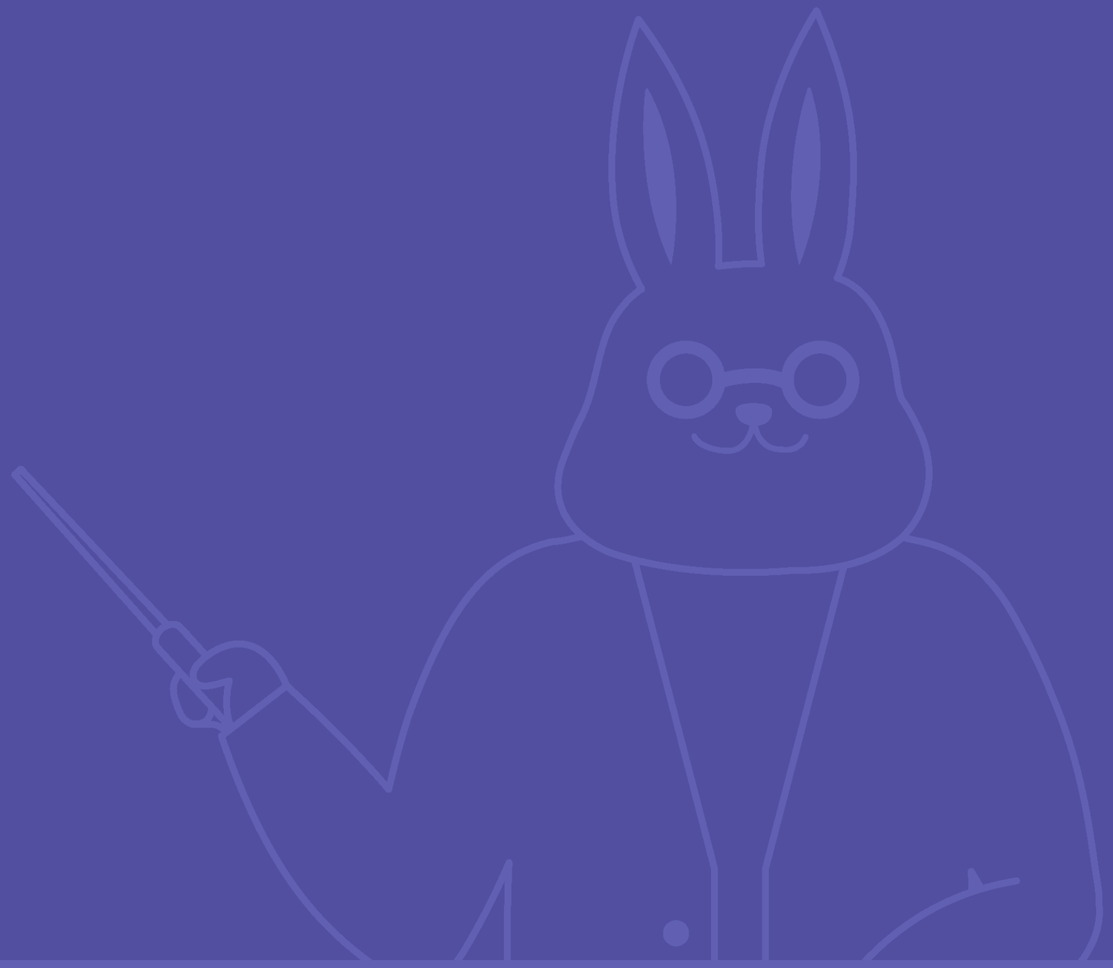
# 알고리즘의 정석 II

## 3장 그래프 알고리즘





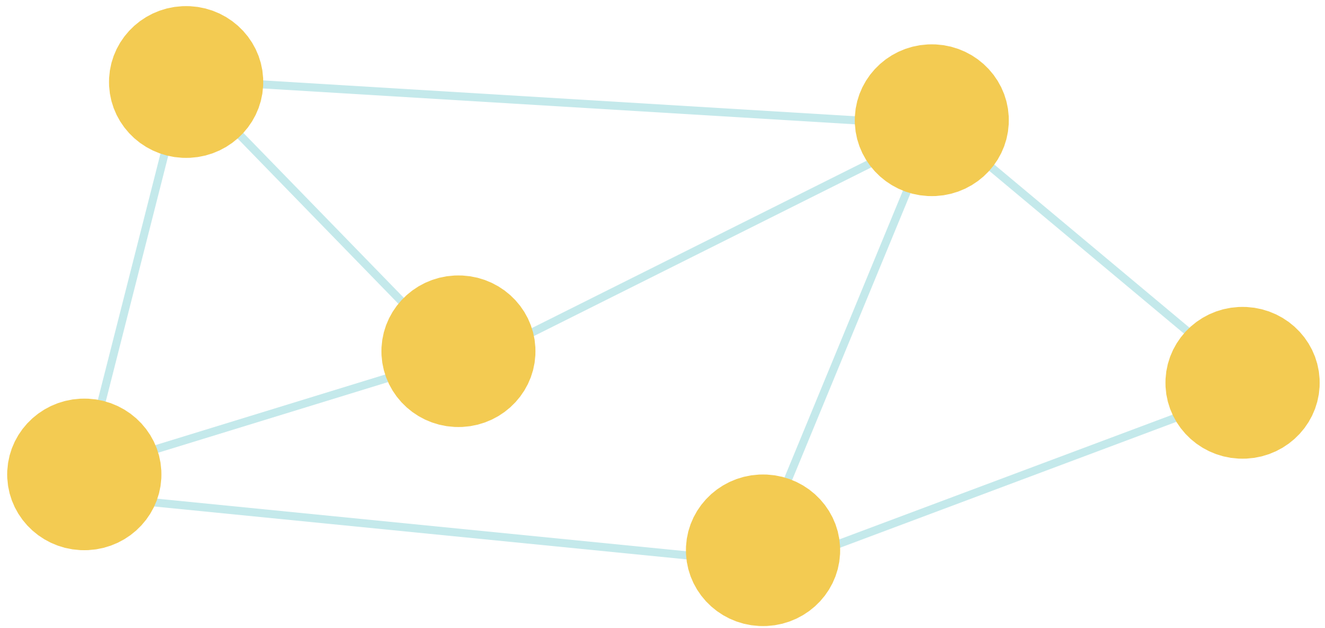
/* elice */

# Contents

# 01 그래프의 개념

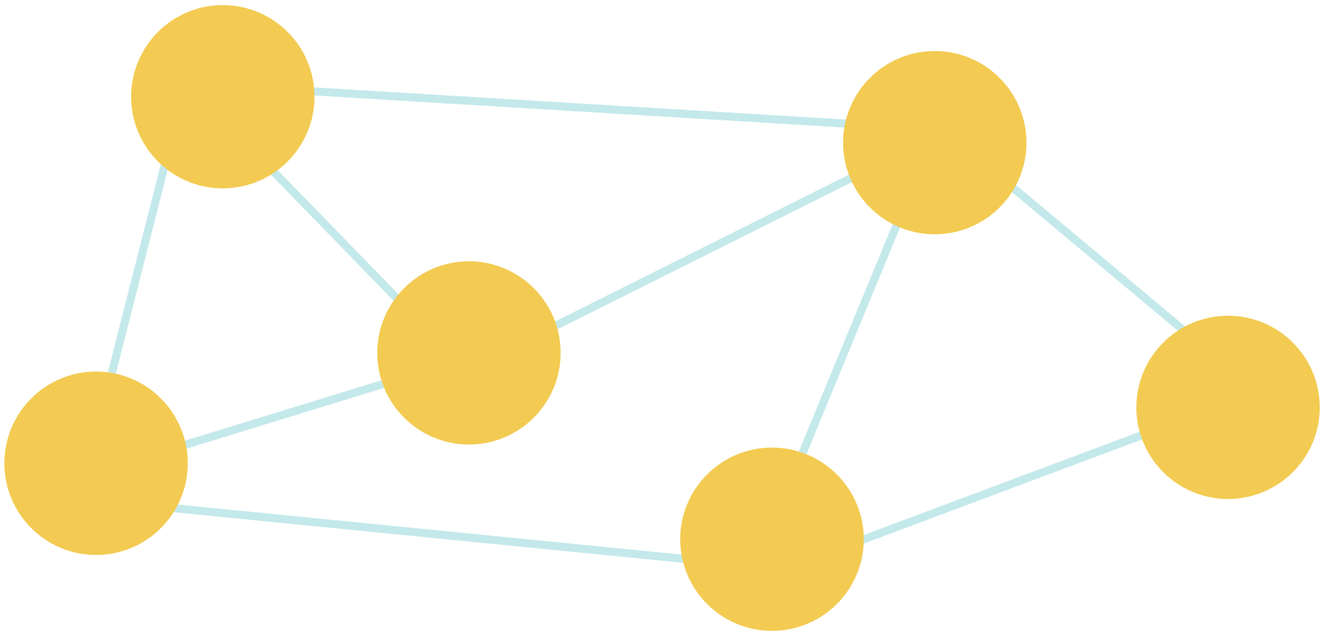
## 그래프란?

**정점**과 **간선**으로 이루어진 자료구조
**정점간의 관계**를 조직도로 표현합니다



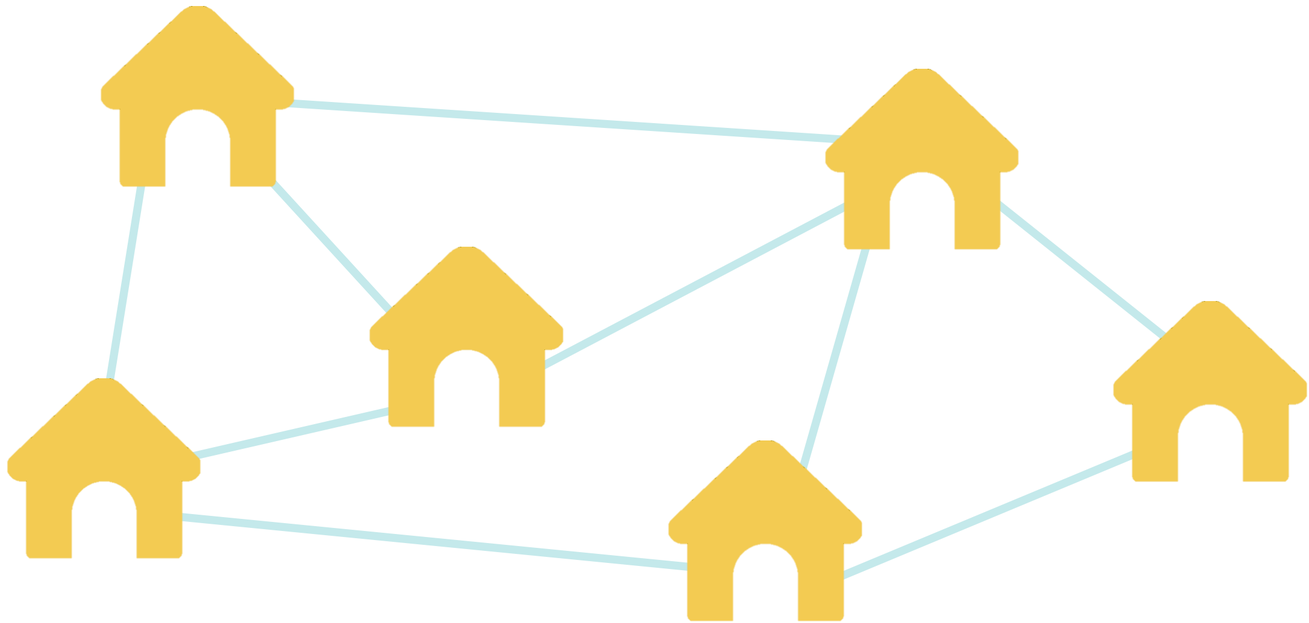
## 정점이란?

**정점**은 **여러가지 특성**을 가질 수 있는 **객체**를 의미합니다



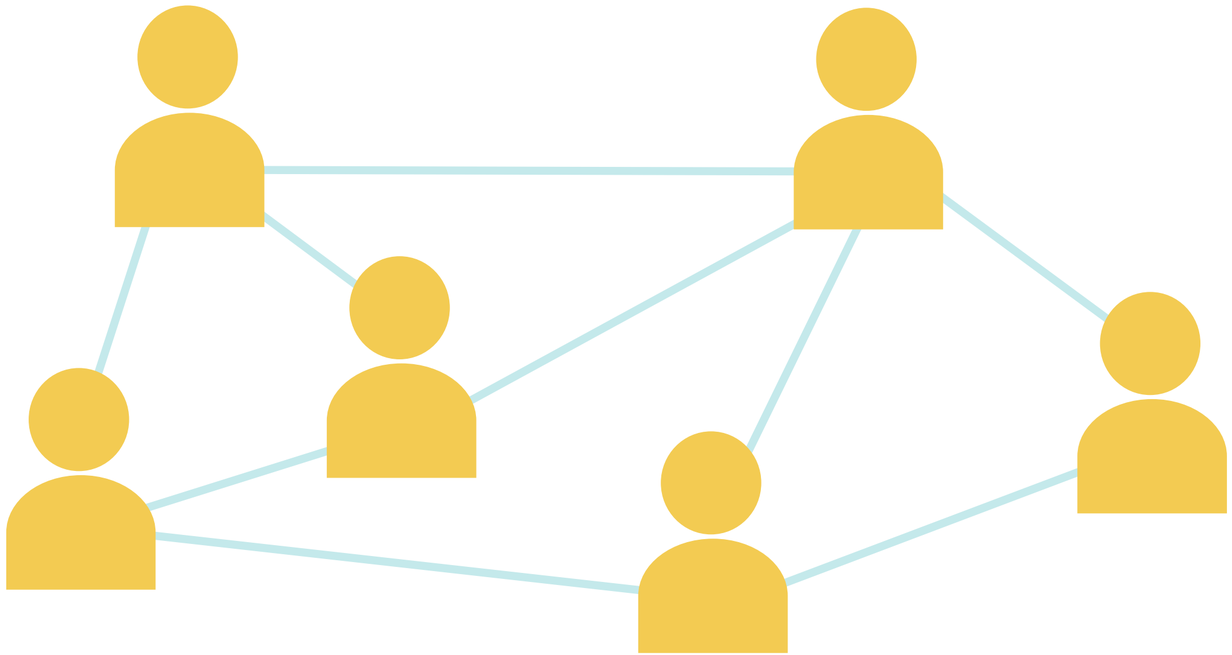
## 정점이란?

**정점**은 **여러가지 특성**을 가질 수 있는 **객체**를 의미합니다



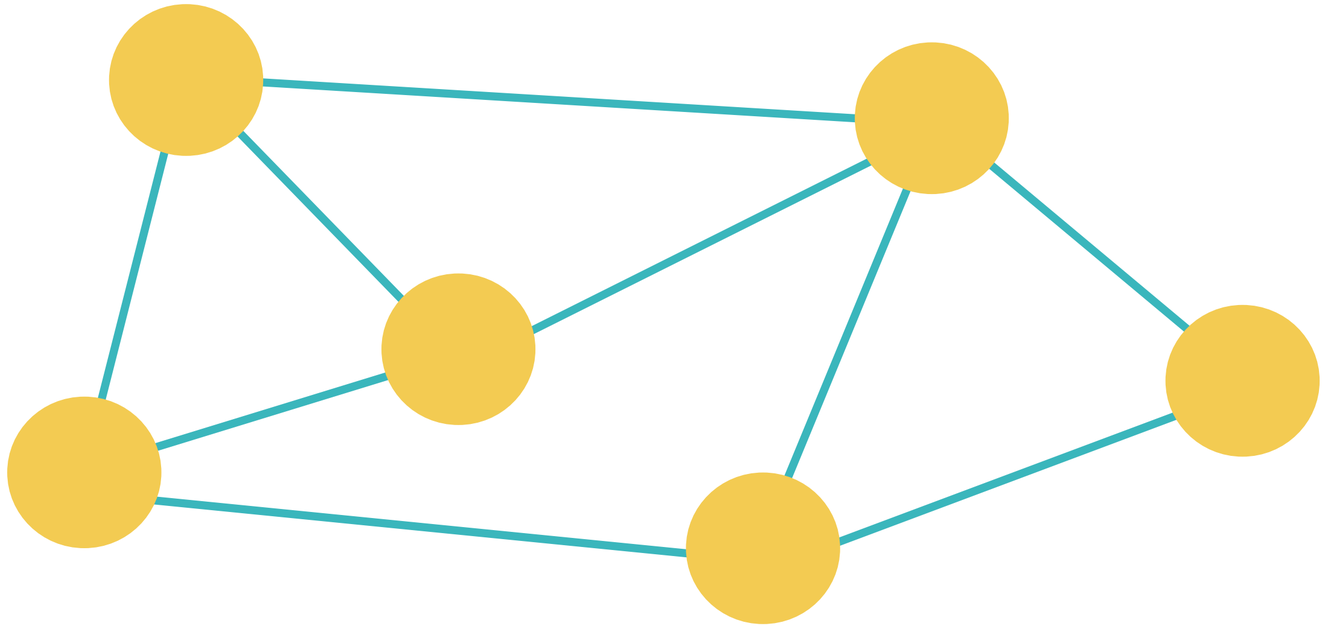
## 정점이란?

