그래프 (Graph)

01 그래프 이론

02 그래프의 활용

신 제 용

01 그래프 이론

가장 표현이 자유로운 그래프 자료구조에 대해 알아보고,
그래프에서 자료를 순회하는 방법을 학습합니다.

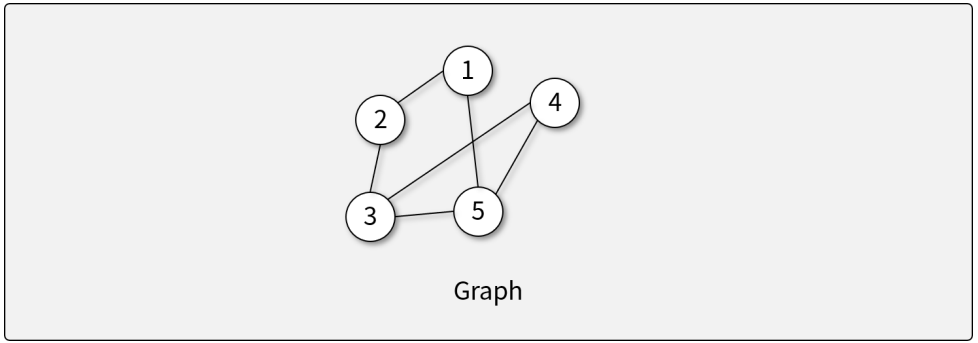
학습 키워드 - 그래프, 노드, 에지, 연결

Chapter 01
그래프 이론

ZEROBASE SCHOOL

그래프 (Graph)

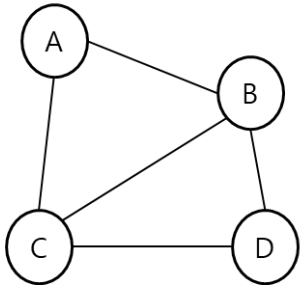
- 객체 간에 짝을 이루는 가장 유연한 자료 구조
- 정점(vertex, node, point)와 이를 잇는 간선(edge)로 구성
- 간선은 무향(undirected) 또는 유향(directed)일 수 있다.
- 간선에는 가중치(weight)가 있을 수 있으며, 이는 연결의 강도나 간선의 길이를 의미한다.



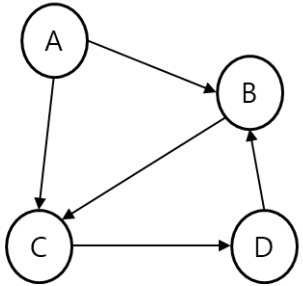
Graph

그래프의 구조

< Graph 예시 1 >



< Graph 예시 2 >

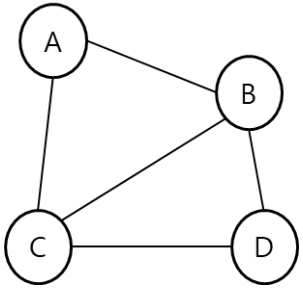


정점(Vertex): 각 노드
간선(Edge): 노드와 노드를 연결하는 선 (link, branch)
인접 정점(Adjacent vertex): 간선 하나를 두고 바로 연결된 정점
정점의 차수(Degree):
- 무방향 그래프에서 하나의 정점에 인접한 정점의 수
- 무방향 그래프 모든 정점 차수의 합 = 그래프 간선의 수 2배
진입 차수(In-degree): 방향 그래프에서 외부에서 오는 간선의 수
진출 차수(Out-degree): 방향 그래프에서 외부로 나가는 간선의 수
경로 길이(Path length): 경로를 구성하는데 사용된 간선의 수
단순 경로(Simple path): 경로 중에서 반복되는 정점이 없는 경우
사이클(Cycle): 단순 경로의 시작 정점과 끝 정점이 동일한 경우

그래프의 구현 (1)

- 인접 행렬 (Adjacency matrix)
  - 2차원 배열 이용
- 인접 행렬의 장단점
  - 간선 정보의 확인과 업데이트가 빠름 $O(1)$
  - 인접 행렬을 위한 메모리 공간 차지

