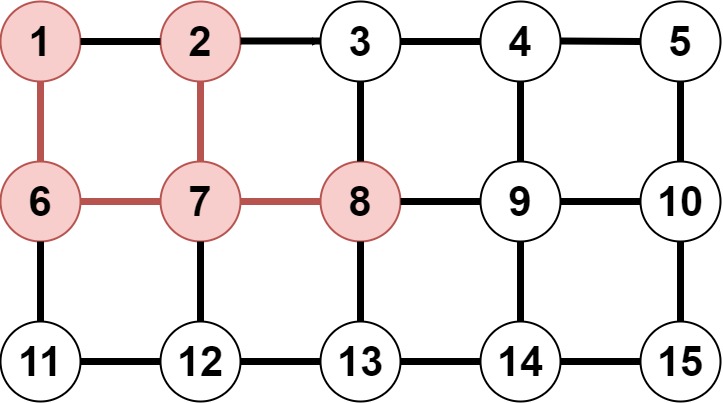
처음에는 정점 1만 Y에 포함되어 있으며 F에는 어떤 간선도 포함되어 있지 않다. V-Y에 속한 2~15 중 1과 인접한 정점은 2, 6이고, 모든 간선의 비용은 1로 동일하므로 둘 중 어떤 정점이든지 선택 가능하다. 6을 선택한다.

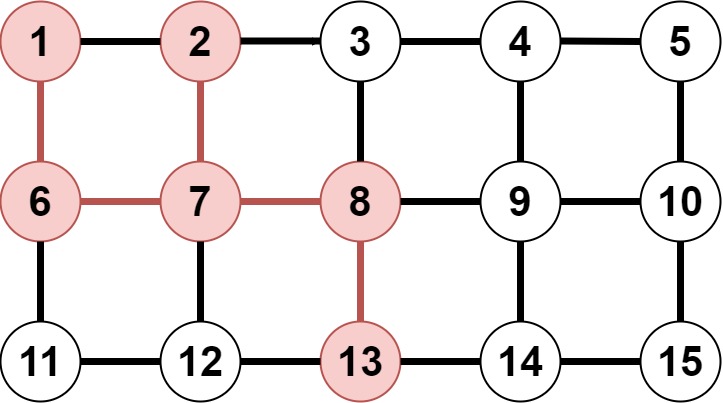


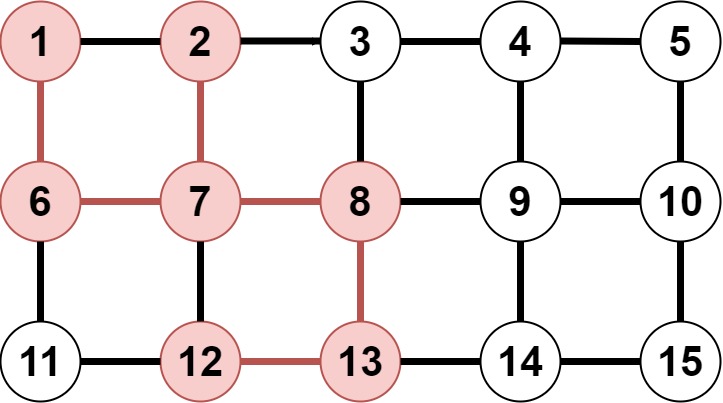
정점 6이 Y에 포함되었고 간선 (1, 6)이 F에 포함되었다. 1에 인접한 정점은 2, 6, 6에 인접한 정점은 1, 7, 11이다. 이 중 1, 6은 Y에 포함되었으므로 제외하면 2, 7, 11 중 하나의 정점을 선택 가능하다. 7을 선택한다.