**정점**은 **여러가지 특성**을 가질 수 있는 **객체**를 의미합니다



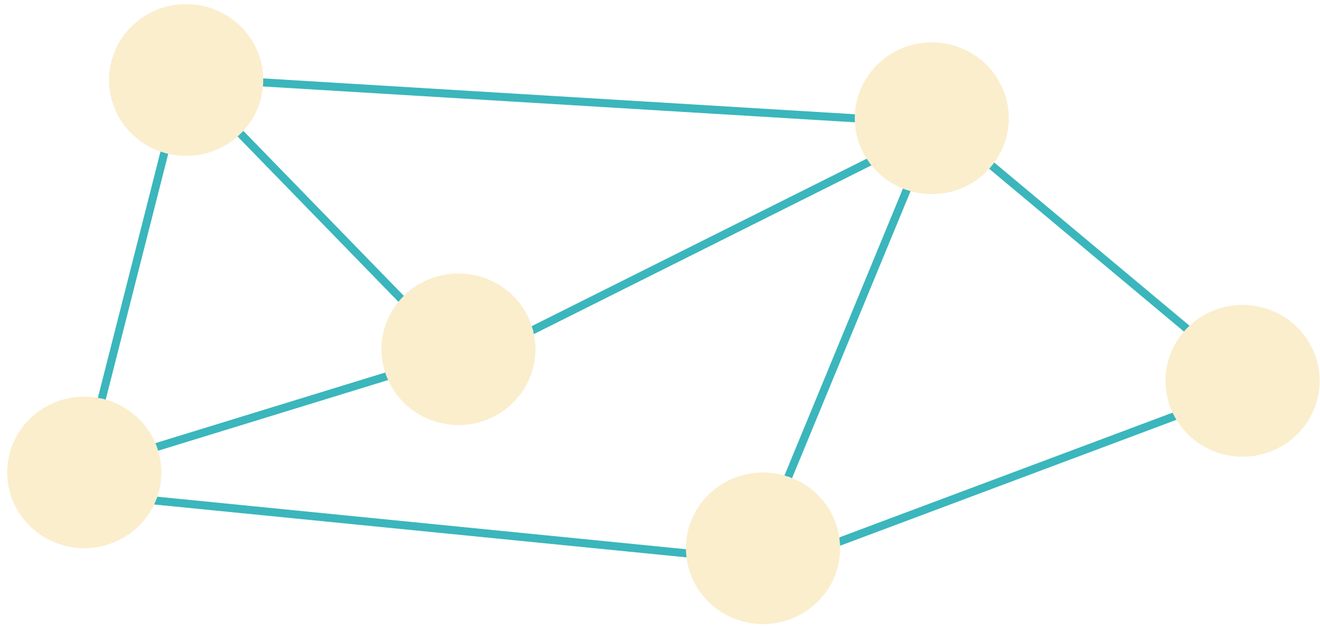
## 간선이란?

**간선**은 **정점들 간의 관계**를 의미합니다.



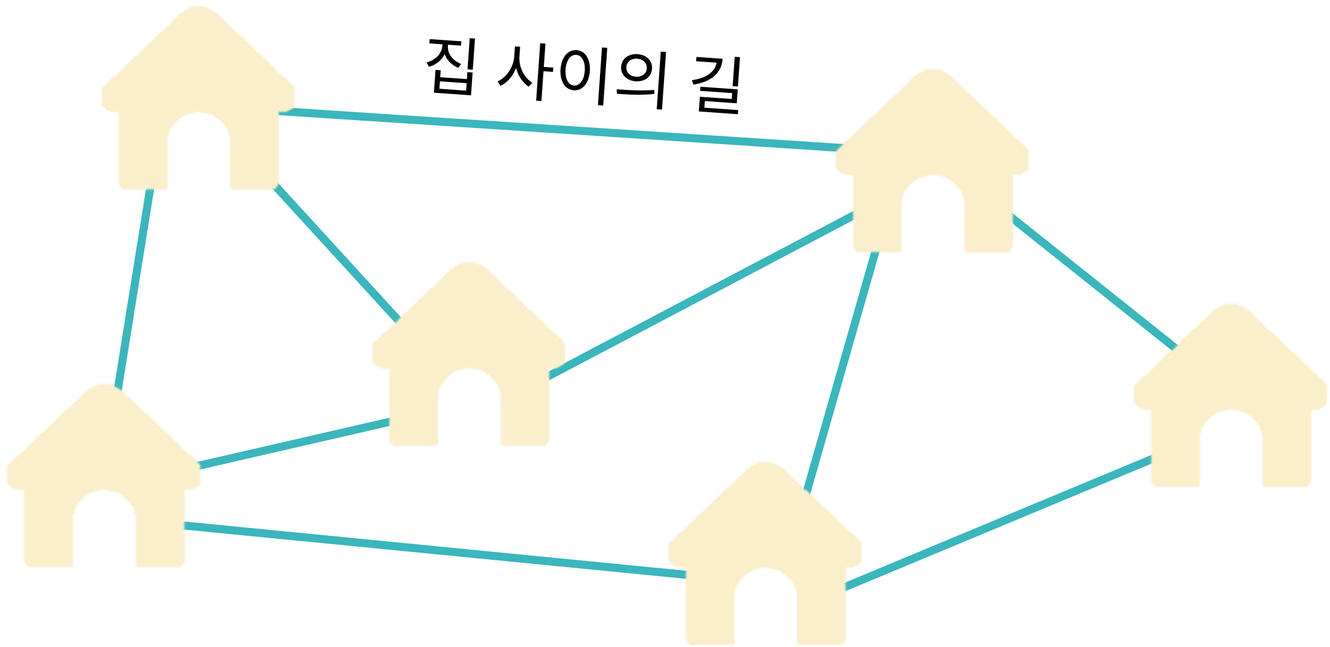
## 간선이란?

**간선**은 **정점들 간의 관계**를 의미합니다.



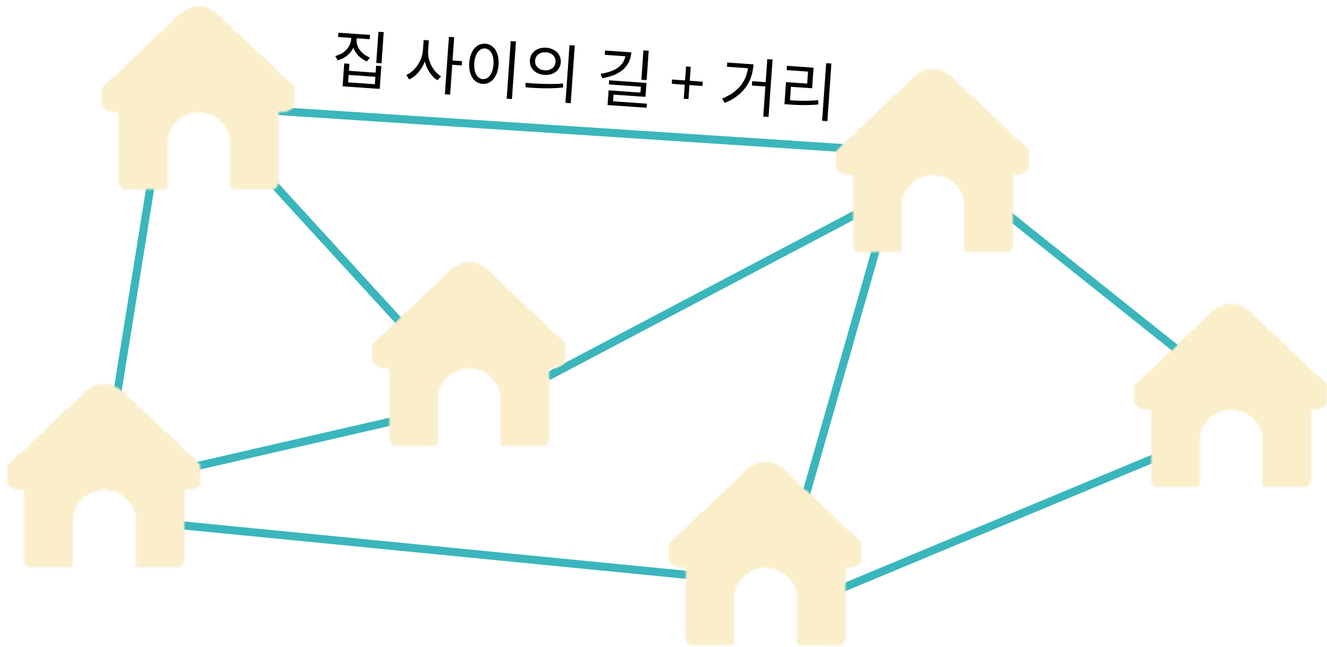
## 간선이란?

**간선**은 **정점들 간의 관계**를 의미합니다.



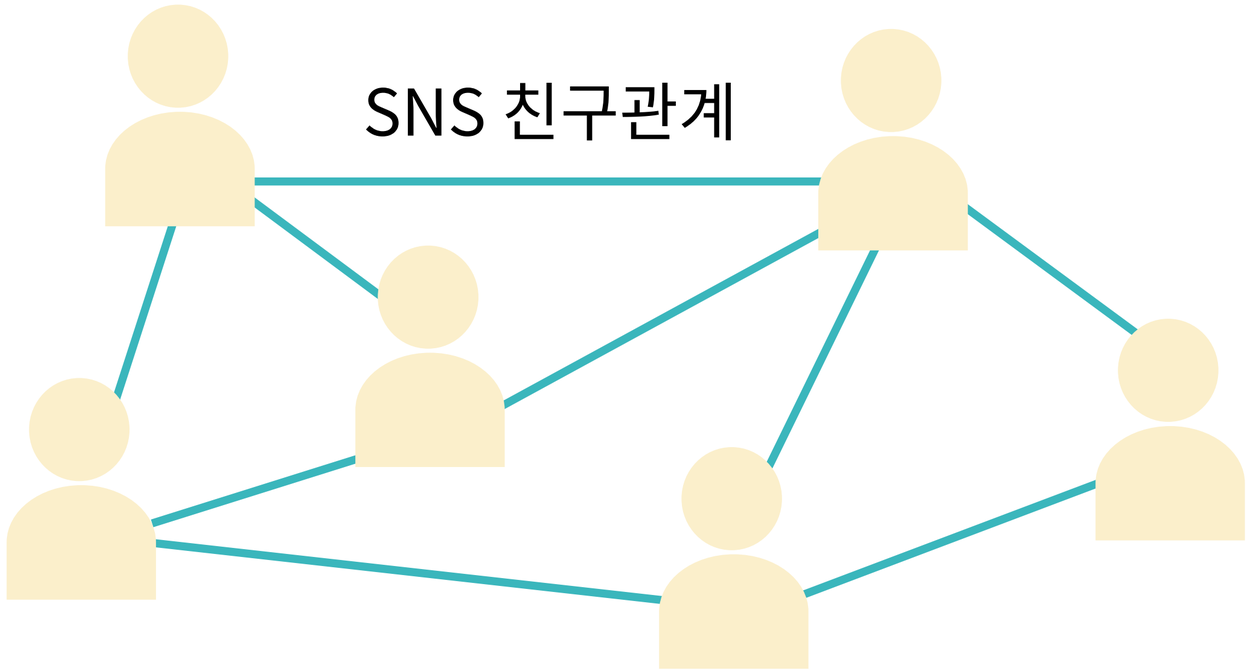
## 간선이란?

**간선**은 **정점들 간의 관계**를 의미합니다.



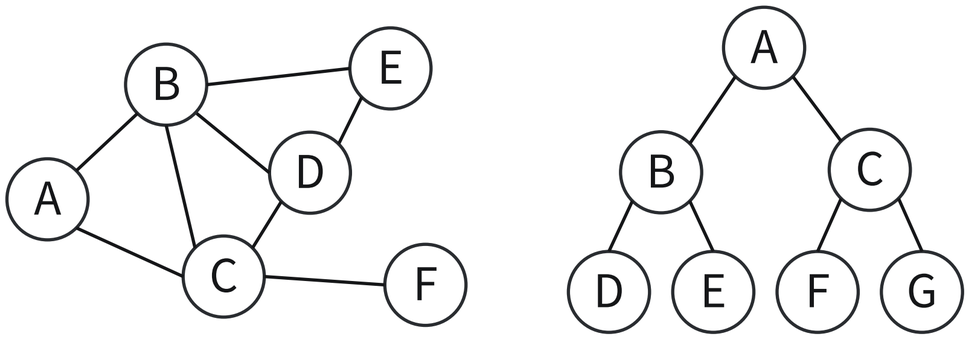
## 간선이란?

**간선**은 **정점들 간의 관계**를 의미합니다.



## 그래프의 종류

- 무방향그래프

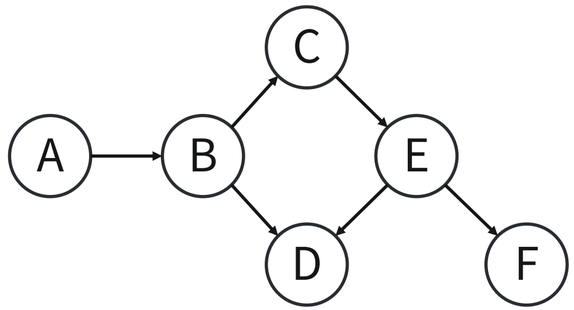


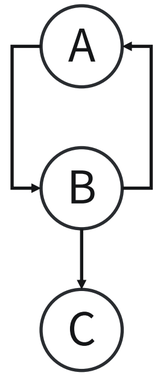




정점간 간선에 **방향이 없는** 그래프

- 방향그래프



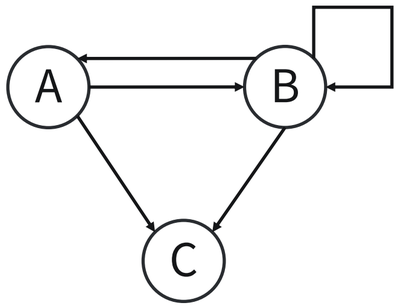




정점간 간선에 **방향이 있는** 그래프

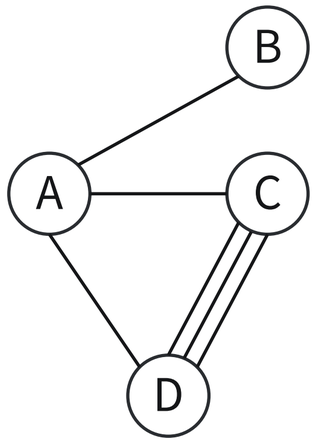
# 01 그래프의 개념

## 그래프의 특징

1) **자기 자신을 향하는** 간선은 없다.



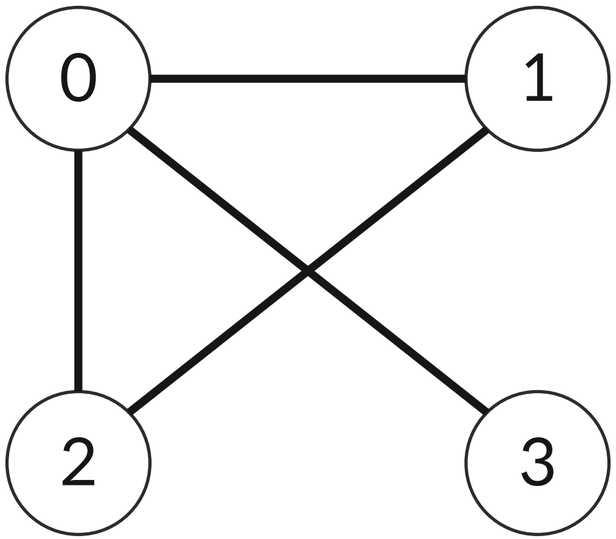

2) **중복된 간선**을 허용하지 않는다.



