



순천향대학교
SOON CHUN HYANG
UNIVERSITY

자료구조 HW 9

자료구조2 실습

담당교수	홍민 교수님
학과	컴퓨터소프트웨어공학과
학번	20204005
이름	김필중
제출일	2021년 11월 03일

| 목 차 |

1. 그래프 깊이 우선 탐색 프로그램

- 1.1 문제 분석
- 1.2 소스 코드
- 1.3 소스 코드 분석
- 1.4 실행화면 + 그래프 그림
- 1.5 느낀점

2. 그래프 너비 우선 탐색 프로그램

- 2.1 문제 분석
- 2.2 소스 코드
- 2.3 소스 코드 분석
- 2.4 실행화면
- 2.5 느낀점

1.1 문제 분석

■ 그래프 깊이 우선 탐색 프로그램

- 382 페이지에 있는 프로그램 10.3의 깊이 우선 탐색 프로그램을 참고하여 파일에 입력되어있는 정점과 간선의 정보를 이용하여 그래프를 구성하고 이 그래프를 깊이 우선 탐색을 통해 출력하는 코드를 작성하시오.
 - 레포트 제출시 그래프가 그려지는 과정과 함께 깊이를 탐색하는 순서를 그린 그림과 함께 제출
(PPT, 한글, WORD 로 그린 그림 제출)

```
C:\WINDOWS\system32\cmd.exe
- 그래프 깊이 우선 탐색 결과 -
< 0 1 4 3 5 2 7 >
계속하려면 아무 키나 누르십시오 . . .
```

```
data - 메모장
파일(F) 편집(E) 서식(O) 보...
v 0          e 2 0
v 1          e 2 5
v 2          e 2 7
v 3          e 3 0
v 4          e 3 4
v 5          e 4 1
v 7          e 4 3
e 0 1        e 5 1
e 0 2        e 5 2
e 0 3        e 5 7
e 1 0        e 7 2
e 1 4        e 7 5
e 1 5
```

이번 1번 과제는 그래프에 시작점에서 출발해 그래프의 깊이를 탐색하는 과제이다. 시작 정점을 방문했다고 체크하고 인접한 정점 중 방문하지 않은 정점을 선택해 방문한다. 이런 식으로 전체 정점을 방문하는 프로그램이다.

인접행렬을 통해 만든다. 우선 필요한 정의를 해준 후 방문했다는 인증을 남기는 visited 배열을 정적 변수로 만든다.

그래프 구조체를 만들어 정점의 개수를 저장하는 변수, 2차원 배열을 선언한다

그래프 초기화 함수를 만들어 앞에서 정의한 크기만큼 2차원 배열을 0으로 초기화 하는 함수를 만든다

정점 삽입 연산 함수를 만들어 현재 정점의 개수가 앞에서 정의한 크기보다 클 경우 종료하고 아닐 경우 정점의 개수를 1개 추가하는 함수를 만든다

간선 삽입 연산 함수를 만들어서 들어온 행, 열의 값이 앞에서 정의한 크기보다 같거나 클 경우 종료하고 아닐 경우는 2차원 그래프의 행과 열을 1로 바꾸는 함수를 만든다

인접 행렬로 만든 그래프이기 때문에 인접 행렬에 맞는 깊이 우선 탐색 함수를 만든다. 우선 시작 정점을 방문했다는 체크를 한 후 정점을 출력한다. 그리고 정점을 탐색하고 정점의 크기보다 작을 때까지 반복하면서 이동할 정점이 방문하지 않은 상태, 그 정점의 행과 열의 위치한 값이 0 일 경우 재귀함수를 호출한다

메인 함수에서는 파일을 읽은 후 가장 큰 값을 찾은 후 그 값에 + 1한 값만큼 정점을 생성한다. 그리고 나머지 함수들을 알맞게 호출한다.

1.2 소스 코드

```
1 // =====
2 // 제작 기간: 21년 10월 25일 ~ 21년 11월 1일
3 // 제작자 : 20204005 김필중
4 // 프로그램명: 그래프 깊이 우선 탐색 프로그램
5 // =====
6
7 // 필요한 헤더파일을 선언한다
8 #include <stdio.h>
9 #include <stdlib.h>
10
11 // 필요한 정보를 정의한다.
12 #define TRUE 1
13 #define FALSE 0
14 #define MAX_VERTICES 50
15
16 // 오류 방지를 선언한다
17 #pragma warning(disable : 4996)
18
19 int visited[MAX_VERTICES]; // 방문한 곳을 저장하는 정적 변수를 선언한다
20
21 // 그래프 구조체를 생성한다
22 typedef struct GraphType
23 {
24     int n; // 정점의 개수
25     int adj_mat[MAX_VERTICES][MAX_VERTICES]; // MAX_VERTICES X MAX_VERTICES 의 2차원 배열을 생성한다
26 }GraphType;
27
28
29 // 그래프 초기화 함수
30 void init(GraphType *g)
31 {
32     int r, c; // 정수형 변수 r, c를 선언한다
33     g->n = 0; // 그래프의 정점의 개수를 0개로 한다
34     for (r = 0; r < MAX_VERTICES; r++) // 행을 MAX_VERTICES 만큼 반복
35         for (c = 0; c < MAX_VERTICES; c++) // 열을 MAX_VERTICES 만큼 반복
36             g->adj_mat[r][c] = 0; // r, c 번째의 2차원 배열을 0으로 바꾼다
37 }
38
39 // 그래프 정점 삽입 함수
40 void insert_vertex(GraphType *g, int v)
41 {
42     if (((g->n) + 1) > MAX_VERTICES) // 만약 현재 정점 + 1개가 MAX_VERTICES 보다 크다면
43     {
44         fprintf(stderr, "그래프 : 정점의 개수 초과\n"); // 오류 발생 후 종료
45         return;
46     }
47     g->n++; // 아닐경우는 정점 1개 증가
48 }
49
```

```

50 // 그래프 간선 삽입 함수
51 void insert_edge(GraphType *g, int start, int end)
52 {
53     if (start >= g->n || end >= g->n) // 만약 행 혹은 열이 정점의 개수보다 크거나 같으면
54     {
55         fprintf(stderr, "그래프 : 정점 번호 오류\n"); // 오류 발생 후 종료
56         return;
57     }
58     g->adj_mat[start][end] = 1; // 받은 행과 열의 2차원 배열을 1로 바꾼다
59 }
60
61 // 깊이 탐색 함수
62 void dfs_mat(GraphType *g, int v)
63 {
64     int w; // 정수형 변수 w 선언
65     visited[v] = TRUE; // v 정점 방문했음을 표시 TRUE
66     printf("Xd ", v); // v 정점을 출력한다
67     for (w = 0; w < g->n; w++) // 만약 w가 정점의 개수보다 작을경우 반복
68     {
69         if (g->adj_mat[v][w] && !visited[w]) // 만약 w가 아직 정점에 방문하지 않았으면서 v행 w열인 2차원 배열이 0일 경우
70             dfs_mat(g, w); // 순환 호출을 한다
71     }
72
73 int main(void)
74 {
75     FILE *fp; // 파일 포인터 fp 선언
76     GraphType *g; // 그래프 포인터 g 선언
77     int temp1, temp2; // 임시 저장 정수형 변수 선언
78     int maximum = 0; // 최대값 저장 변수 선언
79     char tmp = 'a'; // 임시 문자형 변수 선언
80
81     fp = fopen("data01.txt", "r"); // 파일을 오픈한다
82
83     if (fp == NULL) // 만약 오픈에 실패하면
84     {
85         printf("파일 오픈 실패\n"); // 오류 출력 후 종료
86         return 0;
87     }
88
89     // 파일 끝까지 반복한다
90     while (!feof(fp))
91     {
92         fscanf(fp, "%c", &tmp); // 파일의 맨 앞 문자를 읽어온다
93         if (tmp == 'v') // 맨 앞 문자가 v일 경우
94         {
95             fscanf(fp, "%d", &temp1); // temp1에 숫자를 저장한다
96             if (temp1 > maximum) // 만약 temp1이 maximum 보다 클 경우
97                 maximum = temp1; // maximum에 temp1을 저장
98         }
99         if (tmp == 'e') // 맨 앞 문자가 e 일 경우
100         {
101             fscanf(fp, "%d %d", &temp1, &temp2); // temp1, temp2에 숫자를 저장한다
102         }
103     }

```

```

104 rewind(fp); // 파일 포인터를 처음으로 돌린다
105
106 g = (GraphType *)malloc(sizeof(GraphType)); // 그래프 g를 동적할당을 한다
107 init(g); // 그래프 g를 초기화 한다
108 // 최대값 + 1 만큼 반복하면서 정점을 삽입한다
109 for (int i = 0; i < maximum+1; i++)
110     insert_vertex(g, i);
111
112 // 파일 끝까지 반복한다
113 while (!feof(fp))
114 {
115     fscanf(fp, "%c", &tmp); // 파일의 맨 앞 문자를 읽어온다
116     if (tmp == 'v') // 맨 앞 문자가 v일 경우
117     {
118         fscanf(fp, "%d", &temp1); // temp1 에 숫자를 저장한다
119     }
120     if (tmp == 'e') // 맨 앞 문자가 e 일 경우
121     {
122         fscanf(fp, "%d %d", &temp1, &temp2); // temp1, temp2에 숫자를 저장한다
123         insert_edge(g, temp1, temp2); // 두 숫자를 간선 삽입 함수를 호출한다
124     }
125 }
126
127 printf("-그래프 깊이 우선 탐색 결과 -\n < ");
128 dfs_mat(g, 0); // 0번 부터 시작하는 그래프 깊이 우선 탐색 함수를 호출한다
129 printf("> \n");
130 free(g); // 동적할당을 해제한다
131 fclose(fp); // 파일을 닫는다
132 return 0; // 종료한다
133 }

```

1.3 소스 코드 분석

```
// 필요한 헤더파일을 선언한다
#include <stdio.h>
#include <stdlib.h>

// 필요한 정보를 정의한다.
#define TRUE 1
#define FALSE 0
#define MAX_VERTICES 50

// 오류 방지를 선언한다
#pragma warning(disable : 4996)
```

1. 필요한 헤더파일을 선언한다.
2. 필요한 true false를 정의하고 최대 값을 정의한다.
3. 오류 방지 구문을 선언한다

```
int visited[MAX_VERTICES]; // 방문한 곳을 저장하는 정적 변수를 선언한다
```

4. 방문한 곳을 저장하는 정적 행렬을 만든다.

```
// 그래프 구조체를 생성한다
typedef struct GraphType
{
    int n; // 정점의 개수
    int adj_mat[MAX_VERTICES][MAX_VERTICES]; // MAX_VERTICES X MAX_VERTICES 의 2차원 배열을 생성한다
}GraphType;
```

5. 그래프 구조체를 생성한다
6. 정점의 개수와 2차원 정수형 배열을 정의한 크기만큼 생성한다


```

// 그래프 초기화 함수
void init(GraphType *g)
{
    int r, c; // 정수형 변수 r, c를 선언한다
    g->n = 0; // 그래프의 정점의 개수를 0개로 한다
    for (r = 0; r < MAX_VERTICES; r++) // 행을 MAX_VERTICES 만큼 반복
        for (c = 0; c < MAX_VERTICES; c++) // 열을 MAX_VERTICES 만큼 반복
            g->adj_mat[r][c] = 0; // r, c 번째의 2차원 배열을 0으로 바꾼다
}

```

7. 그래프 초기화 함수를 만든다
8. 정수형 변수를 선언한 후 그래프의 정점의 개수를 0개로 만든다
9. 행을 정의한 크기만큼 반복하면서 열 또한 마찬가지로 반복하면서 2차원 배열을 0으로 모두 초기화 한다.

```

// 그래프 정점 삽입 함수
void insert_vertex(GraphType *g, int v)
{
    if (((g->n) + 1) > MAX_VERTICES) // 만약 현재 정점 + 1개가 MAX_VERTICES 보다 크다면
    {
        fprintf(stderr, "그래프 : 정점의 개수 초과\n"); // 오류 발생 후 종료
        return;
    }
    g->n++; // 아닐경우는 정점 1개 증가
}

```

10. 그래프 정점 삽입 함수를 만든다
11. 현재 정점의 1개 추가한 개수가 정의한 크기보다 크다면 종료한다
12. 아닐 경우는 정점 1개를 증가시킨다

```

// 그래프 간선 삽입 함수
void insert_edge(GraphType *g, int start, int end)
{
    if (start >= g->n || end >= g->n) // 만약 행 혹은 열이 정점의 개수보다 크거나 같으면
    {
        fprintf(stderr, "그래프 : 정점 번호 오류\n"); // 오류 발생 후 종료
        return;
    }
    g->adj_mat[start][end] = 1; // 받은 행과 열의 2차원 배열을 1로 바꾼다
}

```

13. 그래프 간선 삽입 함수를 만든다
14. 행 혹은 열이 정점의 크기보다 크거나 같으면 오류 발생 후 종료한다
15. 아닐 경우는 받은 행과 열의 2차원 배열을 1로 바꾼다

```

// 깊이 탐색 함수
void dfs_mat(GraphType *g, int v)
{
    int w; // 정수형 변수 w 선언
    visited[v] = TRUE; // v 정점 방문했음을 표시 TRUE
    printf("%d ", v); // v 정점을 출력한다
    for (w = 0; w < g->n; w++) // 만약 w가 정점의 개수보다 작을경우 반복
    {
        if (g->adj_mat[v][w] && !visited[w]) // 만약 w가 아직 정점에 방문하지 않았으면서 v행 w열인 2차원 배열이 0일 경우
            dfs_mat(g, w); // 순환 호출을 한다
    }
}

```

16. 우선 정수형 변수 w를 선언하고 v 정점을 방문했음을 체크한다
17. v정점을 출력한다
18. 만약 w의 크기가 정점의 개수보다 작을 때 계속 반복한다.
19. w가 정점에 방문하지 않았으면서 v행 w열인 2차원 배열이 0일 경우 w를 깊이 탐색 함수로 재귀 함수를 호출한다

```

FILE *fp; // 파일 포인터 fp 선언
GraphType *g; // 그래프 포인터 g 선언
int temp1, temp2; // 임시 저장 정수형 변수 선언
int maximum = 0; // 최대값 저장 변수 선언
char tmp = 'a'; // 임시 문자형 변수 선언

fp = fopen("data01.txt", "r"); // 파일을 오픈한다

if (fp == NULL) // 만약 오픈에 실패하면
{
    printf("파일 오픈 실패\n"); // 오류 출력 후 종료
    return 0;
}

```

20. 파일 포인터 fp를 선언한다

21. 그래프 포인터 g를 선언한다

22. 임시 저장 정수형 변수를 선언한다. 최대값 저장 변수 선언을 한다. 임시 문자형 변수 선언을 한다.