< 그래프 >



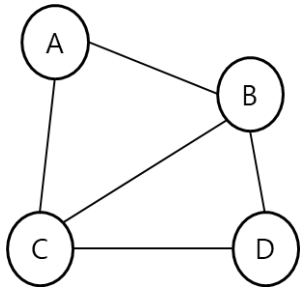
< 인접 행렬 >

	A [0]	B [1]	C [2]	D [3]
A [0]	0	1	1	0
B [1]	1	0	1	1
C [2]	1	1	0	1
D [3]	0	1	1	0

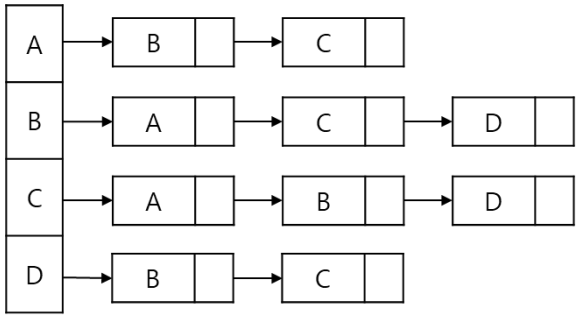
그래프의 구현 (2)

- 인접 리스트 (Adjacency list)
  - 연결리스트 이용
- 인접 행렬의 장단점
  - 메모리 사용량이 상대적으로 적고, 노드의 추가 삭제 빠름
  - 간선 정보 확인이 상대적으로 오래 걸림

< 그래프 >



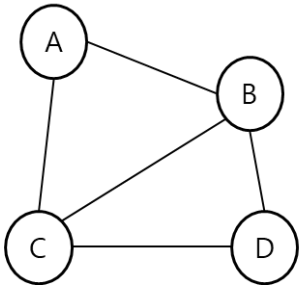
< 인접 리스트 >



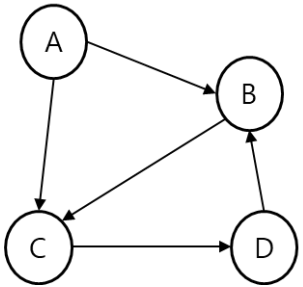
그래프의 종류 (1)

- 무방향 그래프 (Undirected graph)
  - 간선에 방향이 없는 그래프 (양방향 이동 가능)
  - 정점 A - B 간선의 표현: (A, B) = (B, A)
- 방향 그래프 (Directed graph)
  - 간선에 방향이 있는 그래프 (해당 방향으로만 이동 가능)
  - 정점 A → B 간선의 표현: <A, B> ≠ <B, A>

< 무방향 그래프 >



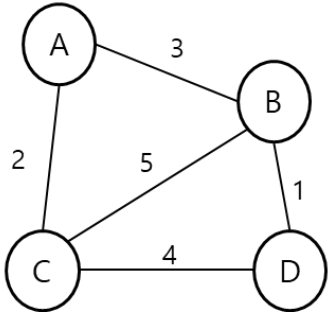
< 방향 그래프 >



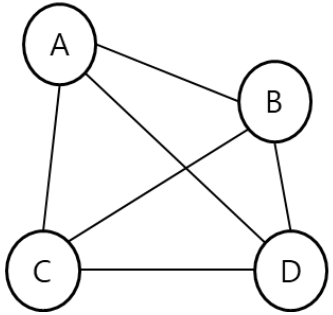
그래프의 종류 (2)

- 가중치 그래프 (Weighted graph)
  - 간선에 가중치가 있는 그래프 (이동 비용 or 연결 강도)
- 완전 그래프 (Complete graph)
  - 모든 정점이 서로 연결되어 있는 그래프
  - 정점이 N개일 경우, 간선의 수는 $N(N-1)/2$ 개

