

그래프

객체들간의 상호관계를 표현하기 위해 고안된 알고리즘

어떤 상호관계도 표현할 수 있음

다른 내용

- 그래프를 표현하는 두가지 방법

↳ 인접행렬 표현

↳ 인접 리스트 표현

- 가장 대표적인 그래프 알고리즘 : 깊이 우선 탐색(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, 플로이드의 최단 경로 알고리즘]

- 그래프의 각 간선 길이 뿐만 아니라 용량을 정의해줄 때 풀수 있는 문제 - 그래프 최대 용량 문제 -

↳ 매우 유명한 최적화 문제, 그래프와 상관 없어보이는 문제도 풀 수 있음

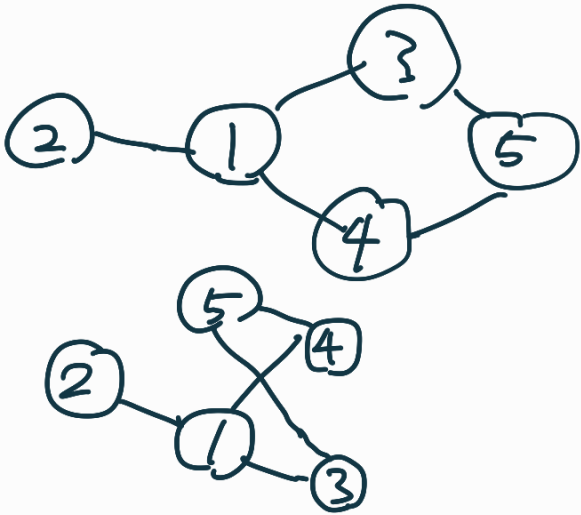
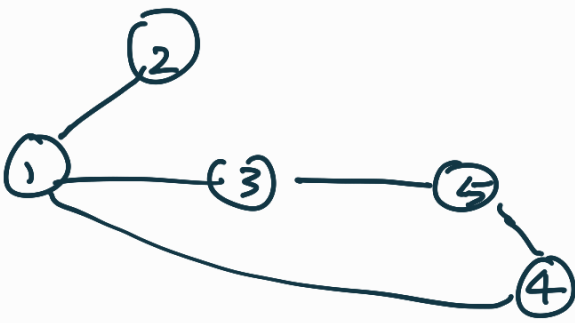
(포드-풀커슨 알고리즘)

그래프의 표현과 정의

그래프의 정의

$G(V, E)$ 는 정점 집합 V 와 간선 집합 E 로 구성된 자료 구조

위치정보, 간선의 순서 등은 포함되지 않음



3개다 같은 그래프

But 표현 대상에 따라 몇가지 변형된 형태가 있음

ex) 방향 그래프, 가중치 그래프, 다중 그래프, 루트 없는 트리
이분 그래프,

중요한 그래프

사이클 없는 방향 그래프 (directed acyclic graph)
DAG

그래프에서 사용되는 가장 중요한 개념

- 경로

→ 일반적으로 하나의 정점이 1번만 지나는 경로인
단순 경로를 지칭함

경로 중 시작점과 끝나는 경로 = 사이클

그래프의 표현 방법

그래프 = 트리보다 정적임 (구조의 변경이 어려워도 간단하고
메모리를 적게 차지하는 방법 있음)
일반적으로
인스턴스 대신 배열로 사용하는 식

간선의 저장 방식에 따른 2가지 구현

1.

인접리스트

가중치, 등지표 표현
구조체로
 $\begin{cases} \text{int vertex} \\ \text{int weight} \end{cases}$

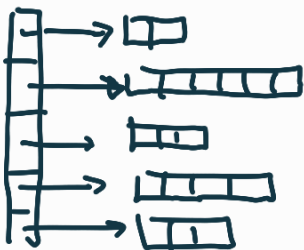
각

정점마다

해당 정점에서 나가는
간선의 목록을 저장해

연결리스트의
형태로 구현

$\text{vector} \langle \text{list} \langle \text{int} \rangle \rangle$
adjacent

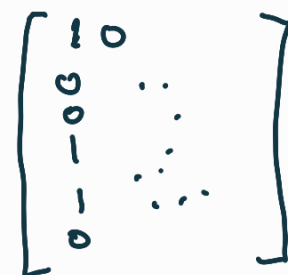


2.

인접행렬

연결리스트를 두/저장하는 인접리스트 방식에
대조

$|V| \times |V|$ 의 2차원 배열로 표현
 $\text{adjacent}[x][y]$ 가
1이면 $x \rightarrow y$ 노드의
간선이 있다.
0이면 없다.



정 반대의 특성

1. 단점이 2. 장점임
2. 단점이 1. 장점임

적절히 선택해야함

인접행렬 방식의 장점: 정점의 번호 u, v 로 간선여부를 바로 알 수 있음

단점: $|V| \times |V|$ 의 공간을 항상 차지함 (밀집그래프유형)

인접리스트의 장점: 실제/간선속판공판공간을 차지함 $|V| + |E|$
(희소 그래프에서 유용)

단점: 간선 (u, v) 의 여부를 바로 알 수 없음

앞서적인 그래프 표현

미로찾기 + 네비게이션 색으로 쉽게 풀림 13번
일일: 그래프를 방문한
번거로움

배열을 만들고 정점이 싱크를 맞추는 방식
(정점이 원소면)

→ 상하 조너를 생각하면 된다.

그래프가 커지면 신제코는 일복만 서둘러는 예시

이런 앞서적 그래프 사용

→ 알고리즘과 변환과정이 합쳐지고 코드가
복잡해질 → 그래프에 대한 복잡한 연산을 할 때는
그래프를 만들어서.