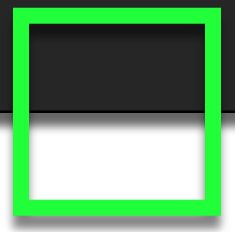
#Algorithm 반
#2주차

BFS/ DFS

T. 최창규
Asst. 강건 남지훈



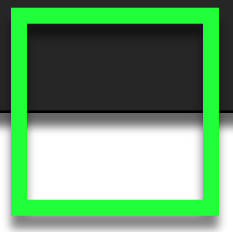
ALOHA
The algorithm club.

목차

0. Graph
1. DFS, BFS
2. 미로찾기

#0

Graph

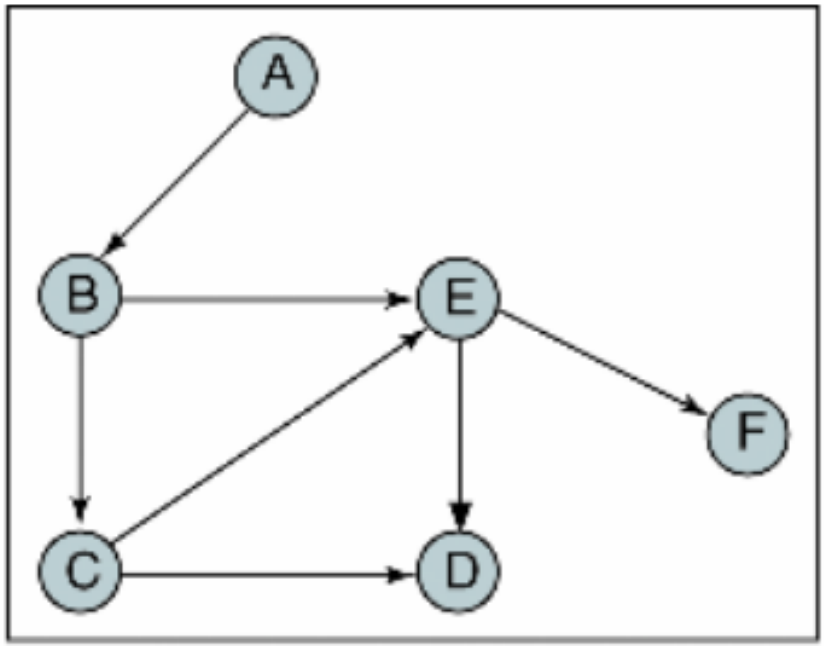


ALOHA
The algorithm club.

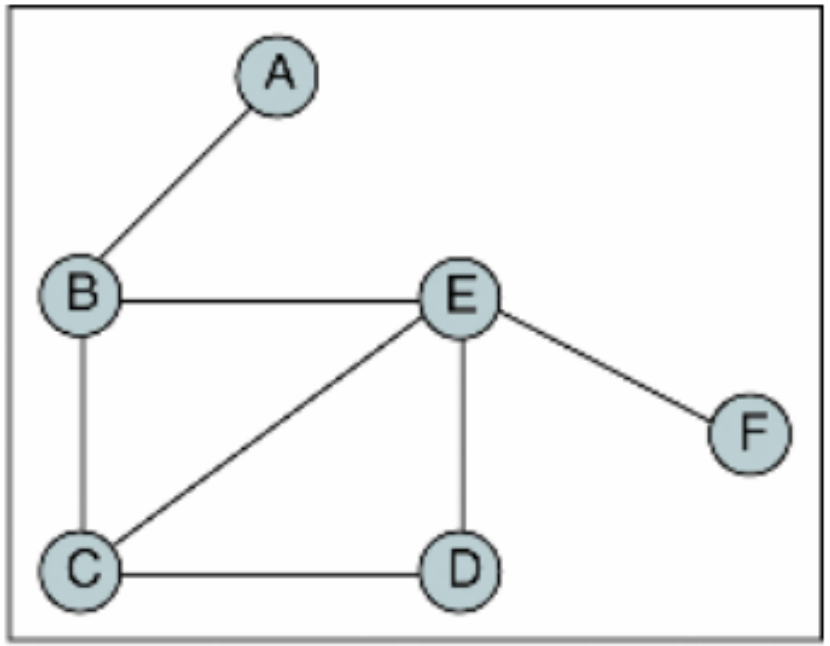
그래프 (Graph) 란?

- 연결되어 있는 정점과 정점간의 관계를 표현할 수 있는 자료 구조.
- 정점(Vertex)과 간선(Edge)로 구성되어 있다.

그래프 (Graph) 란?

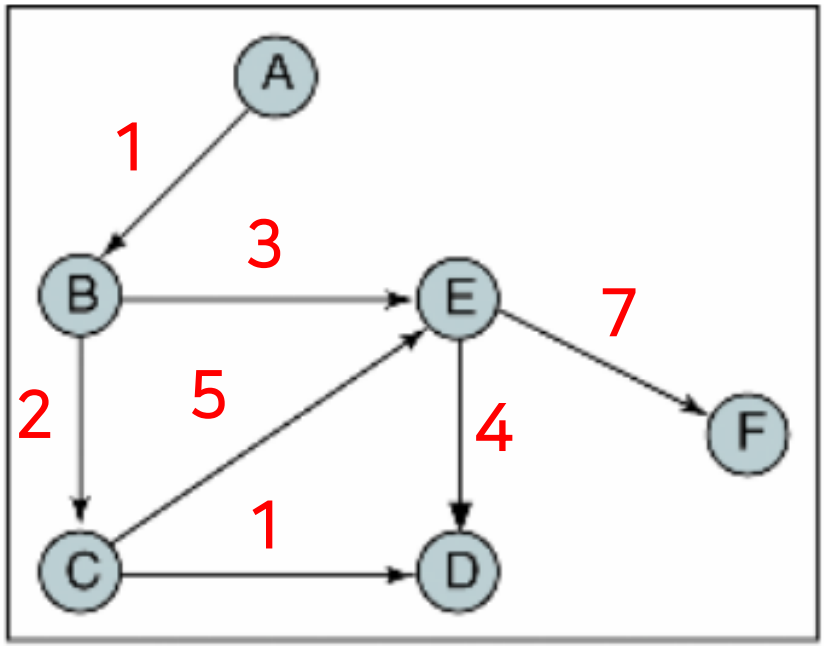


(a) Directed graph

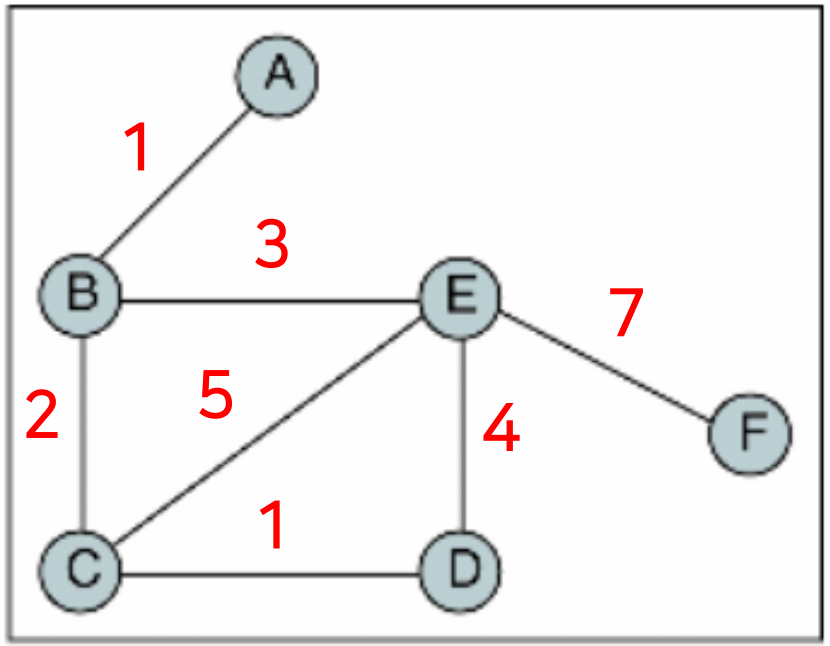


(b) Undirected graph

그래프 (Graph) 란? + 가중치



(a) Directed graph



(b) Undirected graph

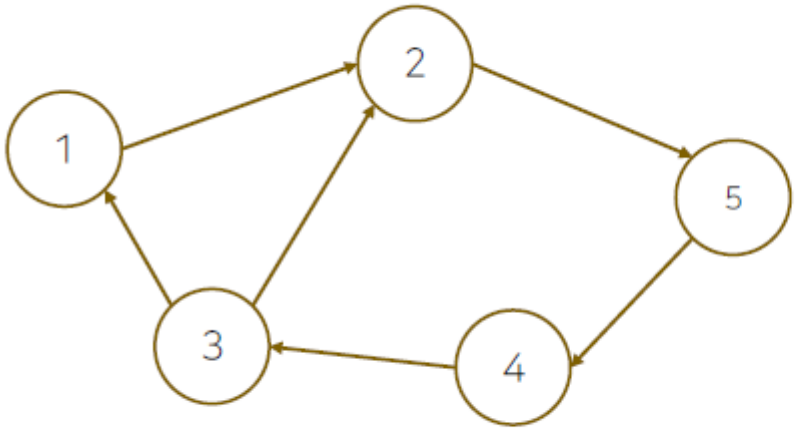
그래프 (Graph) 란?

- 그래프를 표현 할 수 있는 방법 2가지

1. 인접 행렬(Adjacency Matrix)
2. 인접 리스트(Adjacency List)

그래프 (Graph) 란?

