

알고리즘 HW #3

BFS 와 DFS 알고리즘 비교 // Kruskal 과 Prim 알고리즘 비교

보고서 작성 서약서

1. 나는 타학생의 보고서를 베끼거나 여러 보고서의 내용을 짜집기하지 않겠습니다.

2. 나는 보고서의 주요 내용을 인터넷사이트 등을 통해 얻지 않겠습니다.

3. 나는 보고서의 내용을 조작하지 않겠습니다.

4. 나는 보고서 작성에 참고한 문헌의 출처를 밝히겠습니다.

5. 나는 나의 보고서를 제출 전에 타학생에게 보여주지 않겠습니다.

나는 보고서 작성시 윤리에 어긋난 행동을 하지 않고 정보통신공학인으로서 나의 명예를 지킬 것을 맹세합니다.

2018년 5 월 27일

학부 정보통신공학과

학년 3 학년

성명 김 권 중

학번 12141668

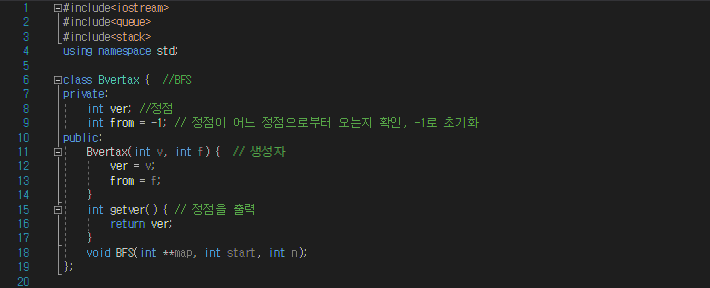


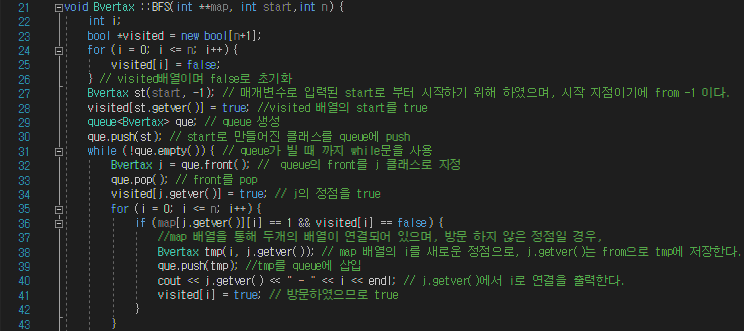
1. 개요

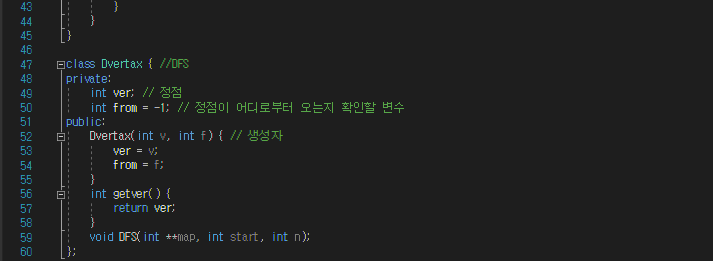
이번 과제는 DFS와 BFS를 이용하여 그래프를 비교하는 문제였습니다. 강의노트에 있는 그래프와 정점이 20개인 그래프를 가지고 비교를 해보았습니다. 정점이 20개인 그래프는 직접 정점과 간선을 설정하고 그려서 비교를 해 보았습니다.

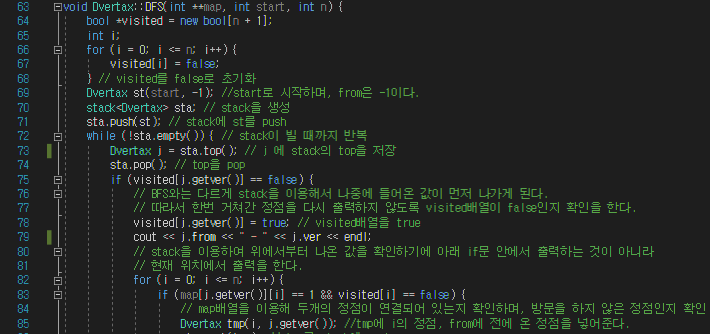
1. 상세 설계내용

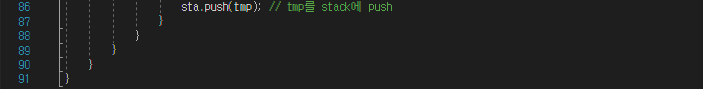
실행 코드.



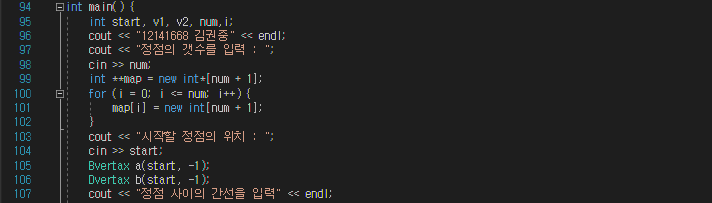


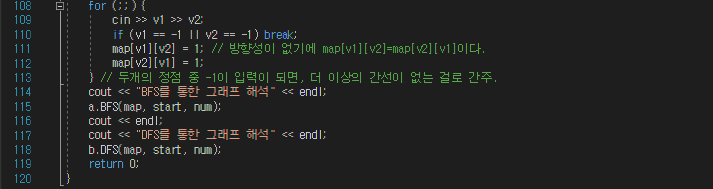






Main.



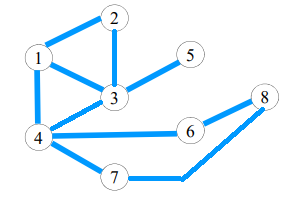


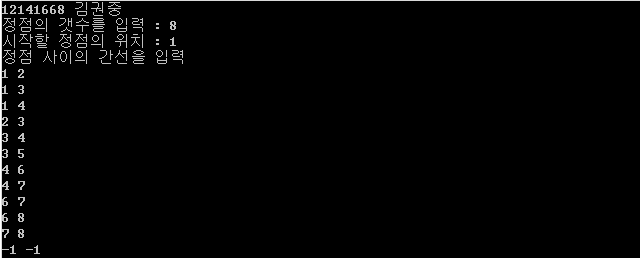
BFS, DFS를 각각 class로 선언하여 만들어주었습니다. Private에 정점을 ver, 어느 정점으로 부터 오는지 알 수 있는 from을 만들어 주었으며, from은 -1로 초기화 해 주었습니다. 두 개의 알고리즘이 거의 비슷하지만 BFS의 경우 Queue를 이용하였으며, DFS의 경우 Stack을 이용하였습니다. BFS의 경우 두 개의 정점이 연결 되어있으며, visited배열의 false인 부분에서 두 개의 정점과 간선을 출력해 주었습니다. 반면에, DFS의 경우 pop을 하고 첫번째 if의 visited 배열에서 방문했는지 확인 하였을 때 출력을 해주었습니다.