23. 파일을 오픈하고 만약 오픈에 실패할 경우는 오류 발생 후 종료한다

```

// 파일 끝까지 반복한다
while (!feof(fp))
{
    fscanf(fp, "%c", &tmp); // 파일의 맨 앞 문자를 읽어온다
    if (tmp == 'v') // 맨 앞 문자가 v일 경우
    {
        fscanf(fp, "%d", &temp1); // temp1에 숫자를 저장한다
        if (temp1 > maximum) // 만약 temp1이 maximum보다 클 경우
            maximum = temp1; // maximum에 temp1을 저장
    }
    if (tmp == 'e') // 맨 앞 문자가 e일 경우
    {
        fscanf(fp, "%d %d", &temp1, &temp2); // temp1, temp2에 숫자를 저장한다
    }
}

rewind(fp); // 파일 포인터를 처음으로 돌린다

```

24. 파일 끝까지 반복하면서 맨 앞의 문자를 읽어 온다

25. 만약 맨 앞 문자가 v일 경우 우선 temp1에 숫자를 저장한다

26. maximum에 값이 temp1보다 작을 경우 maximum에 temp1 값을 저장한다

27. e일 경우는 저장만 한다

28. 파일 포인터를 맨 앞으로 돌린다

```
g = (GraphType *)malloc(sizeof(GraphType)); // 그래프 g를 동적할당을 한다
init(g); // 그래프 g를 초기화 한다
// 최대값 + 1 만큼 반복하면서 정점을 삽입한다
for (int i = 0; i < maximum+1; i++)
    insert_vertex(g, i);
```

29. 그래프 g를 동적할당을 한다

30. 그래프 g를 초기화 한다

31. 그래프 g를 최대값 + 1만큼 반복하면서 정점을 생성한다

```
// 파일 끝까지 반복한다
while (!feof(fp))
{
    fscanf(fp, "%c", &tmp); // 파일의 맨 앞 문자를 읽어온다
    if (tmp == 'v') // 맨 앞 문자가 v일 경우
    {
        fscanf(fp, "%d", &temp1); // temp1 에 숫자를 저장한다
    }
    if (tmp == 'e') // 맨 앞 문자가 e 일 경우
    {
        fscanf(fp, "%d %d", &temp1, &temp2); // temp1, temp2에 숫자를 저장한다
        insert_edge(g, temp1, temp2); // 두 숫자를 간선 삽입 함수를 호출한다
    }
}
```

32. 파일 끝까지 반복하면서 파일 맨 앞을 읽어 온다

33. 맨 앞이 v일 경우 temp1에 저장만 한다

34. 맨 앞이 e일 경우 2개의 숫자를 읽어 온다

35. temp1 temp2 숫자를 간선 삽입 함수를 호출한다

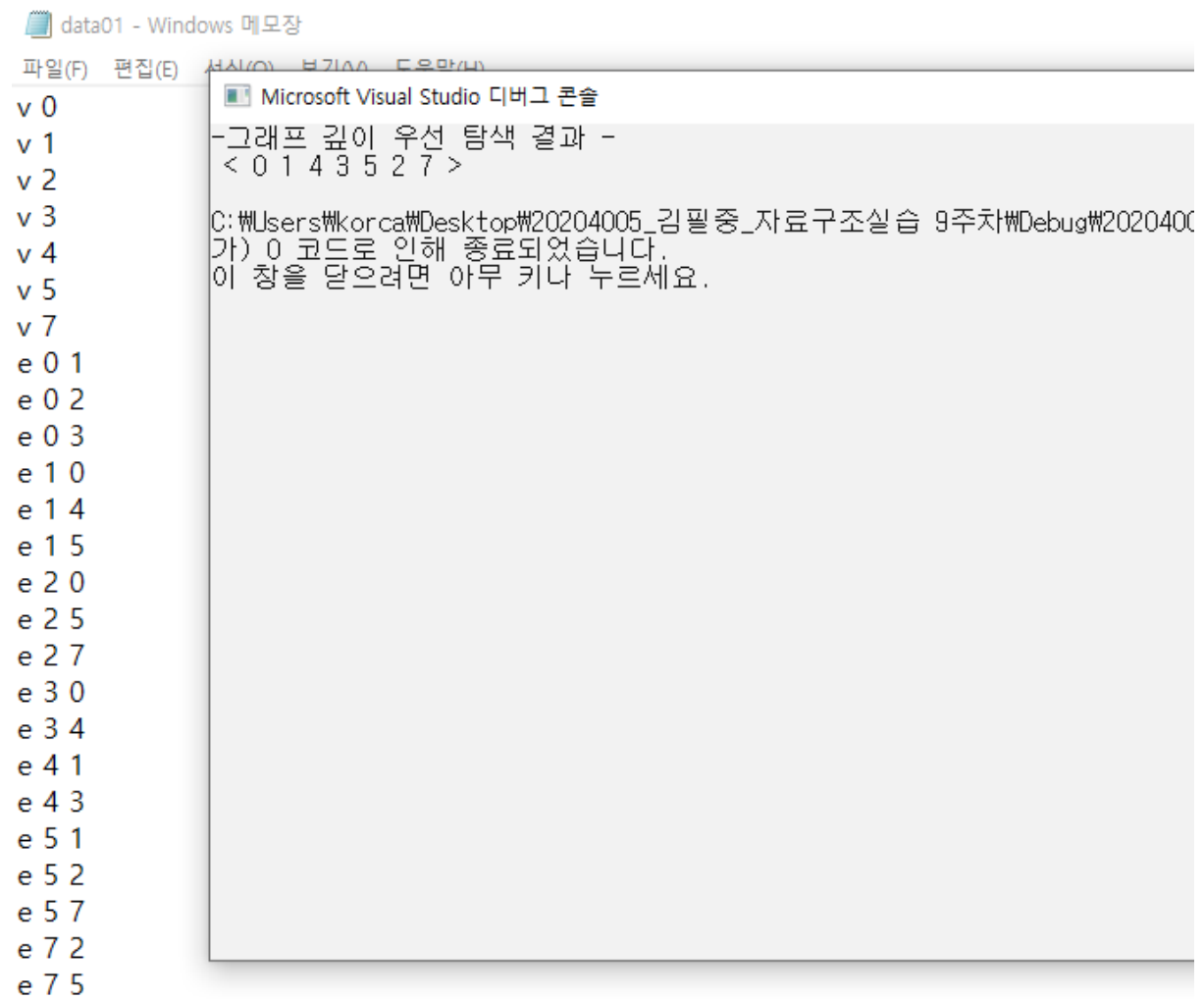
```
printf("-그래프 깊이 우선 탐색 결과 -\n < ");  
dfs_mat(g, 0); // 0번 부터 시작하는 그래프 깊이 우선 탐색 함수를 호출한다  
printf("> \n");  
free(g); // 동적할당을 해제한다  
fclose(fp); // 파일을 닫는다  
return 0; // 종료한다
```

36. 그 후 그래프 깊이 우선 탐색 결과를 출력하고

37. 동적할당을 해제한 후 파일을 닫는다

38. 종료한다.

1.4 실행 화면과 그래프 그림



data01 - Windows 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

Microsoft Visual Studio 디버그 콘솔

-그래프 깊이 우선 탐색 결과 -
< 0 1 4 3 5 2 7 >

C:\Users\korca\Desktop\20204005_김필중_자료구조실습 9주차\Debug\20204005_김필중_자료구조실습 9주차.exe: 0 코드로 인해 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요.

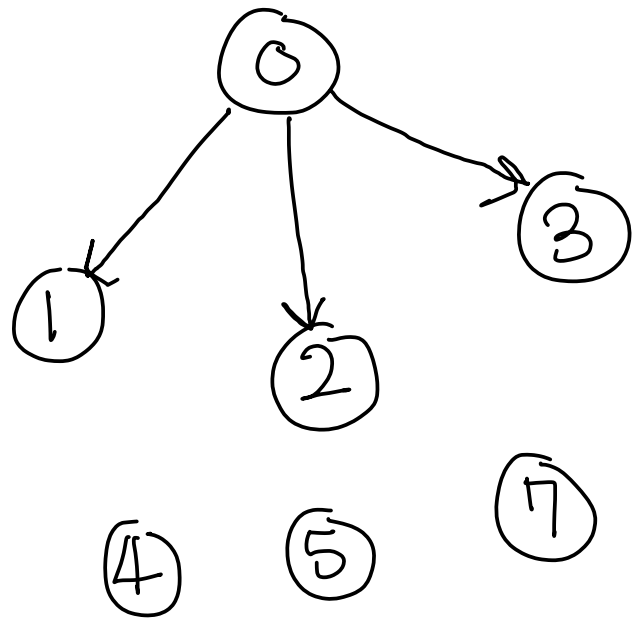
v 0
v 1
v 2
v 3
v 4
v 5
v 7
e 0 1
e 0 2
e 0 3
e 1 0
e 1 4
e 1 5
e 2 0
e 2 5
e 2 7
e 3 0
e 3 4
e 4 1
e 4 3
e 5 1
e 5 2
e 5 7
e 7 2
e 7 5

1.5 느낀점

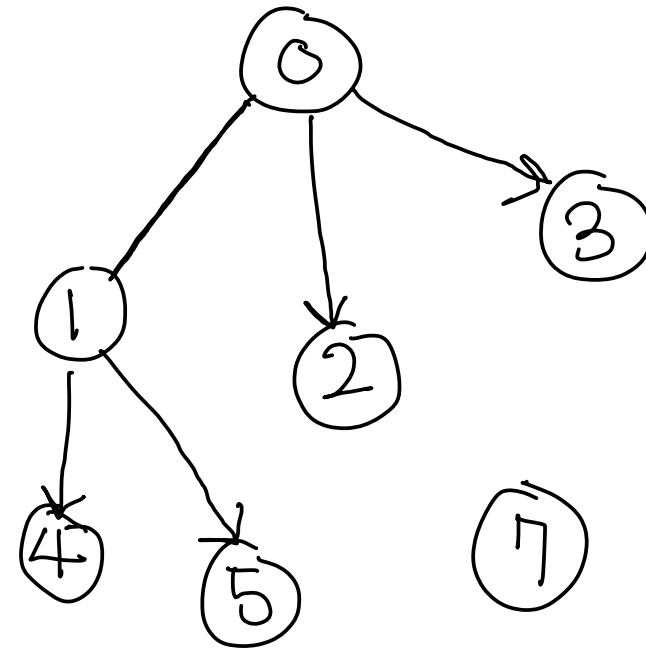
우선 이번 과제를 하면서 저번 과제와 유사하고 거기에 그림으로만 그렸던 깊이 탐색을 프로그램으로 만들어 볼 수 있었다는 점이 좋았다. 직접 프로그램을 만들면서 그래프를 그려보고 그래프와 코드를 비교해가면서 하니까 코드 이해가 더 잘되었다. 나중에는 리스트를 통해 만들 수 있기 때문에 코드를 조금 수정하면 리스트로 만들 수 있겠다는 생각을 했다. 이 프로그램을 만든 후 추후 현실에 필요한 곳에 응용도 가능하겠다는 생각을 했다.

