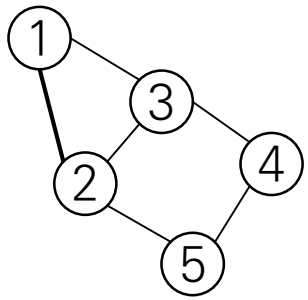
인접행렬

- 무향 그래프



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

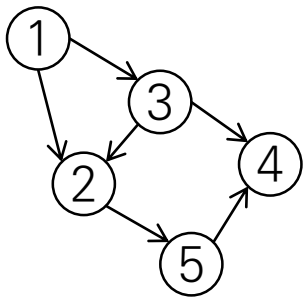
인접은 1, 아닌 경우 0

그래프는 간선 양쪽의 노드 번호로 표현 가능.

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

```
scanf("%d %d", &n1, &n2);
M[n1][n2] = 1;
M[n2][n1] = 1;
```

- 방향성 그래프



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

도착

출발

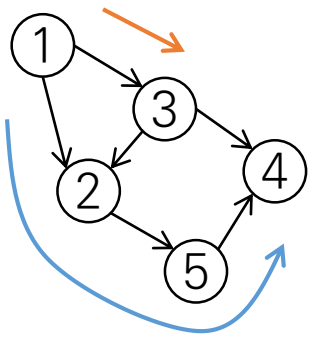
출발 -〉 도착 노드로 주어지는 경우

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

```
scanf("%d %d", &n1, &n2);
M[n1][n2] = 1;
```

탐색

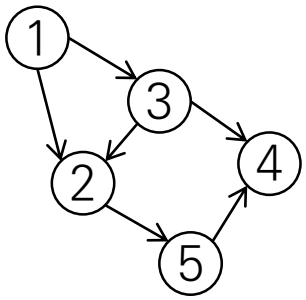
- 깊이 우선 탐색 (DFS)
  - 2개 이상의 선택이 가능할 때, 정해진 순서에 따라 다음 노드 선택.
  - 더 이상 갈 수 없으면 가장 가까운 이전 갈림길에서 다른 방향 선택.
    - 지나온 경로를 저장해야 함.



1 -> 2 -> 5 -> 4 -> 3

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

- 재귀를 사용한 DFS
  - 재귀의 각 단계가 방문중인 노드 번호를 저장.
  - 방문한 노드에서 방문하지 않은 인접 노드 중 번호가 작은 곳으로 이동.

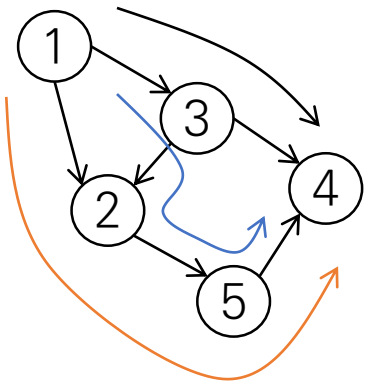



```
DFS(n)
    V[n] = 1                    // 방문 표시
    visit(n)                    // 노드에 대해 처리할 일
    for i : 1 -> N
        if( M[n][i] == 1 && V[i] == 0 )
            DFS(i) // 인접하고 방문하지 않은 노드로 이동
```

- 반복 구조의 DFS
  - 지나온 노드를 스택에 저장.
  - 재귀를 사용한 DFS가 중심.

▪ DFS 응용

- 1에서 4번 노드에 도착할 수 있는 경로의 수 찾기.



가능한 경로

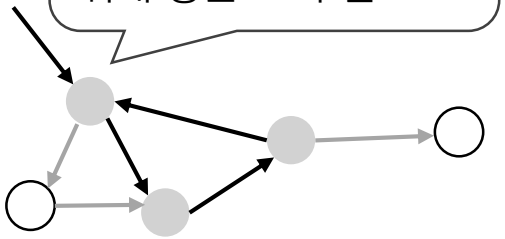
1 -> 2 -> 5 -> 4

1 -> 3 -> 2 -> 5 -> 4

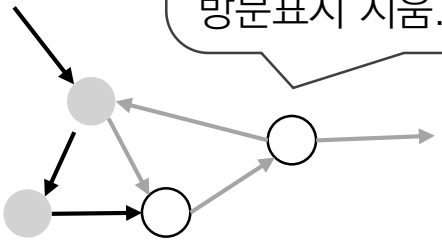
1 -> 3 -> 4

접근하는 경로가 다르면 중복을 허용해야 함.

되돌아 가는 것을 막기 위해 방문 표시 필요.



다른 경로에서의 접근을 위해 갈림길로 되돌아갈 때는 방문표시 지움.



- 1에서 4번 노드에 도착할 수 있는 경로의 수 찾기.

```
// n = 1, k = 4

DFS(n, k)
    if( n == k )
        cnt++;
    else
        V[n] = 1;                   // 방문 표시
        for i : 1 -> N
            if( M[n][i] == 1 && V[i] == 0 )
                DFS(i, k);          // 인접하고 방문하지 않은 노드로 이동
        V[n] = 0;                   //  방문 표시 삭제
```

for문 밖에서 처리

- 1에서 4번 노드에 도착할 수 있는 최단 거리 찾기.
  - 모든 경로를 찾는 것이 기본.
  - 지나온 간선의 수를 인자로 전달.

```
// 호출 조건 : n = 1, k = 4, e = 0, min = INF


DFS(n, k, e)
    if( n == k )
        if( min 〉 e )
            min = e;
    else
        V[n] = 1;                   // 방문 표시
        for i : 1 -〉 N
            if( M[n][i] == 1 && V[i] == 0 )
                DFS(i, k, e+1);          // 인접하고 방문하지 않은 노드로 이동
        V[n] = 0;                   //  방문 표시 삭제
```

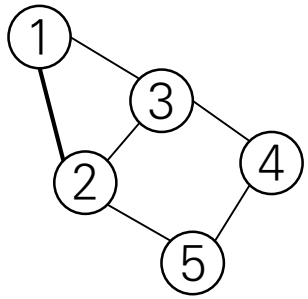
- 1에서 4번 노드에 도착할 수 있는 최단 거리 찾기.
  - 호출을 줄이려면 중단 조건 추가.

```
// 호출 조건 : n = 1, k = 4, e = 0, min = INF

DFS(n, k, e)
    if( n == k )
        if( min 〉 e )
            min = e;
    else if( e 〉= min )          // 현재까지 거리가 기존의 min보다 크면 다른 경로
        return;
    else
        V[n] = 1;                 // 방문 표시
        for i : 1 -〉 N
            if( M[n][i] == 1 && V[i] == 0 )
                DFS(i, k, e+1);          // 인접하고 방문하지 않은 노드로 이동
        V[n] = 0;                 //  방문 표시 삭제
```

연습

- 다음 그래프를 DFS로 탐색하고 방문 순서를 출력하시오.



5 6
1 2 1 3 3 2 3 4 2 5 5 4