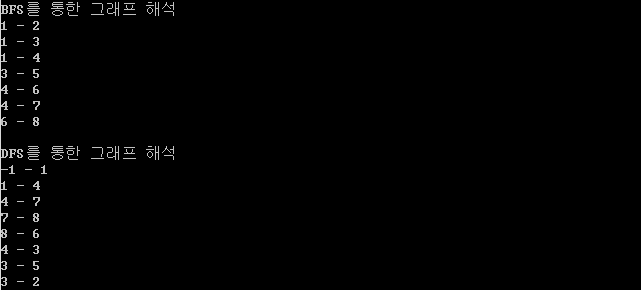
1. 실행 화면

1)21p의 그래프를 이용한 결과 분석

아래의 그래프를 가지고 분석.

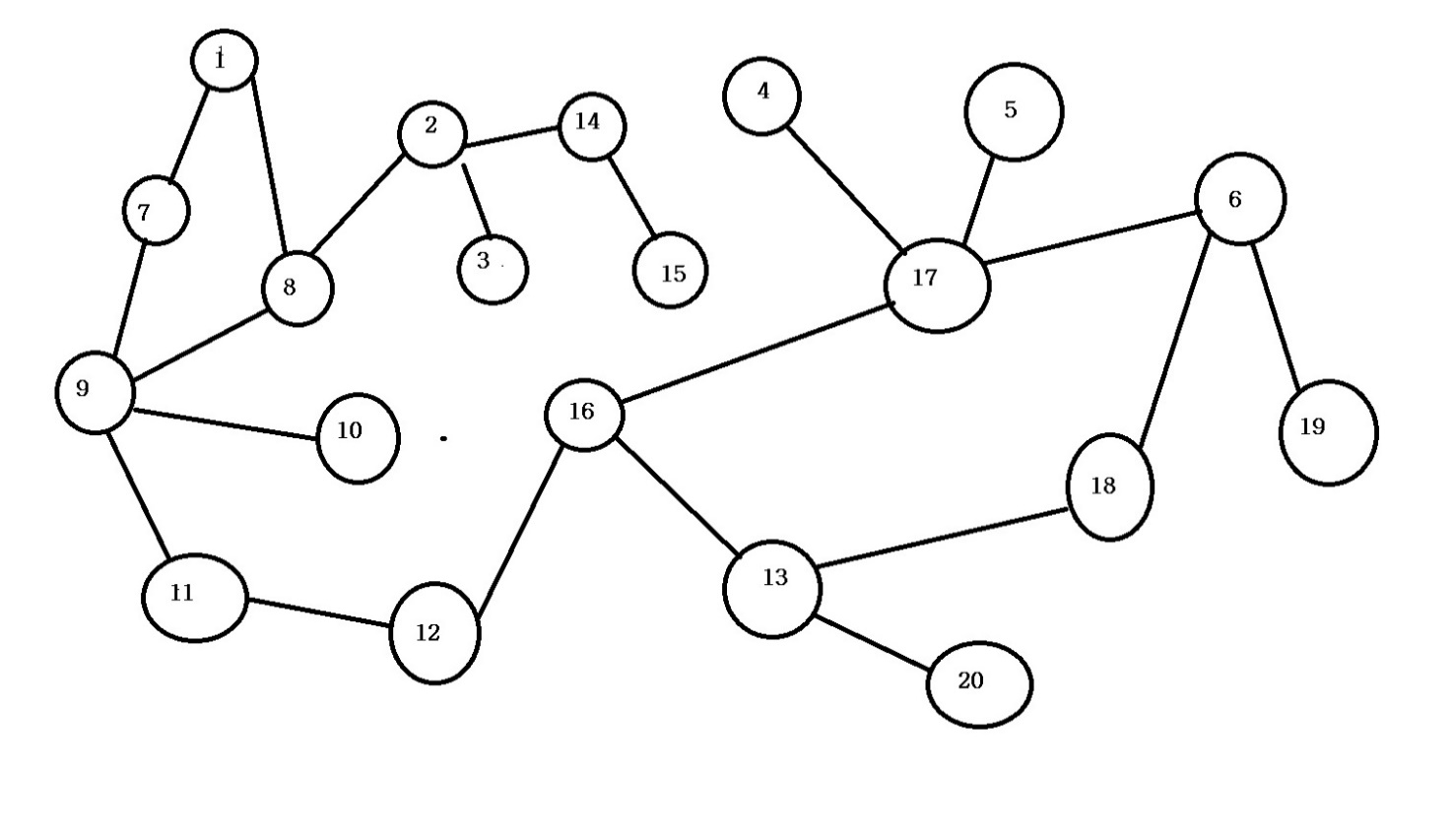




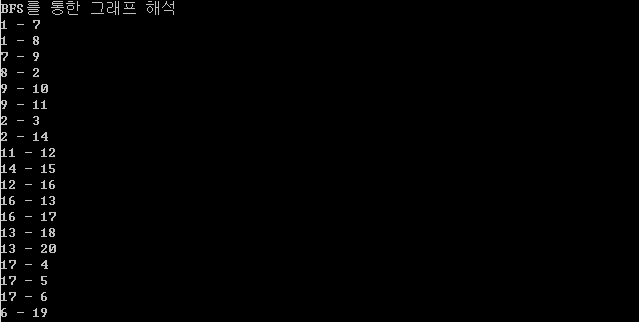


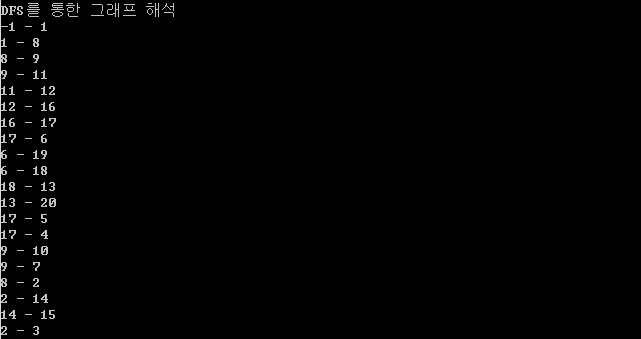
2)20개의 정점을 이용한 실행화면

아래의 그래프를 가지고 분석하였습니다.