1. 인접 행렬(Adjacency Matrix)



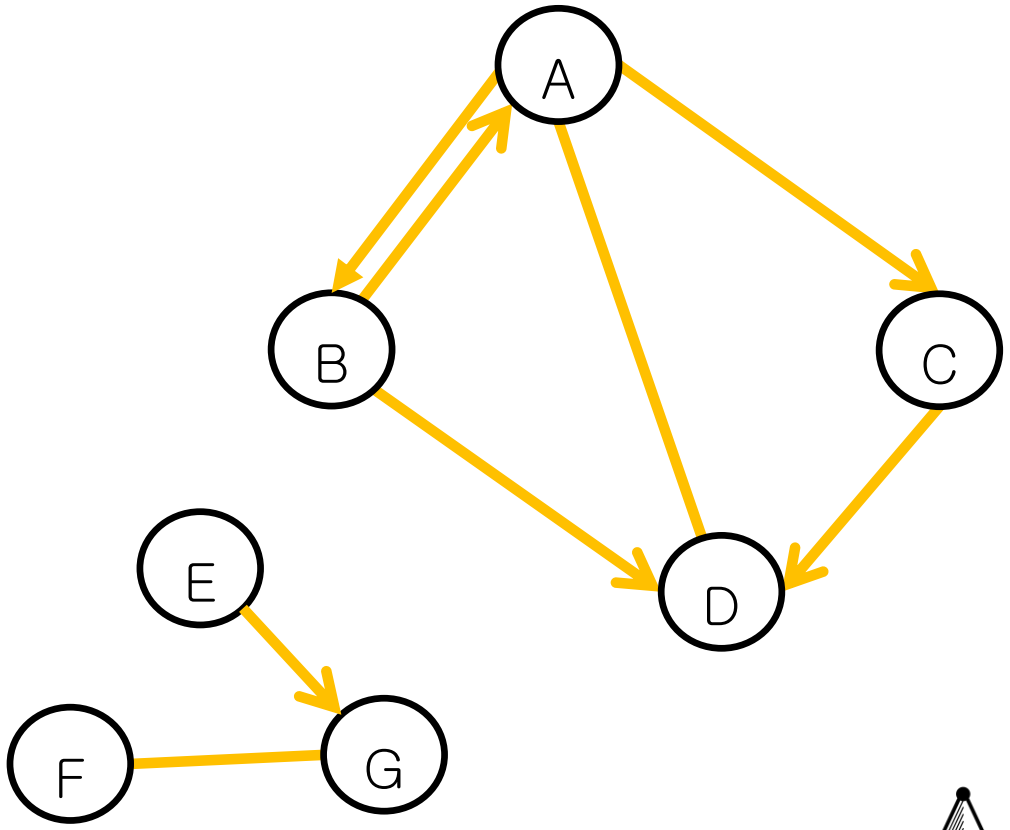
Adjacency Matrix (인접 행렬)

	1	2	3	4	5
1	0	1	0	0	0
2	0	0	0	0	1
3	1	1	0	0	0
4	0	0	0	1	0
5	0	0	0	1	0

↑ (작년 창호 PPT)

그래프 (Graph) 란?

2. 인접 리스트(Adjacency List)



	A	B	C	D	E	F	G
A	0	1	1	1	0	0	0
B	1	0	0	1	0	0	0
C	0	0	0	1	0	0	0
D	1	0	0	0	0	0	0
E	0	0	0	0	0	0	1
F	0	0	0	0	0	0	1
G	0	0	0	0	0	1	0

↑ (윈캠 태현이 PPT)

ALOHA
The algorithm club.

그래프 (Graph) 란?

3(?). Vector

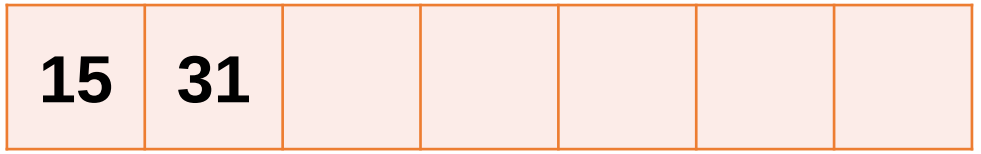
- std::vector is a sequence container that encapsulates dynamic size arrays.

그래프 (Graph) 란?

3(?). Vector

2주차2주차2주차2주차2주차2주차2주차.cpp

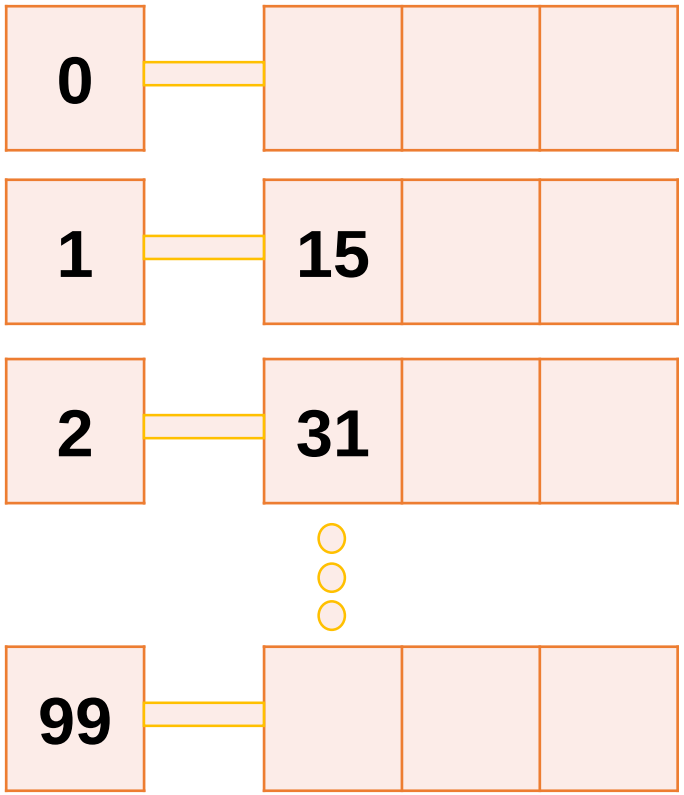
```cpp
#include<vector>
using namespace std;
int main(){
    vector<int> vec;
    vec.push_back(15);
    vec.push_back(31);
    for(int i=0; i<vec.size(); i++){
        printf("%d",vec[i]);
    }
}
```



size = 2

그래프 (Graph) 란?

3(?). Vector



2주차2주차2주차2주차2주차2주차2주차2주차.cpp

```cpp
#include<vector>
using namespace std;
int main(){
    vector<int> vec[100];
    vec[1].push_back(15);
    vec[2].push_back(31);
    for(int i=0; i<100; i++){
        for(int j=0; j<vec[i].size(); j++)
            printf("%d",vec[i]);
    }
}
```

ALOHA
The algorithm club.

그래프 (Graph) 란?

3(?). Vector

입력

첫째 줄에 정점의 개수 N(1 ≤ N ≤ 1,000), 간선의 개수 M(1 ≤ M ≤ 10,000), 탐색을 시작할 정점의 번호 V가 주어진다. 다음 M개의 줄에는 간선이 연결하는 두 정점의 번호가 주어진다. 어떤 두 정점 사이에 여러 개의 간선이 있을 수 있다. 입력으로 주어지는 간선은 양방향이다.

예제 입력 1 복사

```
4 5 1
1 2
1 3
1 4
2 4
3 4
```

ALOHA
The algorithm club.

그래프 (Graph) 란?

3(?). Vector

입력

첫째 줄에 정점의 개수 N(1 ≤ N ≤ 1,000), 간선의 개수 M(1 ≤ M ≤ 10,000), 탐색을 시작할 정점의 번호 V가 주어진다. 다음 M개의 줄에는 간선이 연결하는 두 정점의 번호가 주어진다. 어떤 두 정점 사이에 여러 개의 간선이 있을 수 있다. 입력으로 주어지는 간선은 양방향이다.

예제 입력 1 복사

```
4 5 1
1 2
1 3
1 4
2 4
3 4
```

2주차2주차2주차2주차2주차2주차2주차2주차.cpp

```cpp
int main(){
    int n,m,v,a,b;
    scanf("%d %d %d",&n,&m,&v);
    for(int i=0; i<n; i++){
        scanf("%d %d",&a,&b);
        vec[a].push_back(a);
        vec[b].push_back(a);
    }
}
```

ALOHA
The algorithm club.

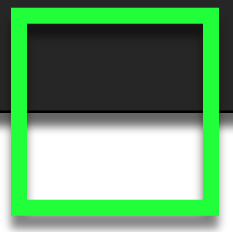
그래프 (Graph) 란?

- 탐색

1. DFS(Depth First Search), 깊이 우선 탐색
2. BFS(Breadth First Search), 너비 우선 탐색

#1

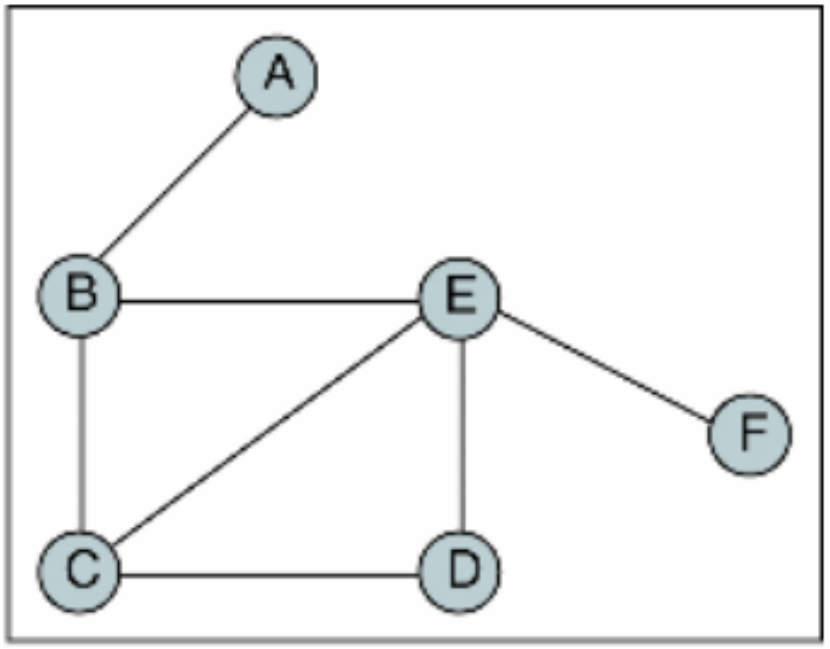
DFS, BFS



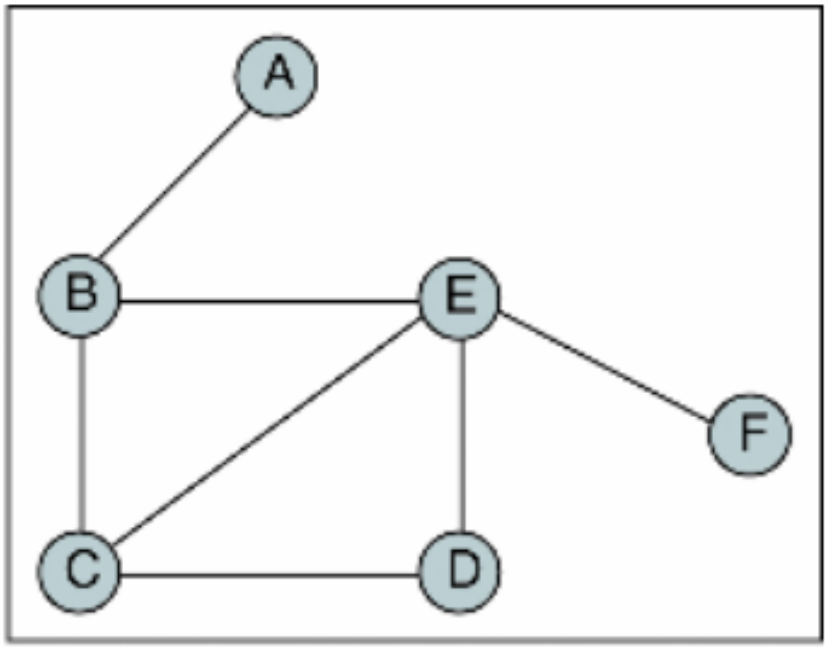
DFS?