# 02 그래프의 표현

## 그래프를 표현하는 방식

- 인접행렬 – 무방향 그래프



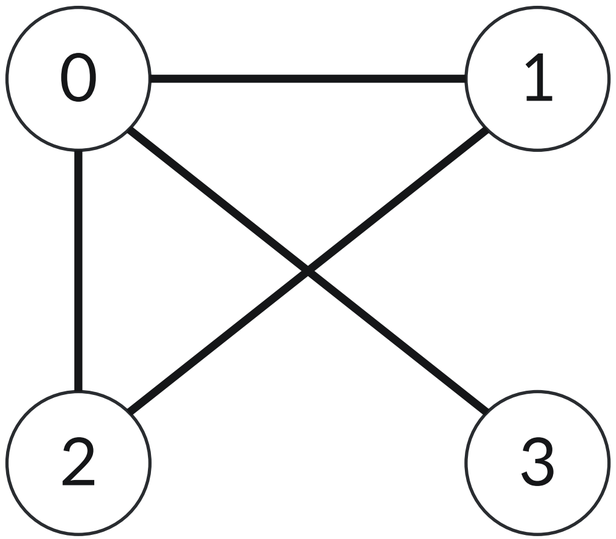

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

/* elice */

# 02 그래프의 표현

## 그래프를 표현하는 방식

- 인접행렬 – 무방향 그래프

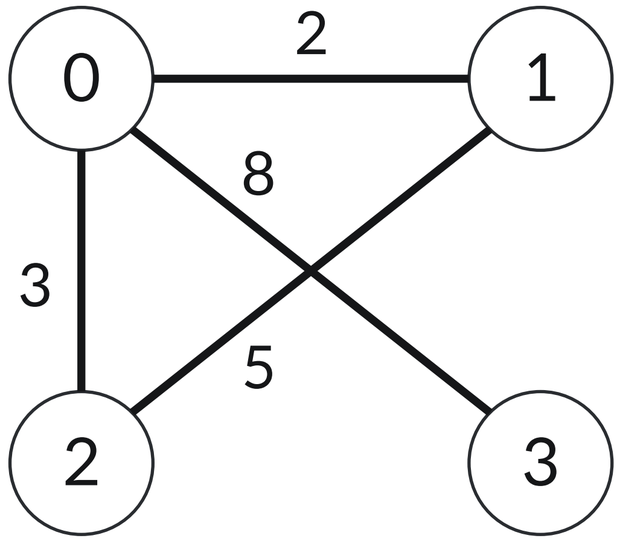



| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 |

# 02 그래프의 표현

## 그래프를 표현하는 방식

- 인접행렬 – 무방향 그래프 with 가중치



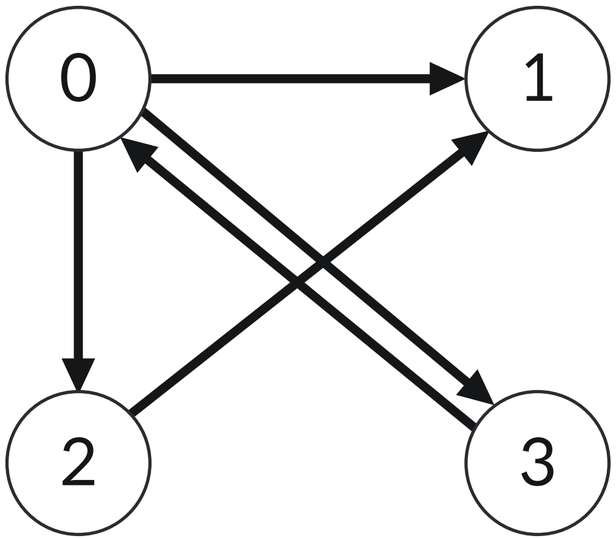

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 2 | 3 | 8 |
| 1 | 2 | 0 | 5 | 0 |
| 2 | 3 | 5 | 0 | 0 |
| 3 | 8 | 0 | 0 | 0 |

/* elice */

# 02 그래프의 표현

## 그래프를 표현하는 방식

- 인접행렬 – 방향 그래프



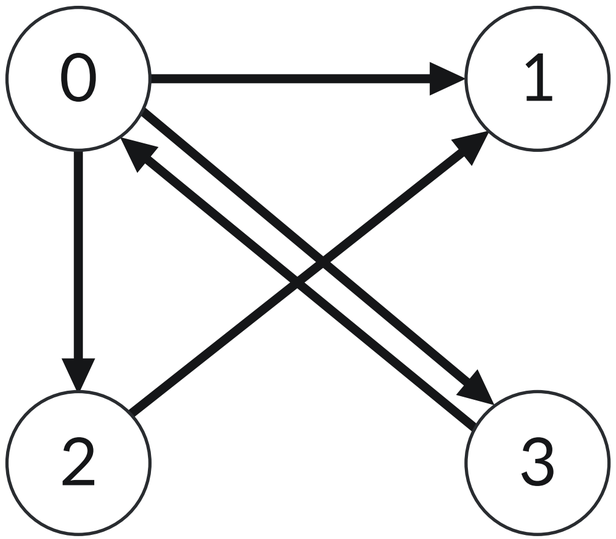

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | | | | |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |

/* elice */

# 02 그래프의 표현

## 그래프를 표현하는 방식

- 인접행렬 – 방향 그래프



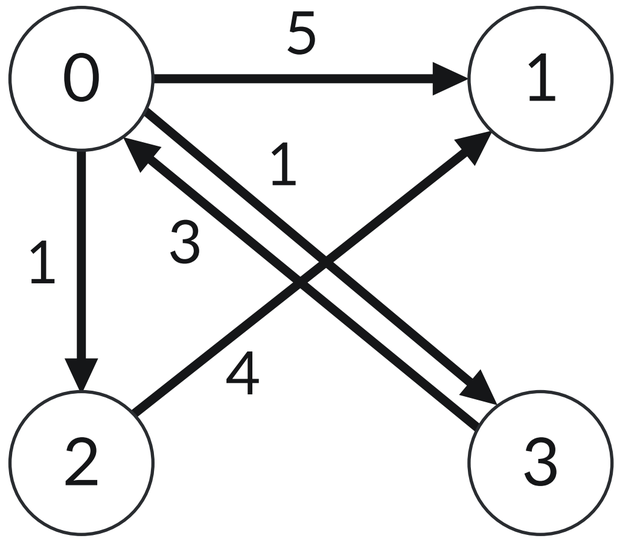

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 1 | 0 | 0 | 0 |

/* elice */

# 02 그래프의 표현

## 그래프를 표현하는 방식

- 인접행렬 – 방향 그래프 with 가중치



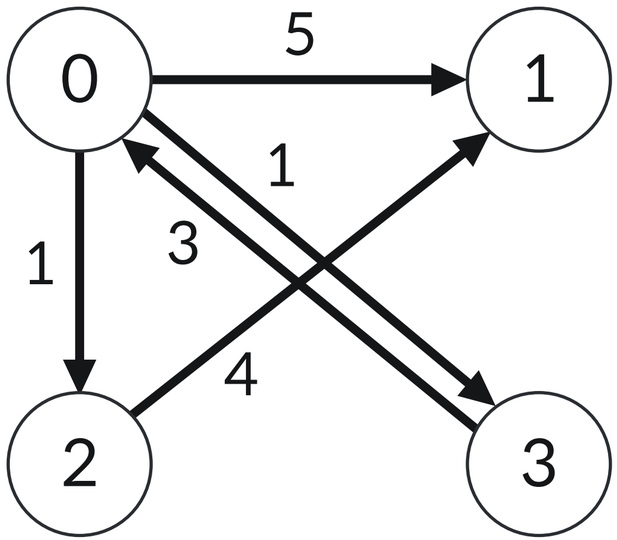

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 5 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 4 | 0 | 0 |
| 3 | 3 | 0 | 0 | 0 |

/* elice */

# 02 그래프의 표현

## 그래프를 표현하는 방식

- 인접행렬



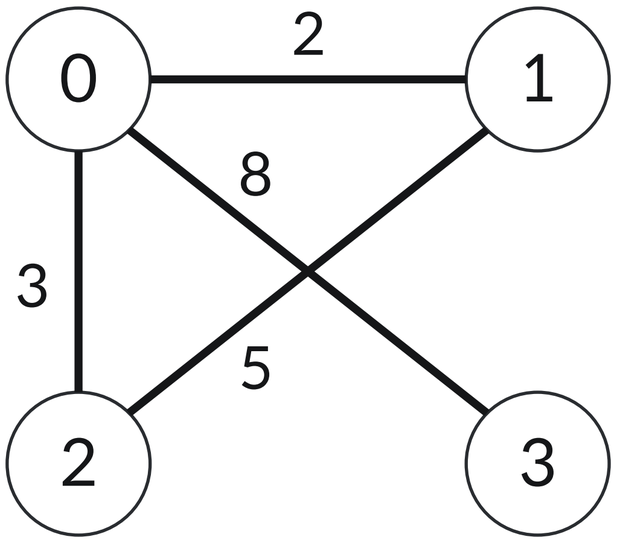

```
graph = {0: {1:5, 2:1, 3:1}, 2: {1: 4}, 3: {0: 3}}
```

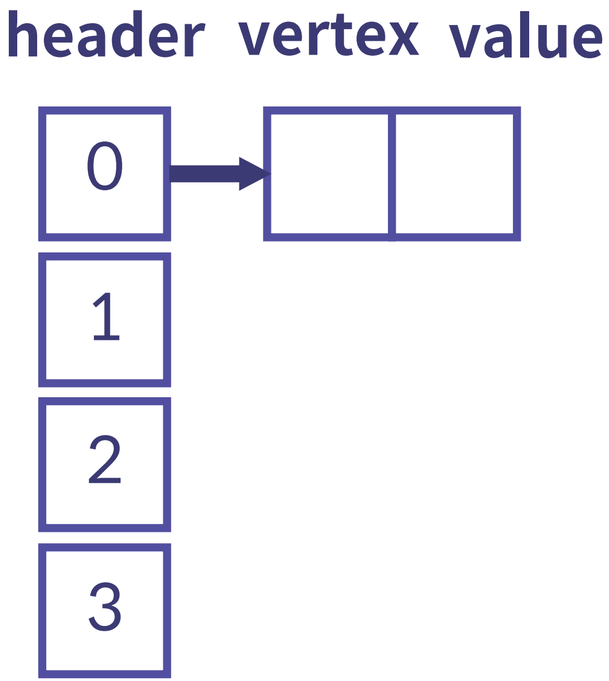
```
graph = [[0,5,1,1],[0,0,0,0],[0,4,0,0],[3,0,0,0]]
```