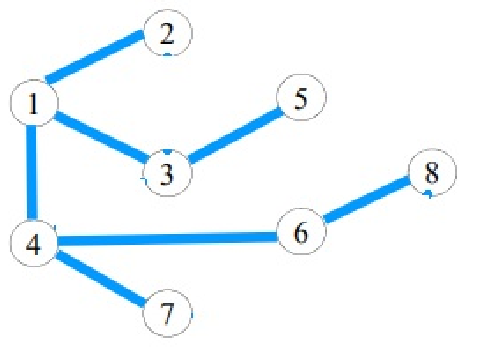






1. 분석 및 결론
2. 20p의 그래프 분석

1)BFS



BFS의 경우 같은 높이의 정점을 먼저 출력해준 것을 알 수 있었습니다.

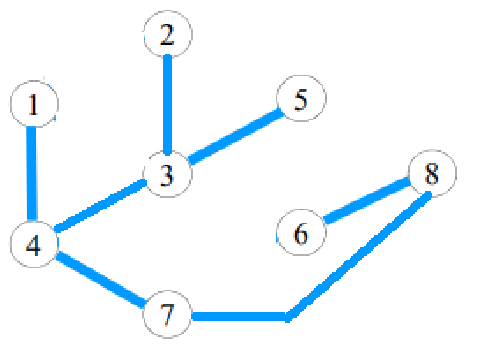
1-2,1-3,1-4

3-5,4-6,4-7

6-8

순으로 출력되는 것을 알 수 있었습니다.

2)DFS



DFS의 경우 가장 깊은 위치까지 내려간 후 다음 한 단계 높은 단계를 출력하는 것을 알 수 있었습니다.

1-4-7-8-6

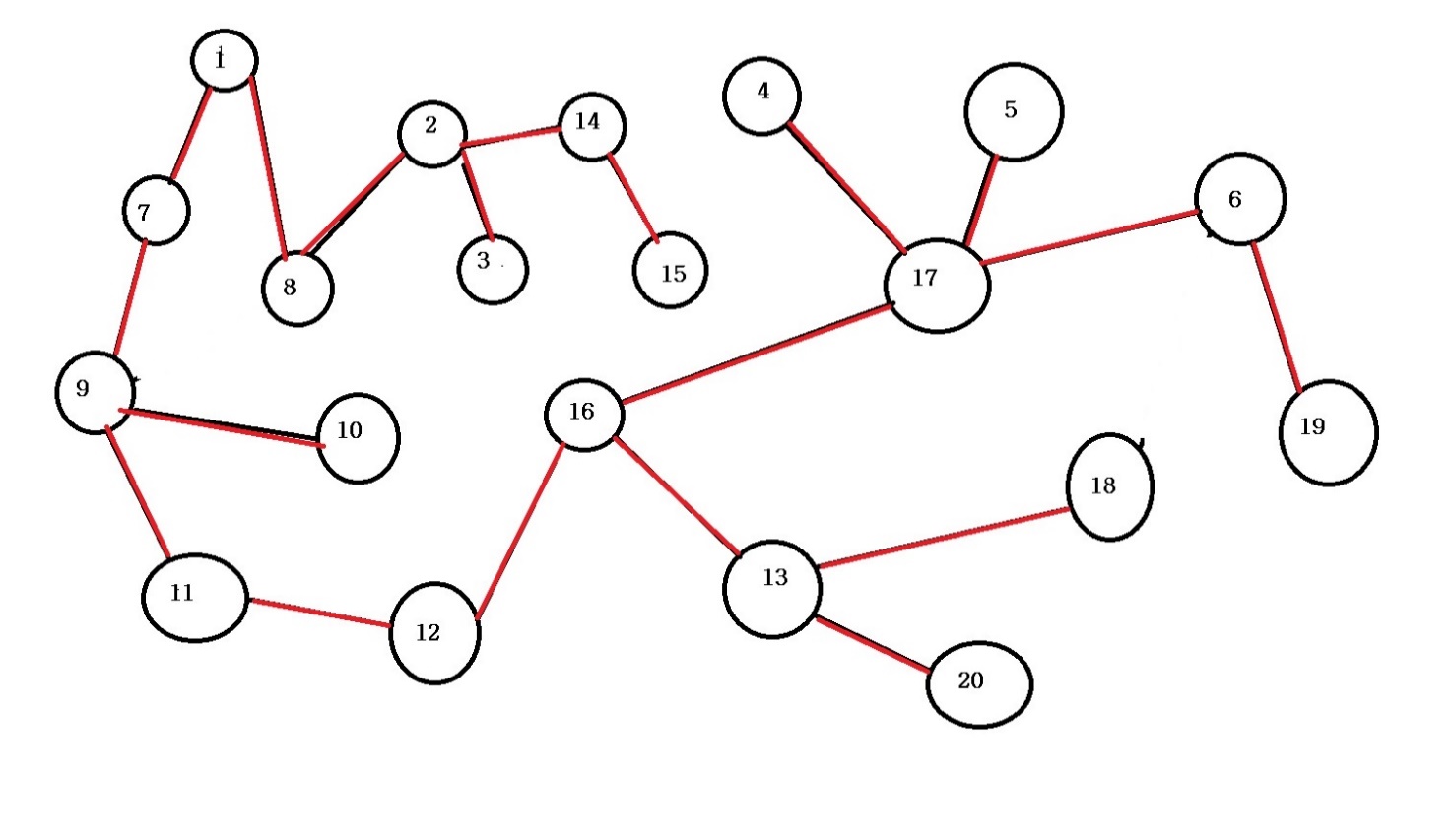
4-3-5

3-2

가 출력 되는 것을 알 수 있었습니다.