- 현재 정점에서 갈 수 있는 정점까지 계속 탐색
- 탐색한 정점은 다시 탐색하지 않음 (visit배열을 활용)
- 주로 재귀함수를 사용(스택도 O)
- 인접 행렬 사용: O(V^2)
  인접 리스트 사용: O(V+E)

DFS?

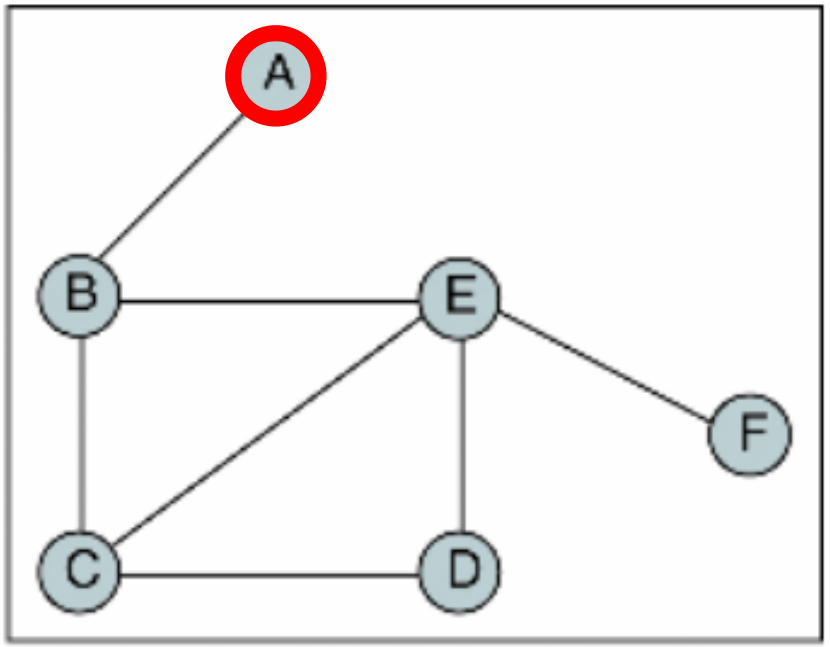


DFS?



- A부터
- 번호가 작은 노드부터 탐색

DFS?



- A부터
- 번호가 작은 노드부터 탐색

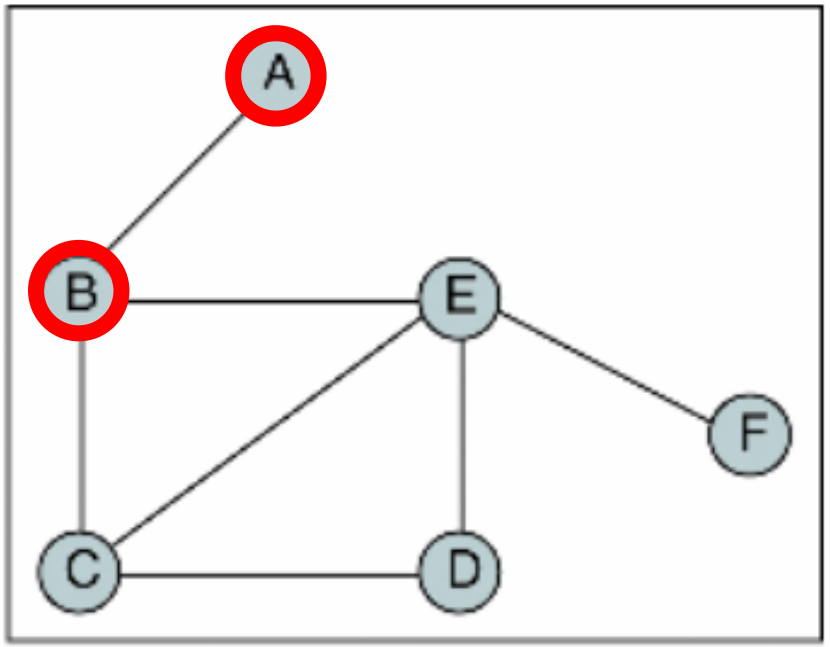
A

now = A

visit

A	1
B	0
C	0
D	0
E	0
F	0

DFS?



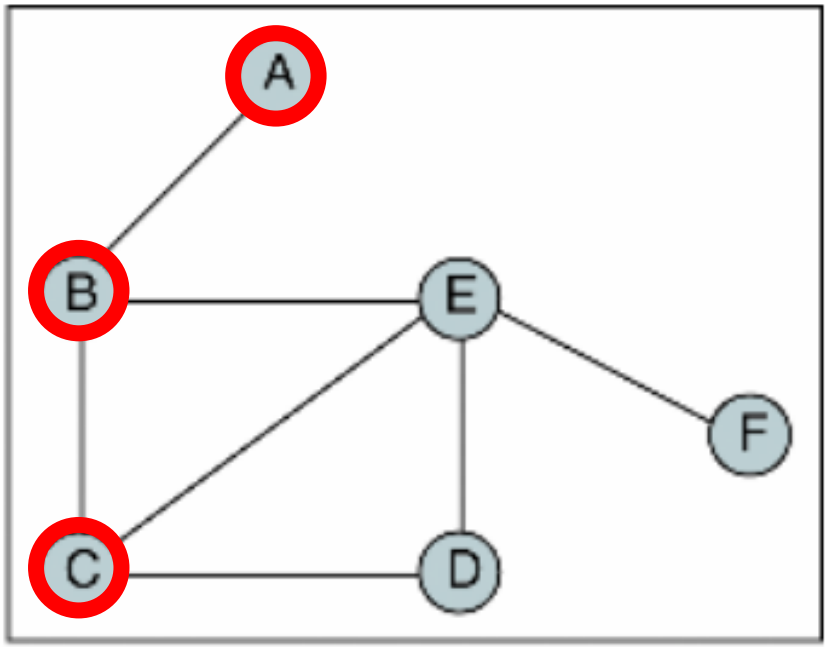
- A부터
- 번호가 작은 노드부터 탐색

A → B

now = B

	visit
A	1
B	1
C	0
D	0
E	0
F	0

DFS?



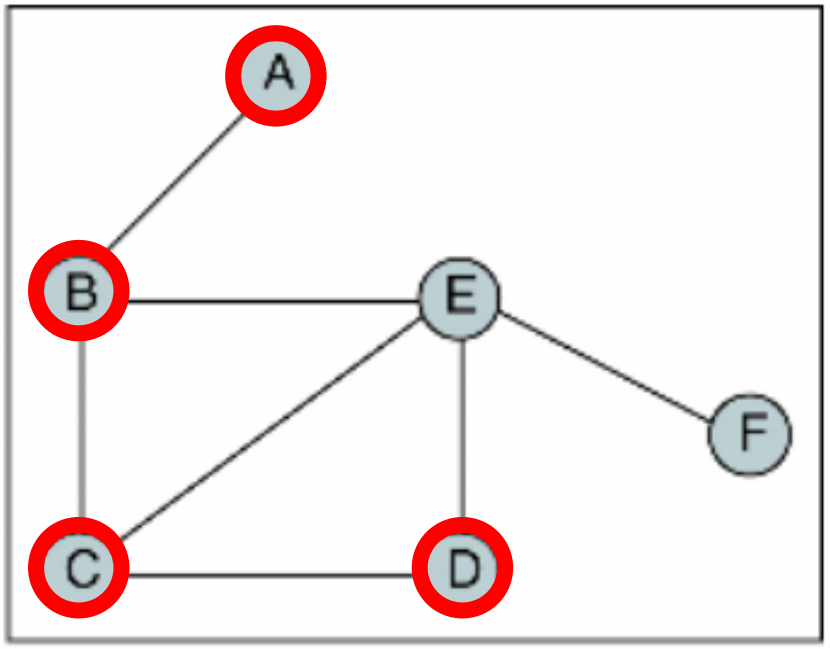
- A부터
- 번호가 작은 노드부터 탐색

A → B → C

now = C

	visit
A	1
B	1
C	1
D	0
E	0
F	0

DFS?



- A부터
- 번호가 작은 노드부터 탐색

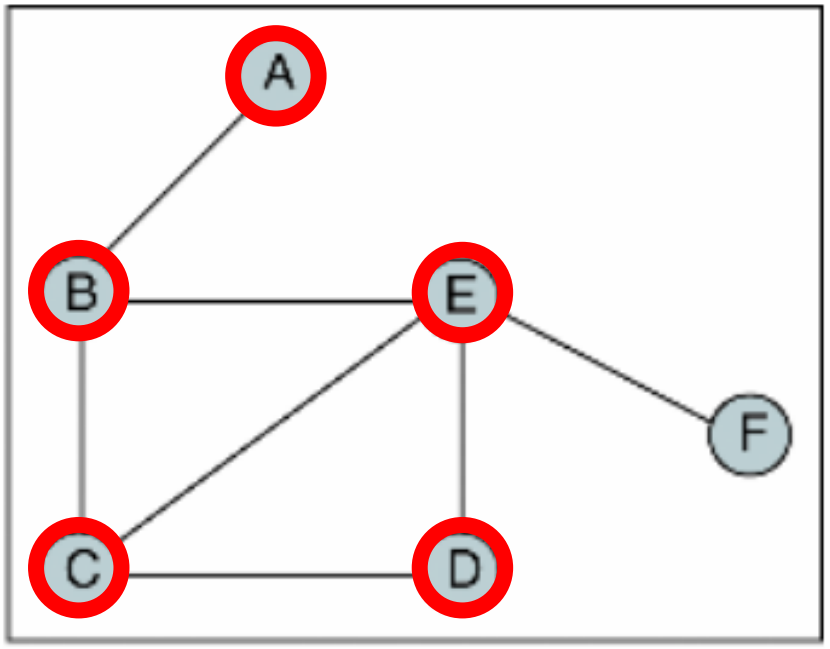
A → B → C → D

now = D

	visit
A	1
B	1
C	1
D	1
E	0
F	0

ALOHA
The algorithm club.

DFS?