# 02 그래프의 표현

## 그래프를 표현하는 방식

- 인접리스트

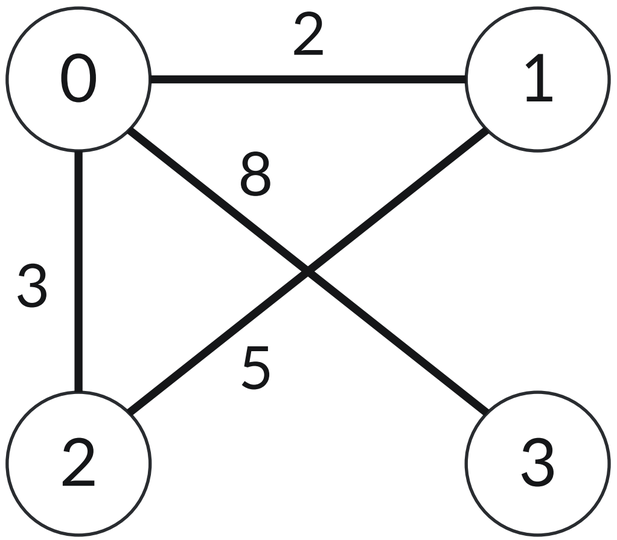


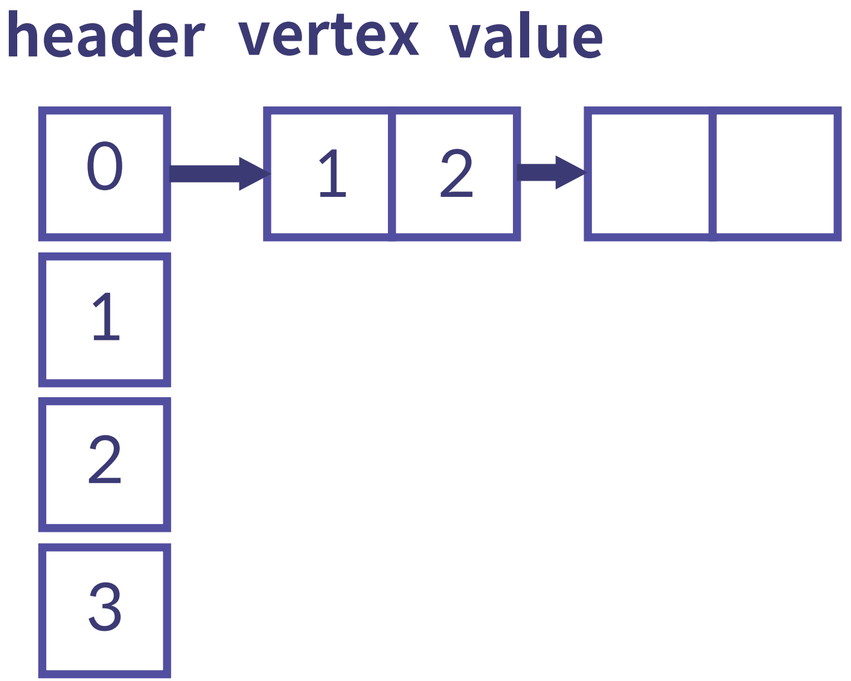



# 02 그래프의 표현

## 그래프를 표현하는 방식

- 인접리스트

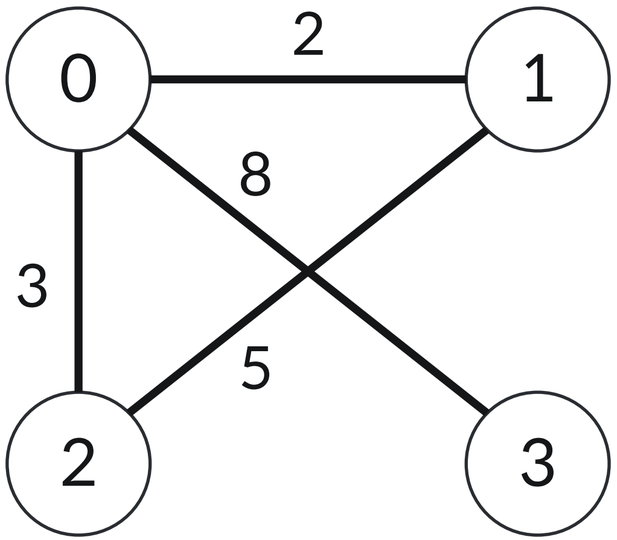


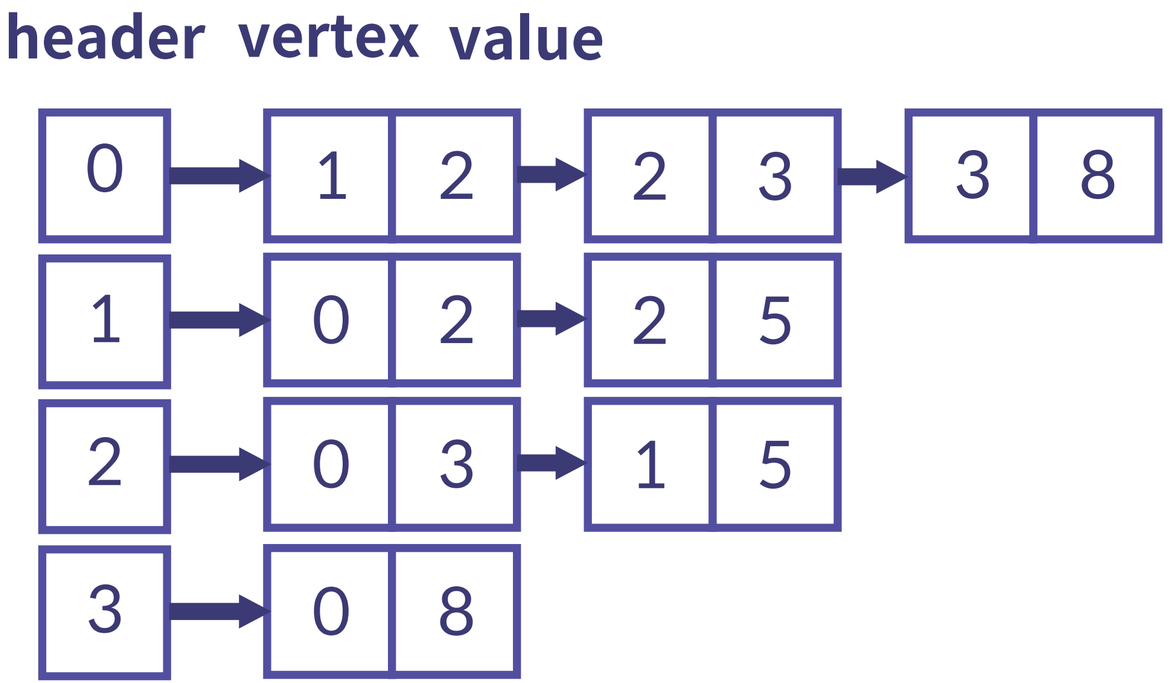



# 02 그래프의 표현

## 그래프를 표현하는 방식

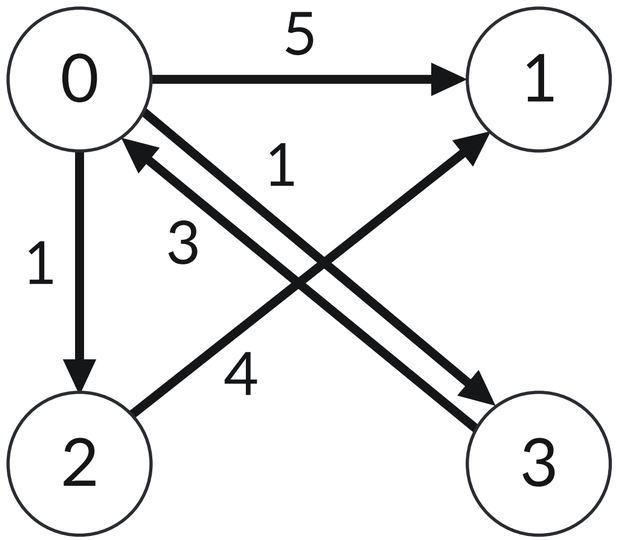
- 인접리스트

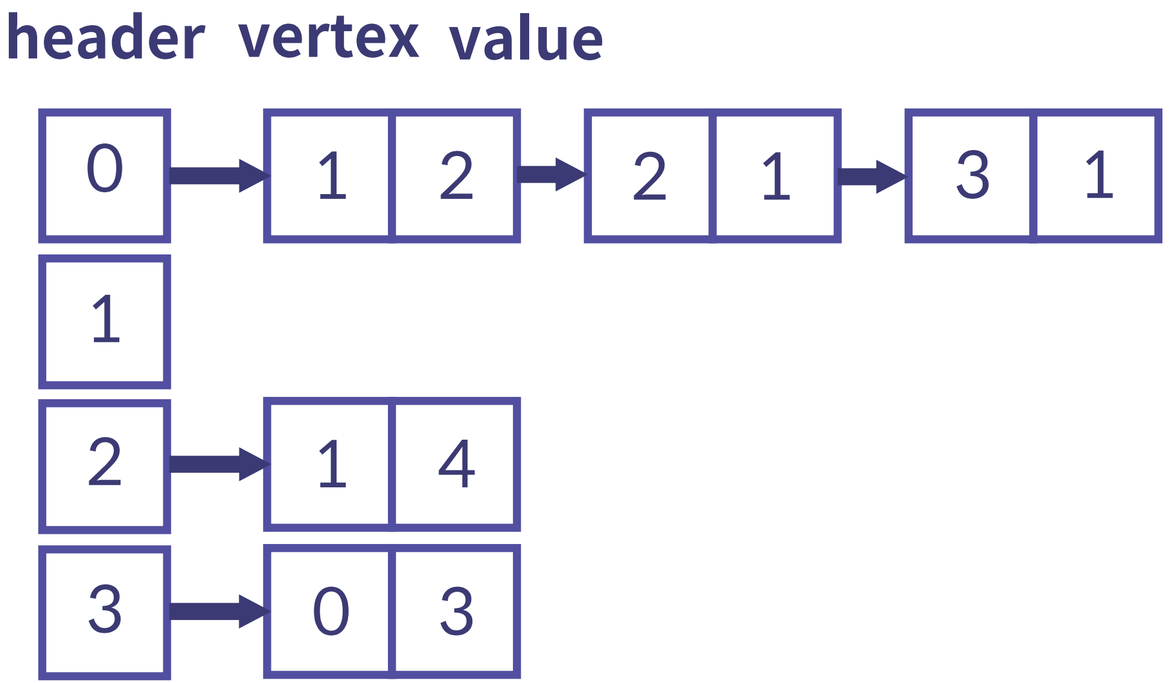


## 그래프를 표현하는 방식

- 인접리스트



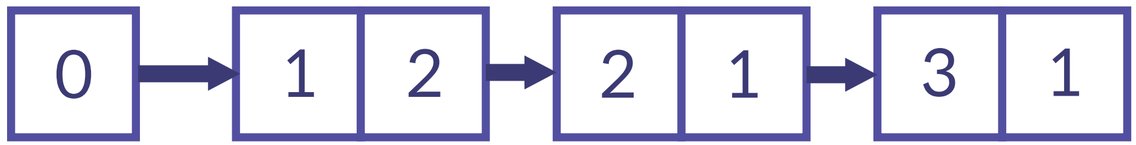
# 02 그래프의 표현

## 그래프를 표현하는 방식

- 인접리스트

**Example**

```
class AdjNode:
    def __init__(self, vertex, value):
        self.vertex = vertex
        self.value = value
        self.next = None
```



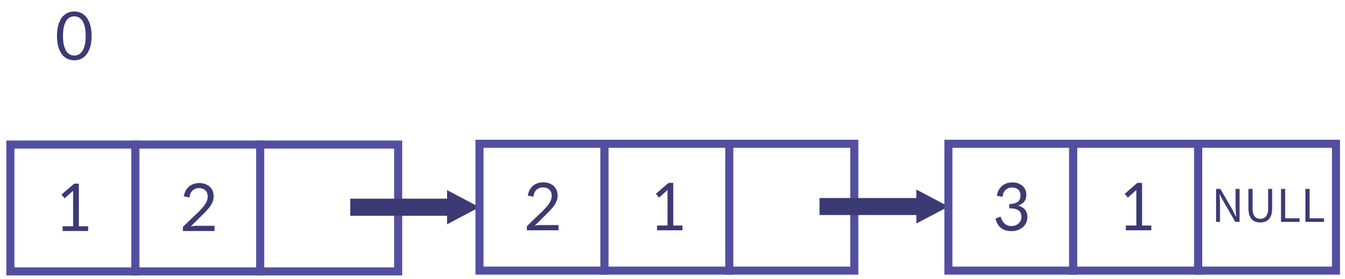
# 02 그래프의 표현

## 그래프를 표현하는 방식

- 인접리스트

**Example**

```
class AdjNode:
    def __init__(self, vertex, value):
        self.vertex = vertex
        self.value = value
        self.next = None
```



# 02 그래프의 표현

## 그래프를 표현하는 방식

- 인접리스트

**Example**

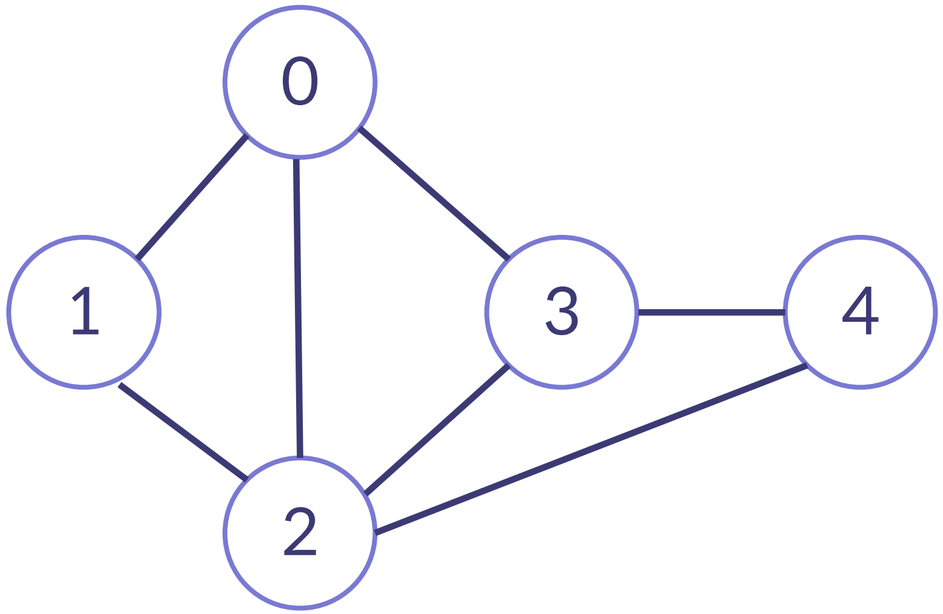
```
class Graph:
    def __init__(self, num):
        self.V = num
        self.graph = [None] * self.V
```

```
def add_edge(self, s, d, w):
        node = AdjNode(d)
        node.next = self.graph[s]
        node.value = w
        self.graph[s] = node
```

/* elice */

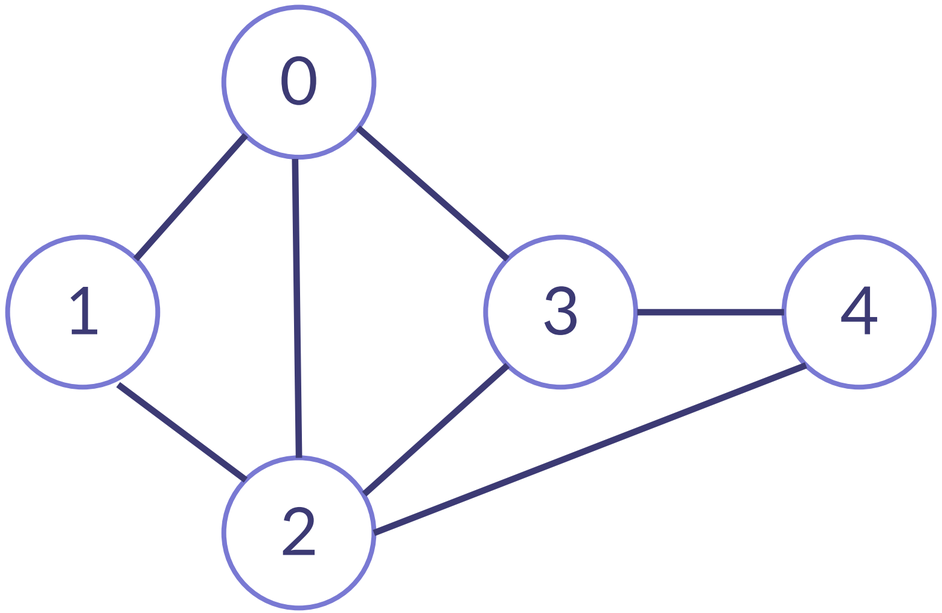
## 너비우선탐색 (Breadth-First Search)

그래프를 탐색하는 기법 중 하나로, **부모를 공유하는 인접 노드**들을 우선적으로 탐색합니다.



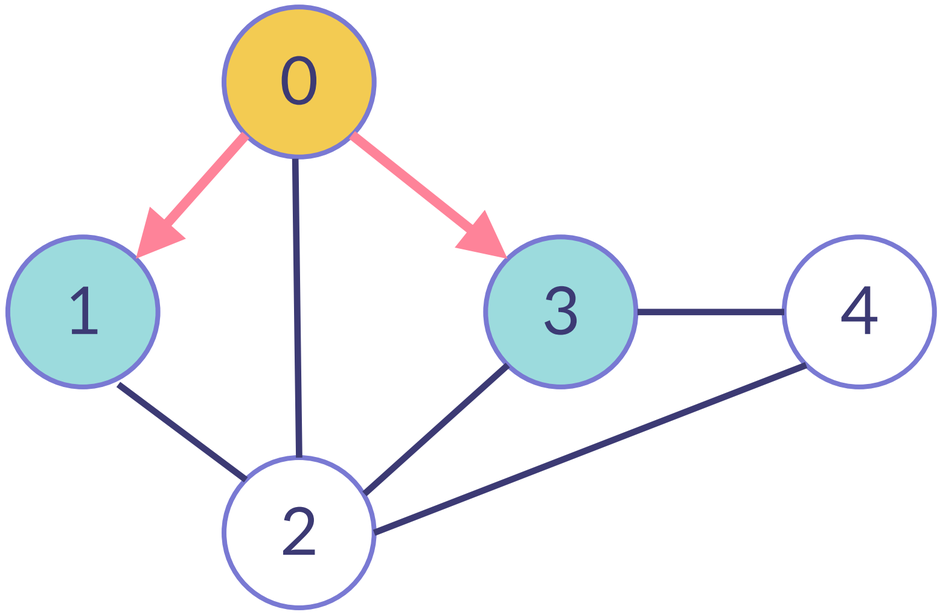
## 너비우선탐색 (Breadth-First Search)

그래프를 탐색하는 기법 중 하나로, **부모를 공유하는 인접 노드**들을 우선적으로 탐색합니다.




## 너비우선탐색 (Breadth-First Search)

그래프를 탐색하는 기법 중 하나로, **부모를 공유하는 인접 노드**들을 우선적으로 탐색합니다.



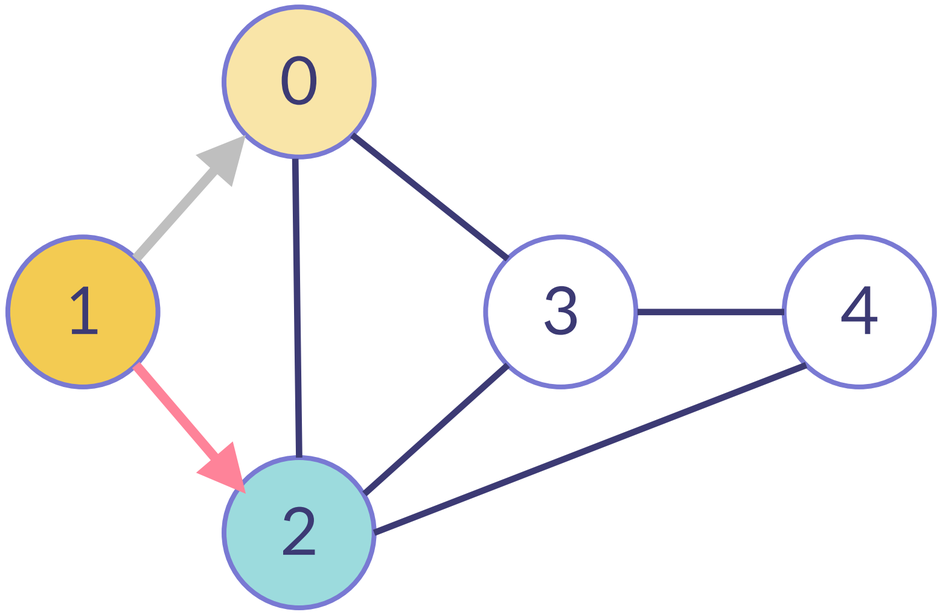

0

Queue

| 1 | 3 | | | | |
|---|---|---|---|---|---|

## 너비우선탐색 (Breadth-First Search)

그래프를 탐색하는 기법 중 하나로, **부모를 공유하는 인접 노드**들을 우선적으로 탐색합니다.