현재 OBS 0 1 2 3 4 5 7

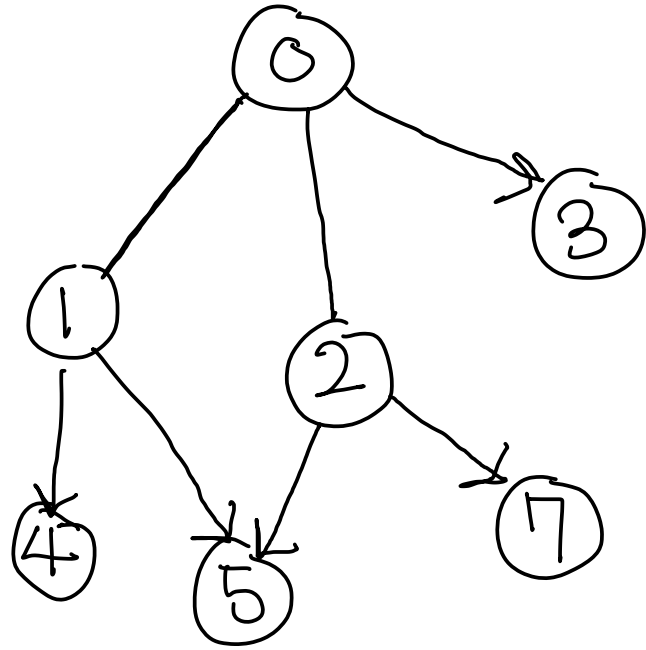
우선 가선 (0,1) (0,2) (0,3) 이 삽입 된다



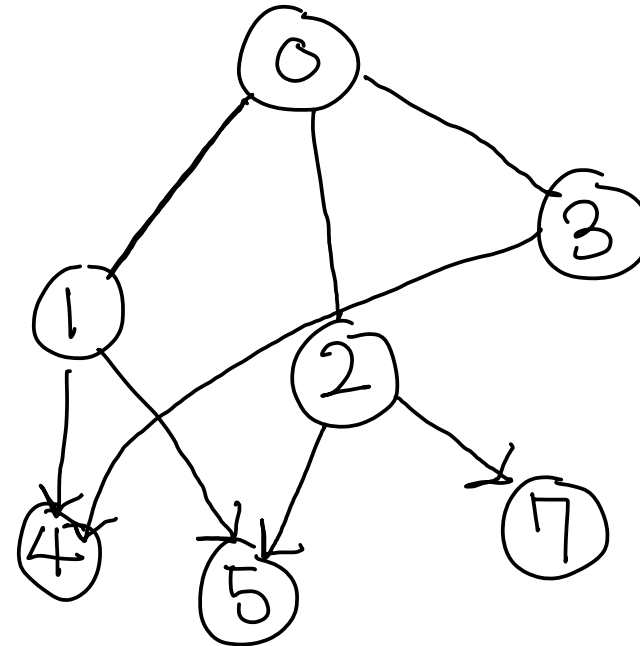
다음으로는 (1,0) (1,4) (1,5) 가 삽입된다



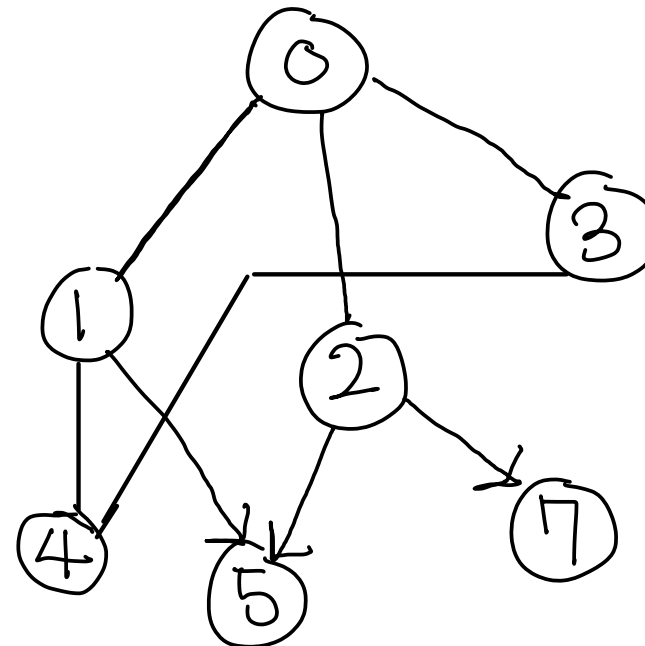
다음으로는 간선 $(2,0)$ $(2,5)$ $(2,7)$ 이
삽입된다



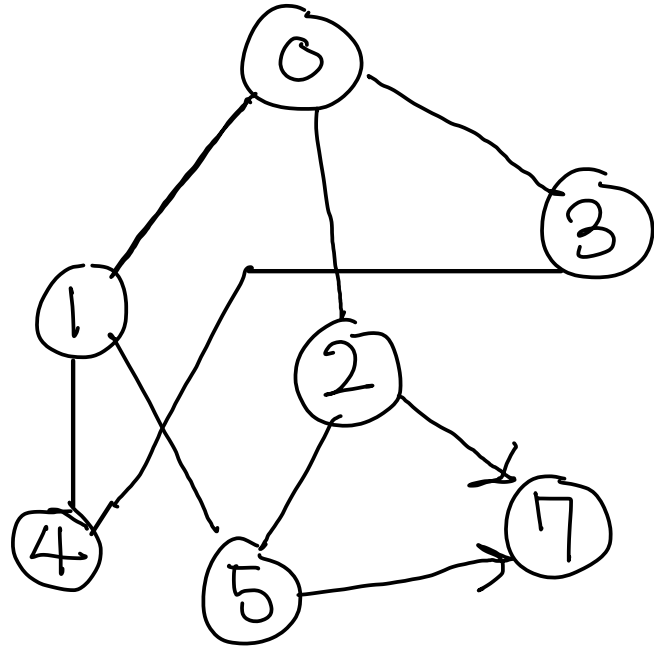
다음으로는 간선 $(3,0)$ $(3,4)$ 가 삽입된다



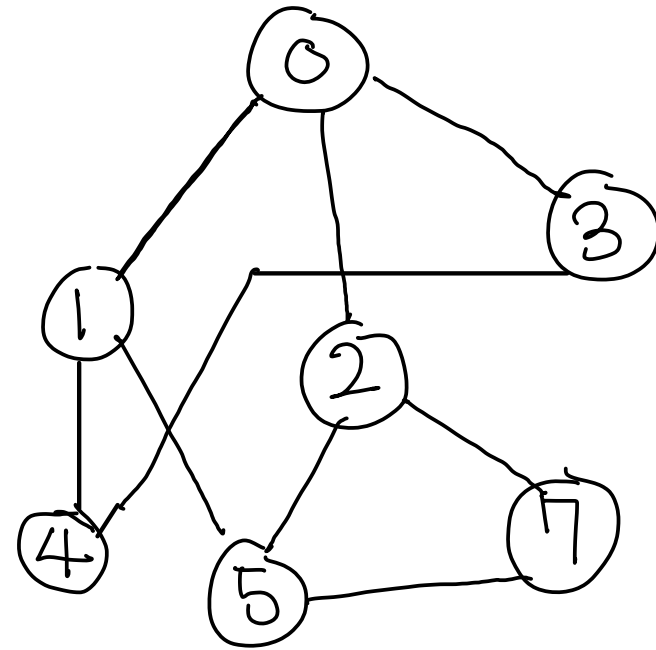