1. 20개의 정점을 가진 그래프

BFS



1-7, 1-8

7-9, 8-2

9-10, 9-11, 2-3, 2-14

11-12, 14-15

12-16

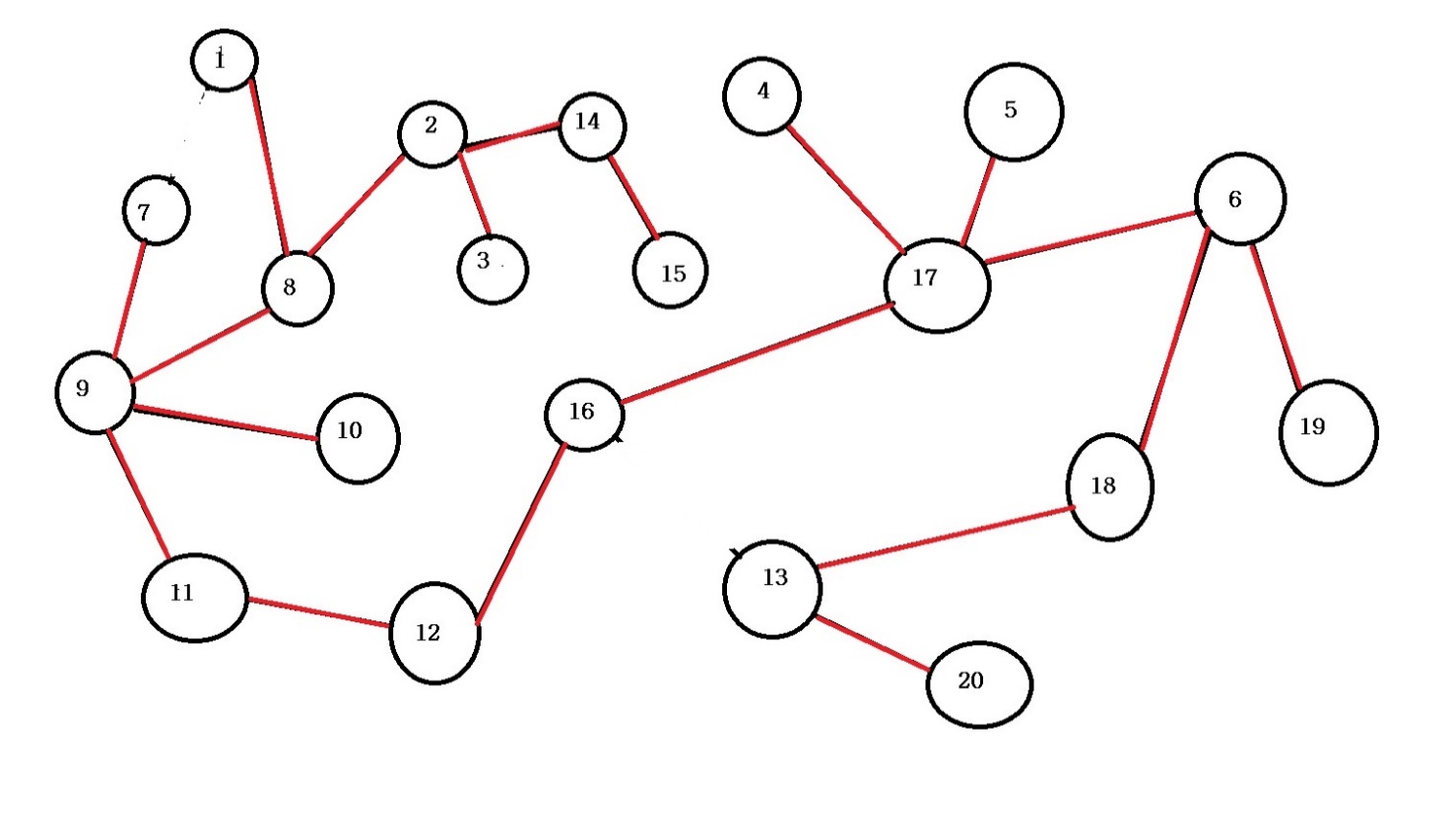
16-13, 16-17

13-18, 13-20, 17-4, 17-5, 17-6

6-19

단계 별로 출력 되는 것을 알 수 있었습니다.

DFS



1-8-9-11-12-16-17-6-19

6-18-13-20

17-5,17-4

9-10, 9-7

8-2-14-15

2-3

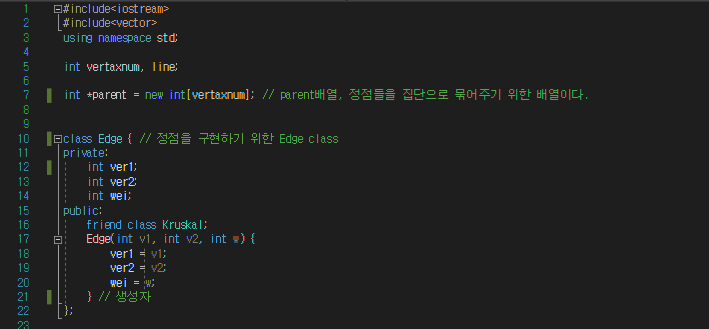
순으로 출력되는 것을 알 수 있었습니다.

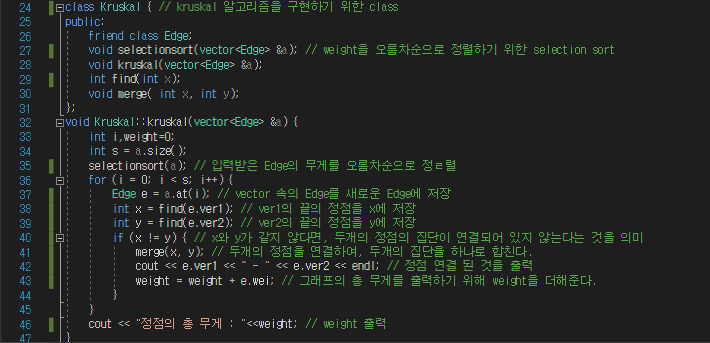
1. BFS의 경우 같은 단계의 있는 정점을 모두 걸친 후, 다음 단계에 있는 정점으로 이동하는 것을 알 수 있었습니다. 이를 위해서 BFS는 Queue를 이용하여 정점을 걸쳐 다음 단계의 정점으로 이동 할 수 있게 해주었습니다.
2. DFS의 경우 가장 깊은 단계까지 내려간 후, 가장 깊은 단계부터 그 전 단계의 높이로 이동하며 정점을 지나가는 것을 확인하였습니다. 이를 위해서 DFS는 Stack을 이용하여 구현을 해 주었습니다.