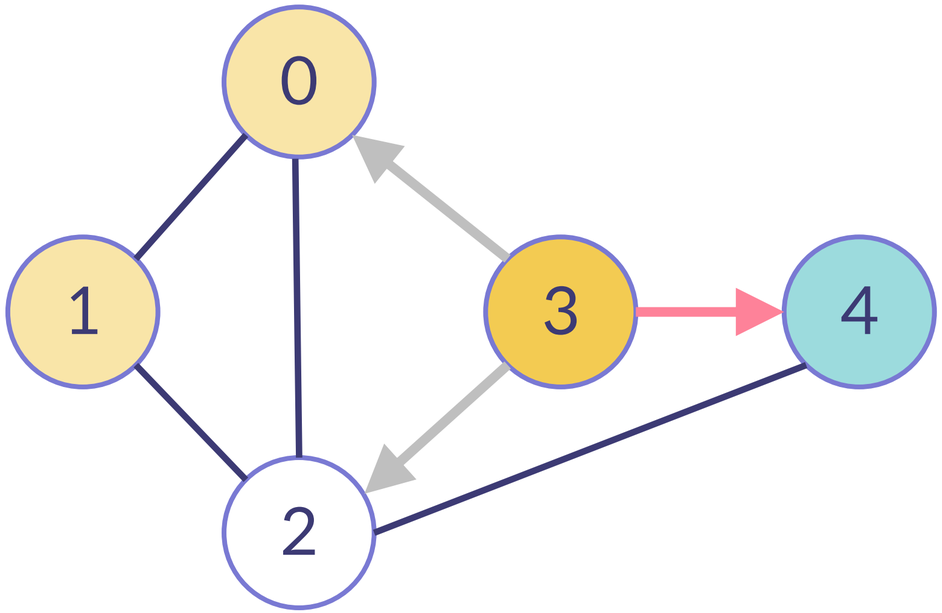

0 1

Queue

| 3 | 2 | | | | |
|---|---|---|---|---|---|

## 너비우선탐색 (Breadth-First Search)

그래프를 탐색하는 기법 중 하나로, **부모를 공유하는 인접 노드**들을 우선적으로 탐색합니다.




0 1 3

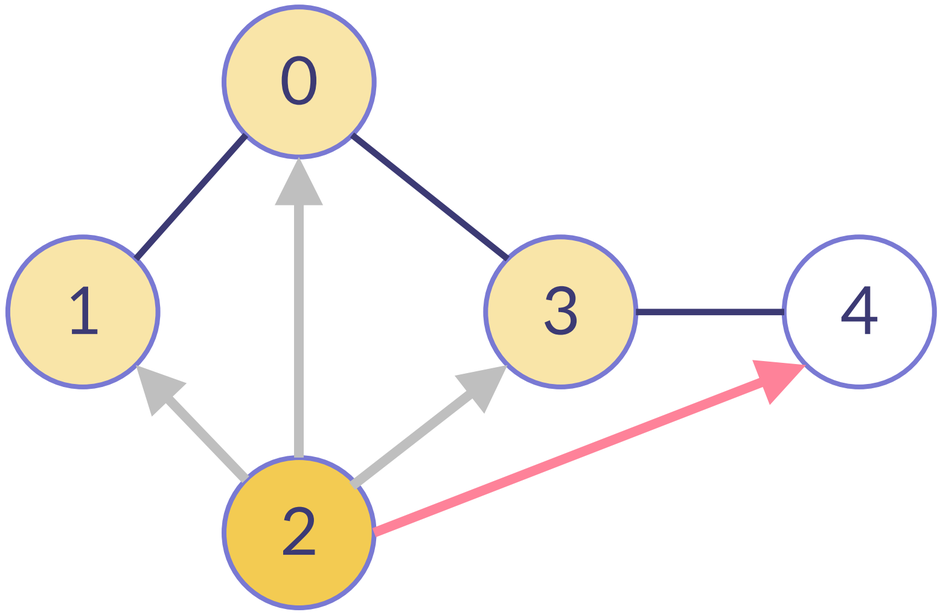
Queue

| 2 | 4 | | | | |
|---|---|---|---|---|---|

/* elice */

## 너비우선탐색 (Breadth-First Search)

그래프를 탐색하는 기법 중 하나로, **부모를 공유하는 인접 노드**들을 우선적으로 탐색합니다.



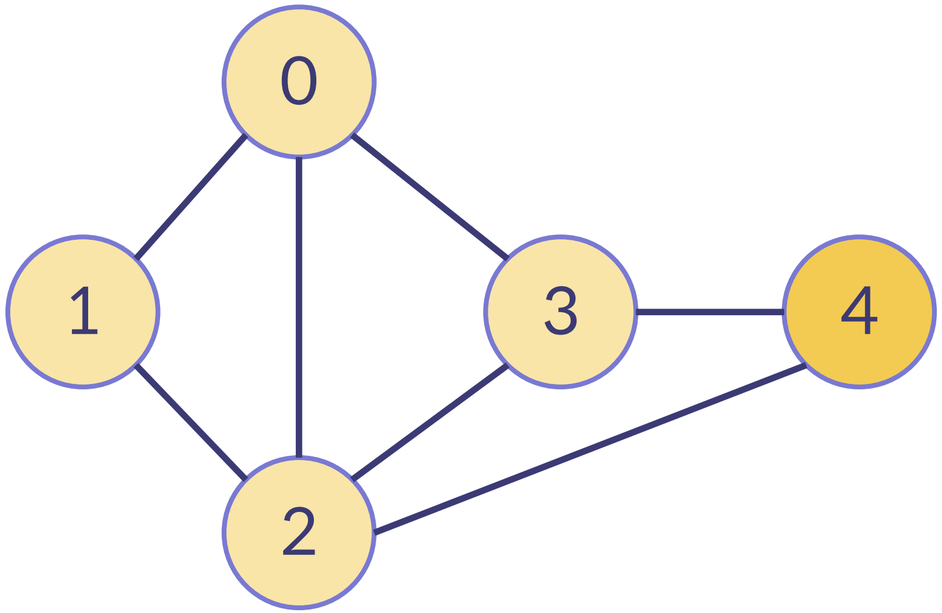

0 1 3 2

Queue

| 4 | | | | | |
|---|---|---|---|---|---|

## 너비우선탐색 (Breadth-First Search)

그래프를 탐색하는 기법 중 하나로, **부모를 공유하는 인접 노드**들을 우선적으로 탐색합니다.




0 1 3 2 4

Queue

/* elice */

# 03 너비우선탐색 (BFS)

## 너비우선탐색 (Breadth-First Search)

그래프를 탐색하는 기법 중 하나로, **부모를 공유하는 인접 노드**들을 우선적으로 탐색합니다.

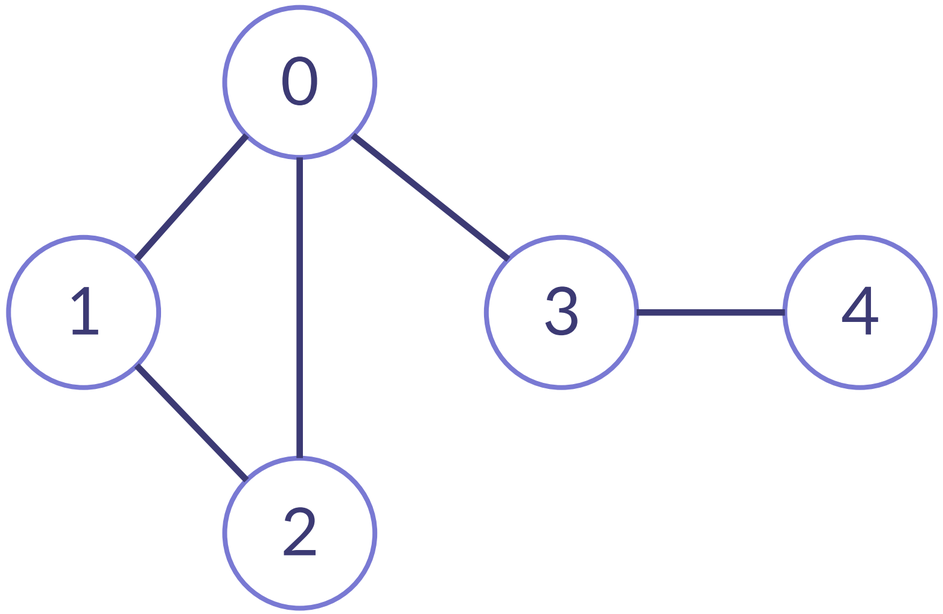
**Example**

```
def BFS (graph, root):
    visited = set([root])
    search = []
    queue = deque([root])
    while queue:
        cur = queue.popleft()
        search.append(cur)
        for node in graph[cur]:
            if node not in visited:
                queue.append(node)
                visited.add(node)
    return search
```

/* elice */

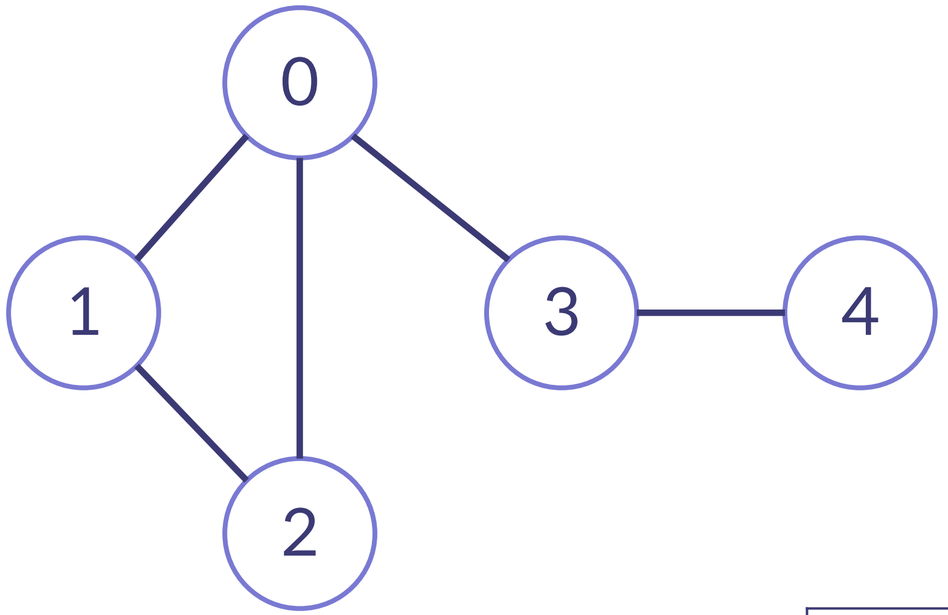
## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.



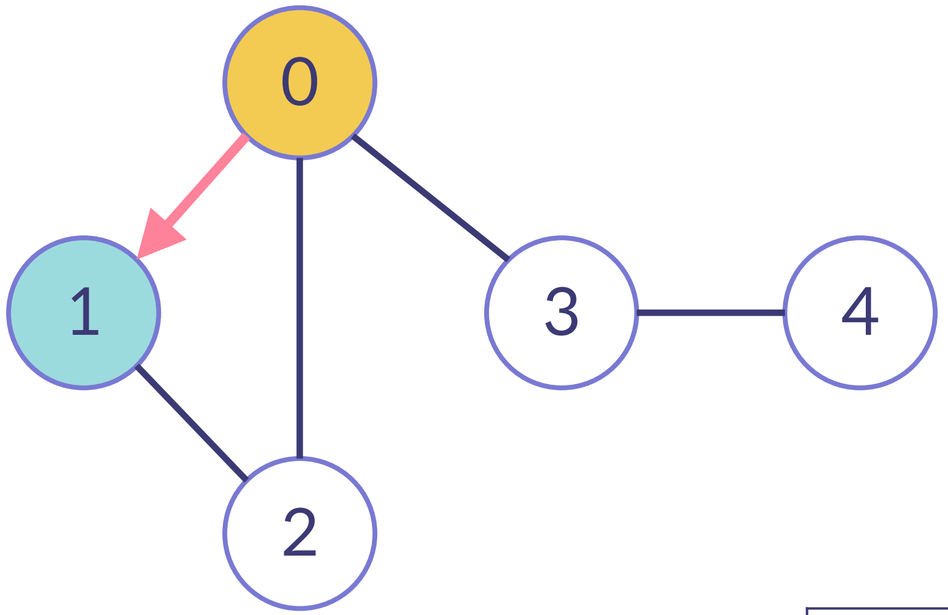
## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.




## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.



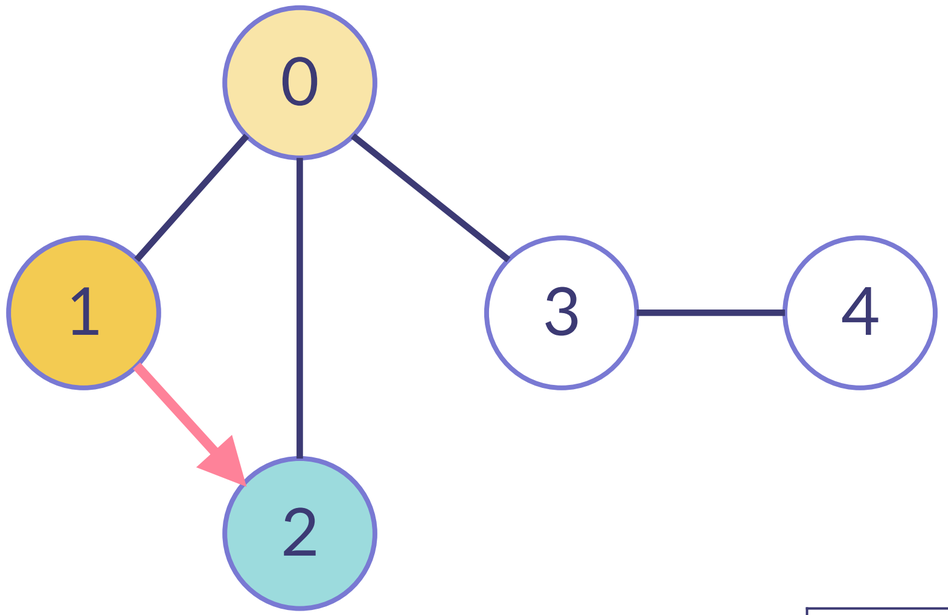

0

Stack

| 0 | | | | | |
|---|---|---|---|---|---|

## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.




0 1

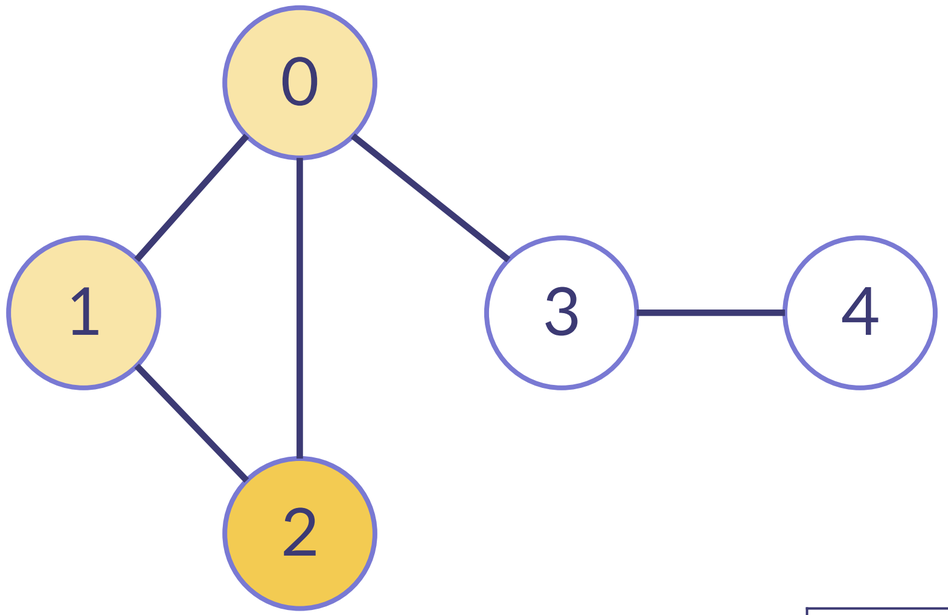
Stack

| 0 | 1 | | | | |
|---|---|---|---|---|---|

/* elice */

## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.



