JavaScript 핵심 자료구조 알아보기

6) 그래프(Graph)의 표현

그래프의 표현 | 다양한 알고리즘의 기본이 되는 자료구조 이해하기

강사 나동빈

JavaScript
핵심 자료구조 알아보기
6) 그래프(Graph)의 표현

그래프(Graph)

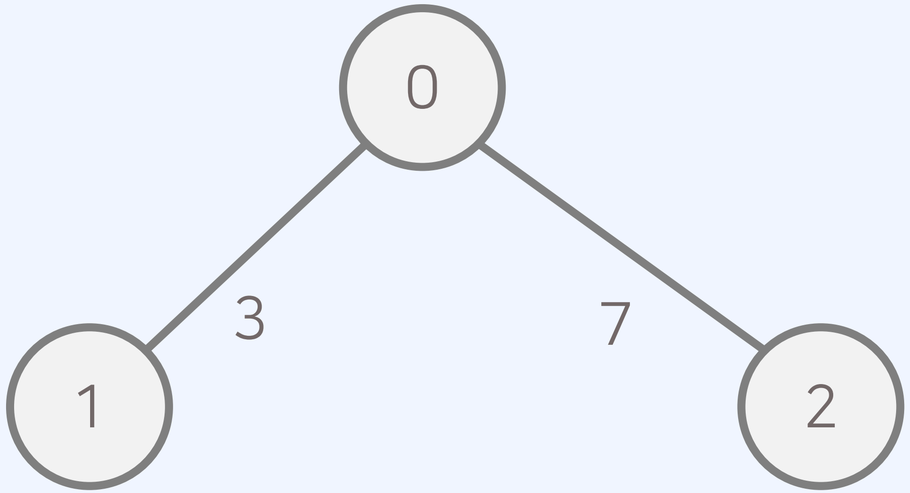
- 그래프(graph)란 사물을 정점(vertex)과 간선(edge)으로 나타내기 위한 도구다.
- 그래프는 **두 가지 방식**으로 구현할 수 있다.

1. **인접 행렬(adjacency matrix)**: 2차원 배열을 사용하는 방식
2. **인접 리스트(adjacency list)**: 연결 리스트를 이용하는 방식

인접 행렬(Adjacency Matrix)

JavaScript
자료구조
그래프의 표현

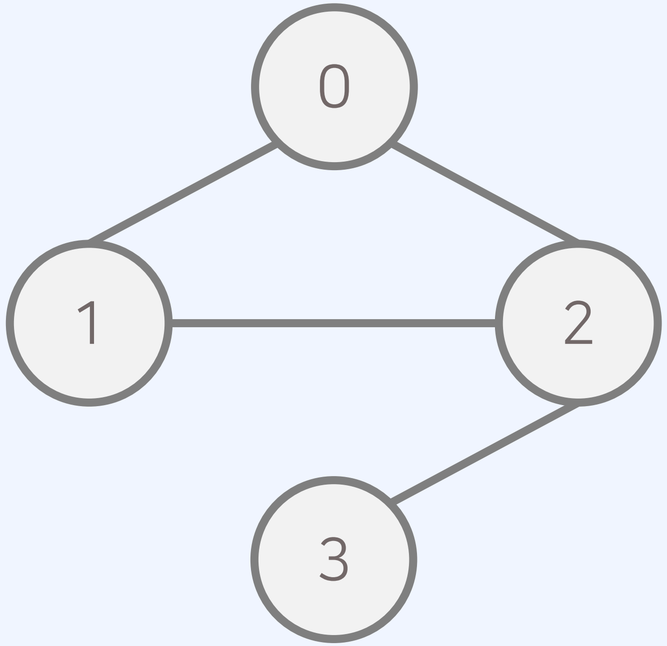
- 인접 행렬(adjacency matrix)에서는 그래프를 2차원 배열로 표현한다.



0	3	7
3	0	무한
7	무한	0

인접 행렬 - 무방향 무가중치 그래프

- 모든 간선이 방향성을 가지지 않는 그래프를 무방향 그래프라고 한다.
- 모든 간선에 가중치가 없는 그래프를 무가중치 그래프라고 한다.
- 무방향 비가중치 그래프가 주어졌을 때 연결되어 있는 상황을 **인접 행렬**로 출력할 수 있다.