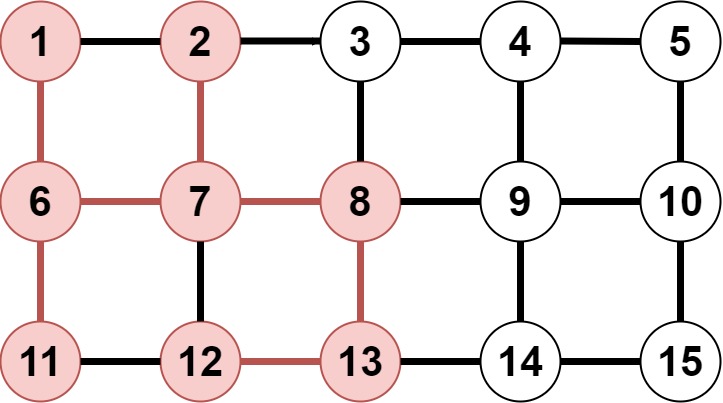


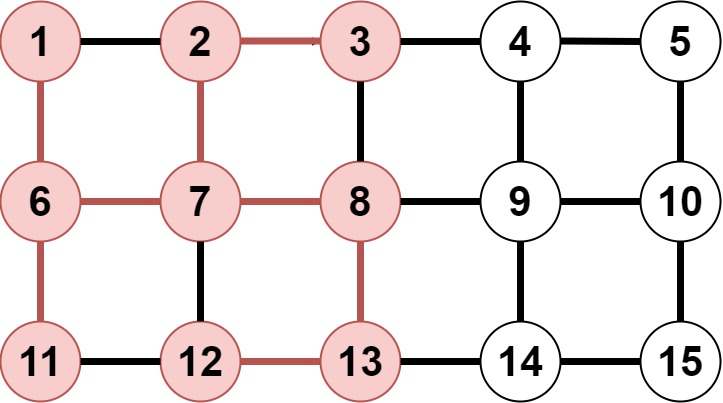
이와 같은 조건을 지켜가며 임의의 정점 선택을 Y와 V가 같아질 때까지 반복한다. 