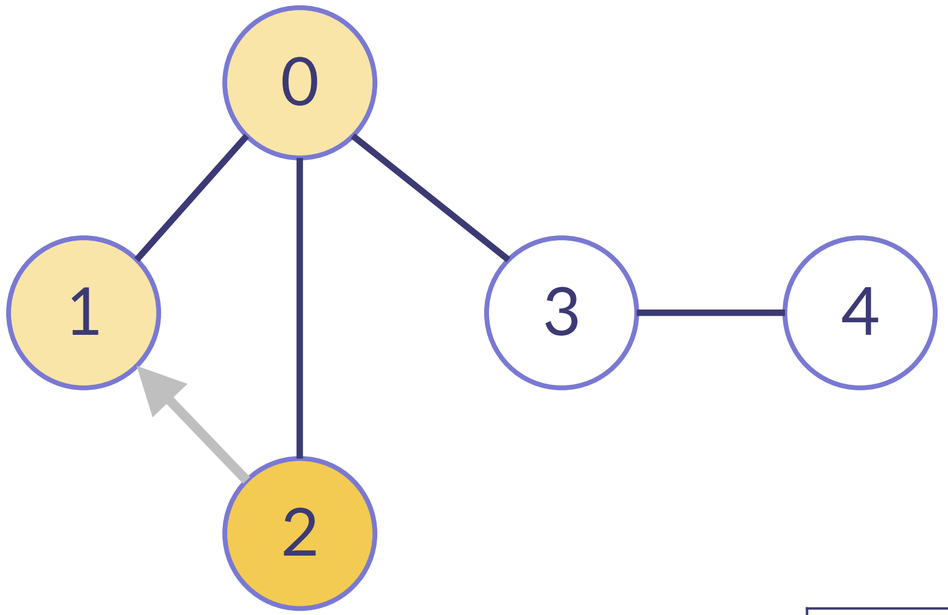

0 1 2

Stack

| 0 | 1 | 2 | | | |
|---|---|---|---|---|---|

## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.



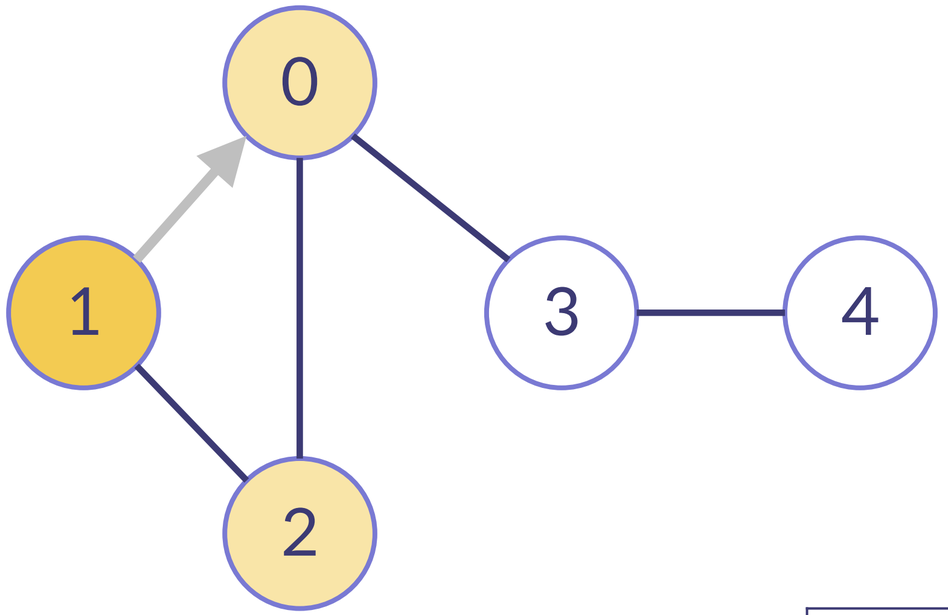

0 1 2

Stack

| 0 | 1 | 2 | | | |
|---|---|---|---|---|---|

## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.




0 1 2

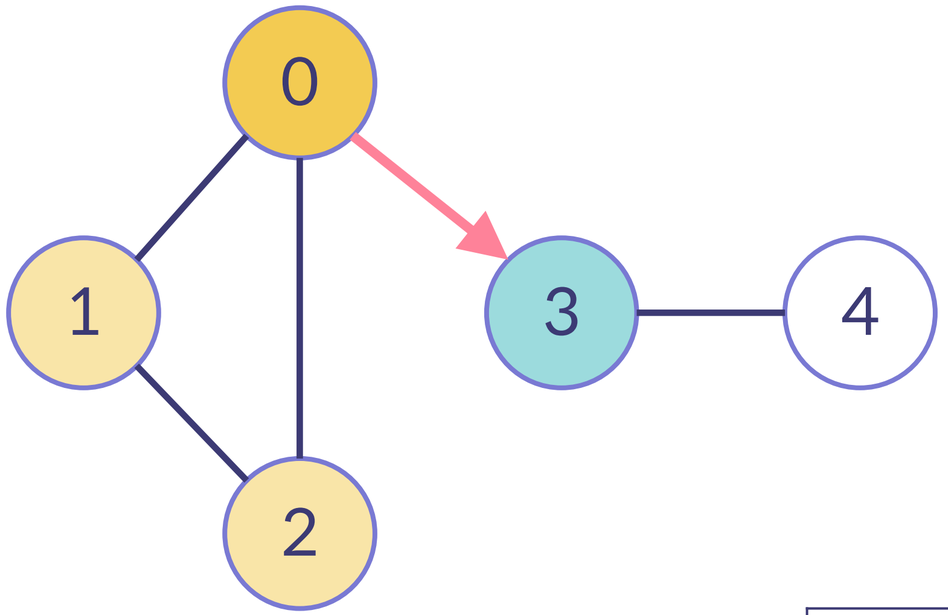
Stack

| 0 | 1 | | | | |
|---|---|---|---|---|---|

/* elice */

## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.

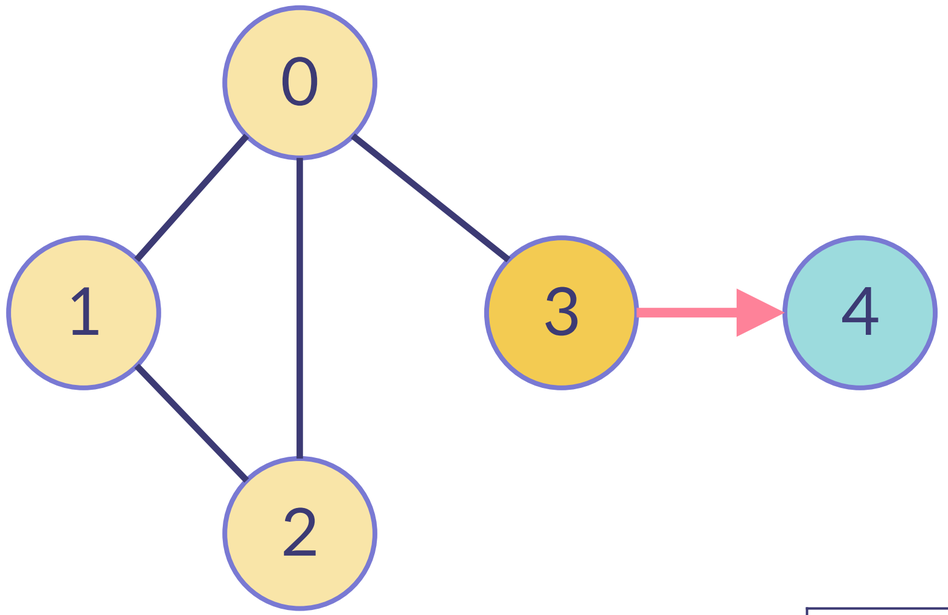



0 1 2


/* elice */

## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.




0 1 2 3

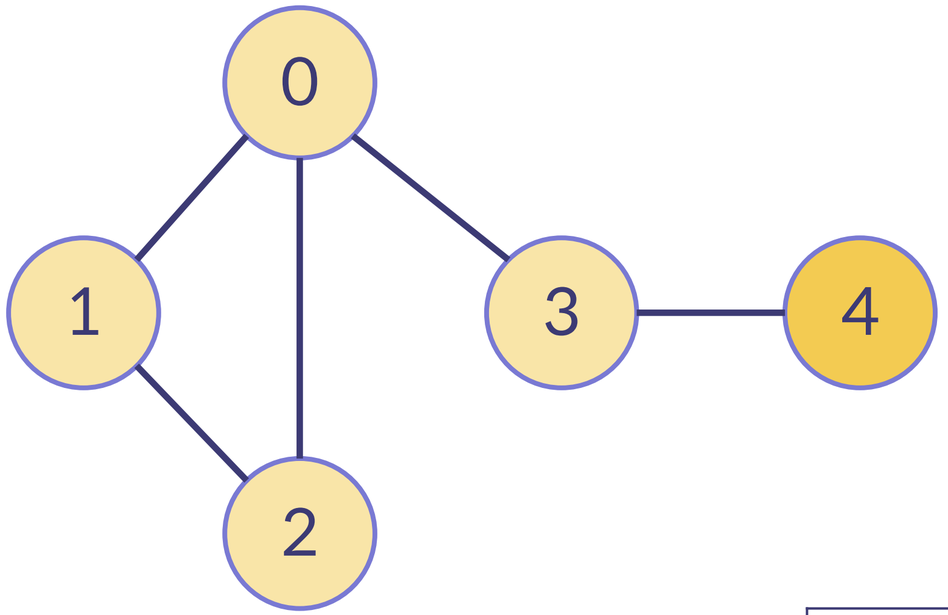
Stack

| 0 | 3 | | | | |
|---|---|---|---|---|---|

/* elice */

## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.



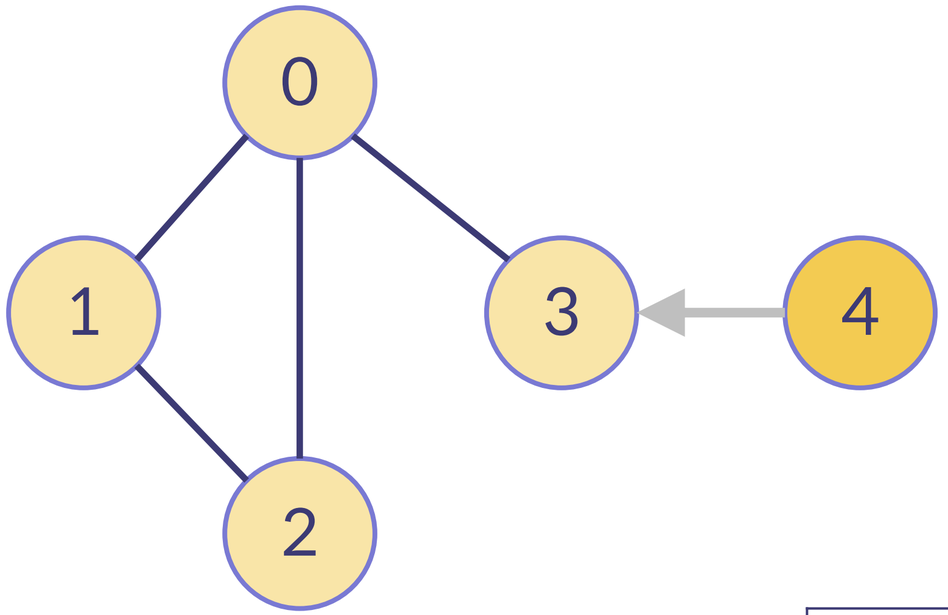

0 1 2 3 4

Stack

| 0 | 3 | 4 | | | |
|---|---|---|---|---|---|

## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.

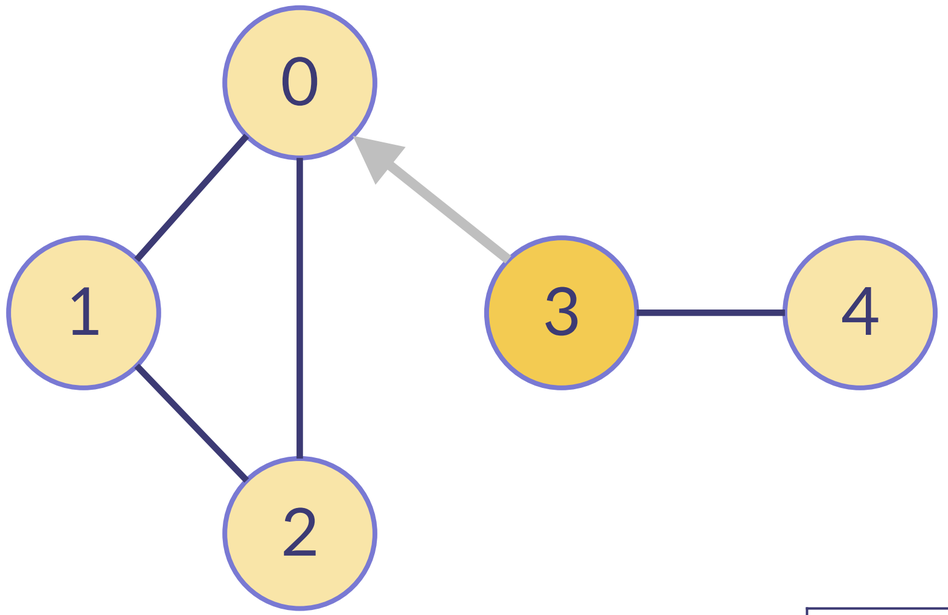



0 1 2 3 4


/* elice */

## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.



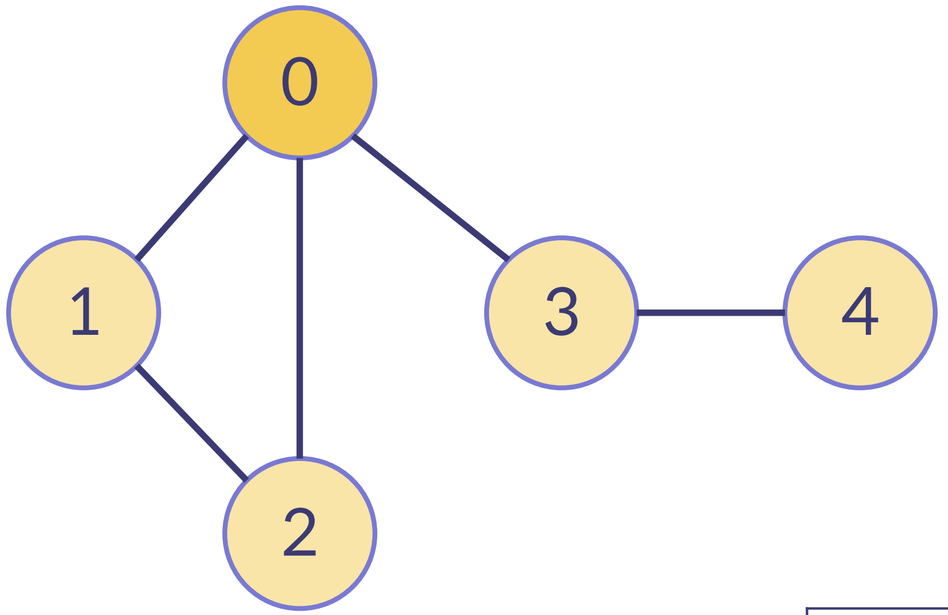

0 1 2 3 4

Stack

| 0 | 3 | | | | |
|---|---|---|---|---|---|

## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.



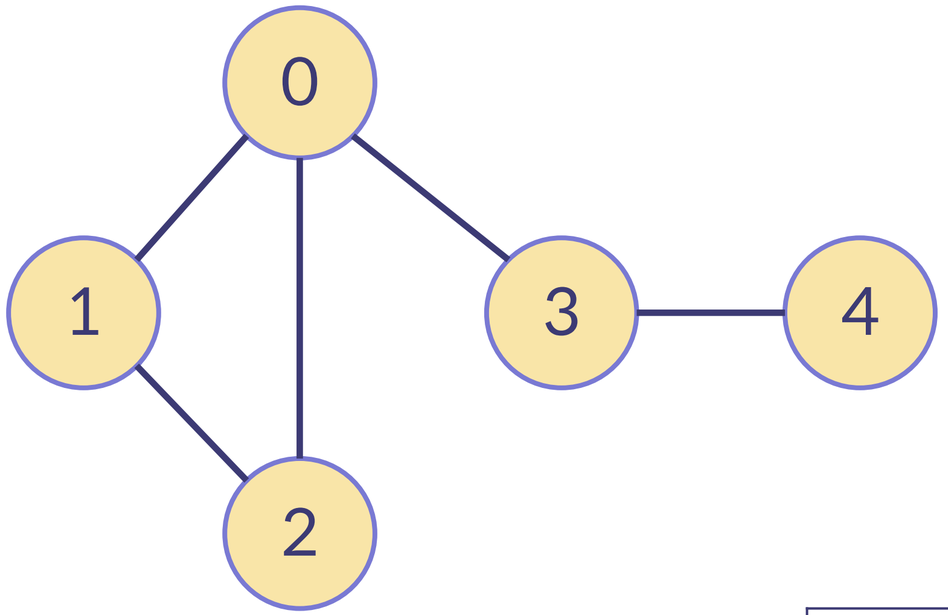

0 1 2 3 4

Stack

| 0 | | | | | |
|---|---|---|---|---|---|

## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.