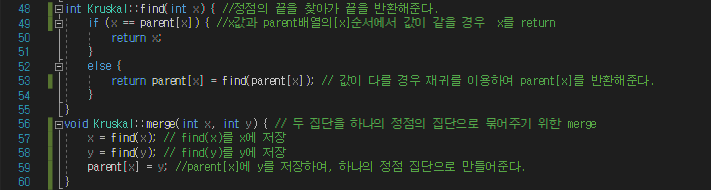
1. 개요

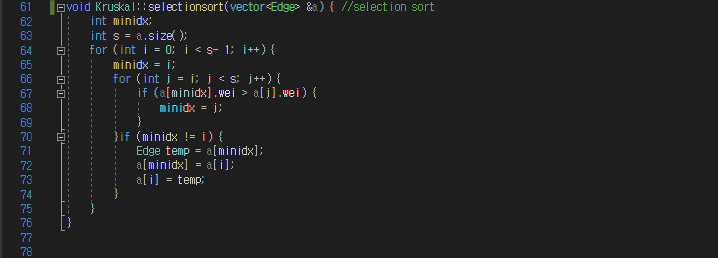
이번에는 Kruskal 알고리즘과 Prim 알고리즘을 구현하고, 강의노트의 그래프와 20개 이상의 노드를 가진 그래프를 가지고 비교하는 내용이었습니다.

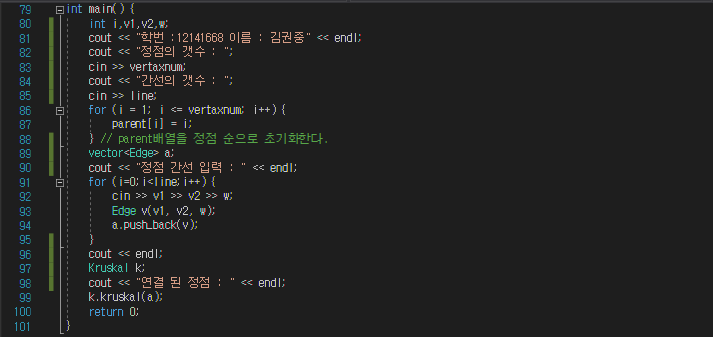
2. 상세 설계내용

Kruskal





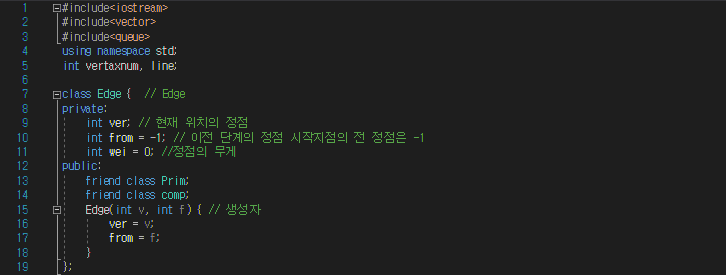


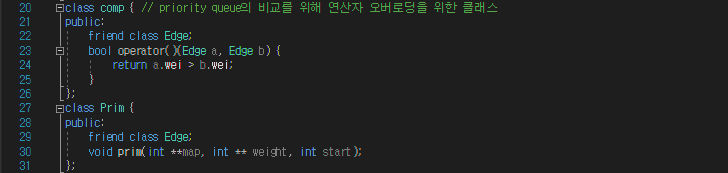


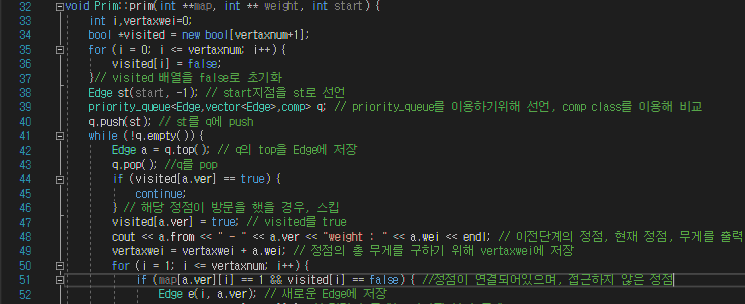
Kruskal 알고리즘 코드입니다.

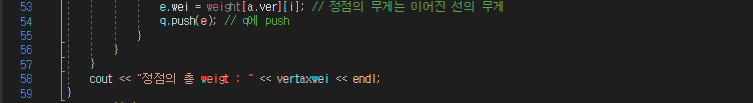
여러 정점을 구현하기 위해 Edge라는 클래스를 만들었습니다. 이 Edge를 저장하기 위해 vector를 사용하였습니다. Kruskal 알고리즘은 간선의 무게를 이용하여 우선순위를 정해주기 때문에, selection sort를 이용했으며, Edge의 wei를 비교하여 Edge를 오름차순으로 정렬해주었습니다. 그래프의 연결 중 싸이클이 형성되면 안되기 때문에, 각 정점의 부모 정점을 찾아주는 함수를 만들었습니다. Find를 통해 정점의 부모를 찾아주었으며, 부모를 저장하기위해 parent라는 배열을 선언해주었습니다. 두 개의 정점을 find를 이용하여 부모 정점을 찾아줍니다. 두개의 부모 정점이 다를 경우, 집단안에 정점이 속해 있지 않은 것이기에 merge를 이용하여 두 개의 집단을 하나의 집단으로 합치는 작업을 반복하면 적은 무게의 그래프가 형성되게 됩니다.

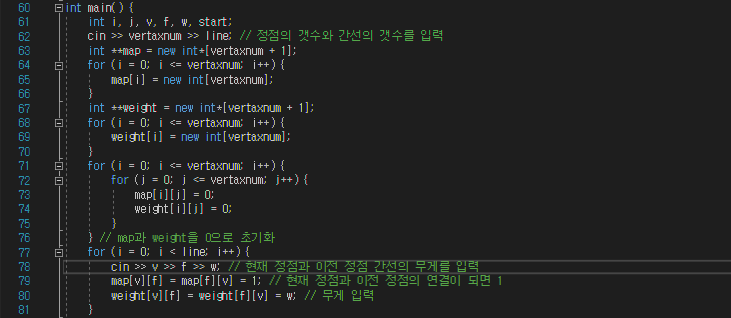
Prim

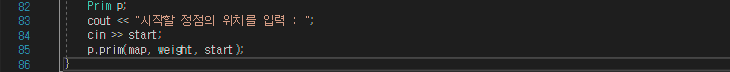












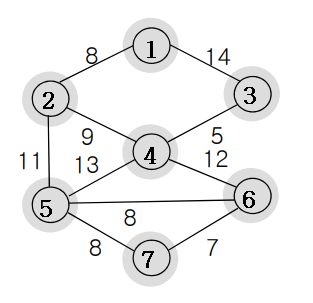
Prim 알고리즘 코드입니다.