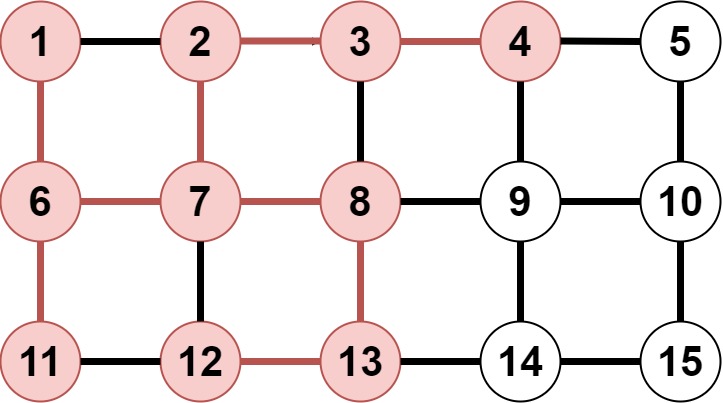


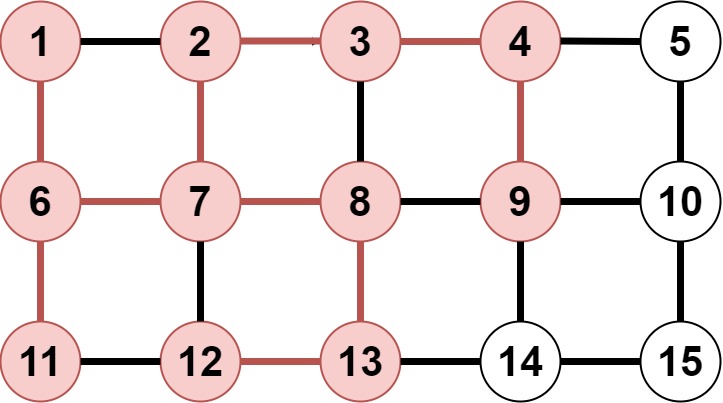


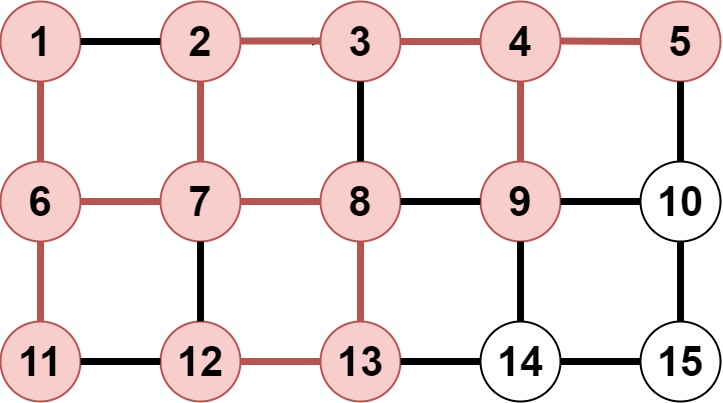


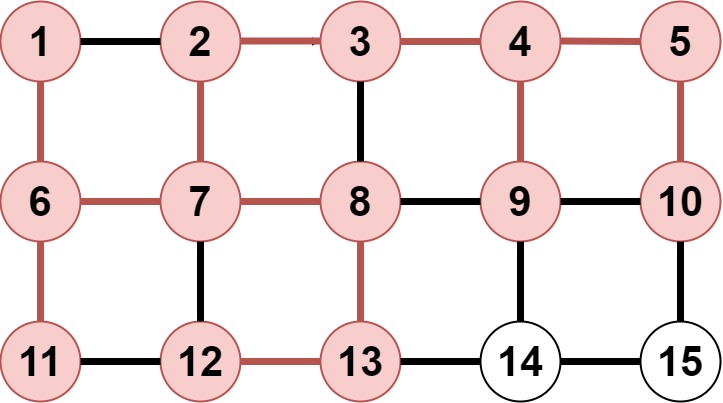


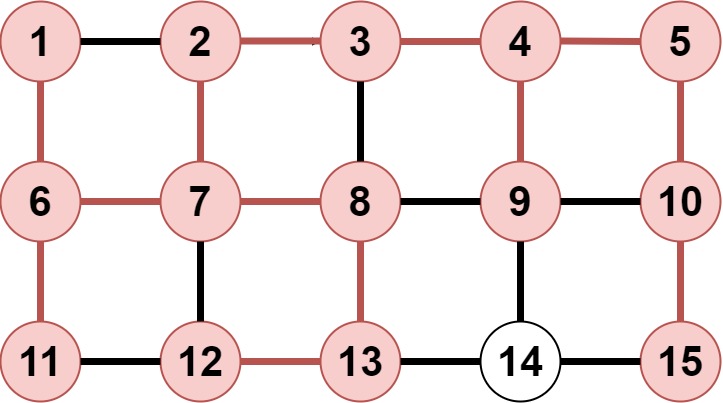


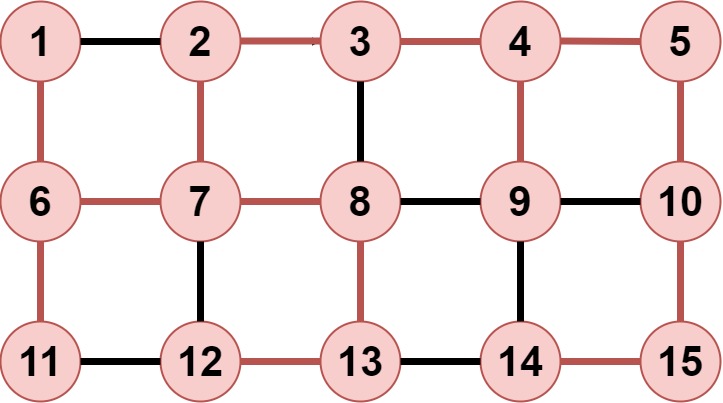




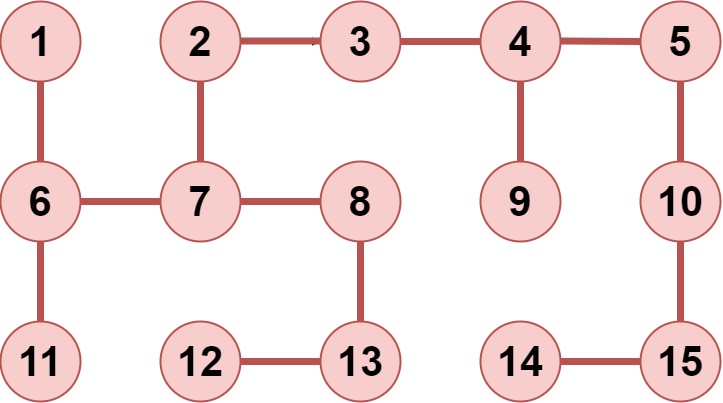






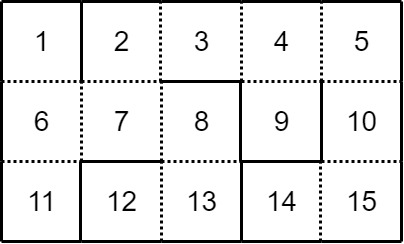


위의 과정을 반복하면 Y가 V가 된다. 정확히 14번 반복 후에 Y가 V가 되어 반복이 종료된다. Y가 V가 되었을 때 F에 포함된 간선들이 최소 비용 신장 트리에 포함되는 간선이다.



완성된 최소 비용 신장 트리(MST)는 위와 같다.

위의 MST로 만들어지는 완전 미로는 다음과 같다.



3. 실습 방법

완전 미로(Perfect Maze)를 만든다고 가정했을 시, Eller’s Algorithm과 2에서 서술한 미로 생성 알고리즘의 자료구조를 제시하고 구현에 필요한 시간 및 공간 복잡도를 비교하여 서술하시오,

Eller’s Algorithm과 Prim’s Algorithm 모두 그래프 자료 구조를 이용한다. 그래프 자료 구조는 정점과 간선으로 이루어져 있다.

정점 자료구조는 다음과 같이 작성 가능하다.

|  |
| --- |