다음으로는 $(4,1)$ $(4,3)$ 이 삽입된다



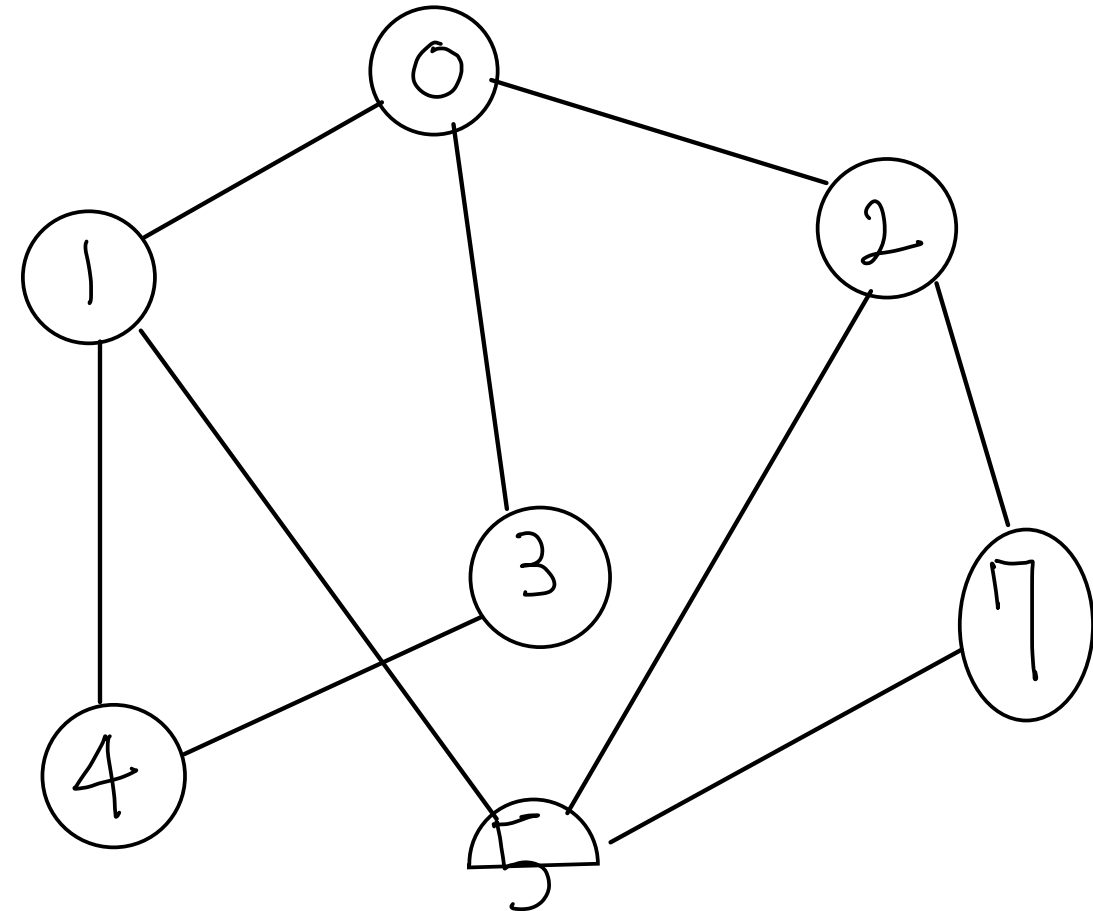
다음으로 (5,2) (5,1) (5,7)이
삽입된다

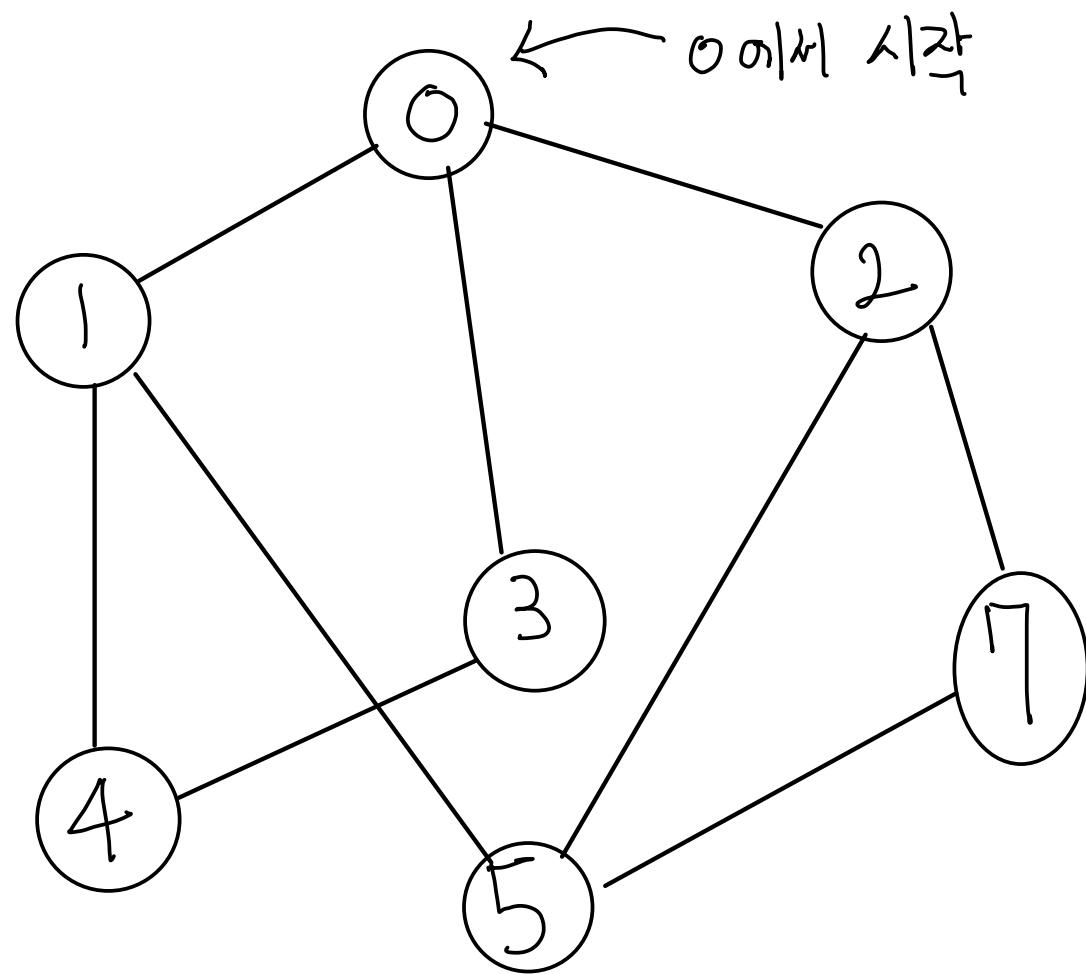


마지막으로 (7,2) (7,5)가 삽입된다



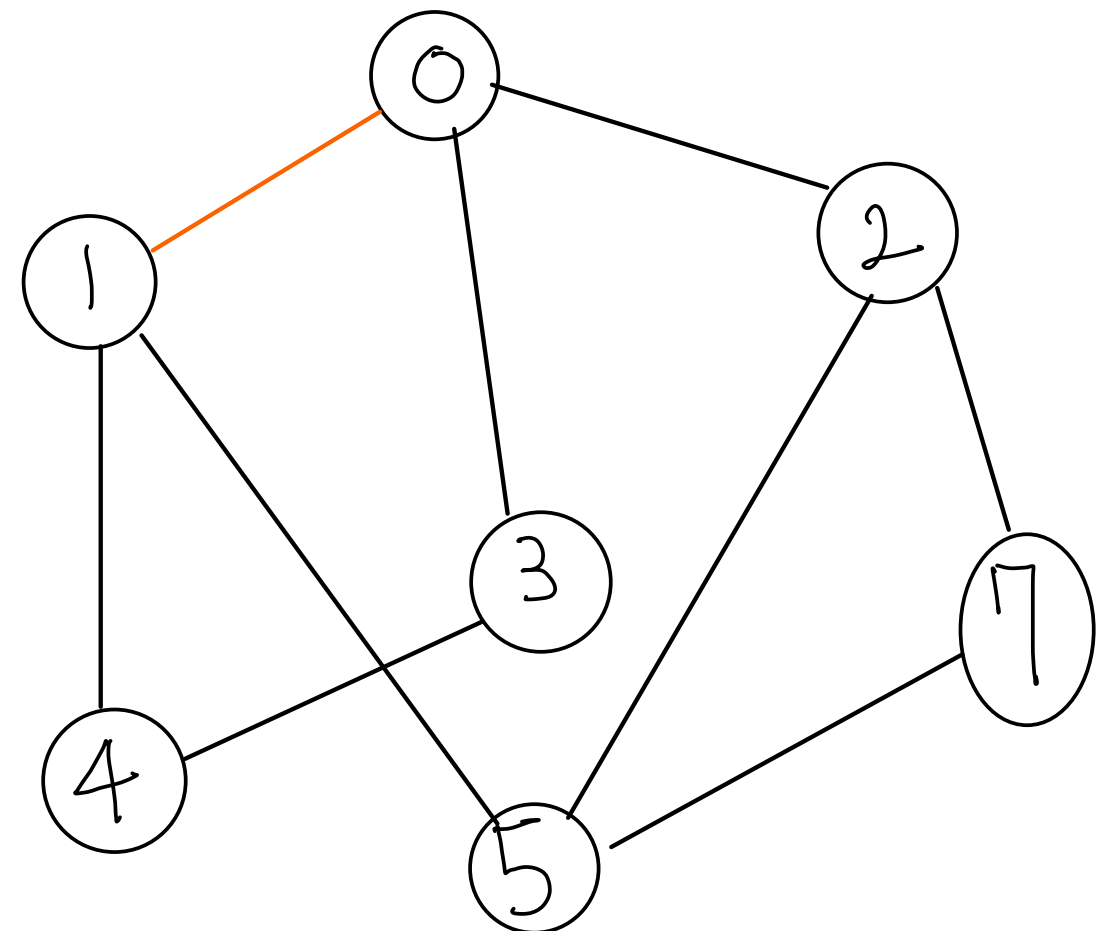
최종 그래프





0이는 1과 2와 연결

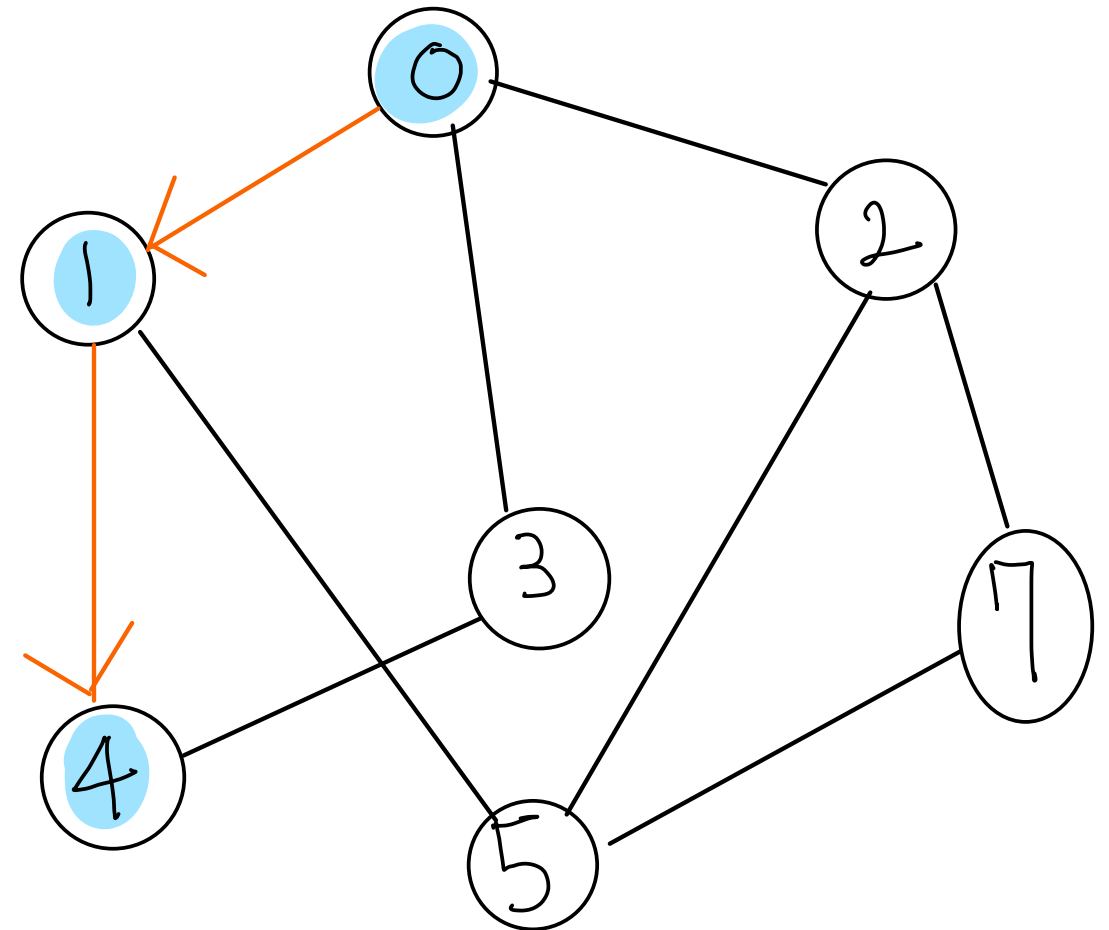
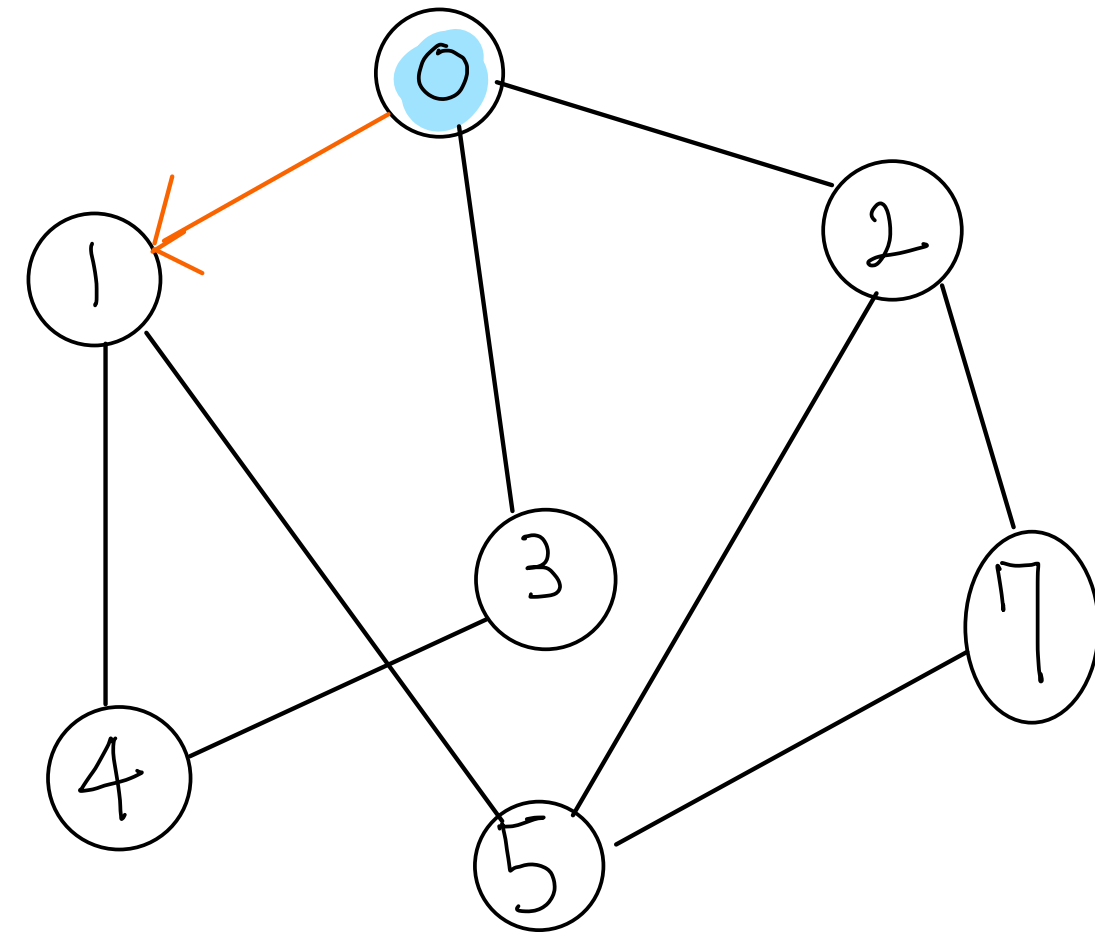
작은 값인 1로 탐색