0 1 2 3 4

Stack

/* elice */

## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.

## 깊이우선탐색 (Depth-Frist Search)

**Example**

```
def DFS(graph, start_node):
    visit = [start_node]
    stack = []

    stack.append(start_node)
    cur = start_node
```

```
while stack:
    for node in graph[cur]:
        if node not in visited:
            stack.append(node)
            visited.append(node)
            cur = node
            break
    else:
        stack.pop()
        if stack : cur = stack[-1]
return visited
```

## 깊이우선탐색 (Depth-Frist Search)

그래프를 탐색하는 기법 중 하나로,
**다음 분기로 넘어가기 전에 해당 분기를 우선적**으로 탐색하는 방법.

**Example**

```
visited = []

def DFS (graph, cur):
    global visited
    if cur not in visited:
        visited.append(cur)
        for node in graph[cur]:
            DFS(graph, node)
```

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

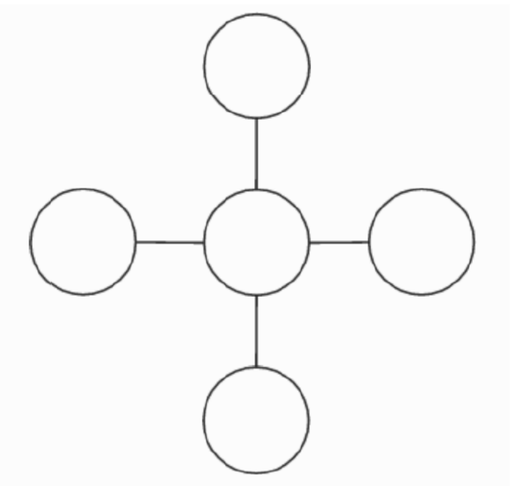
## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

| | | |
|---|---|---|
| 0 | 1 | 0 |
| 1 | 1 | 1 |
| 0 | 1 | 0 |



/* elice */

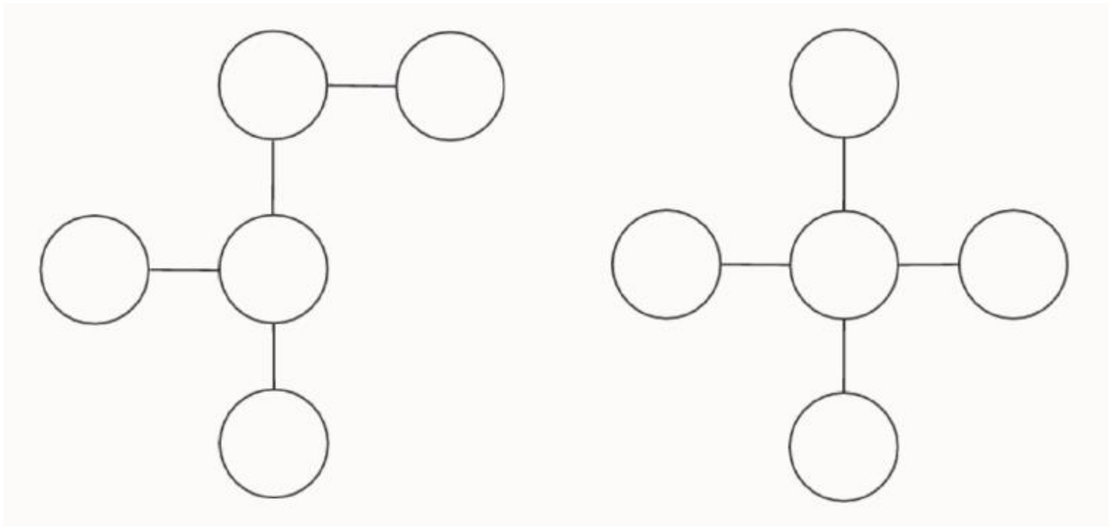
# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |



/* elice */

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.

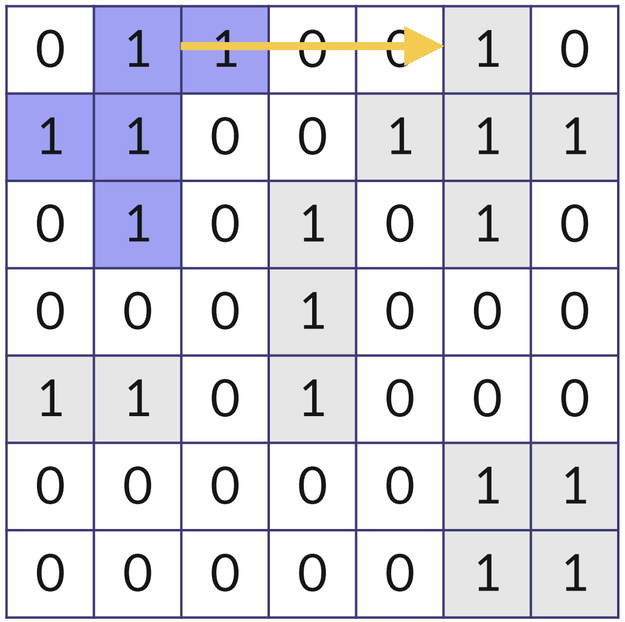
| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.

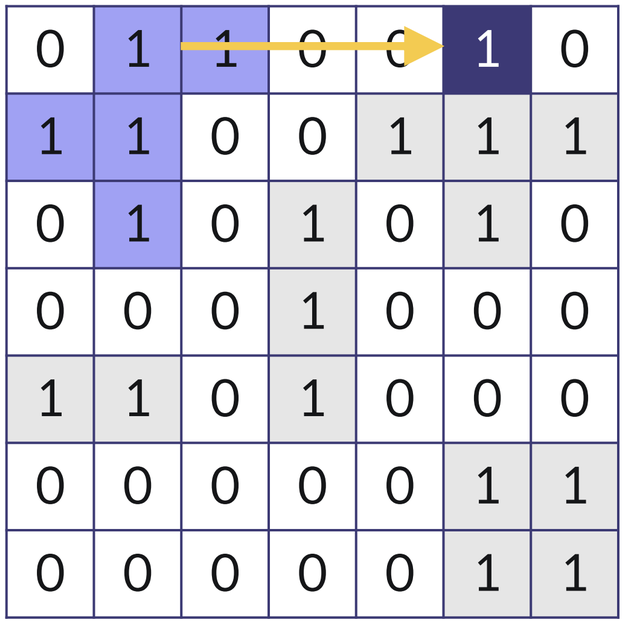


# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.



| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.

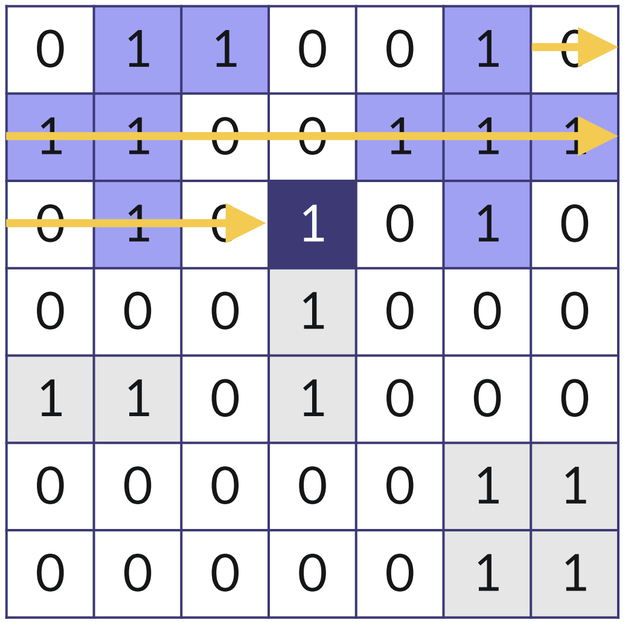
| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.

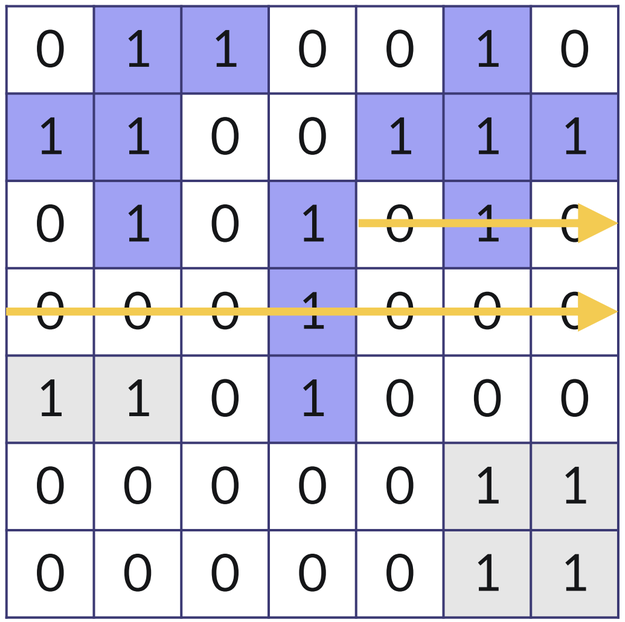


# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.



# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.

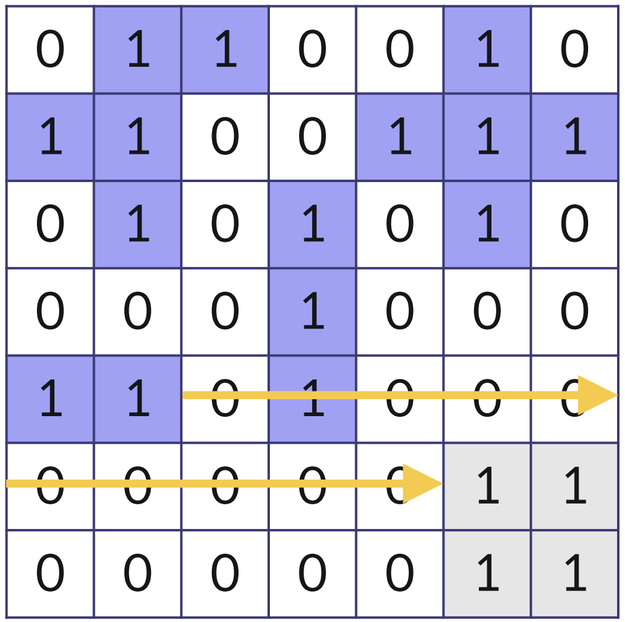
| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.



# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.

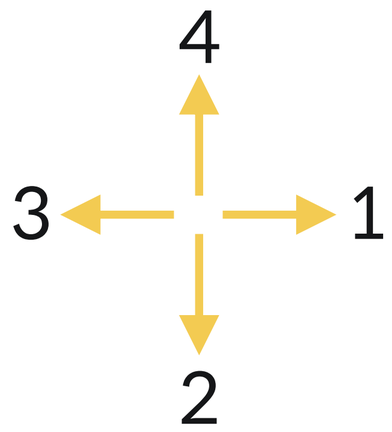
| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.



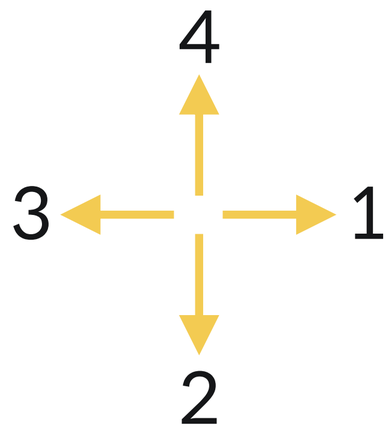

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.



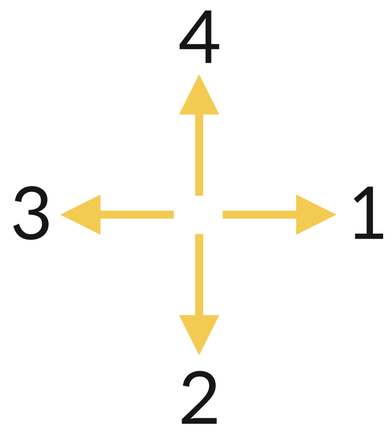

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.



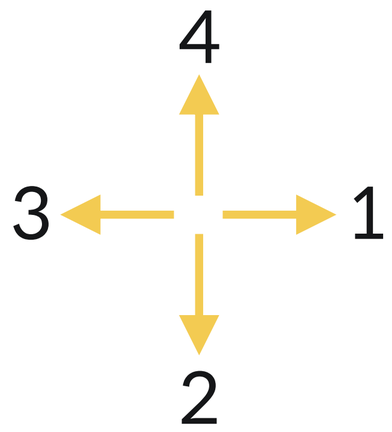

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.



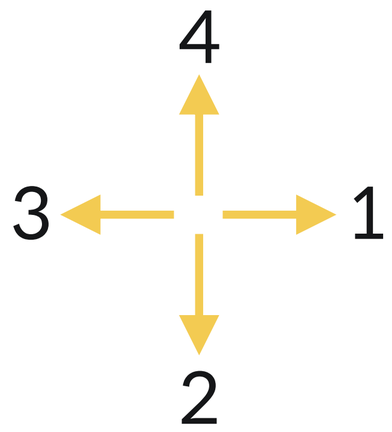

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.



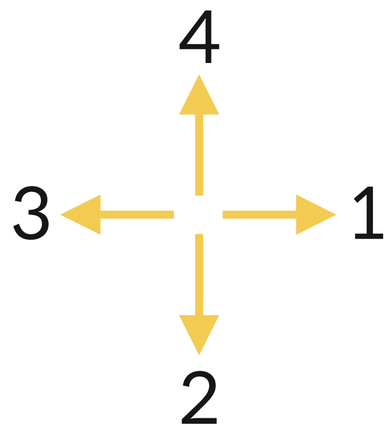

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.



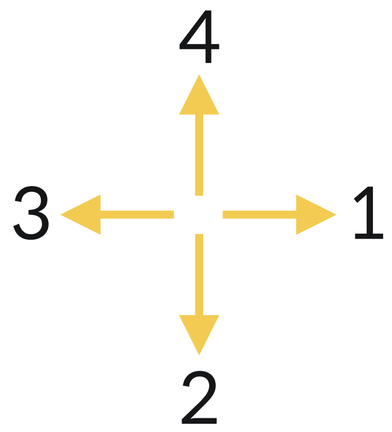

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**문제**

네모 모양 행성에 **1로 되어있는 육지**와 **0으로 되어있는 바다**가 있습니다.
**육지의 좌,우,상,하가 모두 바다라면 섬**이라고 합니다. 섬의 개수를 구하세요.




| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**Example**

```
def count_island(h, w, maps):
    cnt = 0
    isize = []
    for i in range(h):
        for j in range(w):
            if maps[i][j] != 0:
                cnt += 1
                size = 1
                size = dfs(i, j)
            isize.append(size)
    return cnt, isize
```

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

# 05 문제를 그래프로 표현하기 - 1

## 디지털 세계 토지조사

**Example**

```
dir = [(0,1),(1,0),(0,-1),(-1,0)]

def dfs(x,y):
    global maps
    global size
    maps[x][y] = 0
```

```
for i in range(4):
    nx = x + dir[i][0]
    ny = y + dir[i][1]
    if 0 <= nx < h and 0 <= ny < w and maps[nx][ny] == 1:
        size += 1
        maps[nx][ny] = 0
        dfs(nx,ny)
    return size
```

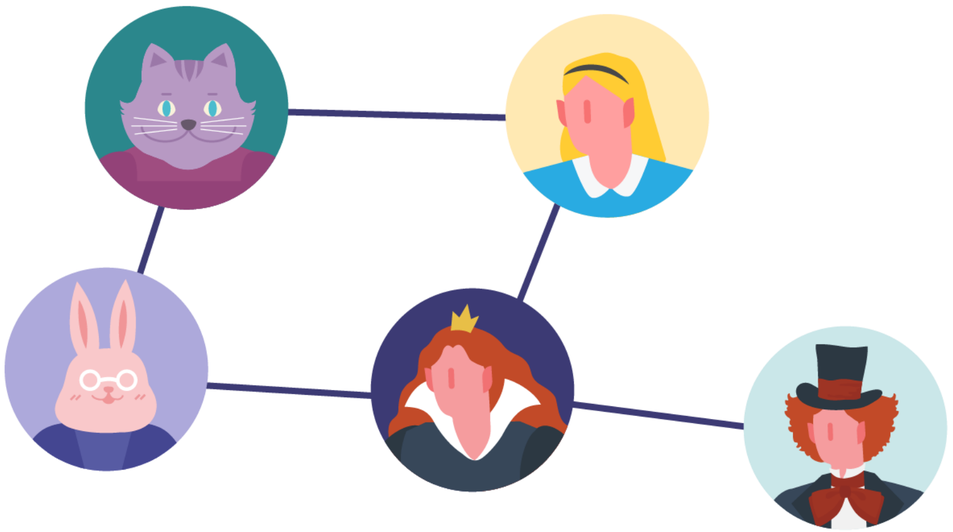
# 05 문제를 그래프로 표현하기 - 2

## 촌수 계산하기

**문제**

새로운 SNS 서비스 **엘리스친구**는 두 명의 사용자 간의 거리를 촌수로 표시합니다.
**직접연결**이 되어있는 친구라면 **1촌**, **N명의 친구를 거쳐** 연결되어있다면 **N+1촌**입니다.
**두 사용자 간의 촌수**를 구하세요.