| typedef struct \_Vertex {  int name;  int parent;  } Vertex; |

각 방(정점)을 구분하기 위해 정수로 이름을 부여한다. 위의 미로에서 1~15에 해당한다. 또한 각 방이 속한 집합을 나타내는 parent를 정의한다. 두 정점이 같은 집합에 포함되면 parent도 같은 값을 가진다. Eller’s Algorithm의 경우 초기에는 모든 정점이 서로 분리되어 있으므로 parent를 각 정점의 name과 같게 하고 Prim’s Algorithm의 경우 초기에는 모든 정점이 연결되어 있으므로 parent를 1로 통일한다.

간선 자료구조는 다음과 같이 작성 가능하다.

|  |
| --- |
| typedef struct \_Edge {  int v1, v2;  bool flag;  } Edge; |

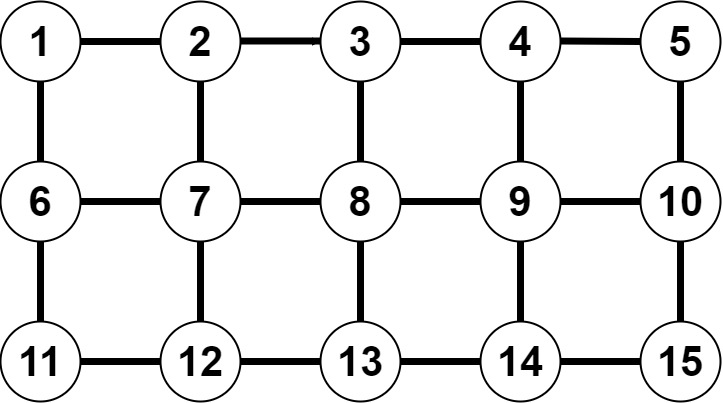
v1, v2는 간선이 연결하는 두 정점의 이름을 나타낸다. flag는 해당 간선이 MST에 포함되는 지를 나타낸다. 포함될 경우 true, 포함되지 않을 경우 false이다.

Eller’s Algorithm과 Prim’s Algorithm 모두 N\*M 미로를 만들기 위해서는 N\*M개의 정점을 생성해야 한다.

Eller’s Algorithm의 경우 간선이 0개인 상태에서 간선을 NM -1 개까지 늘려가며 MST를 완성한다. 따라서 만들어지는 최대 간선은 NM-1개이다. 따라서 NM개의 정점과 NM-1개의 간선을 만들어야 하므로 공간 복잡도는 O(NM)이다.