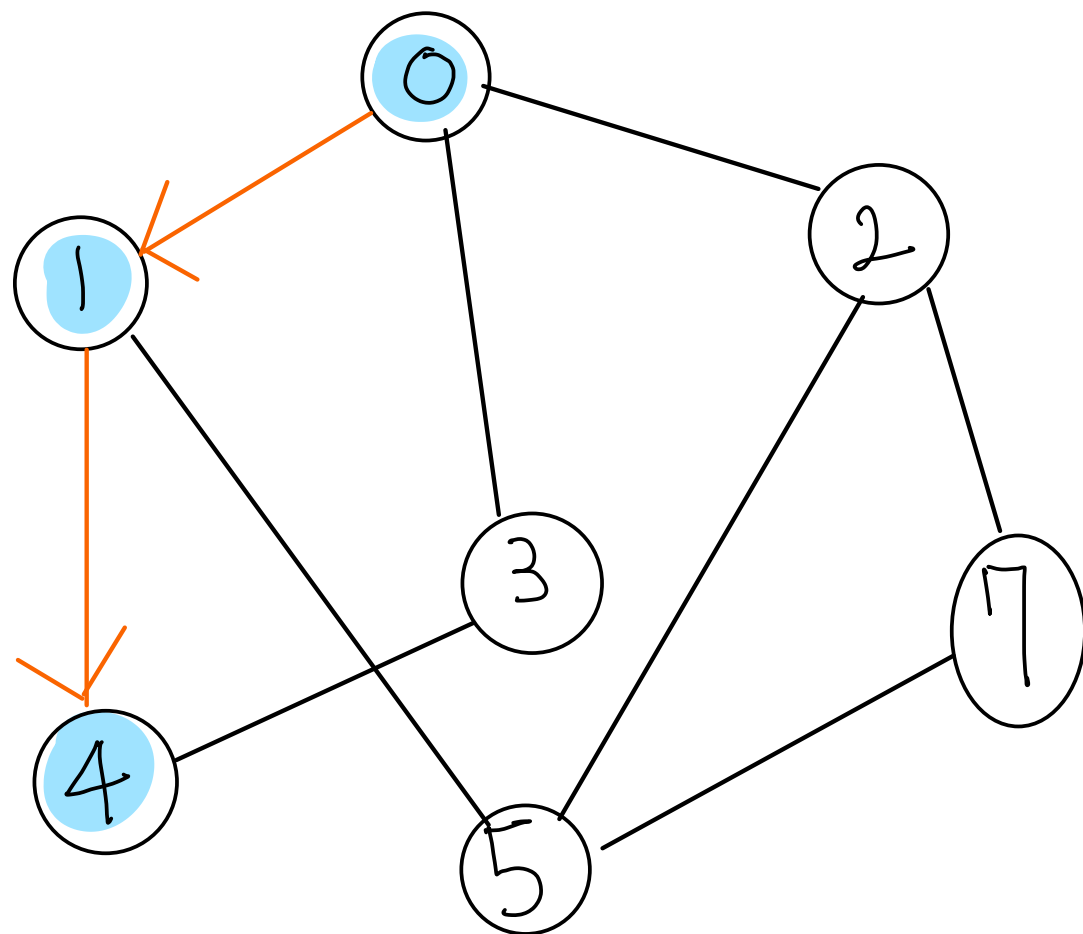


1에서 출발하면 0, 4, 5가 있지만

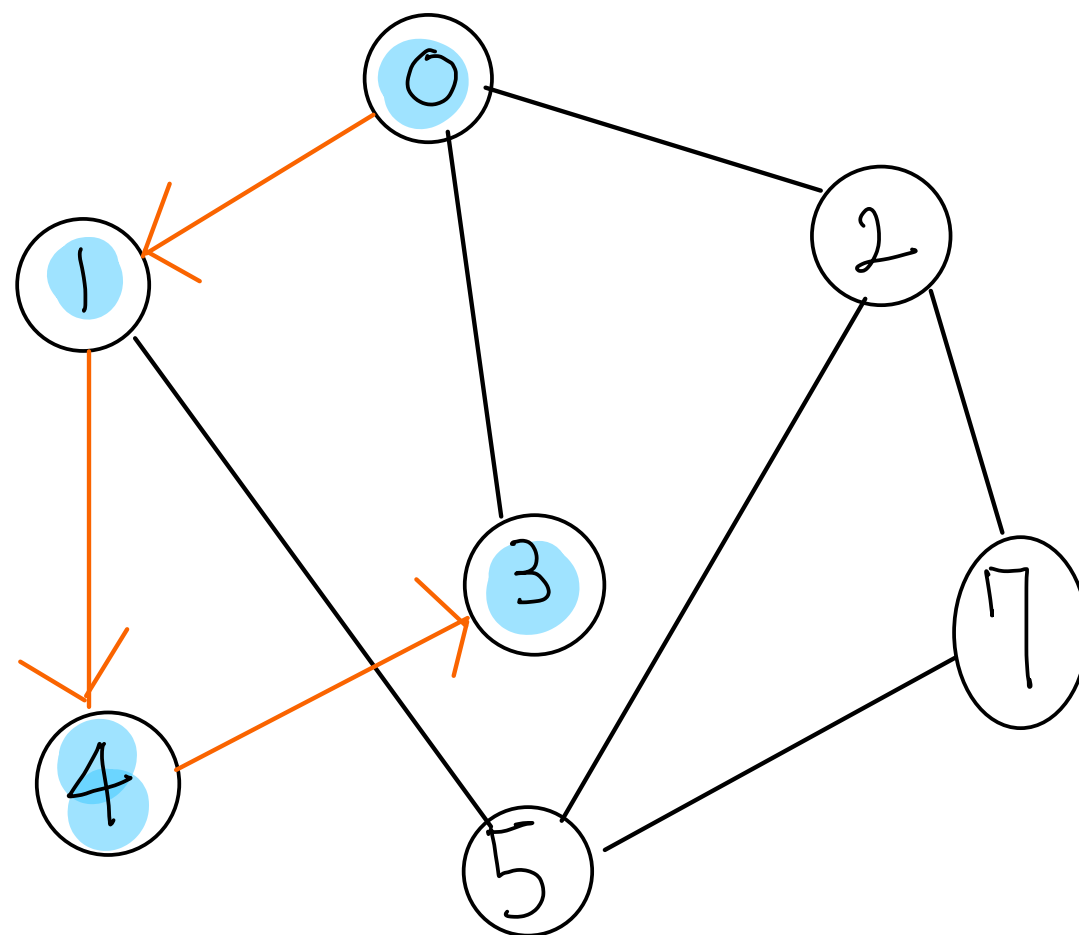
0은 이미 방문한 상태 이므로

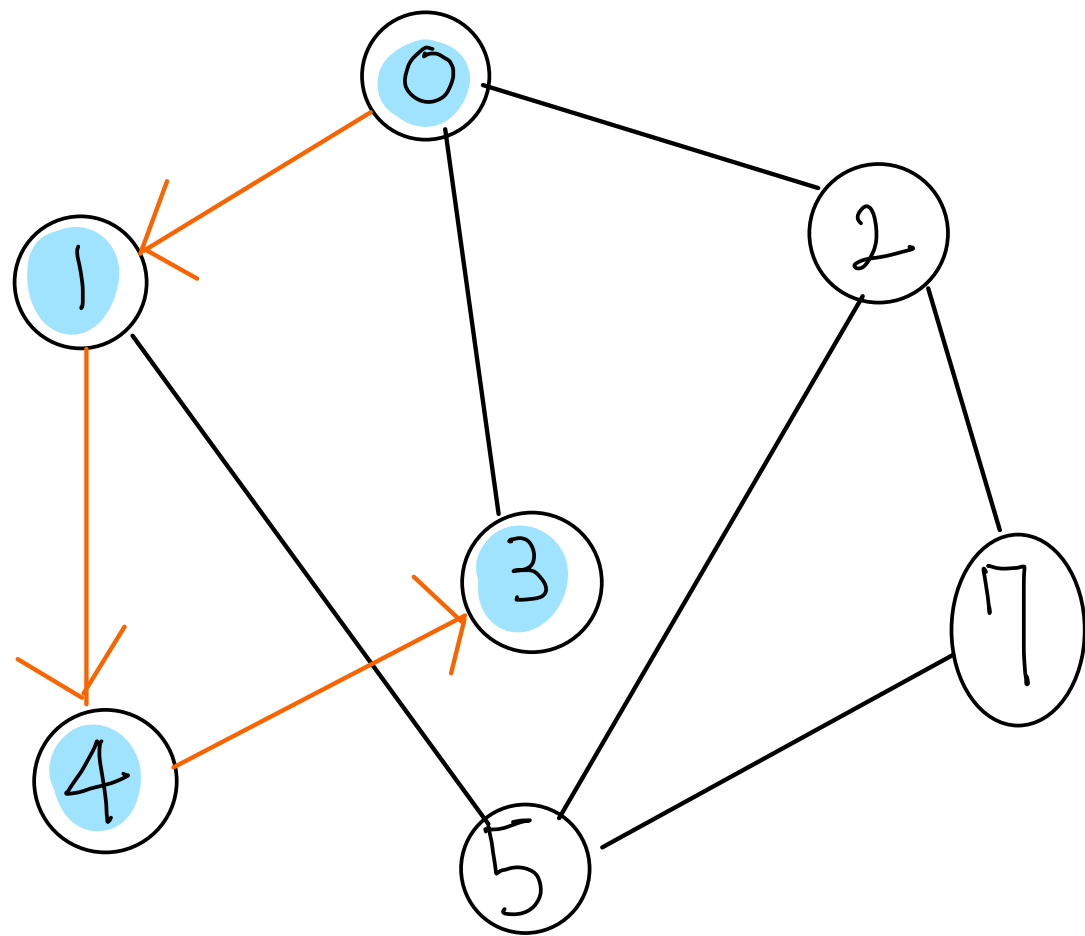
4, 5 중 작은 값인 4를 이동한다



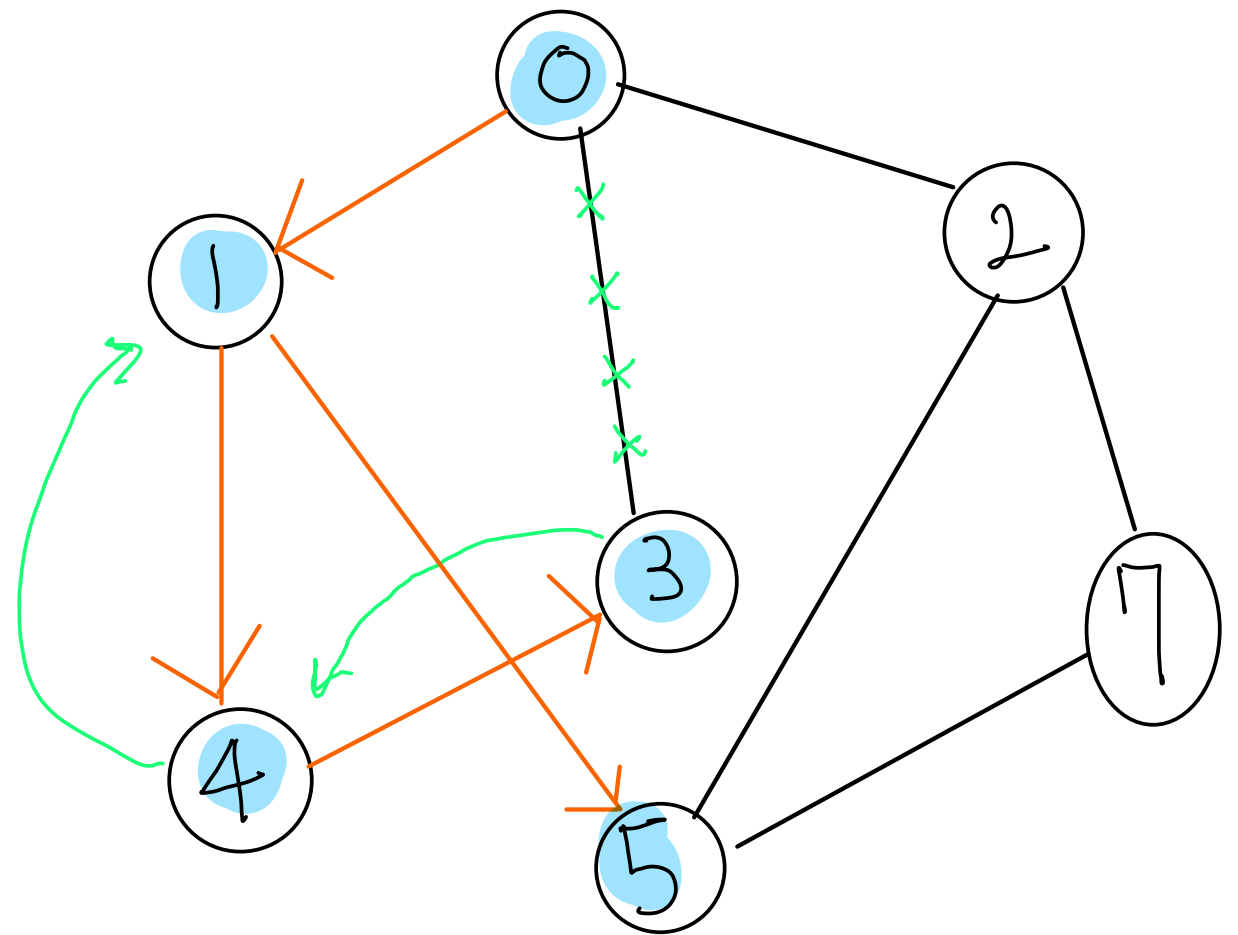
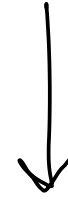