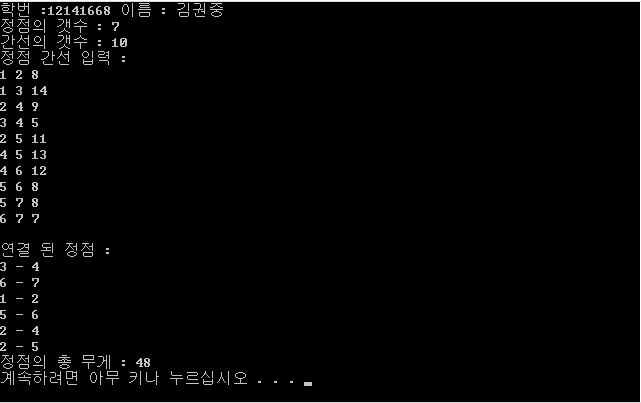
Kruskal 알고리즘과 마찬가지로 여러 정점을 구현하기 위해 Edge라는 class를 구현하였습니다.

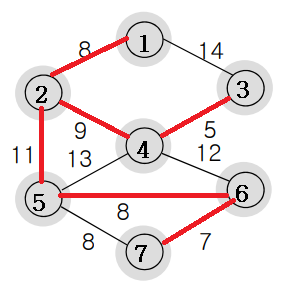
Prim 알고리즘에서는 두개의 정점이 연결 되어있는지 확인 하기 위한 map 이중배열과, 간선의 무게를 저장해줄 weight 이중배열을 선언해주었습니다. 한 정점을 접근했는지 확인 할 수 있는 visited배열을 bool형식으로 선언해주었습니다. Prim 알고리즘에서는 priority queue를 이용하였습니다. Min-heap를 구현하기 위해 comp라는 연산자 오버로딩을 위한 클래스를 따로 만들어 priority queue를 사용하였습니다. Kruskal는 가장 적은 무게의 간선을 우선순위로 하였지만, Prim의 경우 시작지점을 지정 해 주어, st란 첫번째 Edge를 priority queue에 넣어주었습니다. 첫번째 P.Q에 나온 값이 방문해 준 정점일 경우 스킵 해주었습니다. 방문 하지 않은 정점일 경우 visited 배열을 true로 바꾼 후, 간선으로 연결 된 정점들을 P.Q에 push해 주었습니다. 이를 반복하여, 그래프를 만들어 주었으며, 그래프의 총 무게를 출력해주었습니다.

3. 실행 화면

1) Kruskal 강의노트 그래프



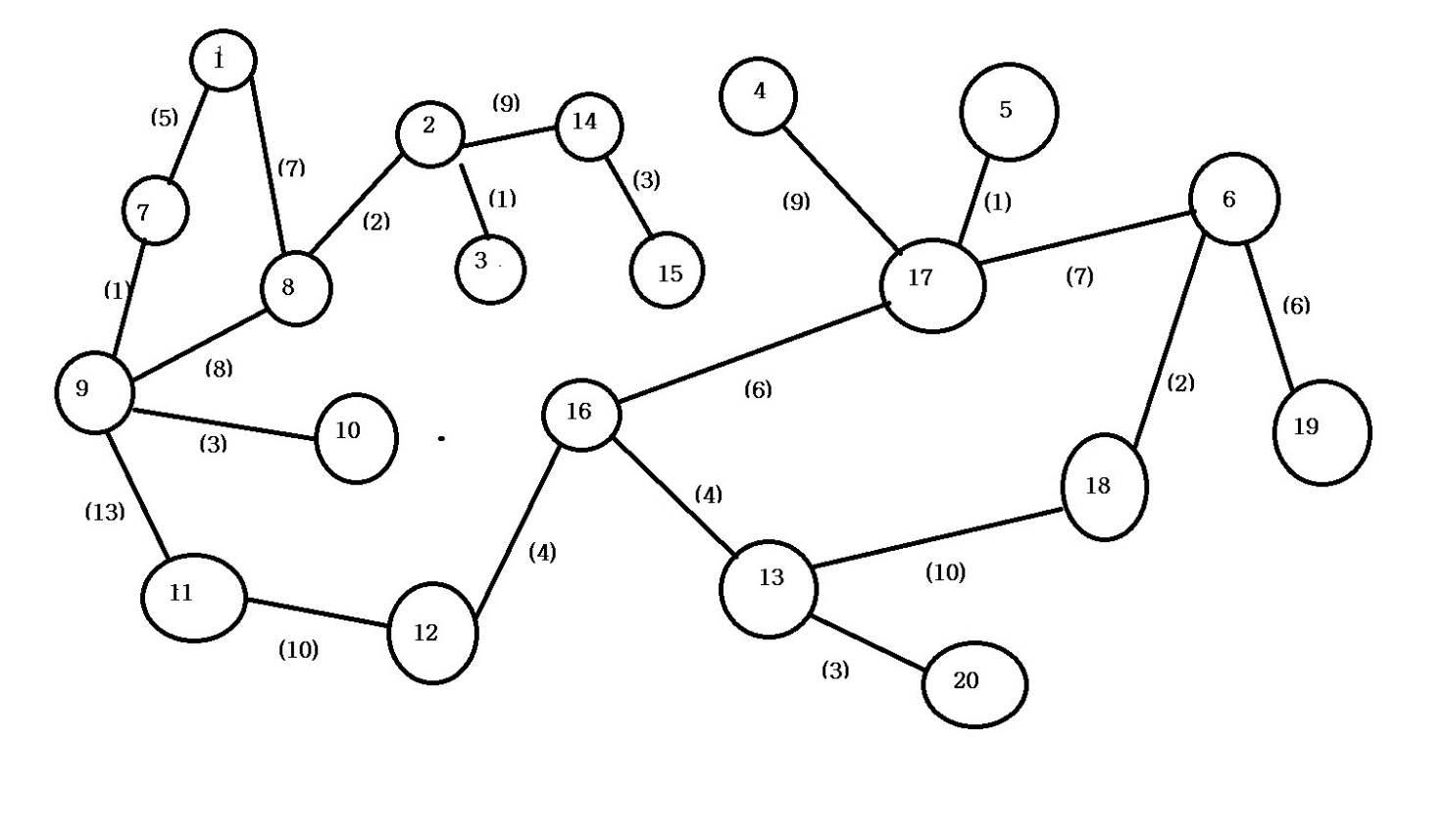


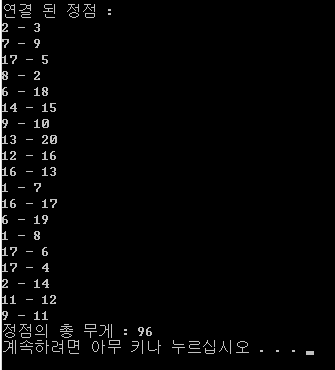
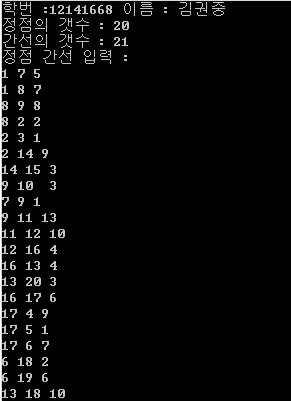