Eller’s Algorithm의 경우 N개의 행 중 1~N-1행 에서는 각 행의 인접 방끼리 집합 비교 M-1번, 각 방의 아래 행의 방과의 비교 M번으로 총 2M-1번 비교한다. N 번째 행에서는 N행의 인접 방끼리의 집합의 비교 M-1번이 이루어진다. 따라서 총 비교의 횟수는 (N-1)\*(2M-1) + M-1 = 2MN-M-N이다. 따라서 시간 복잡도는 O(MN)이다.

Prim’s Algorithm의 경우 미로의 초기 상태는 다음과 같이 인접한 모든 방은 서로 연결되어 있는 상태이다.



일반화해보면 NM 미로의 초기 상태의 경우 정점은 NM개, 간선은 M\*(N-1) + (M-1)\*N=2MN-M-N개이다. 따라서 공간 복잡도는 O(NM)개이다. Eller’s Algorithm과 비교했을 때 초기에 만들어야 하는 간선의 개수가 더 많지만 시간 복잡도의 관점에서는 같다.

Prim’s Algorithm은 최소 비용 신장 트리에 포함될 NM-1개의 간선을 찾기 위한 반복을 NM-1번 한다. 1<=i<=NM-1일 때, i번째 반복에서 Y에 포함되지 않는 NM-i개의 정점을 모두 확인해서 가장 가까운 정점을 찾아낸다. 따라서 i번째 반복의 경우 NM-i번의 반복을 한다.

따라서 총 반복의 횟수는

(NM – 1) + (NM – 2) + … + (NM – NM + 1) = NM(NM-1)/2이다. 따라서 시간 복잡도는 O((NM)2)이다.

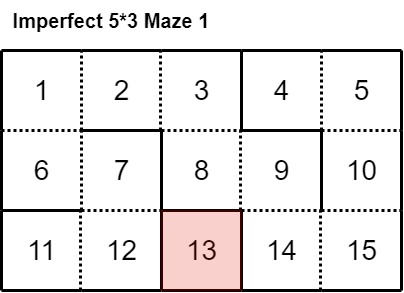
NM은 미로의 방의 개수이다. 두 알고리즘 모두 미로의 방의 개수 X로 시간 복잡도를 나타내보면 다음과 같다.

|  |  |  |
| --- | --- | --- |
|  | **시간 복잡도 Big-OH** | **공간 복잡도 Big-OH** |
| **Eller’s Algorithm** | **X** | **X** |
| **Prim’s Algorithm** | **X2** | **X** |

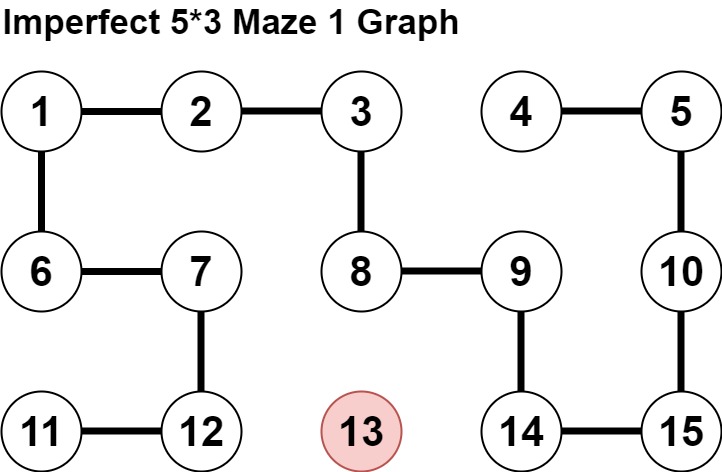
Eller’s Algorithm의 경우, 방의 개수 X에 대하여 미로 생성 시간과 소모 공간이 선형적으로 증가한다. Prim’s Algorithm의 경우, 시간이 제곱으로(quadratic하게) 증가한다.

4. 기타

불완전 미로는 다음과 같은 경우가 있다. 5\*3 미로를 예로 들어보자. 아래 예시에서 방의 번호는 방을 서로 구분하기 위해 붙인 것으로 집합과 무관하다. 연결된 방들은 모두 같은 집합에 속한다.