엘리스 – 체셔
체셔 – 토끼
엘리스 – 하트여왕
모자장수 – 하트여왕
토끼 – 하트여왕



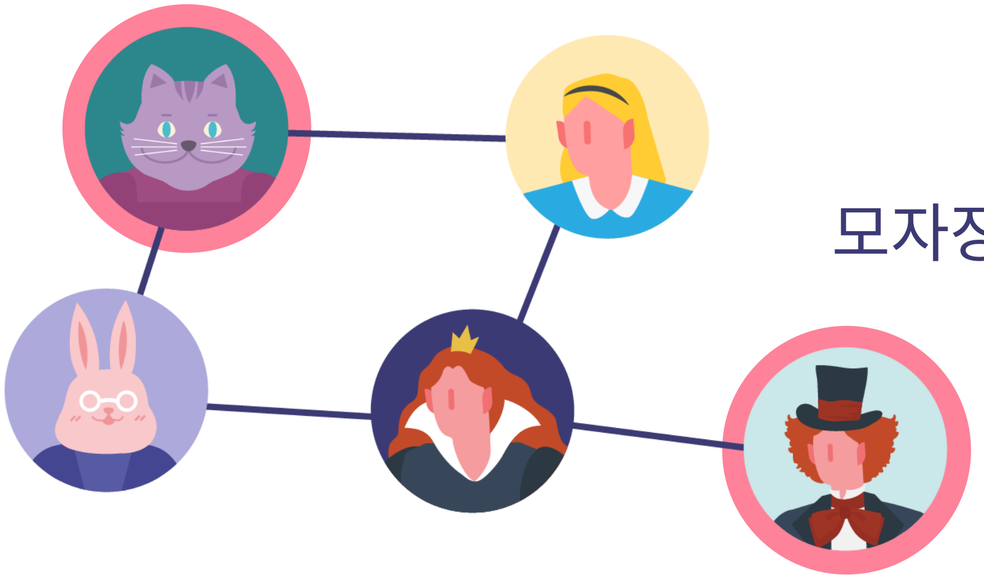
# 05 문제를 그래프로 표현하기 - 2

## 촌수 계산하기

**문제**

새로운 SNS 서비스 **엘리스친구**는 두 명의 사용자 간의 거리를 촌수로 표시합니다.
**직접연결**이 되어있는 친구라면 **1촌**, **N명의 친구를 거쳐** 연결되어있다면 **N+1촌**입니다.
**두 사용자 간의 촌수**를 구하세요.

엘리스 – 체셔
체셔 – 토끼
엘리스 – 하트여왕
모자장수 – 하트여왕
토끼 – 하트여왕



모자장수 – 체셔?

/* elice */

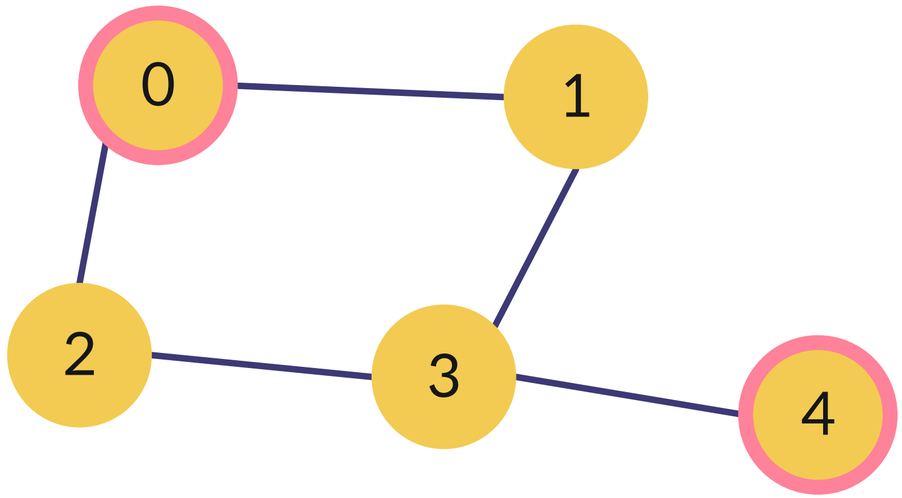
# 05 문제를 그래프로 표현하기 - 2

## 촌수 계산하기

**문제**

새로운 SNS 서비스 **엘리스친구**는 두 명의 사용자 간의 거리를 촌수로 표시합니다.
**직접연결**이 되어있는 친구라면 **1촌**, **N명의 친구를 거쳐** 연결되어있다면 **N+1촌**입니다.
**두 사용자 간의 촌수**를 구하세요.

엘리스 – 체셔
체셔 – 토끼
엘리스 – 하트여왕
모자장수 – 하트여왕
토끼 – 하트여왕



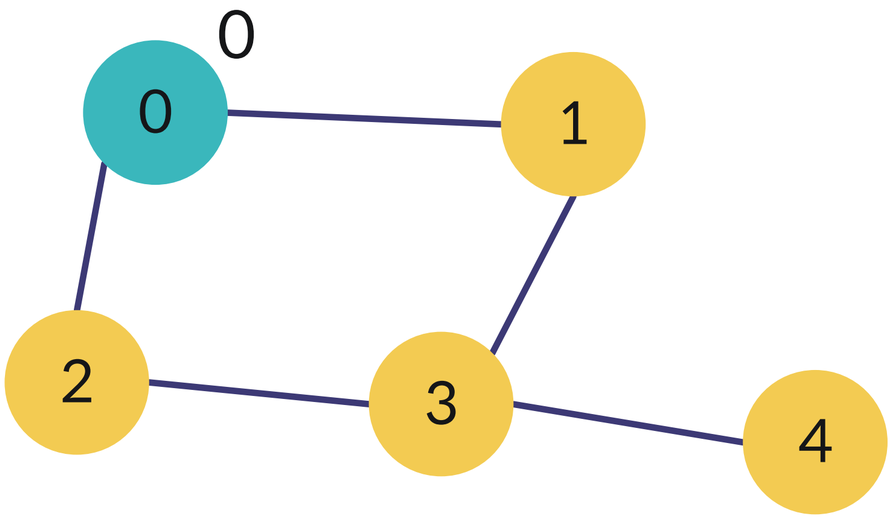

/* elice */

# 05 문제를 그래프로 표현하기 - 2

## 촌수 계산하기

**문제**

새로운 SNS 서비스 **엘리스친구**는 두 명의 사용자 간의 거리를 촌수로 표시합니다.
**직접연결**이 되어있는 친구라면 **1촌**, **N명의 친구를 거쳐** 연결되어있다면 **N+1촌**입니다.
**두 사용자 간의 촌수**를 구하세요.

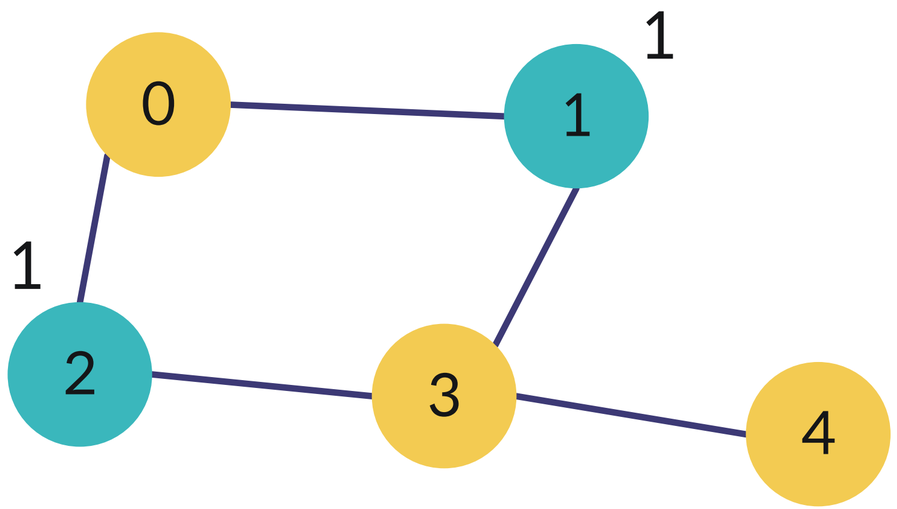


# 05 문제를 그래프로 표현하기 - 2

## 촌수 계산하기

**문제**

새로운 SNS 서비스 **엘리스친구**는 두 명의 사용자 간의 거리를 촌수로 표시합니다.
**직접연결**이 되어있는 친구라면 **1촌**, **N명의 친구를 거쳐** 연결되어있다면 **N+1촌**입니다.
**두 사용자 간의 촌수**를 구하세요.

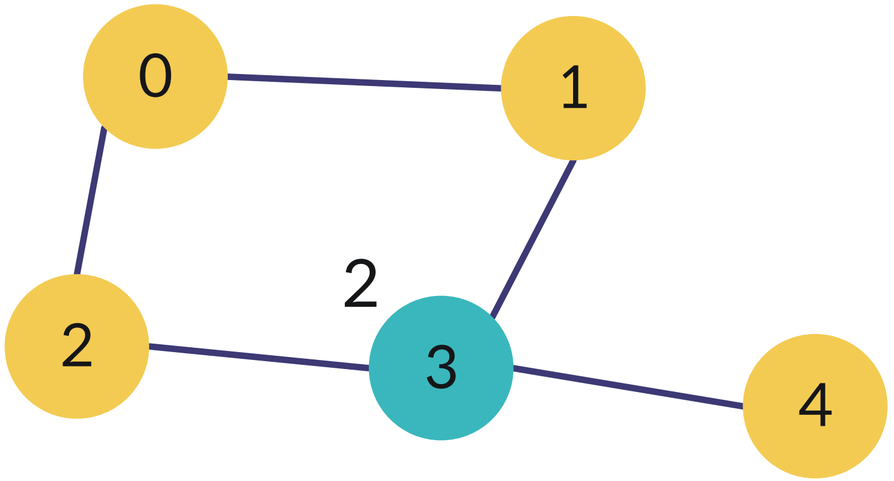


# 05 문제를 그래프로 표현하기 - 2

## 촌수 계산하기

**문제**

새로운 SNS 서비스 **엘리스친구**는 두 명의 사용자 간의 거리를 촌수로 표시합니다.
**직접연결**이 되어있는 친구라면 **1촌**, **N명의 친구를 거쳐** 연결되어있다면 **N+1촌**입니다.
**두 사용자 간의 촌수**를 구하세요.

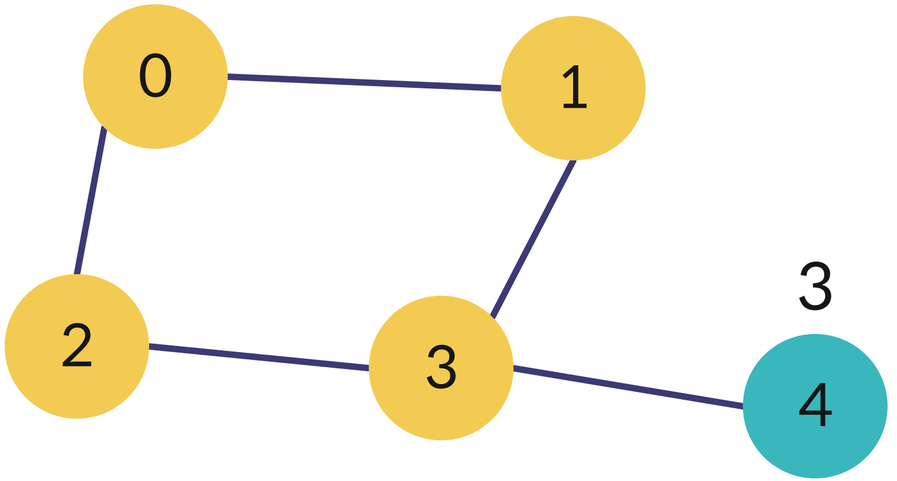


# 05 문제를 그래프로 표현하기 - 2

## 촌수 계산하기

**문제**

새로운 SNS 서비스 **엘리스친구**는 두 명의 사용자 간의 거리를 촌수로 표시합니다.
**직접연결**이 되어있는 친구라면 **1촌**, **N명의 친구를 거쳐** 연결되어있다면 **N+1촌**입니다.
**두 사용자 간의 촌수**를 구하세요.

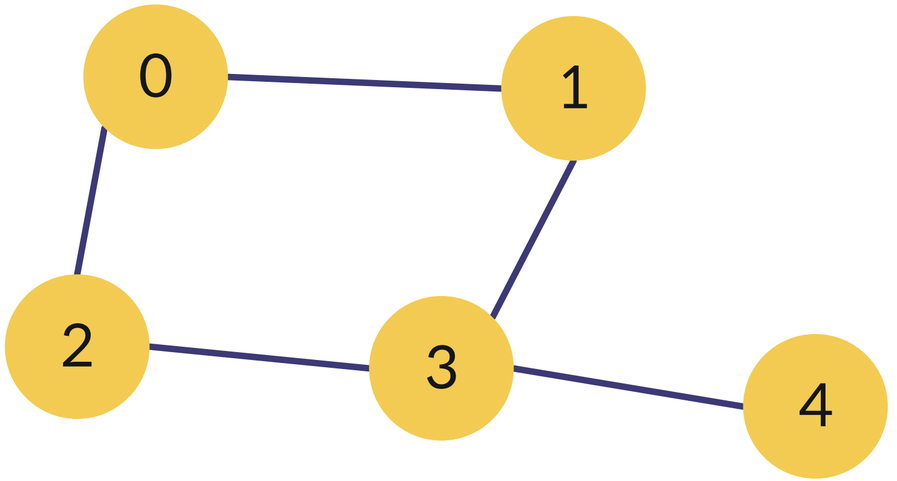


# 05 문제를 그래프로 표현하기 - 2

## 촌수 계산하기

**Example**

```
def bfs(x) :
    queue = deque([x])
    visit = [-1] * n
    visit[x] = 0;
    while queue :
        cur = queue.popleft()
        for I in range(n) :
            if friends[cur][i] == 1 and visit[i] == -1 :
                visit[i] = visit[cur] + 1
                queue.append(i)
```




/* elice */

# 07 정리

## 그래프 알고리즘에서는 …

- 그래프는 정점과 간선으로 이루어져 있습니다.
- 인접행렬, 인접리스트 등의 방법으로 그래프를 구현할 수 있습니다.
- 그래프 탐색방법으로는 BFS와 DFS가 있습니다.
  - BFS는 너비우선탐색으로 같은 루트를 공유하는 노드를 우선적으로 탐색합니다.
  - DFS는 특정 분기를 모두 탐색을 마친 후 다음 분기로 넘어가 탐색합니다.

# Contact


**TEL**
070-4633-2015

**WEB**
https://elice.io

**E-MAIL**
contact@elice.io