그래프

객체들간의 상호관계를 표현하기 위해 고안된 알고리즘

어떤 상호관계도 표현할 수 있음

다룰 내용

- 그래프를 표현하는 두가지 방법
  - 인접행렬 표현
  - 인접 리스트 표현
- 가장 대표적인 그래프 알고리즘 : 깊이 우선탐색(DFS)
- 가중치 없는 그래프 상에서 두점 사이의 최단경로를 알기 유용한 알고리즘 : 너비 우선 탐색(BFS)
- 가중치가 있는 그래프 상에서 정점 사이의 최단 경로를 찾는 알고리즘들 [다익스트라, 벨만-포드, 플로이드의최단경로 알고리즘]
- 그래프의 각 간선 길이 뿐만 아니라 용량을 정의해놨을때 풀수 있는 문제 - 그래프 최대 유량 문제 -
  - ↳ 매우 유명한 최적화 문제, 그래프와 상관 없어보이는 문제도 풀 수 있음

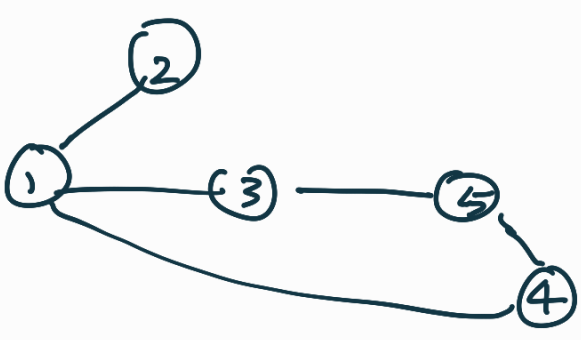
(포드-풀커슨 알고리즘)

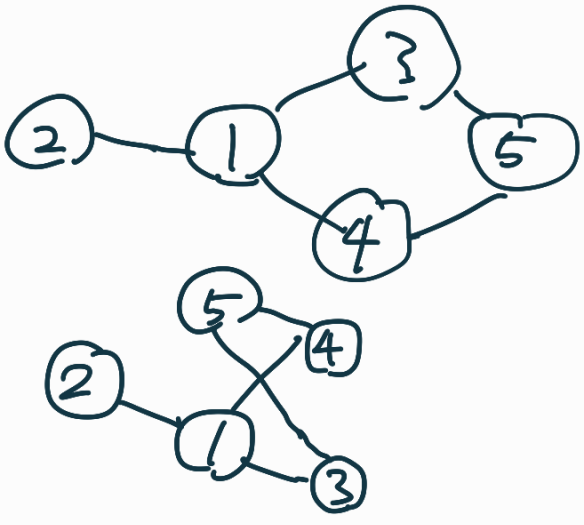
그래프의 표현과 정의

그래프의 정의

$G(V, E)$ 는 정점 집합 V 와 간선집합 E 로 구성된 자료구조

위치정보, 간선의 순서등은 포함되지 않음





⇑

3개다 같은 그래프

But 표현 대상에 따라 몇가지 변형된 형태가 있음

ex) 방향그래프, 가중치그래프, 다중그래프, 루트없는트리

이분그래프, · · · · ·

중요한 그래프

사이클 없는 방향그래프 (directed acyclic graph)

D A G

그래프에서 사용되는 가장 중요한 개념

- 경로
  - 일반적으로 하나의 정점이 1번만 지나는 경로인 단순 경로를 지칭함
- 경로中 시작점에서 끝나는 경로 = 사이클

그래프의 표현 방법

그래프 = 트리보다 정적임 (구조의 변경이 어려워도 간단하고 메모리를 적게 차지하는 방법 선택)

일반적으로

↕ 인스턴스 대신 배열로 사용하는 식

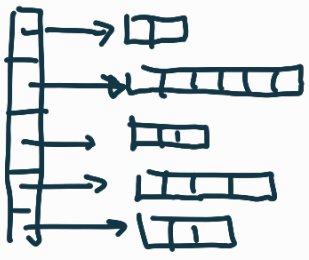
간선의 저장 방식에 따른 2가지 구현

1. 인접리스트

가중치 등의 표현 구조체로 { int vertex, int weight }

각 정점마다 해당 정점에서 나가는 간선의 목록을 저장해 연결리스트의 형태로 구현

vector<list<int>> adjacent

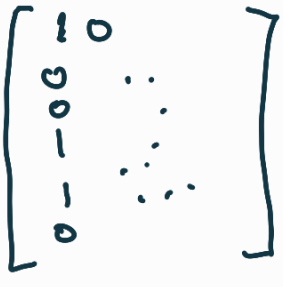


2. 인접행렬

연결리스트를 뒤져야하는 인접리스트 방식의 해결

|V| x |V| 의 2차원 배열로 표현

adjacent[x][y]가 1이면 x → y로의 간선이 있다. 0이면 없다.



정 반대의 특성

1. 단점이 2. 장점임
2. 단점이 1. 장점임

적절히 선택해야함

인접행렬 방식의 장점: 정점의 번호 u, v로 간선여부를 바로 알수있음

단점: |V|x|V|의 공간을 항상 차지함 (밀집그래프유용)

인접리스트의 장점 : 실제간선수만큼만 공간을 차지함 |V|+|E|
(희소 그래프에서 유용)

단점 : 간선(u, v)의 여부를 바로 알 수 없음

암시적인 그래프 표현

미로찾기 + 너비우선탐색으로 쉽게 풀림 But 일일이 그래프로 만들어야함 → 번거로움

배열을 만들고 적당히 싱크를 맞추는 방식
(정점이 완성되면)
↳ 상하 좌우를 검사하면된다.

그래프가 개념적 실제로는 일부만 사용하는 예시

이런 암시적 그래프 사용

↳ 알고리즘과 변환과정이 합쳐지고 코드가 복잡해짐 → 그래프에 대한 복잡한 연산을 한다면 그래프로 만들어라.

깊이 우선 탐색

모든 정점을 특정 순서에 따라 방문하는 알고리즘

- 탐색