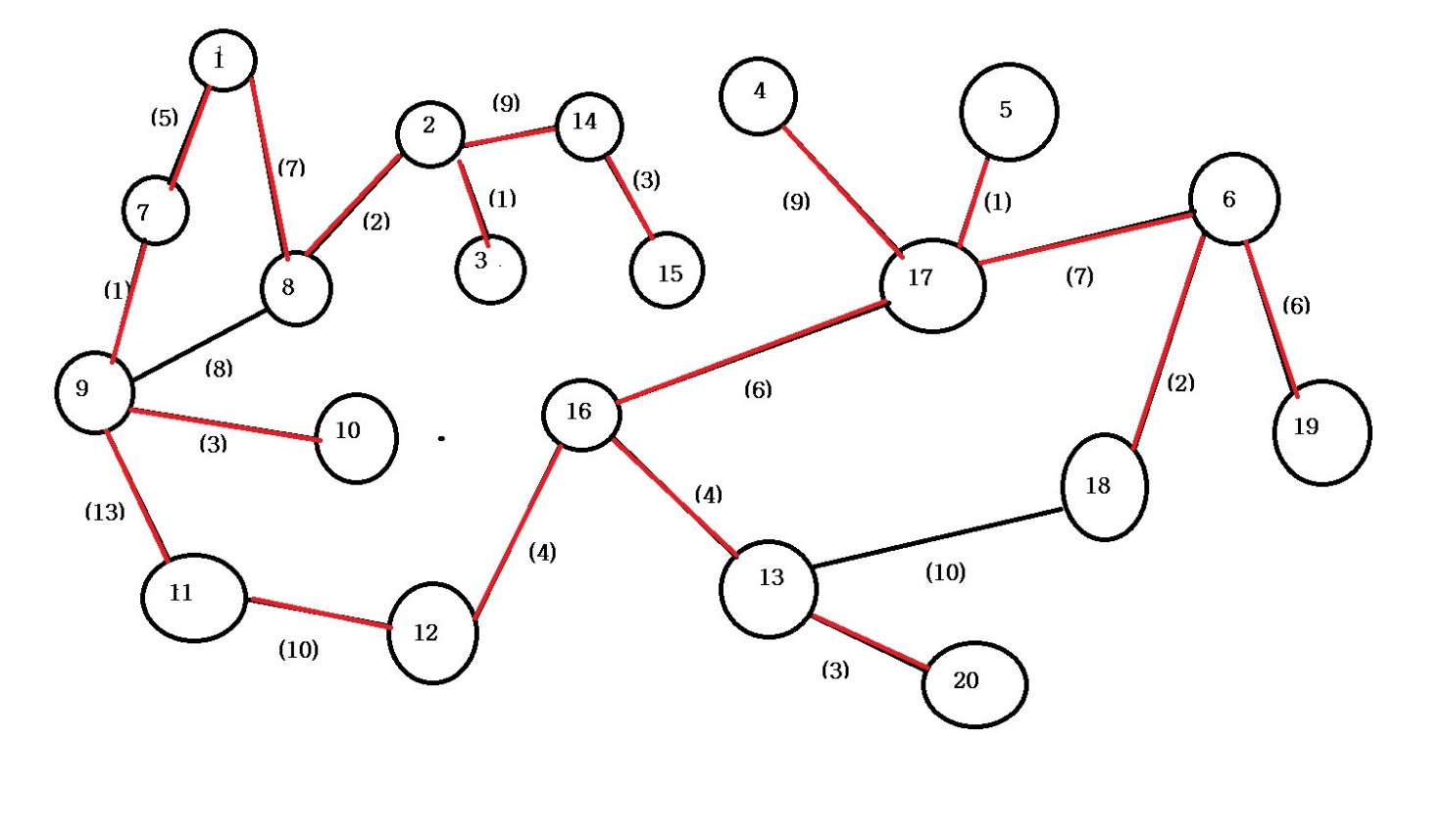
3-4, 6-7, 1-2, 5-6, 2-4, 2-5 순으로 그래프의 간선이 연결 되는 것을 확인하였습니다.

그래프의 싸이클이 형성되지 않았으며, 강의 노트랑 완성된 그래프가 다르 긴 하지만 5-6, 5-7의 간선의 무게가 같기 때문에 그래프가 완성이 된 것을 확인하였습니다.

2) 20개의 정점을 가진 그래프



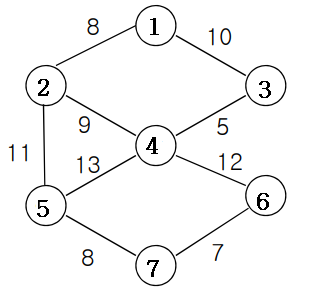


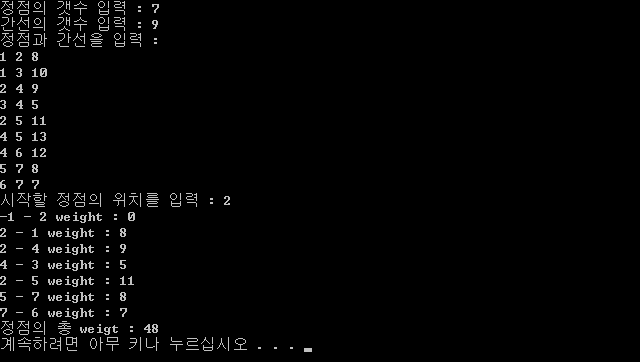


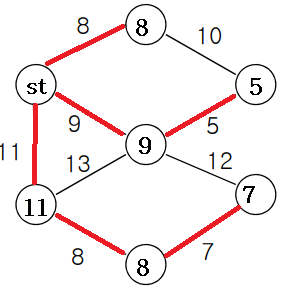
20개의 정점을 가지고 Kruskal 알고리즘을 실행시켰습니다. 실행 화면 창 오른쪽 순서대로 그래프의 간선이 연결되었으며, 실제 그림을 보았을 때, 싸이클이 형성되지 않은 것을 확인하였습니다.