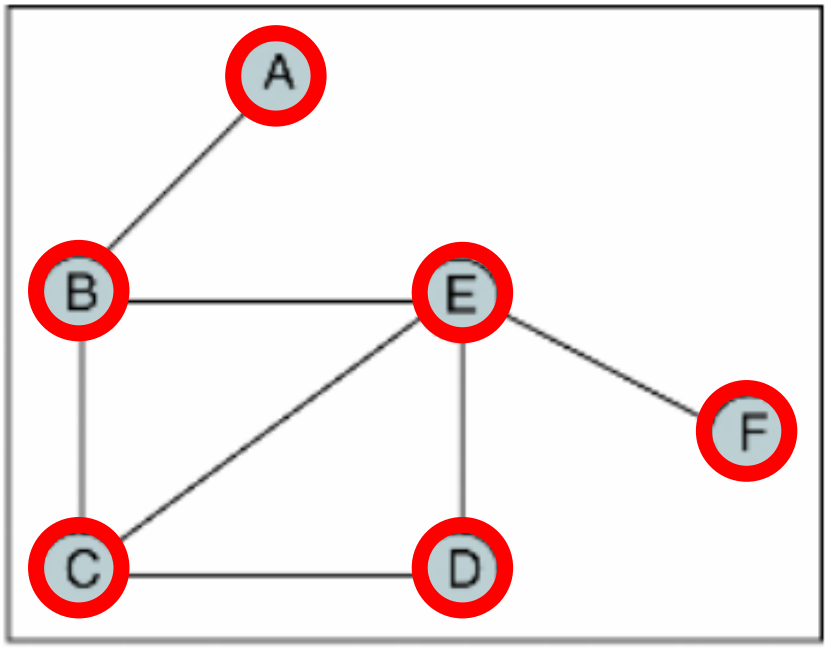
- A부터
- 번호가 작은 노드부터 탐색

A → B → C → D → E

now = E

	visit
A	1
B	1
C	1
D	1
E	1
F	0

DFS?



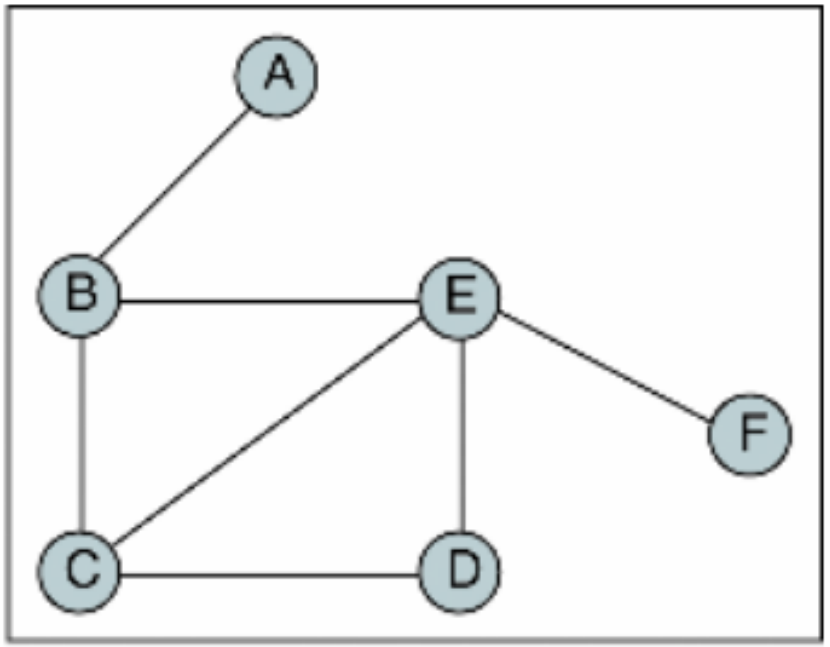
- A부터
- 번호가 작은 노드부터 탐색

A → B → C → D → E →F

now = F

	visit
A	1
B	1
C	1
D	1
E	1
F	1

DFS?



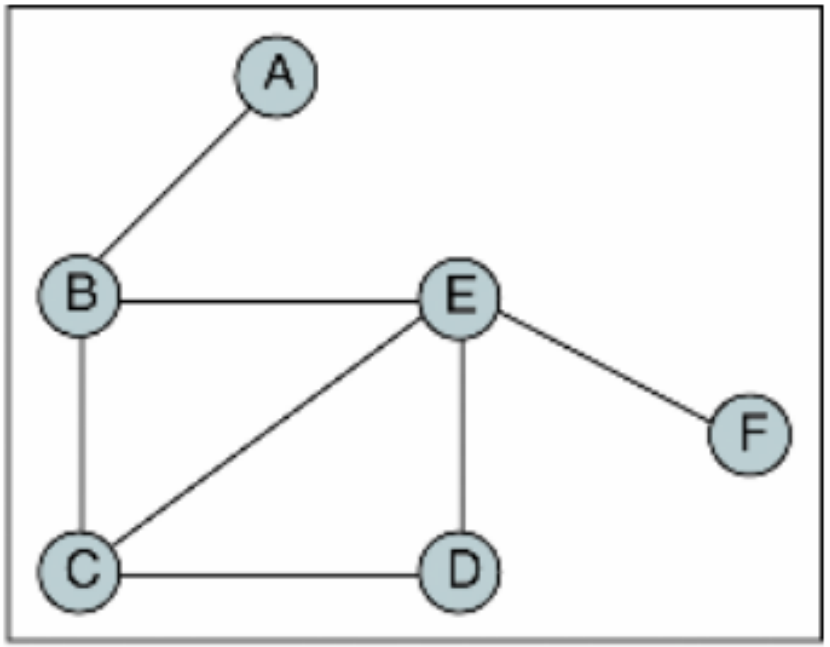
- B부터
- 번호가 작은 노드부터 탐색

B → A → C → D → E → F

BFS?

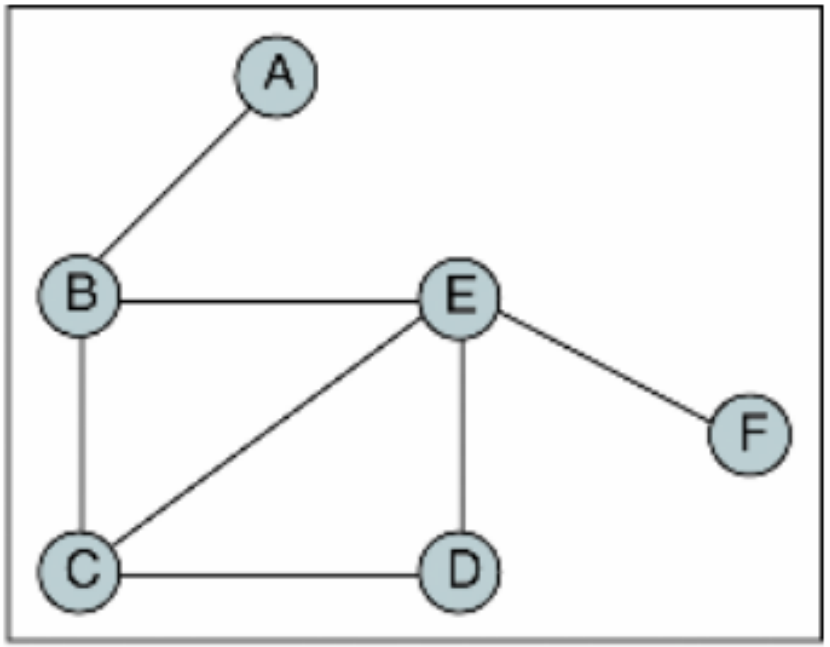
- 현재 정점에 연결되어 있는 정점을 Queue에 넣고 탐색
- 탐색한 정점은 다시 탐색하지 않음 (visit배열을 활용)
- Queue이용 (재귀도 O)
- 인접 행렬 사용: O(V^2)
  인접 리스트 사용: O(V+E)

BFS?



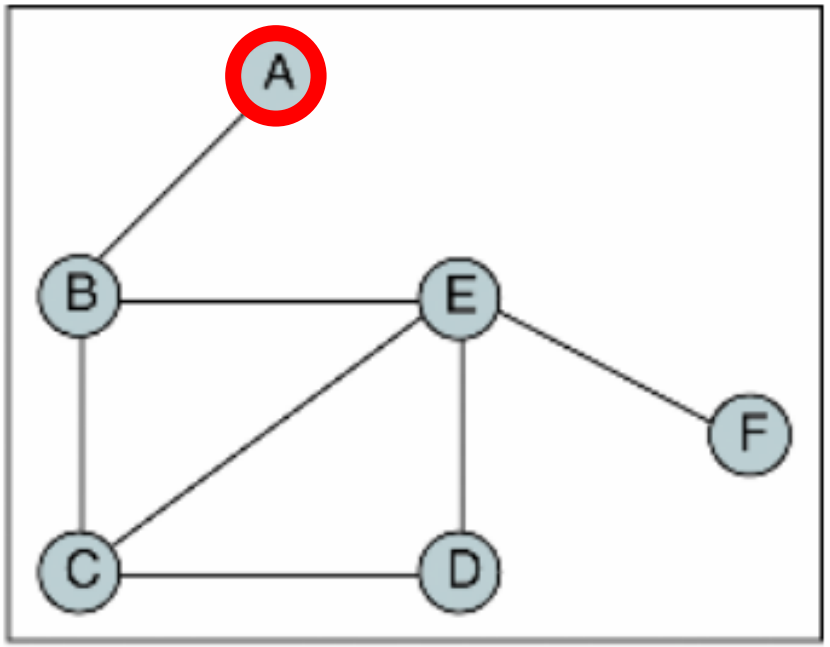
- A부터
- 번호가 작은 노드부터 탐색

BFS?



Queue

BFS?