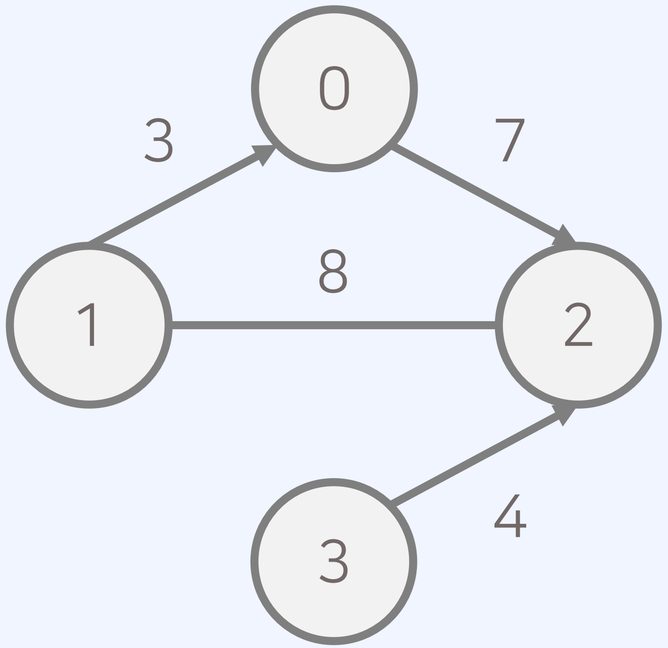

```
let graph = [
  [0, 1, 1, 0],
  [1, 0, 1, 0],
  [1, 1, 0, 1],
  [0, 0, 1, 0]
];

console.log(graph);
```

인접 행렬 - 방향 가중치 그래프

JavaScript
자료구조
그래프의 표현

- 모든 간선이 방향을 가지는 그래프를 방향 그래프라고 한다.
- 모든 간선에 가중치가 있는 그래프를 가중치 그래프라고 한다.
- 방향 가중치 그래프가 주어졌을 때 연결되어 있는 상황을 **인접 행렬**로 출력할 수 있다.

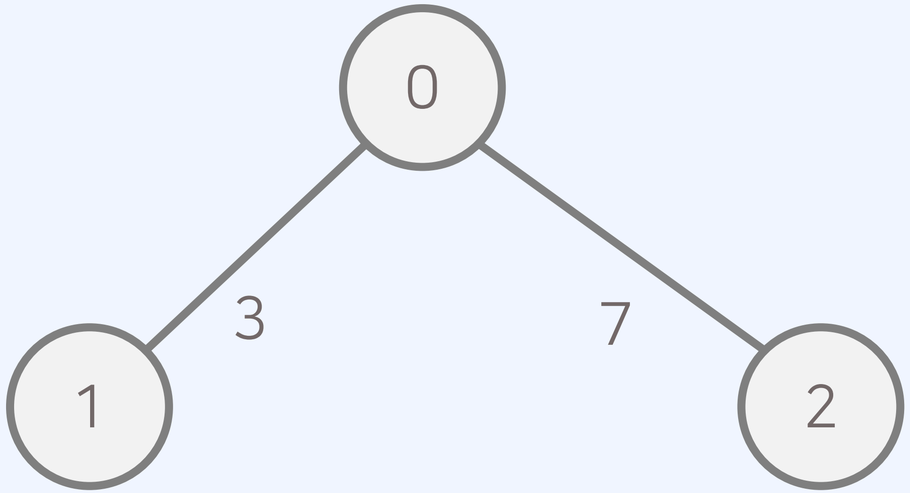



```
let graph = [
  [0, 0, 7, 0],
  [3, 0, 8, 0],
  [0, 8, 0, 0],
  [0, 0, 4, 0]
];

console.log(graph);
```

인접 리스트(Adjacency List)

- 인접 리스트(adjacency list)에서는 그래프를 리스트로 표현한다.