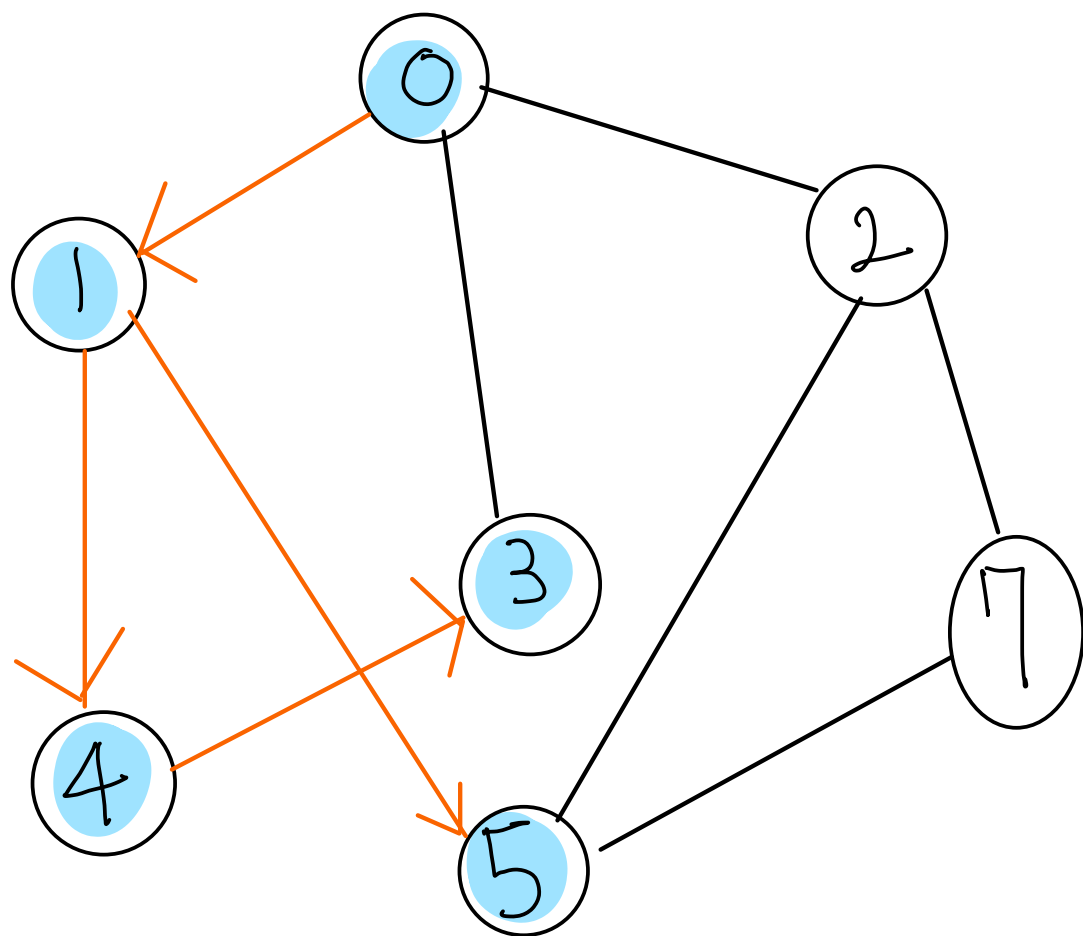
4에서 는 1, 3으로 이동 가능하지만
1은 이미 방문 했으므로 3으로 이동



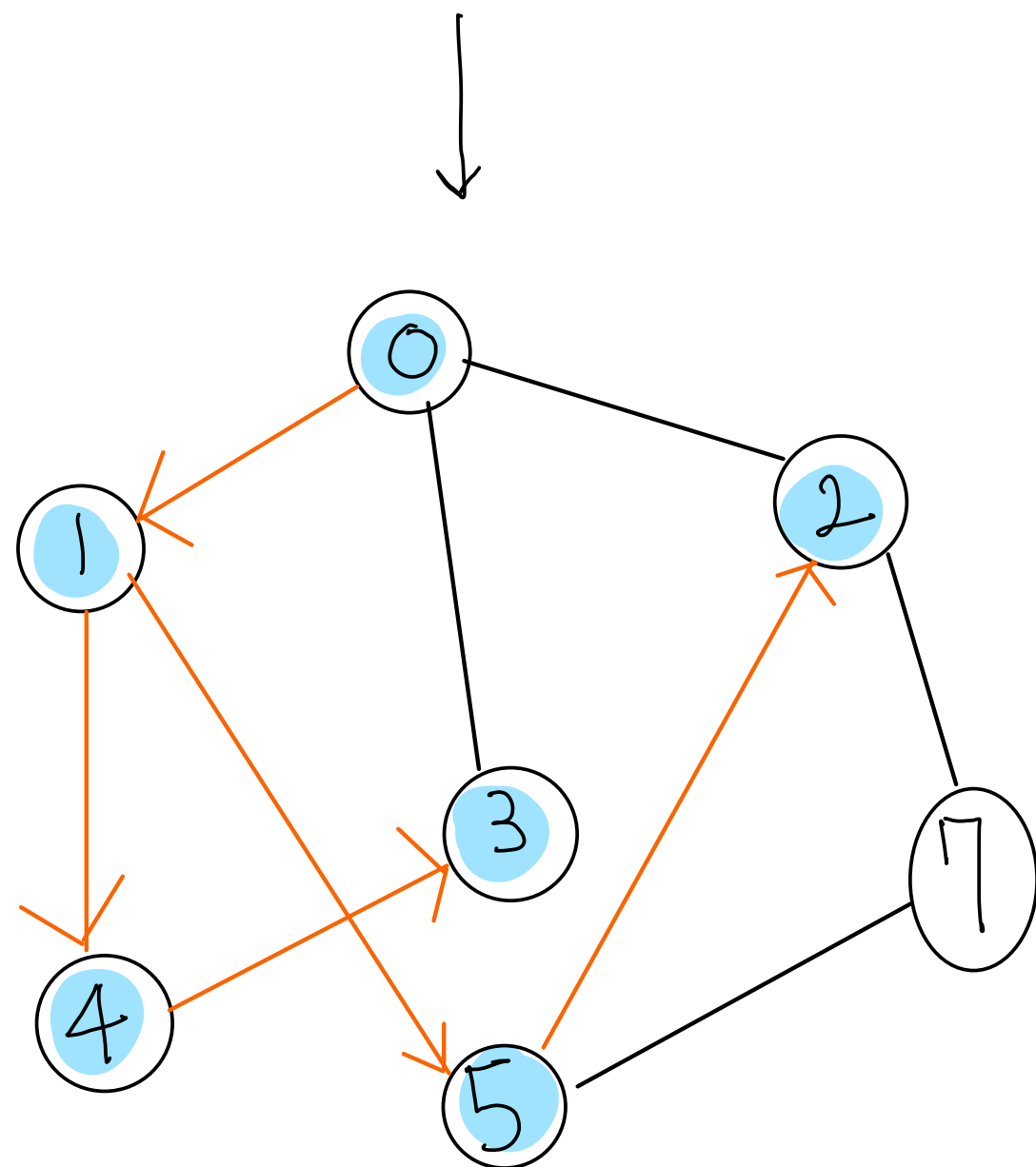


3에서는 0과 4로 이동 가능하지만
이미 모두 방문했으므로 4, 1을 거쳐
돌아간 후 1에서 방문 안한 5를 방문





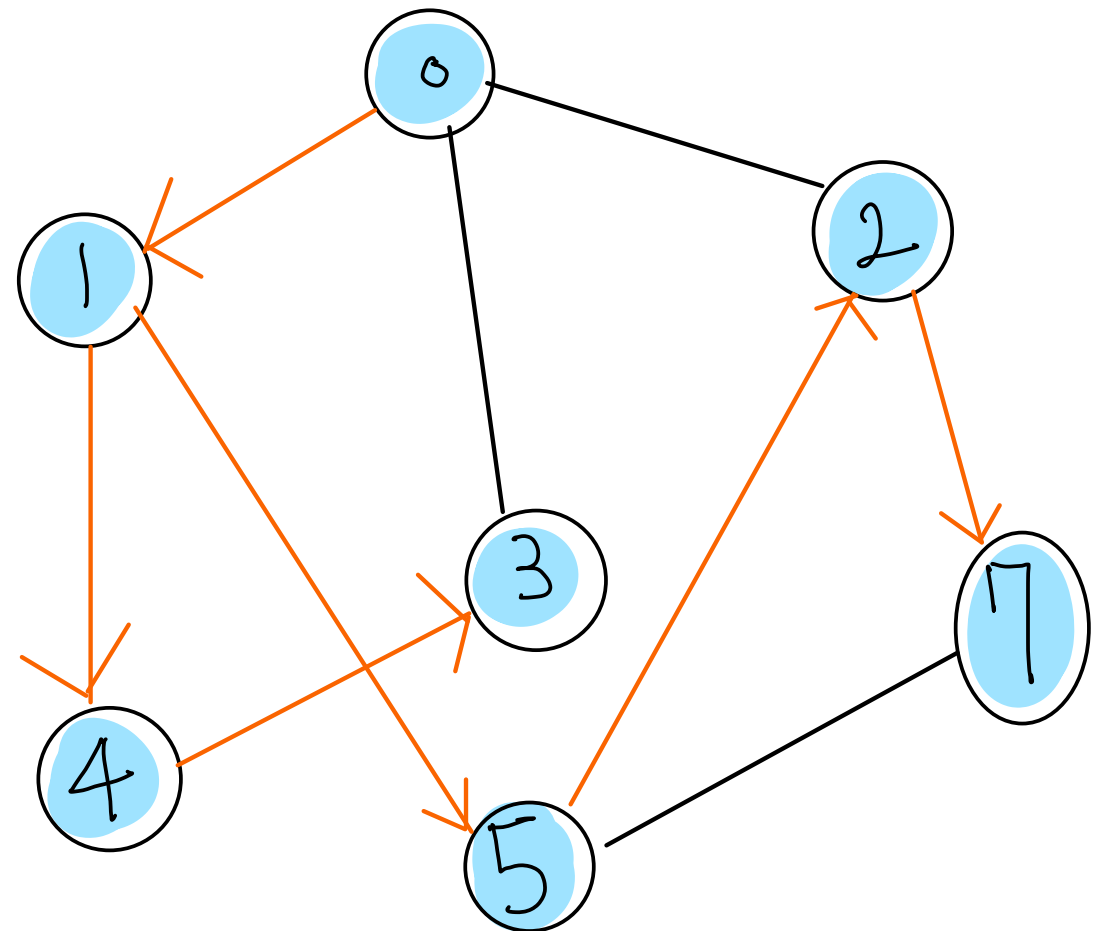
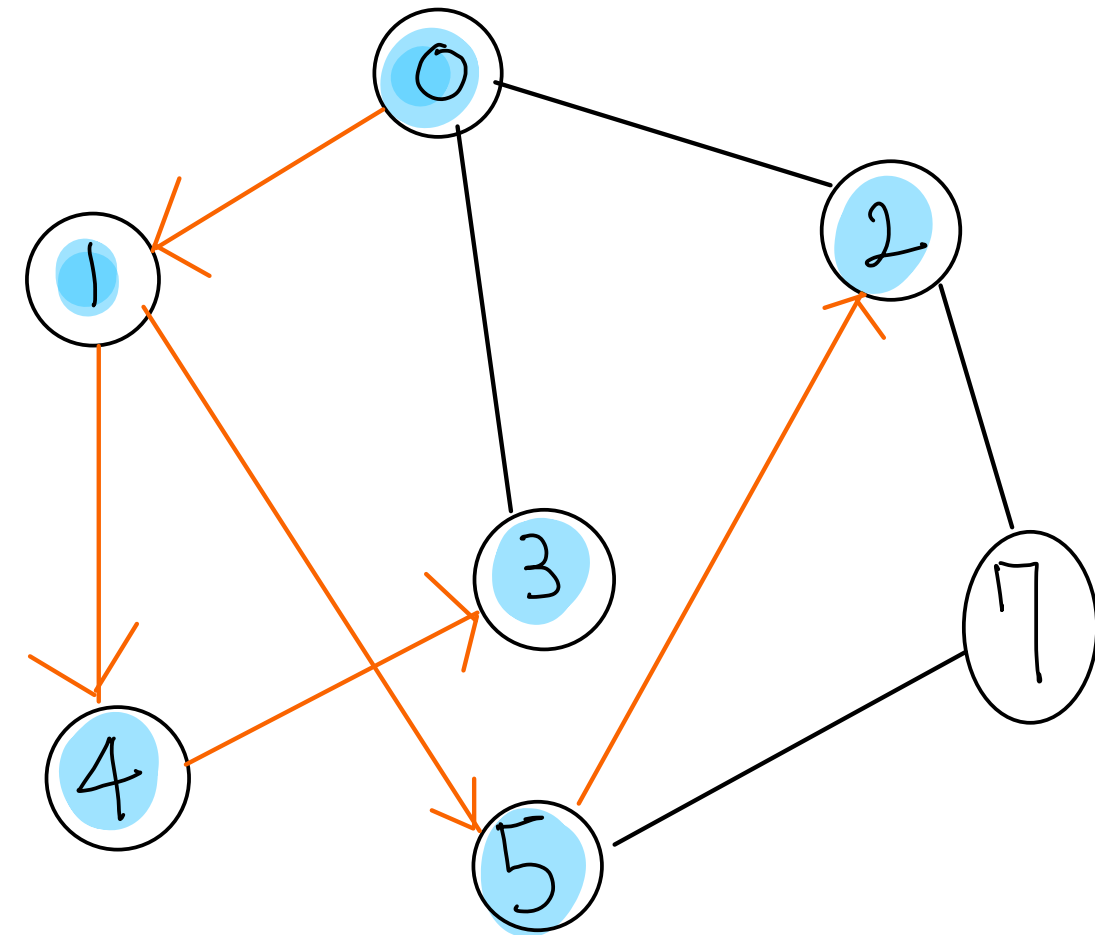
5에서는 2와 1과 7을 방문이 가능하지만
 1은 방문한 상태이고 2와 7은 작은 노드인
 2를 방문한다