Prim

1) 강의노트 그래프

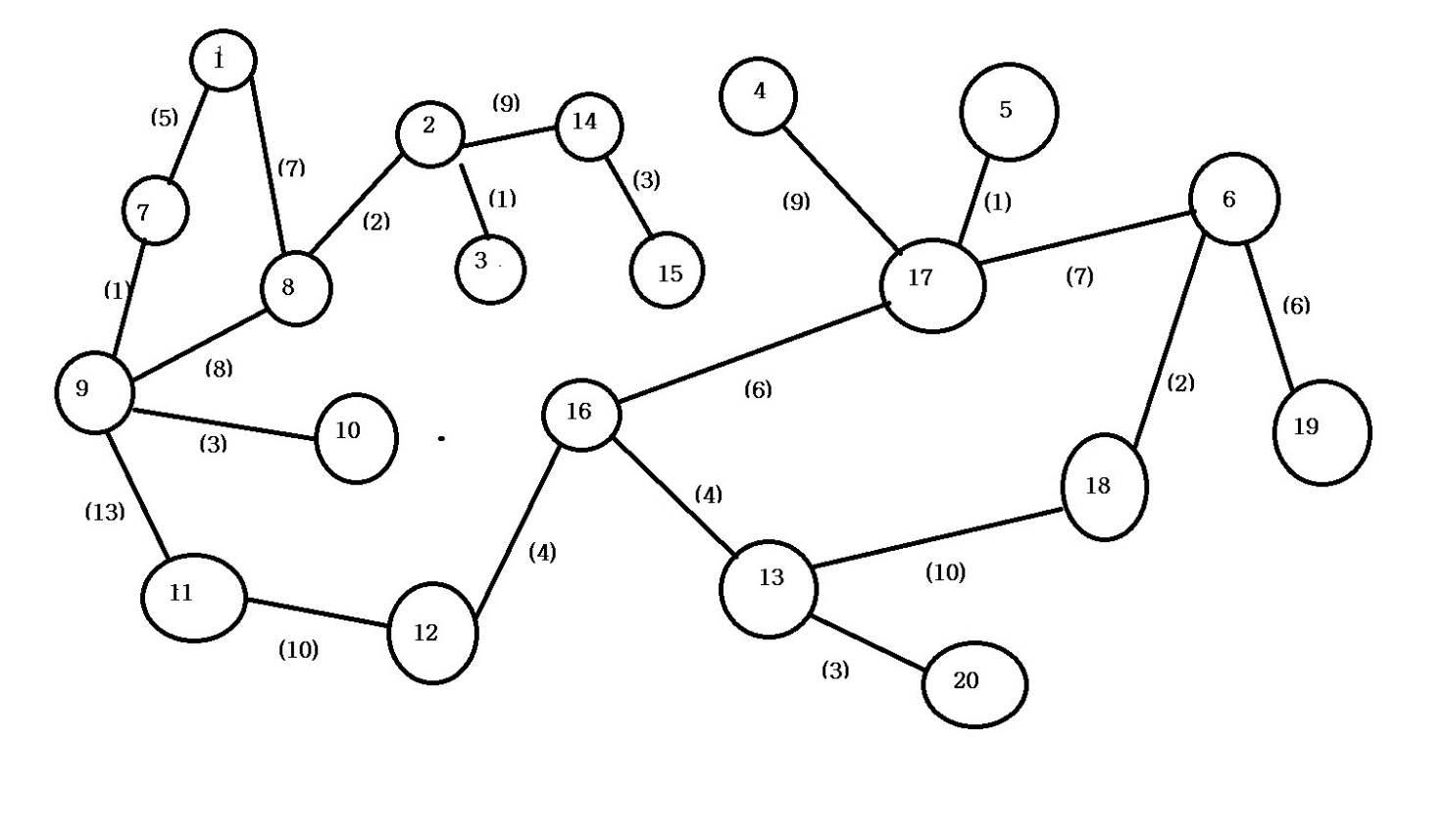




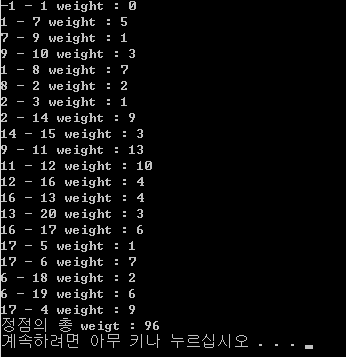
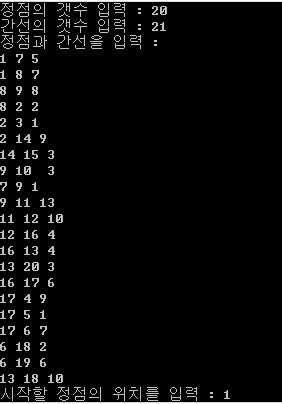


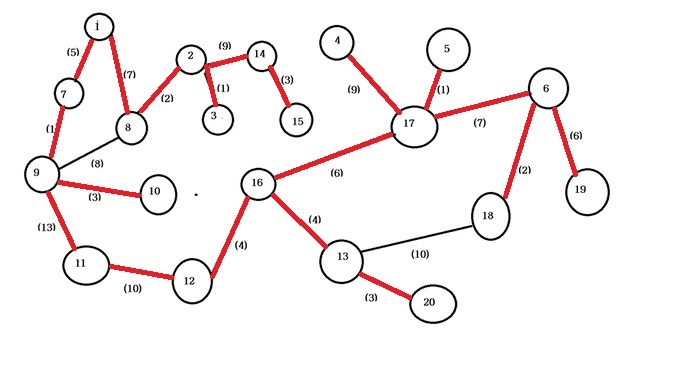
위 그래프처럼 그래프가 연결되었으며, 강의노트와 일치하는 것을 확인하였습니다.

2) 20개의 정점 그래프



시작 지점은 1로 하였습니다.





실행 화면 창에 오른쪽 사진 순으로 간선이 연결되는 것을 확인 하였습니다. 또한 순환 싸이클이 없는 것을 확인하였습니다.

Kruskal 알고리즘의 경우 전체 그래프에서 가장 작은 무게의 간선부터 연결하여 정점의 집단을 만든 후, 하나의 집단으로 만들어 가며 그래프를 완성시켜주는 것을 보았습니다.

Prim 알고리즘의 경우 하나의 시작 지점에서 연결되어 있는 정점을 P.Q를 이용하여 가장 무게가 적은 간선을 찾은 후, 이어주는 식으로 그래프를 완성시켜 주었습니다.