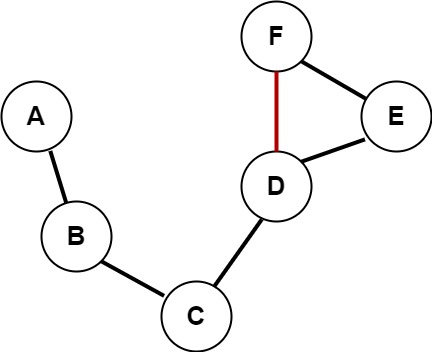


위의 미로는 방 13이 사면이 벽으로 가로막혀 어떤 방에서도 방 13으로 접근할 수 없고, 반대로 방 13에서 어떠한 다른 방으로 접근할 수도 없다. 방 13은 폐쇠된 공간이므로 위의 미로는 불완전 미로이다. 그래프로 표현하면 아래와 같다.

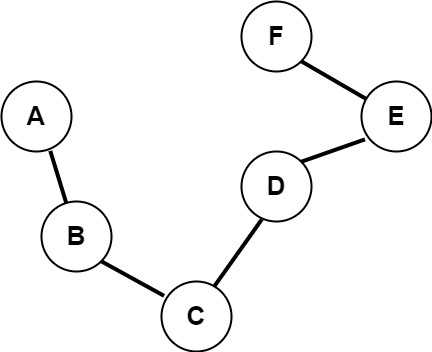


정점 13이 홀로 떨어져서 다른 정점들과 연결되지 않은 것을 알 수 있다. 정점 13으로의 경로가 존재하지 않는 연결되지 않은 그래프(Disconnected Graph)이다.

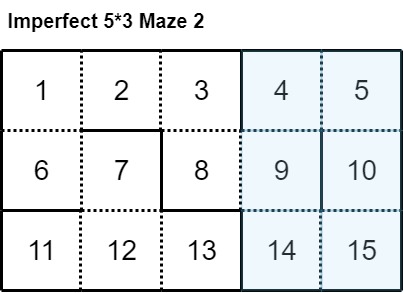
또한 완전 미로에서 임의의 두 방 사이의 경로는 오직 하나여야 한다. 그래프에서 임의의 두 정점 사이의 경로가 오직 하나여야 한다. 임의의 두 정점 사이의 경로에 포함된 정점들 중 시작 정점과, 끝 정점을 제외한 모든 정점이 서로 다를 경우 해당 경로는 단순 경로(simple path)라고 한다.[1] 시작 정점과 끝 정점은 같을 수도 있다. 사이클은 단순 경로 중 시작 정점과 끝 정점이 같은 단순 경로를 의미한다.[1] 만약 그래프는 사이클(cycle)을 가지고 있으면, 이 경우 두 정점 사이의 경로가 유일하지 않은 어떤 두 정점이 있다. 아래 그림에서 D와 F 사이의 경로는 D-F도 있지만 D-E-F도 있다.



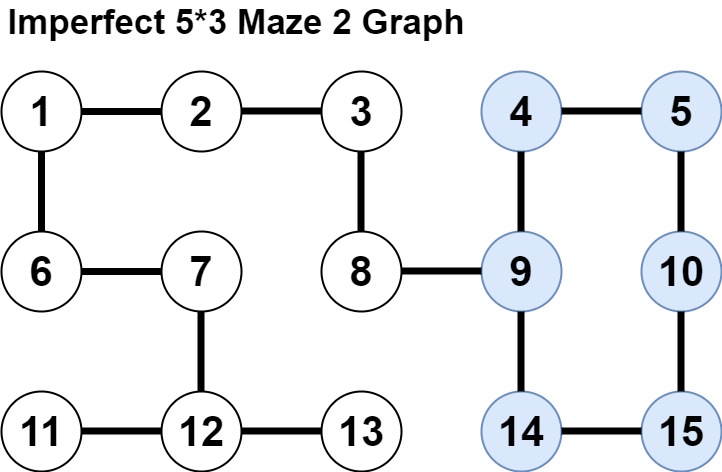
정점 A에서 F에서 가는 경로가 A->B->C->D->E->F일 때 F에서 A로 돌아 가는 경로로 F->E->D->C->B->A 외에도 F->D->C->B->A가 가능하다. A에서 F로 갈 때의 경로에 포함되지 않는 간선 (F, D)가 존재한다. 해당 그래프는 사이클 D-E-F를 가지고 있다. 사이클에 포함되는 간선 (D, E), (E, F), (F, D) 중 어느 하나를 제거해도 그래프는 연결 그래프가 유지되며 사이클이 사라진다.