< 가중치 그래프 >

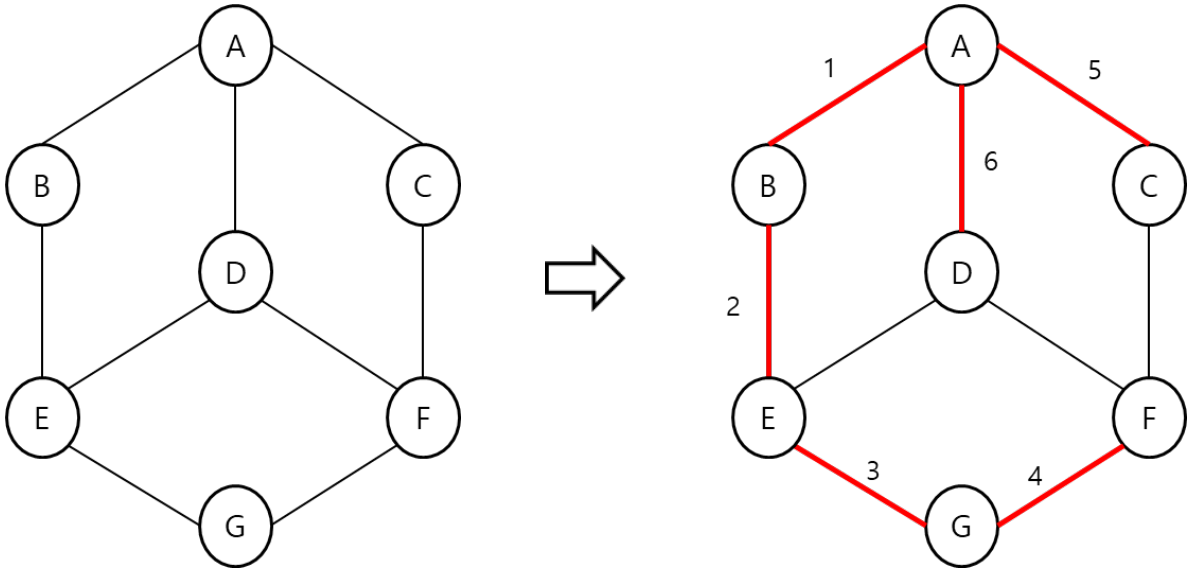


< 완전 그래프 >



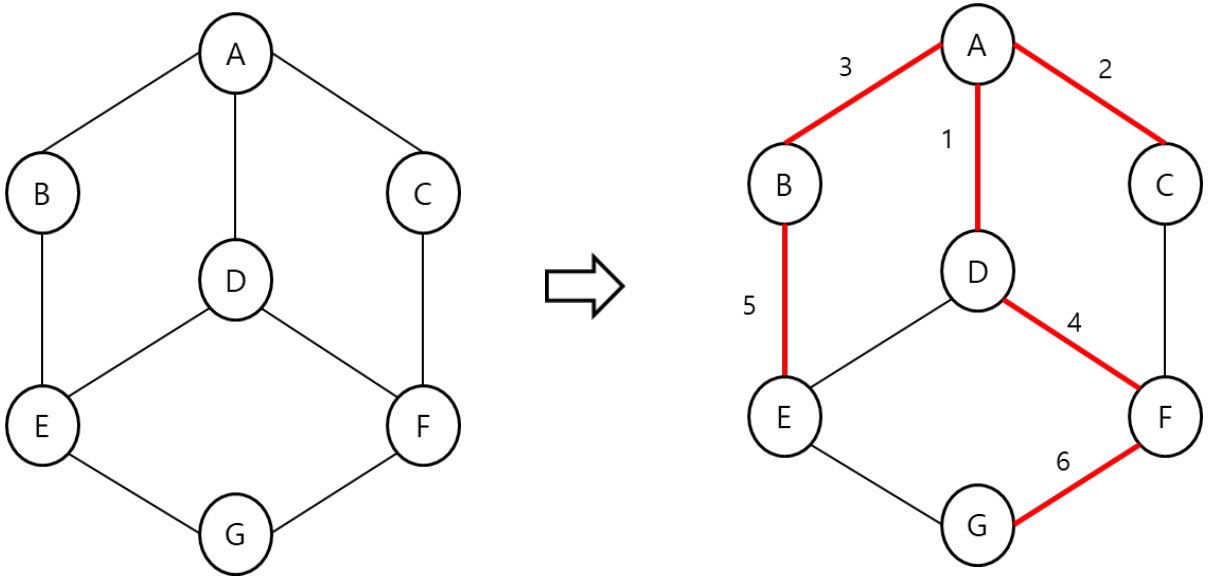
그래프의 탐색 (1)

- 깊이 우선 탐색 (Depth first search; DFS)
  - visited 배열과 스택 또는 재귀적으로 구현(Preorder)



그래프의 탐색 (2)

- 너비 우선 탐색 (Breath first search; BFS)
  - 각 노드에 방문했는지 여부를 체크할 배열과 큐 이용하여 구현



02 그래프의 구현

다음 챕터에서는 그래프로 자료를 표현해 보고,
직접 자료를 순회하는 코드를 구현해 봅시다.

Chapter 01
그래프 이론

02 그래프의 구현

그래프로 자료를 직접 표현해 보고, 기초적인 순회 방법을 구현하는 방법을 학습합니다.

학습 키워드 - 그래프, 순회, 구현

Chapter 02
그래프의 구현

ZEROBASE SCHOOL

```python
# 인접 리스트로 그래프의 bfs와 dfs를 구현하시오.

class Vertex:
    def __init__(self, value, adj_list=None):
        self.value = value
        self.adj_list = adj_list if adj_list else []


class Graph:
    def __init__(self):
        self.vertices = []

    def insert(self, value, adj_list):
        self.vertices.append(Vertex(value, adj_list))
        for adj_ind in adj_list:
            self.vertices[adj_ind].adj_list.append(len(self.vertic
es) - 1)

    def bfs(self, vert_ind, value):
        return False

    def dfs(self, vert_ind, value):
        return False
```

Chapter 02
그래프의 구현

ZEROBASE SCHOOL