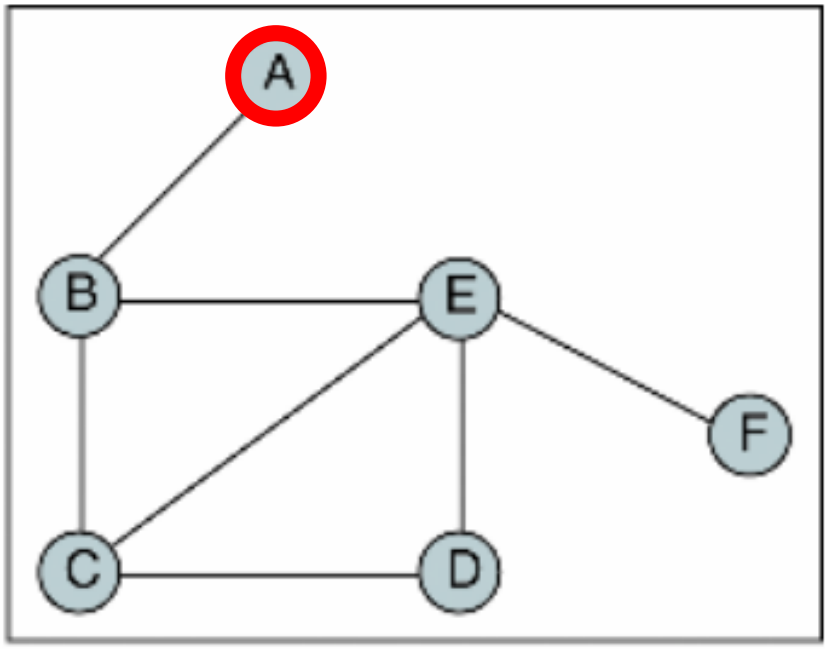


Queue

A				

	visit
A	1
B	0
C	0
D	0
E	0
F	0

BFS?



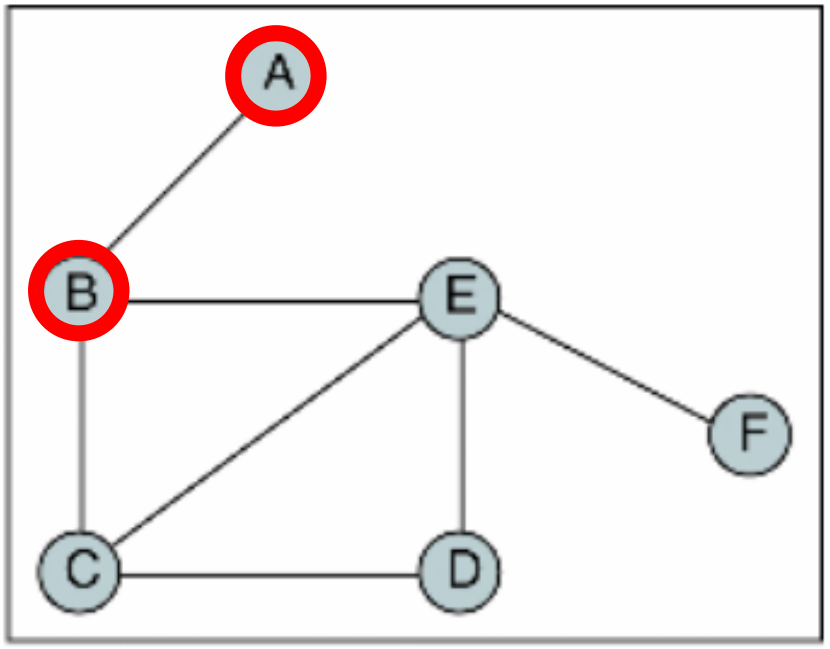
Queue

B				

A

	visit
A	1
B	1
C	0
D	0
E	0
F	0

BFS?



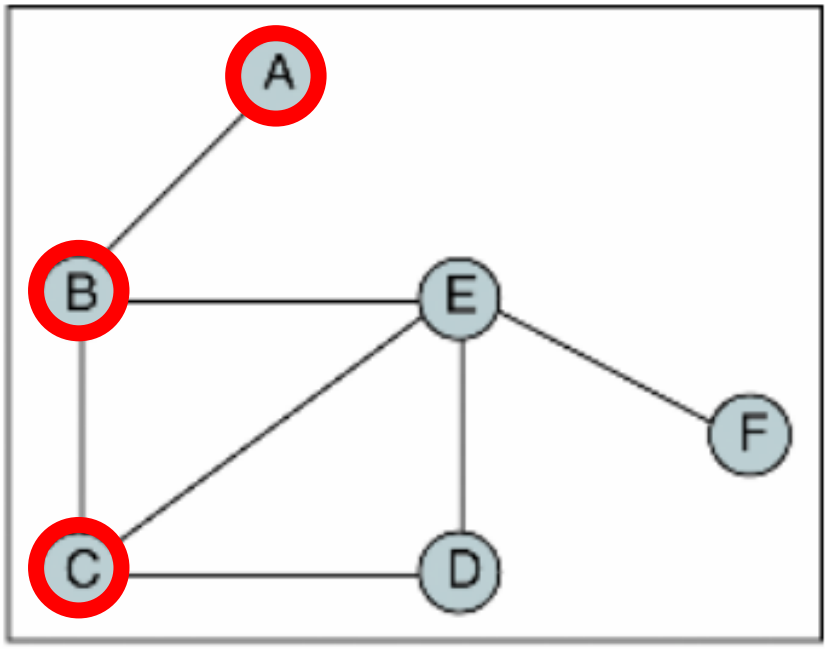
Queue

C	E			

A → B

	visit
A	1
B	1
C	1
D	0
E	1
F	0

BFS?



Queue

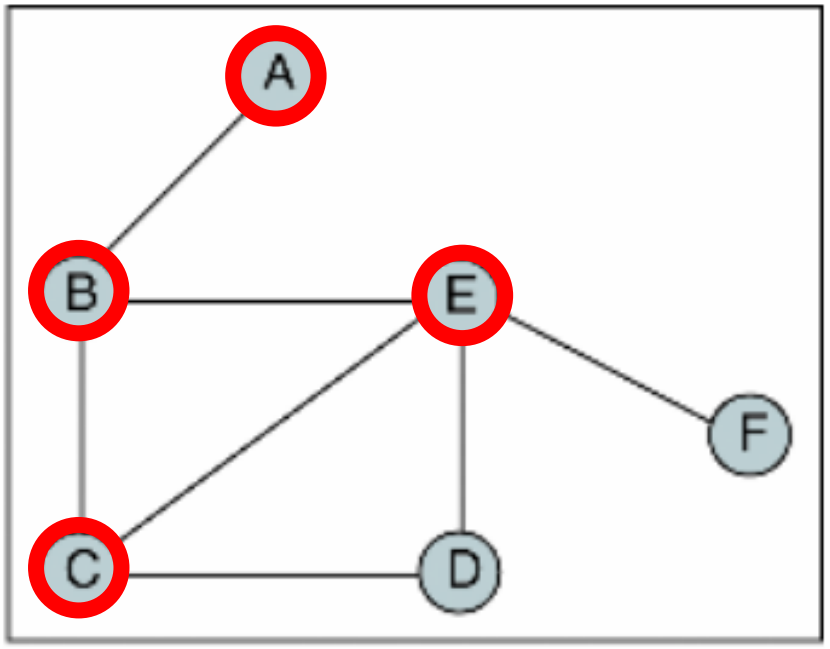
E	D			

A → B → C

visit

A	1
B	1
C	1
D	1
E	1
F	0

BFS?



Queue

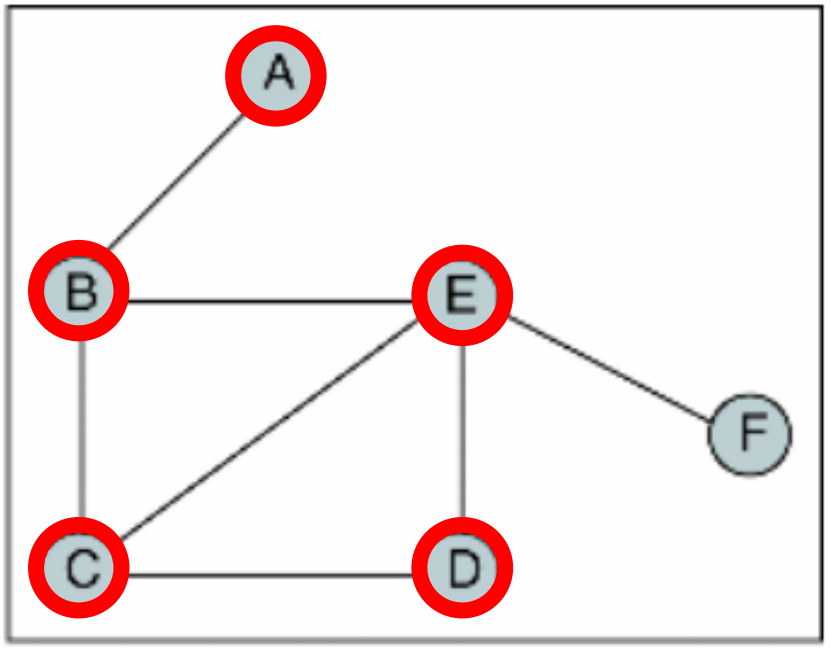
D	F			

A → B → C → E

visit

A	1
B	1
C	1
D	1
E	1
F	1

BFS?



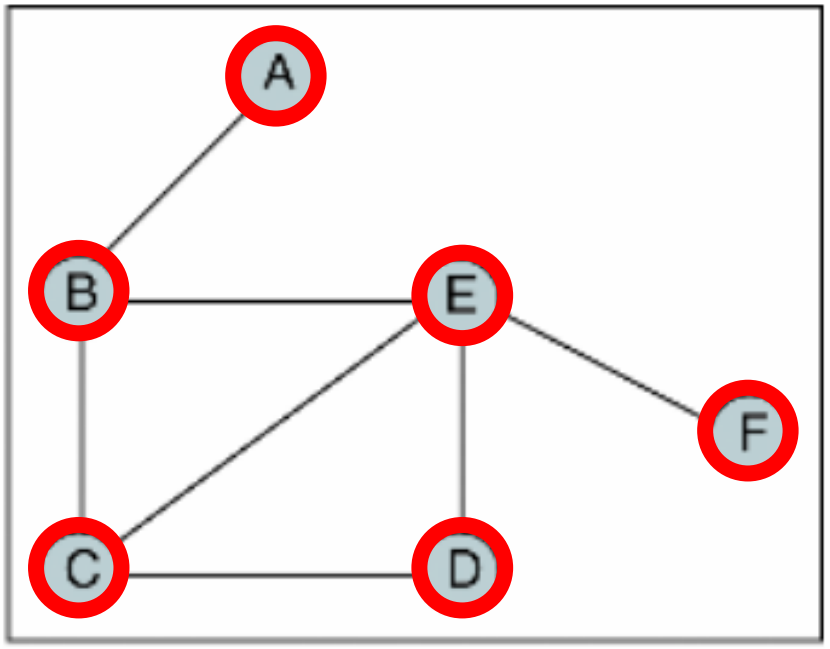
Queue

F				

A → B → C → E → D

	visit
A	1
B	1
C	1
D	1
E	1
F	1

BFS?