사이클에 포함된 간선을 하나 제거하면 사이클이 없는(acyclic) 연결 그래프가 된다.

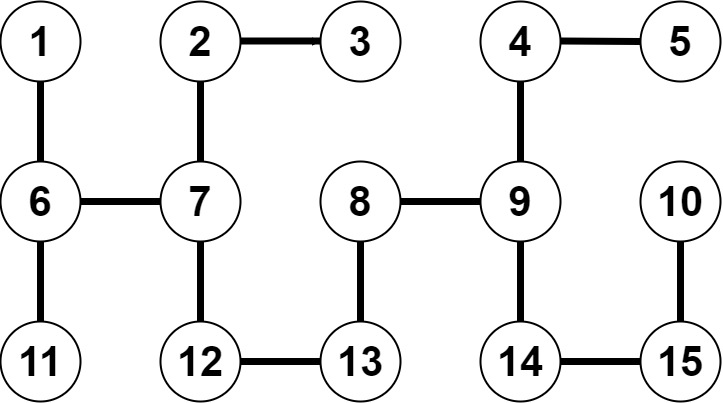
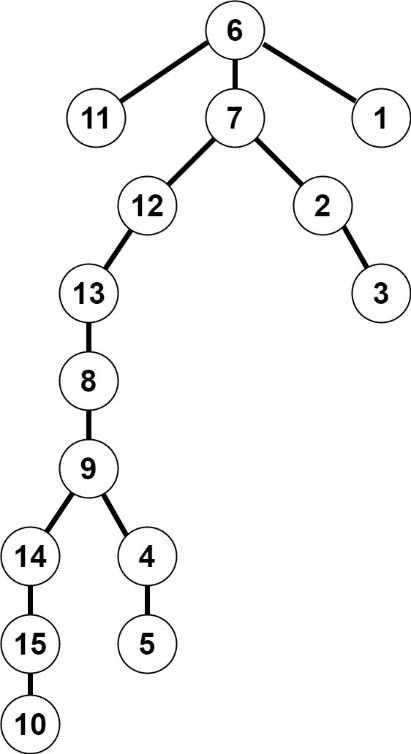


위의 미로는 8번 방에서 10번방으로 가는 경로가 유일하지 않다. 그 외에도 임의의 두 방을 정했을 때 두 방 사이의 경로가 유일하지 않은 경우는 더 존재한다. 따라서 불완전 미로이다.



그래프로 나타내면 사이클이 존재하는 것을 알 수 있다.

완전 미로를 만들기 위해서 그래프는 사이클이 없는 연결된 그래프, 즉 트리가 되어야 한다. 2-1.의 완전 미로에 해당하는 그래프를 6번 노드를 루트 노드로 하는 트리로 변형하면 다음과 같이 3진 트리로 그릴 수 있다.

5. 참고 문헌

[1] [Ellis Horowitz](http://www.yes24.com/SearchCorner/Result?domain=ALL&author_yn=Y&query=Ellis+Horowitz), [Sartaj Sahni](http://www.yes24.com/SearchCorner/Result?domain=ALL&author_yn=Y&query=Sartaj+Sahni), [Susan Anderson-Freed](http://www.yes24.com/SearchCorner/Result?domain=ALL&author_yn=Y&query=Susan+Anderson-Freed), “Fundamentals of Data Structures in C”, [*Silicon Valley Publishers Group*](javascript:void(0);), 2007

[2] Richard Neapolitan, “Foundation of Algorithms, 5/E”, *Jones & Bartlett Publishers*, 2014