깊이 우선 탐색

NEXTERS

그래프의 정의

그래프 G(V, E)는 어떤 자료나 개념을 표현하는 정점들의 집합 V와 이들을 연결하는 간선(edge)들의 집합 E로 구성된 자료 구조

정점의 위치 정보나 간선의 순서 등은 그래프의 정의에 포함되지 않음

그래프의 종류

방향 그래프 ,유향 그래프 : 두 정점 u, v 있을 때 u 와 v로 가는 간선과 v에서 u로 가는 간선이 서로 다른 간선

가중치 그래프 : 간선마다 가중치가 존재하는 그래프

다중 그래프 : 두 정점 사이에 두 개 이상의 간선이 있을 때

단순 그래프 : 한 개의 간선만 있을 때

이분 그래프 : 그래프의 정점들을 겹치지 않는 두 개의 그룹으로 나눠서 서로 다른 그룹에 속한 정점들 간에만 간선이 존재 할 수 있도록 만들 수 있는 그래프

그래프의 표현 방법

인접 리스트 표현 : 각 정점마다 해당 정점에서 나가는 간선의 목록을 리스트로 저장해서 표현

- `vector<list<int>> adjacent`

인접 행렬 표현 : |V| X |V| 크기의 행렬, 2차원 배열을 이용해 그래프의 간선 정보를 저장

- `vector<vector<bool>> adjacent`

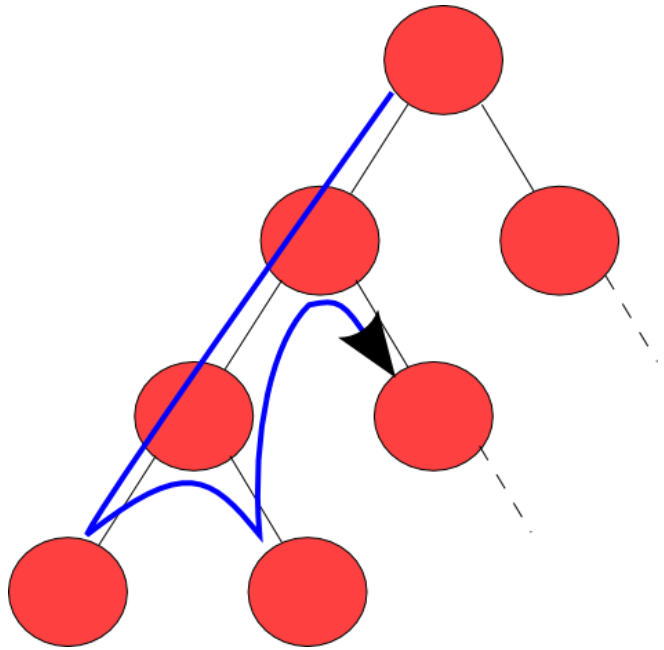
깊이 우선 탐색 (1)

현재 정점과 인접한 간선들을 하나씩 검사하면서 아직 방문하지 않은 정점이 있으면 그 정점으로 이동한다. 더 이상 이동할 정점이 없다면 마지막에 따라왔던 간선을 따라 뒤로 돌아간다.

인접리스트 사용 시 모든 간선을 한 번씩 돌아보기 때문에 O(V) + 모든 간선을 확인해보기 때문에 O(E) = O(V+E) 인접행렬 사용 시 두 정점 사이에 간선이 있는가를 확인해야 하기 때문에 O(V)가 걸린다. dfs가 V번 호출되기 때문에 O(V^2)

깊이 우선 탐색 (2)

깊이 제한에 도달할 때까지 목표노드가 발견되지 않으면 최근에 방문한 부모노드로 되돌아와서, 부모노드에 이전과는 다른 경로를 적용하여 새로운 자식노드를 생성한다.



위상 정렬

의존성 그래프(DAG: Directed Acycle Graph)

정점 간에 의존 관계를 간선으로 표현한 그래프를 의존성 그래프라 한다. 의존성 그래프는 사이클이 없다.

깊이 우선 탐색을 한 결과를 뒤집으면 위상 정렬 결과를 얻을 수 있다.

오일러 서킷

그래프의 모든 간선을 정확히 한 번씩 지나서 시작점으로 돌아오는 경로

모든 정점의 차수가 짝수여야 한다. 들어오는 간선 나가는 간선의 갯수가 같지 않으면 제자리로 갈 수 없기 때문.

오일러 트레일

그래프의 모든 간선을 정확히 한 번씩 지나지만 시작점과 끝점이 다른 경로

a -> ... -> b의 오일러 트레일을 찾는다고 할 때 b -> a 간선을 만들어 오일러 서킷을 구한 후 b -> a를 끊으면 된다.

위와 같이 오일러 서킷과 조건이 비슷하다. b -> a 간선을 추가한 상태로 오일러 서킷이 되려면 모든 정점의 차수는 짝수가 되어야 한다. b -> a는 제외해야 하므로 시작, 끝점은 홀수여야 한다.

문제 - 고대어 사전

알파벳의 순서들이 주어졌을 때 각 알파벳의 상대적 순서를 알아내는 방법

깊이 우선 탐색을 한 뒤 이를 뒤집어 위상정렬을 구현 한다.

✓문제

https://algospot.com/judge/problem/read/DICTIONARY

✓풀이코드

https://github.com/Nexters/algorithmStudy/blob/master/seokjoong/Chapter28/DICTIONARY.cpp

문제 – 단어 제한 끝말잇기

주어진 단어를 모두 사용하고 끝말잇기를 마칠 수 있는지 알아보는 문제

첫글자와 마지막 글자를 이용하여 그래프를 표시하고 이 그래프의 오일러 트레일 혹은 오일러 서킷을 찾으면 된다.

✓문제

https://algospot.com/judge/problem/read/WORDCHAIN

✓풀이코드

https://github.com/Nexters/algorithmStudy/blob/master/seokjoong/Chapter28/WordChain.cpp