2에서는 0과 7 방문이 가능하지만

0은 이미 방문한 상태이므로

7로 이동

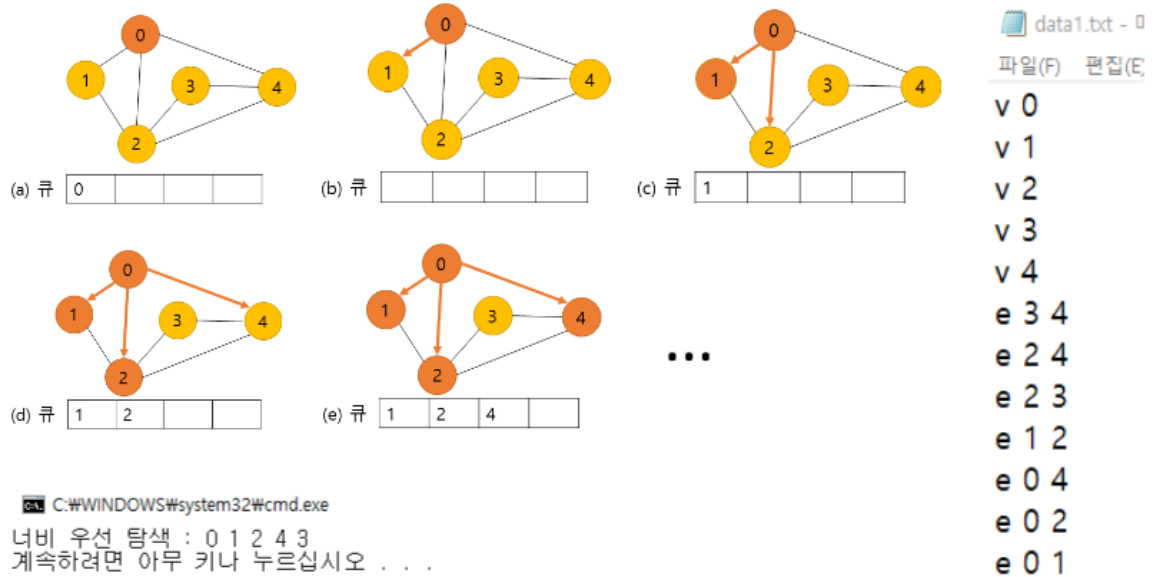


2.1 문제 분석



■ 그래프 너비 우선 탐색 프로그램

- data.txt에서 데이터를 읽어와 프로그램을 작성하시오. 이때 v는 정점, e는 간선을 의미한다. 프로그램의 실행과정은 아래의 그림과 같다.



2.2 소스 코드

```
1 // =====
2 // 제작 기간: 21년 10월 25일 ~ 21년 11월 1일
3 // 제작자 : 20204005 김필중
4 // 프로그램명: 그래프 너비 우선 탐색 프로그램
5 // =====
6
7 // 필요한 헤더파일을 선언한다
8 #include <stdio.h>
9 #include <stdlib.h>
10
11 // 오류 방식을 선언한다
12 #pragma warning(disable : 4996)
13
14 // 필요한 정보를 정의한다.
15 #define TRUE 1
16 #define FALSE 0
17 #define MAX_QUEUE_SIZE 10
18 #define MAX_VERTICES 50
19
20 int visited[MAX_VERTICES]; // 방문한 곳을 저장하는 정적 변수를 선언한다
21
22 typedef int element; // element 를 정수형으로 정의한다
23
24 // 큐 구조체를 생성한다
25 typedef struct
26 { // 큐 타입
27     element queue[MAX_QUEUE_SIZE]; // 정수형 큐 MAX_QUEUE_SIZE 크기의 배열을 만든다
28     int front, rear; // 첫번째 마지막 요소를 가리키는 정수형 변수를 선언한다
29 } QueueType;
30
31 // 오류 함수
32 void error(char *message)
33 {
34     fprintf(stderr, "%s\n", message); // 오류 출력 후 종료
35     exit(1);
36 }
37
38 // 큐 초기화 함수
39 void queue_init(QueueType *q)
40 {
41     q->front = q->rear = 0; // 큐의 앞과 뒤를 0으로 초기화 한다
42 }
43
44 // 공백 상태 검출 함수
45 int is_empty(QueueType *q)
46 {
47     return (q->front == q->rear); // 앞과 뒤가 같을 경우 1을 반환한다
48 }
49
50 // 포화 상태 검출 함수
51 int is_full(QueueType *q)
52 {
53     return ((q->rear + 1) % MAX_QUEUE_SIZE == q->front); // 큐의 마지막 요소 + 1 MAX_QUEUE_SIZE로 값을 나눈 나머지 값이 맨 앞 요소와 동일하면 1
54 }
55
```

```

56 // 삽입 함수
57 void enqueue(QueueType *q, element item)
58 {
59     if (is_full(q)) // 만약 q가 포화일경우
60         error("큐가 포화상태입니다"); // 종료
61     q->rear = (q->rear + 1) % MAX_QUEUE_SIZE; // q의 마지막 요소를 옮긴다
62     q->queue[q->rear] = item; // 마지막 요소에 item을 넣는다
63 }
64
65 // 삭제 함수
66 element dequeue(QueueType *q)
67 {
68     if (is_empty(q)) // 만약 q가 공백일경우
69         error("큐가 공백상태입니다"); // 종료
70     q->front = (q->front + 1) % MAX_QUEUE_SIZE; // q의 맨 앞을 찾은 후
71     return q->queue[q->front]; // 값을 리턴한다
72 }
73
74 // 그래프 구조체 선언
75 typedef struct GraphType
76 {
77     int n; // 정점의 개수
78     int adj_mat[MAX_VERTICES][MAX_VERTICES]; // 2차원 행렬 그래프를 만든다
79 } GraphType;
80
81
82
83 // 그래프 초기화
84 void graph_init(GraphType* g)
85 {
86     int r, c; // 정수형 변수 r, c를 선언한다
87     g->n = 0; // 그래프의 정점의 개수를 0개로 한다
88     for (r = 0; r < MAX_VERTICES; r++) // 행을 MAX_VERTICES 만큼 반복
89         for (c = 0; c < MAX_VERTICES; c++) // 열을 MAX_VERTICES 만큼 반복
90             g->adj_mat[r][c] = 0; // r, c 번째의 2차원 배열을 0으로 바꾼다
91 }
92
93 // 정점 삽입 연산
94 void insert_vertex(GraphType* g, int v)
95 {
96     if (((g->n) + 1) > MAX_VERTICES) // 만약 현재 정점 + 1개가 MAX_VERTICES 보다 크다면
97     {
98         fprintf(stderr, "그래프: 정점의 개수 초과"); // 오류 발생 후 종료
99         return;
100     }
101     g->n++; // 아닐경우는 정점 1개 증가
102 }
103
104 // 간선 삽입 연산
105 void insert_edge(GraphType* g, int start, int end)
106 {
107     if (start >= g->n || end >= g->n) // 만약 행 혹은 열이 정점의 개수보다 크거나 같으면
108     {
109         fprintf(stderr, "그래프: 정점 번호 오류"); // 오류 발생 후 종료
110         return;
111     }
112     g->adj_mat[start][end] = 1; // 받은 행과 열의 2차원 배열을 1로 바꾼다
113 }
114
115 // 너비 우선 탐색 함수
116 void bfs_mat(GraphType* g, int v)
117 {
118     int w; // w 변수를 선언한다
119     QueueType q; // 큐 구조체를 선언한다
120
121     queue_init(&q); // 큐 초기화
122     visited[v] = TRUE; // 정점 v 방문 표시를 한다
123     printf("%d ", v); // v를 출력한다
124     enqueue(&q, v); // 시작 정점을 큐에 저장한다
125
126     while (!is_empty(&q)) // 만약 q 큐가 비어있지 않다면
127     {
128         v = dequeue(&q); // 큐에 정점 추출을 한다
129         for (w = 0; w < g->n; w++) // 인접 정점 탐색을 한다
130             if (g->adj_mat[v][w] && !visited[w]) // 만약 w가 아직 정점에 방문하지 않았으면서 v행 w열인 2차원 배열이 0일 경우
131             {
132                 visited[w] = TRUE; // 방문 표시를 남긴다
133                 printf("%d ", w); // w를 출력한다
134                 enqueue(&q, w); // 방문한 정점을 큐에 저장
135             }
136     }
137 }
138

```

```

139 int main(void)
140 {
141     FILE *fp; // 파일 포인터 fp 선언
142     GraphType *g; // 그래프 포인터 g 선언
143     int temp1, temp2; // 임시 저장 정수형 변수 선언
144     int cnt = 0; // 정점 개수 확인 변수
145     char tmp = 'a'; // 임시 문자형 변수 선언
146
147
148     fp = fopen("data02.txt", "r"); // 파일을 오픈한다
149
150     if (fp == NULL) // 만약 오픈에 실패하면
151     {
152         printf("파일 오픈 실패\n"); // 오류 출력 후 종료
153         return 0;
154     }
155
156     // 파일 끝까지 반복한다
157     while (!feof(fp))
158     {
159         fscanf(fp, "%c", &tmp); // 파일의 맨 앞 문자를 읽어온다
160         if (tmp == 'v') // 맨 앞 문자가 v일 경우
161         {
162             fscanf(fp, "%d", &temp1); // temp1 에 숫자를 저장한다
163             cnt++; // cnt를 증가시킨다
164         }
165         if (tmp == 'e') // 맨 앞 문자가 e 일 경우
166         {
167             fscanf(fp, "%d %d", &temp1, &temp2); // temp1, temp2에 숫자를 저장한다
168         }
169     }
170
171     rewind(fp); // 파일 포인터를 처음으로 돌린다
172
173     g = (GraphType *)malloc(sizeof(GraphType)); // 그래프 g를 동적할당을 한다
174     graph_init(g); // 그래프 g를 초기화 한다
175     for (int i = 0; i < cnt; i++) // cnt 개수 만큼 반복해서 정점을 생성한다
176         insert_vertex(g, i);
177
178
179     // 파일 끝까지 반복한다
180     while (!feof(fp))
181     {
182         fscanf(fp, "%c", &tmp); // 파일의 맨 앞 문자를 읽어온다
183         if (tmp == 'v') // 맨 앞 문자가 v일 경우
184         {
185             fscanf(fp, "%d", &temp1); // temp1 에 숫자를 저장한다
186         }
187         if (tmp == 'e') // 맨 앞 문자가 e 일 경우
188         {
189             fscanf(fp, "%d %d", &temp1, &temp2); // temp1, temp2에 숫자를 저장한다
190             insert_edge(g, temp1, temp2); // 두 숫자를 간선 삽입 함수를 호출한다
191         }
192     }
193
194
195
196     printf("너비 우선 탐색\n");
197     bfs_mat(g, 0); // 0번 부터 시작하는 그래프 너비 우선 탐색 함수를 호출한다
198     printf("\n");
199     free(g); // 동적할당을 해제한다
200     fclose(fp); // 파일을 닫는다
201     return 0; // 종료한다
202 }

```

2.3 소스 코드 분석

```
// 필요한 헤더파일을 선언한다
#include <stdio.h>
#include <stdlib.h>

// 오류 방지를 선언한다
#pragma warning(disable : 4996)

// 필요한 정보를 정의한다.
#define TRUE 1
#define FALSE 0
#define MAX_QUEUE_SIZE 10
#define MAX_VERTICES 50

int visited[MAX_VERTICES]; // 방문한 곳을 저장하는 정적 변수를 선언한다

typedef int element; // element 를 정수형으로 정의한다
```

1. 필요한 헤더파일을 선언한다.
2. 필요한 true false를 정의하고 최대 값을 정의한다.
3. 오류 방지 구문을 선언한다
4. 방문한 곳을 저장하는 정적 행렬을 만든다.

```
// 큐 구조체를 생성한다
typedef struct
{
    // 큐 타입
    element queue[MAX_QUEUE_SIZE]; // 정수형 큐 MAX_QUEUE_SIZE 크기의 배열을 만든다
    int front, rear; // 첫번째 마지막 요소를 가리키는 정수형 변수를 선언한다
} QueueType;
```

5. 큐 구조체를 생성한다
6. 위에서 정의한 element 타입의 정의한 크기만큼 배열을 만든다
7. 첫번째 요소와 마지막 요소를 가리키는 정수형 변수를 선언한다

```
// 오류 함수
void error(char *message)
{
    fprintf(stderr, "%s\n", message); // 오류 출력 후 종료
    exit(1);
}
```

8. 오류 함수를 만들어 준다

```
// 공백 상태 검출 함수
int is_empty(QueueType *q)
{
    return (q->front == q->rear); // 앞과 뒤가 같을 경우 1을 반환한다
}
```

9. 공백 상태 검출 함수를 만들어 준다

10. 첫번째 요소와 마지막 요소가 같을 경우 공백이라는 뜻이므로 1을 반환한다

```
// 포화 상태 검출 함수
int is_full(QueueType *q)
{
    return ((q->rear + 1) % MAX_QUEUE_SIZE == q->front); // 큐의 마지막 요소 + 1 MAX_QUEUE_SIZE로 값을 나눈 나머지 값이 맨 앞 요소와 동일하면 1
}
```

11. 포화 상태 검출 함수를 만들어 준다

12. 큐의 마지막 요소의 1을 더한 값을 정의한 크기로 나눈 나머지 값이 맨 앞 요소와 동일하면 1을 반환한다

```
// 삽입 함수
void enqueue(QueueType *q, element item)
{
    if (is_full(q)) // 만약 q와 포화일경우
        error("큐가 포화상태입니다"); // 종료
    q->rear = (q->rear + 1) % MAX_QUEUE_SIZE; // q의 마지막 요소를 옮긴다
    q->queue[q->rear] = item; // 마지막 요소에 item을 넣는다
}
```