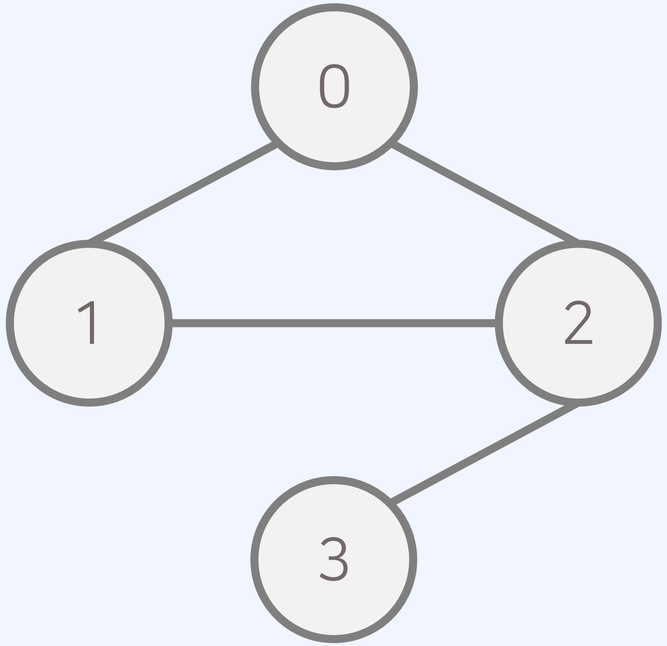
0: [(1, 3), (2, 7)]

1: [(0, 3)]

2: [(0, 7)]

인접 리스트 - 무방향 무가중치 그래프

- 모든 간선이 방향성을 가지지 않는 그래프를 무방향 그래프라고 한다.
- 모든 간선에 가중치가 없는 그래프를 무가중치 그래프라고 한다.
- 무방향 비가중치 그래프가 주어졌을 때 연결되어 있는 상황을 **인접 리스트**로 출력할 수 있다.

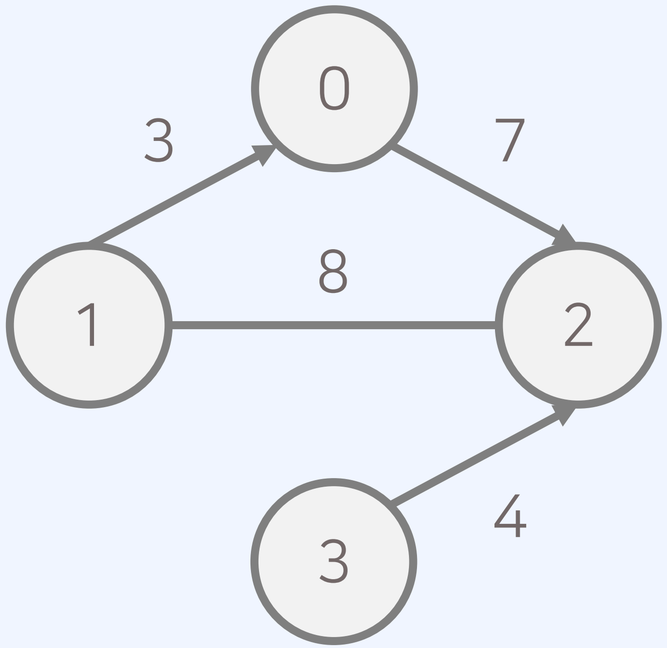



```
let graph = [
  [1, 2],
  [0, 2],
  [0, 1, 3],
  [2]
];

console.log(graph);
```

인접 리스트 - 방향 가중치 그래프

- 모든 간선이 방향을 가지는 그래프를 방향 그래프라고 한다.
- 모든 간선에 가중치가 있는 그래프를 가중치 그래프라고 한다.
- 방향 가중치 그래프가 주어졌을 때 연결되어 있는 상황을 **인접 리스트**로 출력할 수 있다.




```
let graph = [
  [(2, 7)],
  [(0, 3), (2, 8)],
  [(1, 8)],
  [(2, 4)]
];

console.log(graph);
```

그래프의 시간 복잡도

JavaScript
자료구조
그래프의 표현

1. **인접 행렬**: 모든 정점들의 연결 여부를 저장해 $O(V^2)$의 공간을 요구한다.
- 공간 효율성이 떨어지지만, 두 노드의 연결 여부를 $O(1)$에 확인할 수 있다.
2. **인접 리스트**: 연결된 간선의 정보만을 저장하여 $O(V+E)$의 공간을 요구한다.
- 공간 효율성이 우수하지만, 두 노드의 연결 여부를 확인하기 위해 $O(V)$의 시간이 필요하다.

	필요한 메모리	연결 여부 확인
인접 행렬	$O(V^2)$	$O(1)$
인접 리스트	$O(V+E)$	$O(V)$

인접 행렬 vs. 인접 리스트

- **최단 경로 알고리즘**을 구현할 때, 어떤 자료구조가 유용할까?
- 각각 근처의 노드와 연결되어 있는 경우가 많으므로, 간선 개수가 적어 **인접 리스트**가 유리하다.