Queue

A → B → C → E → D → F

	visit
A	1
B	1
C	1
D	1
E	1
F	1

연습 문제

BOJ 1260
DFS와 BFS

BOJ 1260
DFS와 BFS

2주차2주차2주차2주차2주차2주차2주차2주차.cpp

```cpp
int main()
{
    for(int i=0; i<n; i++) // 작은 번호부터
        sort(vec[i].begin(),vec[i].end());
    dfs(v);
    printf("\n");
    bfs();
}

void dfs(int x)
{
    visit_dfs[x] = true;
    printf("%d ",x);
    for(int i=0; i<vec[x].size(); i++)
        if(!visit_dfs[vec[x][i]])
            dfs(vec[x][i]);
}

void bfs(){
    q.push(v);
    visit_bfs[v] = true;
    while(!q.empty()){
        int now = q.front();
        q.pop();
        printf("%d ",now);
        for(int i=0; i<vec[now].size(); i++){
            if(!visit_bfs[vec[now][i]]){
                q.push(vec[now][i]);
                visit_bfs[vec[now][i]] = true;
    }}}
}
```

ALOHA
The algorithm club.

BOJ 1260
DFS와 BFS

2주차2주차2주차2주차2주차2주차2주차.cpp

```cpp
int main()
{
    for(int i=0; i<n; i++) // 작은 번호부터
        sort(vec[i].begin(),vec[i].end());
    dfs(v);
    printf("\n");
    bfs();
}
```

- 작은 번호부터 방문
- dfs에는 시작점을 인자로

ALOHA
The algorithm club.

BOJ 1260
DFS와 BFS

```
10  void dfs(int x)
11  {
12      visit_dfs[x] = true;
13      printf("%d ",x);
14      for(int i=0; i<vec[x].size(); i++)
15          if(!visit_dfs[vec[x][i]])
16              dfs(vec[x][i]);
17  }
```

- 함수 호출 후 바로 visit 체크
- 연결된 다른 node중 visit이 안된 node를 재귀로 호출

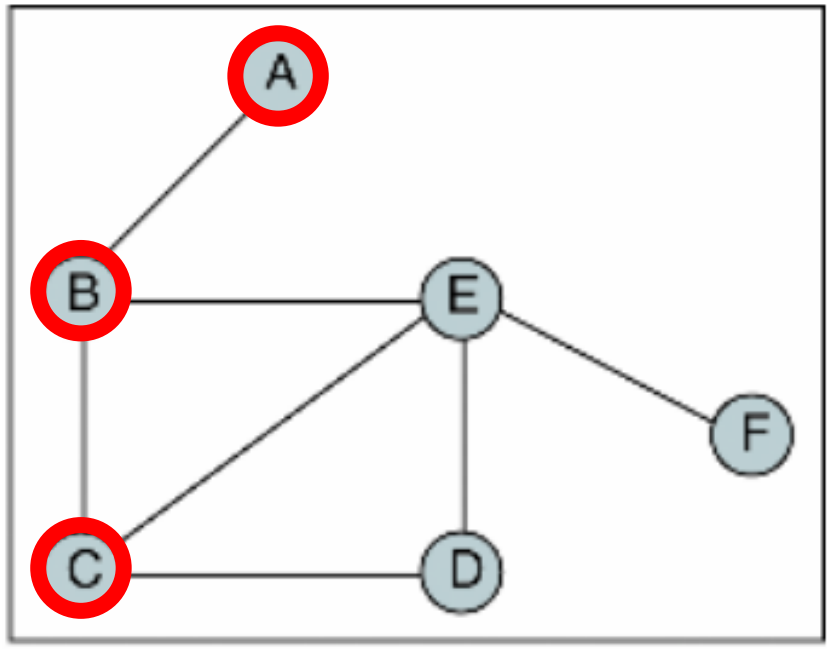
BOJ 1260
DFS와 BFS

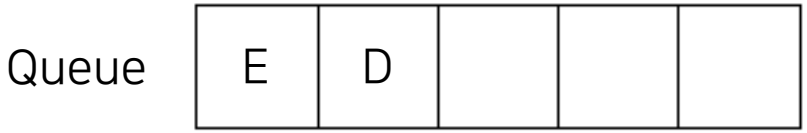
- dfs와는 다르게 push전 visit 체크
- Queue를 pop할 때 visit을 체크하면 중복 방문이 일어날 수 있음

```
19 void bfs(){
20     q.push(v);
21     visit_bfs[v] = true;
22     while(!q.empty()){
23         int now = q.front();
24         q.pop();
25         printf("%d ",now);
26         for(int i=0; i<vec[now].size(); i++){
27             if(!visit_bfs[vec[now][i]]){
28                 q.push(vec[now][i]);
29                 visit_bfs[vec[now][i]] = true;
30     }}}
31 }
```

BOJ 1260
DFS와 BFS

- dfs와는 다르게 push전 visit 체크
- Queue를 pop할 때 visit을 체크하면
  중복 방문이 일어날 수 있음





A → B → C

now = E
D가 다시 push

ALOHA
The algorithm club.

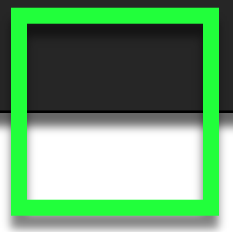
연습 문제

BOJ 11724
연결 요소의 개수

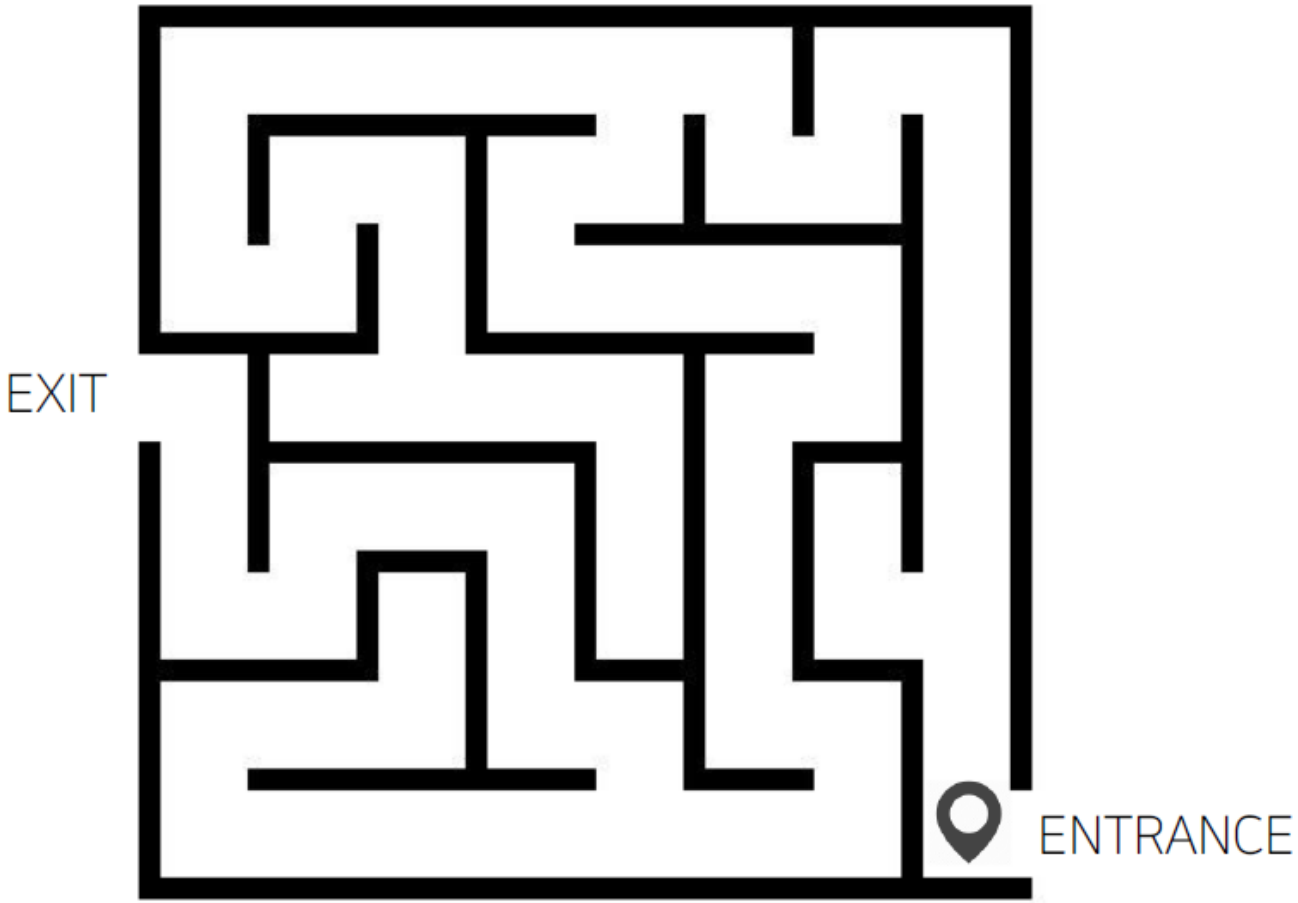
== 그래프의 개수

#2

미로찾기



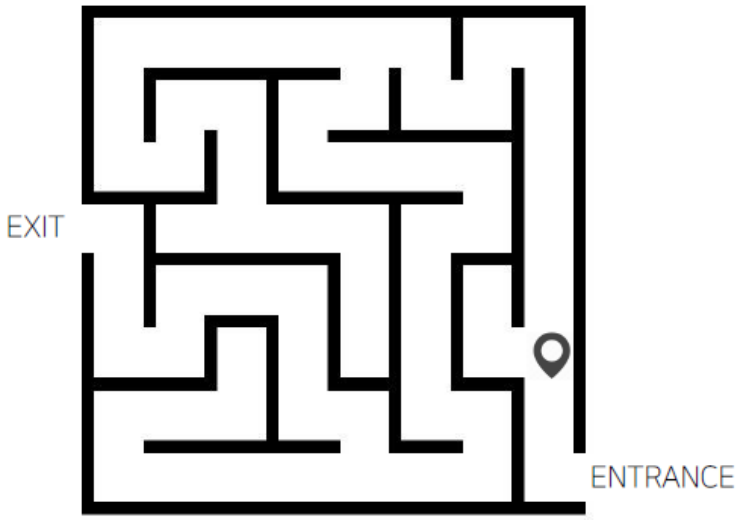
미로찾기?