13. 큐 삽입 함수를 만들어 준다

14. 만약 q를 포화 함수에 호출하고 참일 경우 포화이므로 종료한다

15. 아닐 경우는 q의 마지막 요소를 나눈 나머지 값의 위치로 가리킨 후

그 위치에 item 값을 넣는다

```
// 삭제 함수
element dequeue(QueueType *q)
{
    if (is_empty(q)) // 만약 q가 공백일경우
        error("큐가 공백상태입니다"); // 종료
    q->front = (q->front + 1) % MAX_QUEUE_SIZE; // q의 맨 앞을 찾은 후
    return q->queue[q->front]; // 값을 리턴한다
}
```

16. 큐 제거 함수를 만든다

17. q를 공백 확인 함수에 호출하고 공백이면 종료한다

18. 아닐 경우는 q의 앞 요소의 + 1한 값을 나눈 나머지 값을 가리키게 한 후
그 값을 리턴 한다

```
// 그래프 구조체 선언
typedef struct GraphType
{
    int n; // 정점의 개수
    int adj_mat[MAX_VERTICES][MAX_VERTICES]; // 2차원 행렬 그래프를 만든다
} GraphType;
```

19. 그래프 구조체를 생성한다

20. 정점의 개수와 2차원 정수형 배열을 정의한 크기만큼 생성한다

```
// 그래프 초기화
void graph_init(GraphType* g)
{
    int r, c; // 정수형 변수 r, c를 선언한다
    g->n = 0; // 그래프의 정점의 개수를 0개로 한다
    for (r = 0; r < MAX_VERTICES; r++) // 행을 MAX_VERTICES 만큼 반복
        for (c = 0; c < MAX_VERTICES; c++) // 열을 MAX_VERTICES 만큼 반복
            g->adj_mat[r][c] = 0; // r, c 번째의 2차원 배열을 0으로 바꾼다
}
```

21. 그래프 초기화 함수를 만든다

22. 정수형 변수를 선언한 후 그래프의 정점의 개수를 0개로 만든다

23. 행을 정의한 크기만큼 반복하면서 열 또한 마찬가지로 반복하면서 2차원 배열을 0으로 모두 초기화 한다.

```
// 정점 삽입 연산
void insert_vertex(GraphType* g, int v)
{
    if (((g->n) + 1) > MAX_VERTICES) // 만약 현재 정점 + 1개가 MAX_VERTICES 보다 크다면
    {
        fprintf(stderr, "그래프: 정점의 개수 초과"); // 오류 발생 후 종료
        return;
    }
    g->n++; // 아닐 경우는 정점 1개 증가
}
```

24. 그래프 정점 삽입 함수를 만든다

25. 현재 정점의 1개 추가한 개수가 정의한 크기보다 크다면 종료한다

26. 아닐 경우는 정점 1개를 증가시킨다

```
// 간선 삽입 연산
void insert_edge(GraphType* g, int start, int end)
{
    if (start >= g->n || end >= g->n) // 만약 행 혹은 열이 정점의 개수보다 크거나 같으면
    {
        fprintf(stderr, "그래프: 정점 번호 오류"); // 오류 발생 후 종료
        return;
    }
    g->adj_mat[start][end] = 1; // 받은 행과 열의 2차원 배열을 1로 바꾼다
}
```

27. 그래프 간선 삽입 함수를 만든다

28. 행 혹은 열이 정점의 크기보다 크거나 같으면 오류 발생 후 종료한다

29. 아닐 경우는 받은 행과 열의 2차원 배열을 1로 바꾼다


```

// 너비 우선 탐색 함수
void bfs_mat(GraphType* g, int v)
{
    int w; // w 변수를 선언한다
    QueueType q; // 큐 구조체를 선언한다

    queue_init(&q); // 큐 초기화
    visited[v] = TRUE; // 정점 v 방문 표시를 한다
    printf("%d ", v); // v를 출력한다
    enqueue(&q, v); // 시작 정점을 큐에 저장한다

    while (!is_empty(&q)) // 만약 q 큐가 비어있지 않다면
    {
        v = dequeue(&q); // 큐에 정점 추출을 한다
        for (w = 0; w < g->n; w++) // 인접 정점 탐색을 한다
            if (g->adj_mat[v][w] && !visited[w]) // 만약 w가 아직 정점에 방문하지 않았으면서 v행 w열인 2차원 배열이 0일 경우
            {
                visited[w] = TRUE; // 방문 표시를 남긴다
                printf("%d ", w); // v를 출력한다
                enqueue(&q, w); // 방문한 정점을 큐에 저장
            }
    }
}

```

30. 우선 정수형 변수 w를 선언한다
31. 큐 구조체를 선언한다
32. 큐를 초기화 하고, v 정점을 방문했음을 체크한다
33. v를 출력하고 시작 정점을 큐에 저장한다
34. 만약 큐가 비어 있지 않을 때까지 반복한다
35. 큐에 정점 제거 함수를 호출해서 리턴 되는 값을 v에 저장한다
36. w의 크기가 정점의 개수보다 작을 때 계속 반복한다.
37. w가 정점에 방문하지 않았으면서 v행 w열인 2차원 배열이 0일 경우 w에 방문 표시를 남긴 후 v를 출력하고 방문한 정점을 큐에 저장한다

```

FILE *fp; // 파일 포인터 fp 선언
GraphType *g; // 그래프 포인터 g 선언
int temp1, temp2; // 임시 저장 정수형 변수 선언
int cnt = 0; // 정점 개수 확인 변수
char tmp = 'a'; // 임시 문자형 변수 선언

fp = fopen("data02.txt", "r"); // 파일을 오픈한다

if (fp == NULL) // 만약 오픈에 실패하면
{
    printf("파일 오픈 실패\n"); // 오류 출력 후 종료
    return 0;
}

```

38. 파일 포인터 fp를 선언한다

39. 그래프 포인터 g를 선언한다

40. 임시 저장 정수형 변수를 선언한다. 최대값 저장 변수 선언을 한다. 임시 문자형 변수 선언을 한다.

41. 파일을 오픈하고 만약 오픈에 실패할 경우는 오류 발생 후 종료한다

```

// 파일 끝까지 반복한다
while (!feof(fp))
{
    fscanf(fp, "%c", &tmp); // 파일의 맨 앞 문자를 읽어온다
    if (tmp == 'v') // 맨 앞 문자가 v일 경우
    {
        fscanf(fp, "%d", &temp1); // temp1에 숫자를 저장한다
        cnt++; // cnt를 증가시킨다
    }
    if (tmp == 'e') // 맨 앞 문자가 e일 경우
    {
        fscanf(fp, "%d %d", &temp1, &temp2); // temp1, temp2에 숫자를 저장한다
    }
}

rewind(fp); // 파일 포인터를 처음으로 돌린다

```

42. 파일 끝까지 반복하면서 맨 앞의 문자를 읽어 온다

43. 만약 맨 앞 문자가 v일 경우 우선 temp1에 숫자를 저장한다

44. cnt값을 증가시킨다

45. e일 경우는 저장만 한다

46. 파일 포인터를 맨 앞으로 돌린다

```
g = (GraphType *)malloc(sizeof(GraphType)); // 그래프 g를 동적할당을 한다
graph_init(g); // 그래프 g를 초기화 한다
for (int i = 0; i < cnt; i++) // cnt 개수 만큼 반복해서 정점을 생성한다
    insert_vertex(g, i);
```

47. 그래프 g를 동적할당을 한다

48. 그래프 g를 초기화 한다

49. 그래프 g를 최대값 + 1만큼 반복하면서 정점을 생성한다

```
// 파일 끝까지 반복한다
while (!feof(fp))
{
    fscanf(fp, "%c", &tmp); // 파일의 맨 앞 문자를 읽어온다
    if (tmp == 'v') // 맨 앞 문자가 v일 경우
    {
        fscanf(fp, "%d", &temp1); // temp1 에 숫자를 저장한다
    }
    if (tmp == 'e') // 맨 앞 문자가 e 일 경우
    {
        fscanf(fp, "%d %d", &temp1, &temp2); // temp1, temp2에 숫자를 저장한다
        insert_edge(g, temp1, temp2); // 두 숫자를 간선 삽입 함수를 호출한다
    }
}
```

50. 파일 끝까지 반복하면서 파일 맨 앞을 읽어 온다

51. 맨 앞이 v일 경우 temp1에 저장만 한다

52. 맨 앞이 e일 경우 2개의 숫자를 읽어 온다

53. temp1 temp2 숫자를 간선 삽입 함수를 호출한다

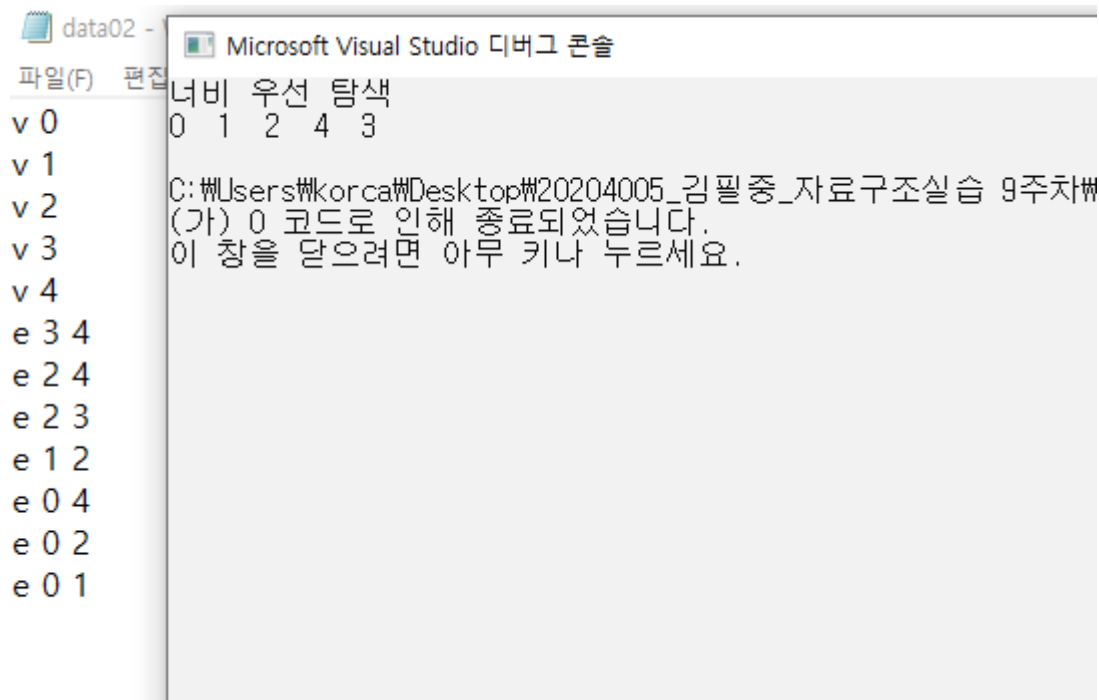
```
printf("너비 우선 탐색\n");  
bfs_mat(g, 0); // 0번 부터 시작하는 그래프 너비 우선 탐색 함수를 호출한다  
printf("\n");  
free(g); // 동적할당을 해제한다  
fclose(fp); // 파일을 닫는다  
return 0; // 종료한다
```

54. 그 후 그래프 너비 우선 탐색 결과를 출력하고

55. 동적할당을 해제한 후 파일을 닫는다

56. 종료한다.

2.4 실행 화면



```
data02 - Microsoft Visual Studio 디버그 콘솔
파일(F) 편집
너비 우선 탐색
v 0 0 1 2 4 3
v 1
v 2 C:\Users\korca\Desktop\20204005_김필중_자료구조실습 9주차
v 3 (가) 0 코드로 인해 종료되었습니다.
v 4 이 창을 닫으려면 아무 키나 누르세요.
e 3 4
e 2 4
e 2 3
e 1 2
e 0 4
e 0 2
e 0 1
```

2.5 느낀점

이번 과제는 큐를 이용해서 우선 너비 탐색을 이용한 방법이다

이번 과제 문제에서는 그림이 주어진 상태로 문제를 해결해야하는 방식이었기 때문에 훨씬 더 이해가 잘 되는 상태로 문제를 수행했다. 처음에는 큐를 왜 이용해야 하는지 모르고있었으나 너비는 도착한 정점에서 차례로 뺀어 나가는 방법이기 때문에 큐에 넣은 후 꺼낸 다음 꺼낸 정점에서 연결된 정점을 차례로 다시 넣고 하는 방식으로 진행해야 한다는 사실을 알고 큐를 써야하는 이유를 명확히 알게 되었다. 그래서 이번 과제를 하면서 다시한번 큐를 공부하고 너비 우선 탐색을 공부하면서 언제 사용할지에 대해 계속 고민하고 나중에 프로그램을 만들 때 사용하게겠다고 느꼈다