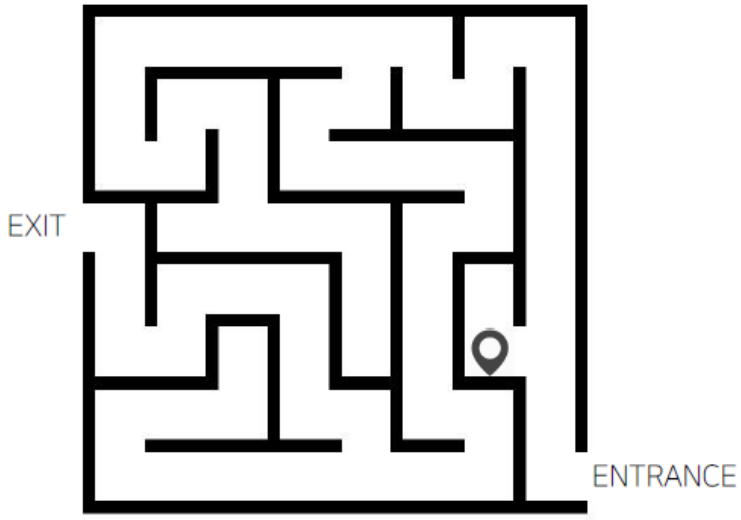


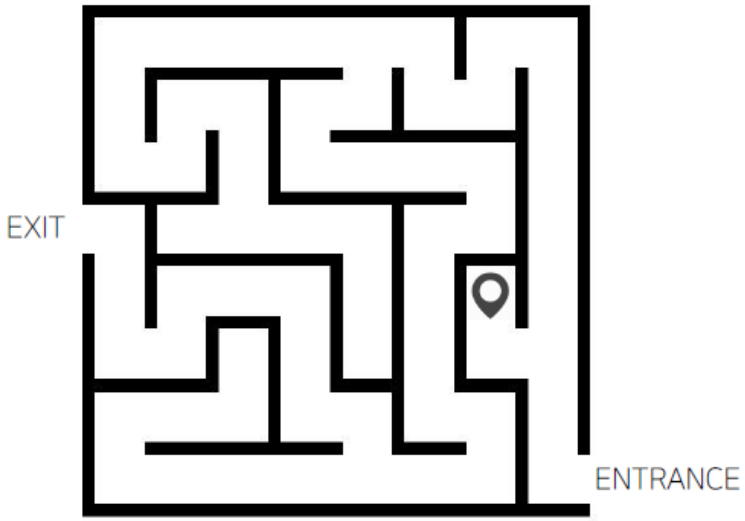
↑ (작년 창호 PPT 댕꿀)

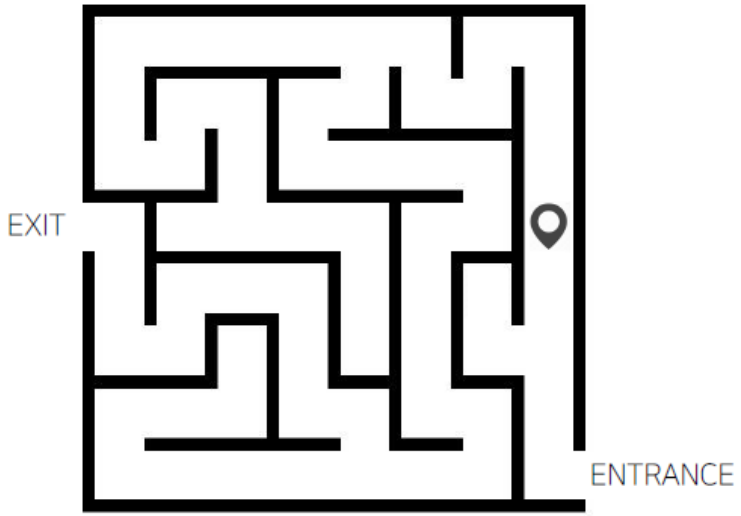
ALOHA
The algorithm club.

미로찾기?









- Node번호 대신 x, y좌표
- dfs는 재귀
- bfs는 queue에 pair<int, int>를 인자로 사용
- ★ 거리에 대한 배열 ★

미로찾기 // 2178 – 미로 탐색

입력

첫째 줄에 두 정수 N, M(2 ≤ N, M ≤ 100)이 주어진다. 다음 N개의 줄에는 M개의 정수로 미로가 주어진다. 각각의 수들은 **붙어서** 입력으로 주어진다.

출력

첫째 줄에 지나야 하는 최소의 칸 수를 출력한다. 항상 도착위치로 이동할 수 있는 경우만 입력으로 주어진다.

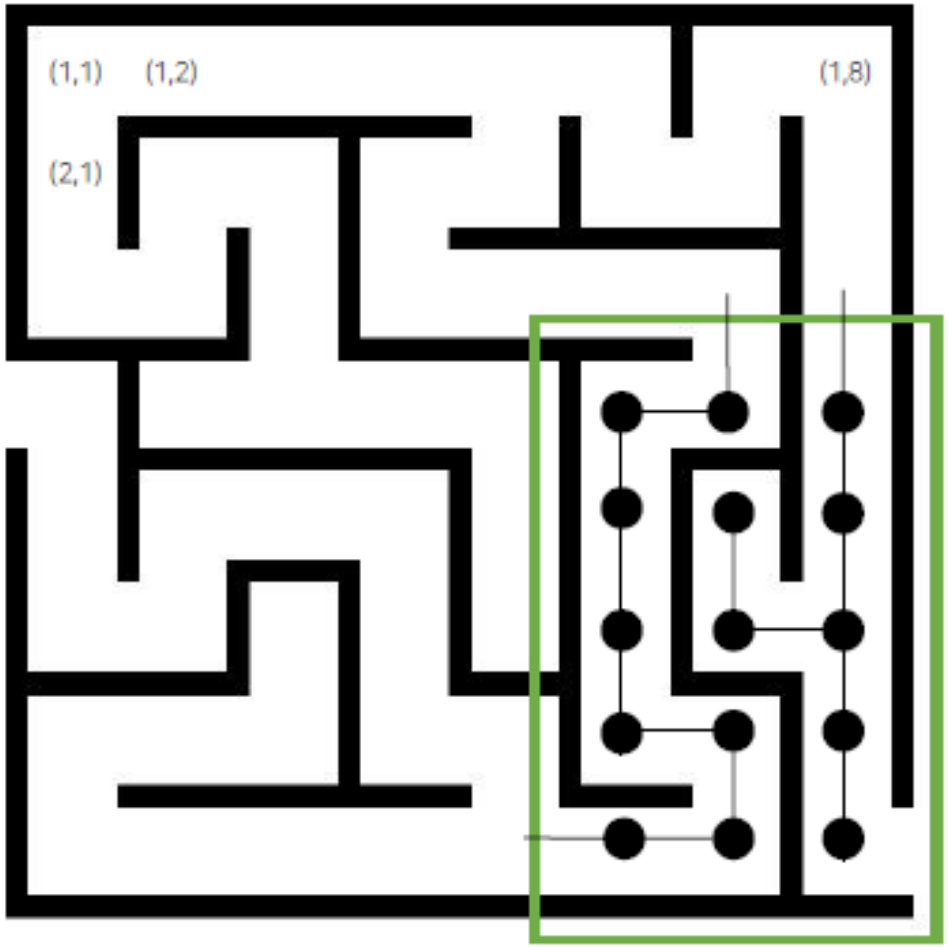
예제 입력 1 복사

```
4 6
101111
101010
101011
111011
```

예제 출력 1 복사

```
15
```

미로찾기

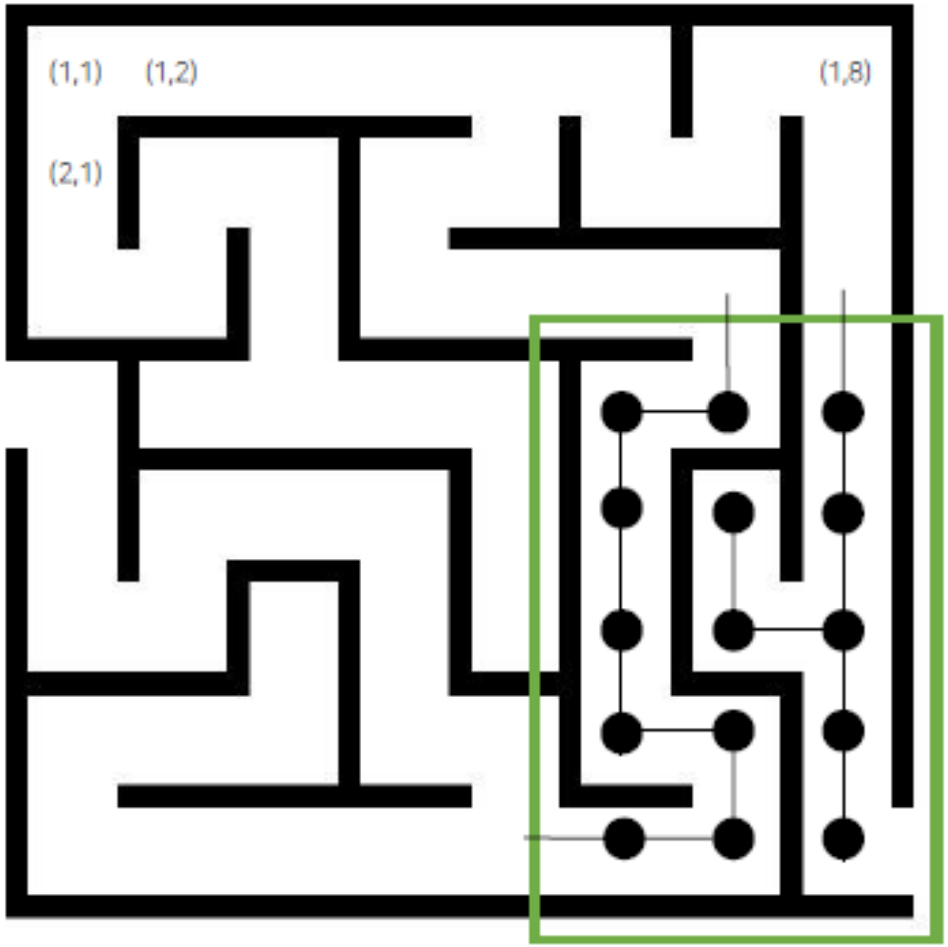


- scanf("%1d",&MAP[i][j]);
  > 한 자리 수 단위로 scanf

- MAP형식으로 주어지기에
  vector보단 array사용

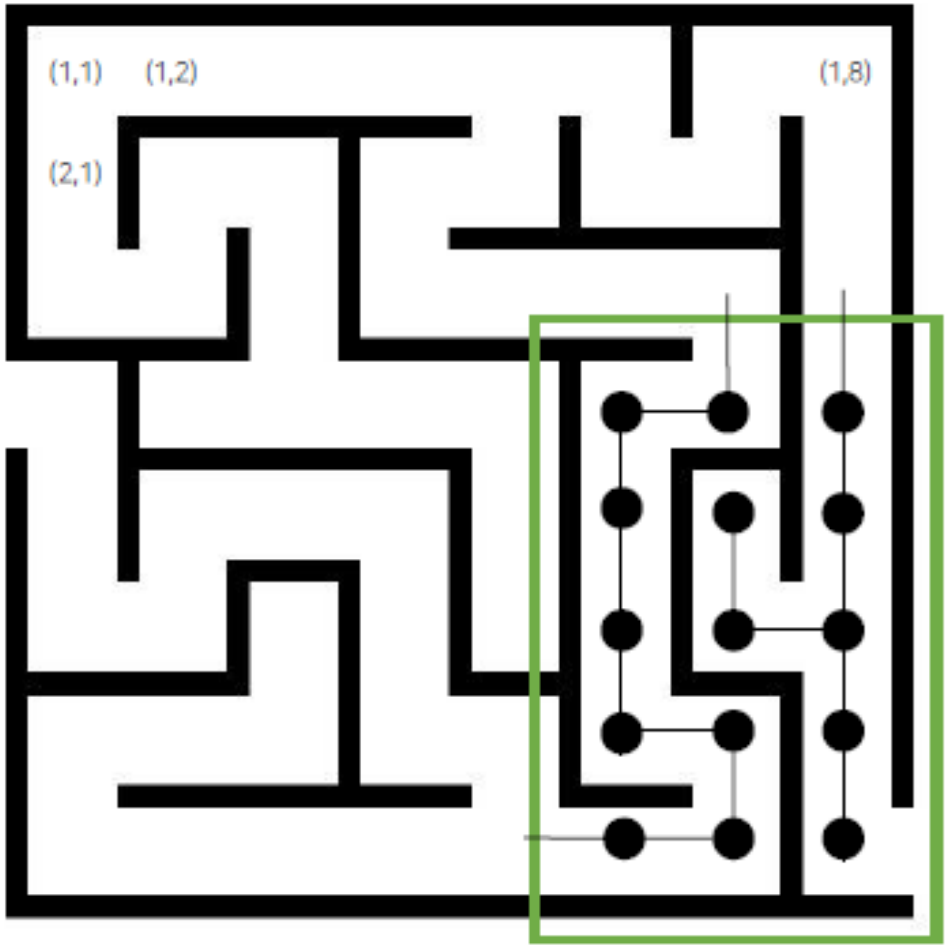
ALOHA
The algorithm club.

미로찾기




```
1  void dfs(int x)
2  {
3      visit_dfs[x] = true;
4      printf("%d ",x);
5      for(int i=0; i<vec[x].size(); i++)
6          if(!visit_dfs[vec[x][i]])
7              dfs(vec[x][i]);
8  }
```

미로찾기




```
1  void dfs(int x)
2  {
3      visit_dfs[x] = true;
4      printf("%d ",x);
5      for(int i=0; i<vec[x].size(); i++)
6          if(!visit_dfs[vec[x][i]])
7              dfs(vec[x][i]);
8  }
```

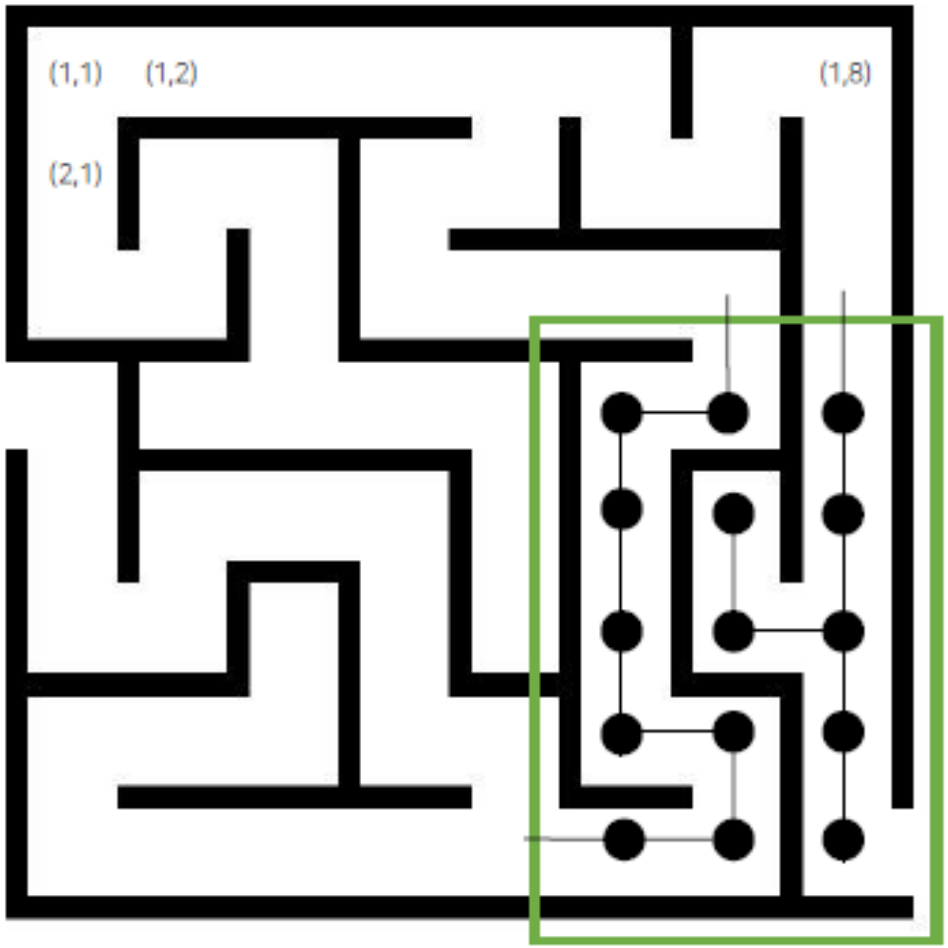
↑이거를
이렇게 ↓

[*] 2주차2주차2주차2주차2주차2주차2주차2주차.cpp

```
1   int dx[4] = {-1,1,0,0};
2   int dy[4] = {0,0,-1,1};
3   void dfs(int x, int y){
4       visit_dfs[x] = true;
5       printf("%d ",x);
6       for(int i=0; i<4; i++){
7           int nx = x + dx[i];
8           int ny = y + dy[i];
9           if(visit[ny][nx] == false){
10              dfs(ny,nx);
11          }
12      }
13  }
```

ALOHA
The algorithm club.

미로찾기



↓이렇게 하면 BFS로도 할 수 있다
queue<pair<int,int> > q;

2주차2주차2주차2주차2주차2주차2주차2주차.cpp

```cpp
void bfs(){
    q.push({1,1});
    while(!q.empty()){
        int size = q.size();
        int x = q.front().first;
        int y = q.front().second;
        q.pop();
        visit[x][y] = true;
        printf("%d ",now);
        for(int i=0; i<size; i++){
            for(int j=0; j<4; j++){
                int nx = x+nx[i];
                int ny = y+ny[i];
                if(!visit_bfs[ny][nx]){
                    q.push({nx,ny});
    }}}}
}
```

미로찾기?

```
4   for(int i=0; i<n; i++){
5       for(int j=0; j<m; j++)
6           dis[i][j] = 987654321;
7   }
```

- 단순한 방문은 visit배열만 활용

- **최단거리**를 구하는 문제의 경우 distance배열에 INF값을 채운 후 작은 값으로 교체

  ex) if(dis[next] > dis[now] +1)
  dis[next] = dis[now]+1;

1	0	1	1	1	1
1	0	1	0	1	0
1	0	1	0	1	1
1	1	1	0	1	1

미로찾기?

+) array방문 팁

-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1

```
1  #include<string.h>
2  main(){
3      memset(MAP,-1,sizeof(MAP));
4
```

미로찾기?

+) array방문 팁

-1	-1	-1	-1	-1	-1	-1	-1
-1	1	0	1	1	1	1	-1
-1	1	0	1	0	1	0	-1
-1	1	0	1	0	1	1	-1
-1	1	1	1	0	1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1

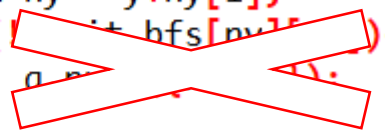
```cpp
#include<string.h>
main(){
    memset(MAP,-1,sizeof(MAP));

```

- 정보 받고

2주차2주차2주차2주차2주차2주차2주차2주차.cpp

```cpp
void bfs(){
    q.push({1,1});
    while(!q.empty()){
        int size = q.size();
        int x = q.front().first;
        int y = q.front().second;
        q.pop();
        visit[x][y] = true;
        printf("%d ",now);
        for(int i=0; i<size; i++){
            for(int j=0; j<4; j++){
                int nx = x+nx[i];
                int ny = y+ny[i];
                if(
    }}}}
}
```



ALOHA
The algorithm club.

미로찾기?

+) array방문 팁

-1	-1	-1	-1	-1	-1	-1	-1
-1	1	0	1	1	1	1	-1
-1	1	0	1	0	1	0	-1
-1	1	0	1	0	1	1	-1
-1	1	1	1	0	1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1

// dis[1][1] = 1;

```cpp
#include<string.h>
main(){
    memset(MAP,-1,sizeof(MAP));

```

- 정보 받고

2주차2주차2주차2주차2주차2주차2주차2주차.cpp

```cpp
void bfs(){
    q.push({1,1});
    while(!q.empty()){
        int size = q.size();
        int x = q.front().first;
        int y = q.front().second;
        q.pop();
        visit[x][y] = true;
        printf("%d ",now);
        for(int i=0; i<size; i++){
            for(int j=0; j<4; j++){
                int nx = x+nx[i];
                int ny = y+ny[i];
                If(MAP[ny][nx] == 1 && dis[ny][nx] > dis[y][x]+1){
                    dis[ny][nx] = dis[y][x]+1;
                    p.push({nx,ny});
    }}}}
}
```

ALOHA
The algorithm club.

미로찾기?

+) queue의 인자를 구조체로

```
10 typedef struct P{
11     P(int a, int b, int c):x(a), y(b), z(c){} // q.push(P(1,1,1));
12     int x,y,z;
13 }P;
14 queue<P> q;
```

```
27     q.push(P(1,1,0));
```

```
29         P now = q.front();
```

```
32             int nx = now.x+dx[i];
33             int ny = now.y+dy[i];
```

연습문제

BOJ 2178
미로탐색

BOJ 7576
토마토

과제야

7562 나이트의 이동
2667 단지번호붙이기
1726 로봇
2206 벽 부수고 이동하기

(1726 이나 2206중에 하나만 풀어도 성공했다)

수고했어요