자료구조 (Data Structure
8주차: 그래프

선형 (일차원) 자료구조 정리

- 리스트: 데이터를 추가하면 순서가 유지되는 구조
- 스택: 가장 최신 데이터만 꺼낼 수 있는 구조
- 큐: 가장 오래된 데이터만 꺼낼 수 있는 구조

계층적 자료구조 정리

- 트리: 루트로부터 부모 자식 관계로 연결된 구조
- 이진 트리: 모든 노드의 자식 노드 숫자가 두개 이하인 트리
- 최대힙: 모든 노드의 데이터가 자식 노드 데이터들보다 큰 이진 트리
- 힙 정렬: 최대힙으로 정렬되지 않은 구간의 최대값을 반복적으로 꺼내서 정렬하는 방법

이번 시간 목차

- 트리보다 더 일반적인 자료구조 소개
- 트리보다 더 일반적인 자료구조에서 반복하는 방법
- 트리보다 더 일반적인 자료구조 예제

트리보다 더 일반적인 자료 구조

- 리스트의 노드마다 리스트를 가지고 있다면?
- 노드 사이에 계층이 없이 자유롭게 연결되어 있다면?
- 예시) 인터넷, 도로

트리보다 더 일반적인 자료 구조

- 노드들은 선으로 연결. 다대다 관계
- 루트가 없기 때문에 노드 리스트 저장

그래프란?

- 그래프 정의: 각 노드에 다른 노드들이 연결된 구조

그래프

- 그래프는 노드와 연결선을 모은 것이다

  - vertexes: 전체 노드 리스트 (타입: 그래프 노드)

그래프 노드 타입

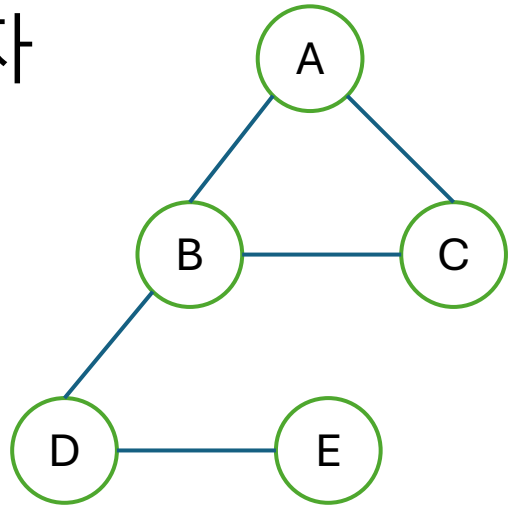
- 데이터와 이웃 노드 정보의 묶음

  - adj: 이웃 노드 리스트

  - value: 저장된 데이터

그래프 예시

- 다음과 같은 그래프를 만들어보자



그래프 노드 방문

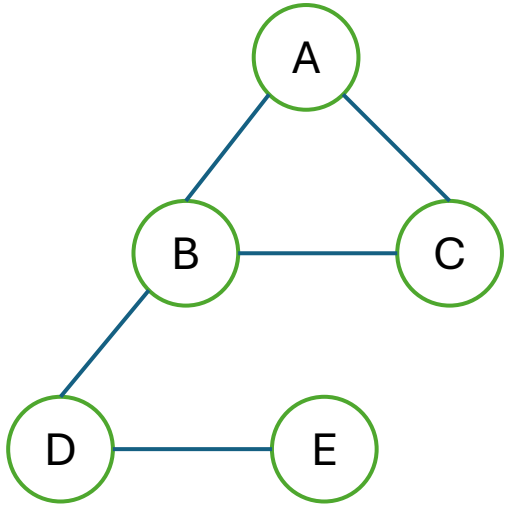
- 목적: 특정 노드와 연결된 모든 노드에 대해 반복하기

- 깊이 우선: 한 이웃의 모든 이웃들을 먼저 방문
  데이터 처리 타이밍:
  - Preorder: 방문전 처리할 작업
  - Postorder: 방문 후 처리할 작업

- 너비 우선: 모든 이웃을 먼저 방문

그래프 노드 방문 예시

- 다음 그래프에서 깊이 우선 방문 순서는?
- 너비 우선 방문 순서는?



깊이 우선 방문 구현

- 그래프 노드 타입에 추가할 정보:

  - status : 방문 상태 (방문 전, 중, 후)

깊이 우선 방문 구현 - 사전 작업

함수 DFS_PREP(그래프 G):

- 반복: G->vertexes의 모든 노드들 u에 대해,

  u->status = 방문전

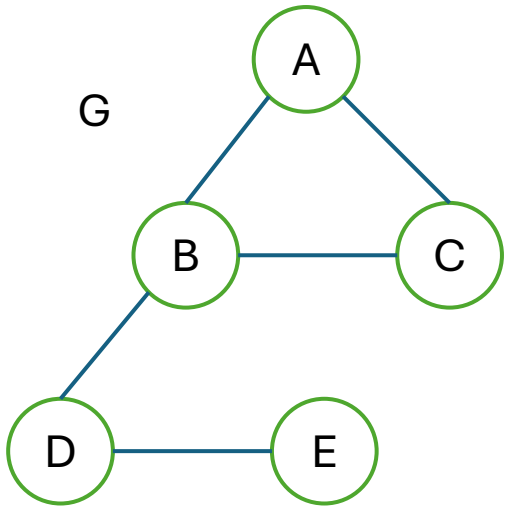
깊이 우선 방문 구현 - 방문 함수

함수 DFS_VISIT(시작 노드 u):

- u->status = 방문중
- pre_task(u)
- 반복: u->adj의 모든 노드들 v에 대해,
  만약: v->status가 방문전이면,
  DFS_VISIT(v)
- u->status = 방문후
- post_task(u)

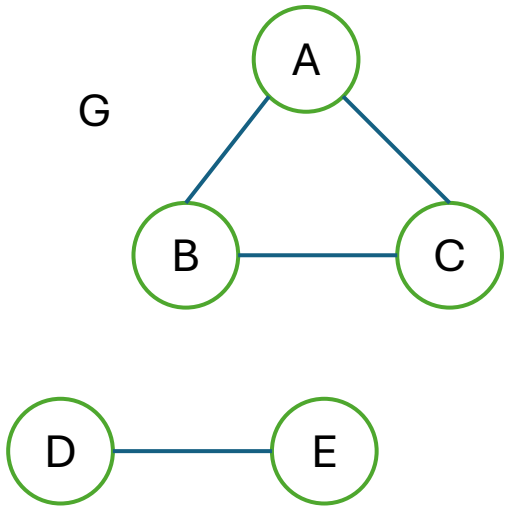
깊이 우선 방문 구현 - 예시

- DFS_PREP(G)를 한 후, DFS(A)를 했을때, pre_task, post_task의 실행 순서는?



깊이 우선 방문 구현 예제 - 그룹핑

- 다음 그래프를 연결된 노드들로 그룹핑하면?



깊이 우선 방문 구현 예제 - 그룹핑

- 그래프 노드 타입에 group 항목 추가 (초기값 0)
- 전역 변수: n (초기값 0)
- pre_task(u): u->group = n
- post_task(u): 없음

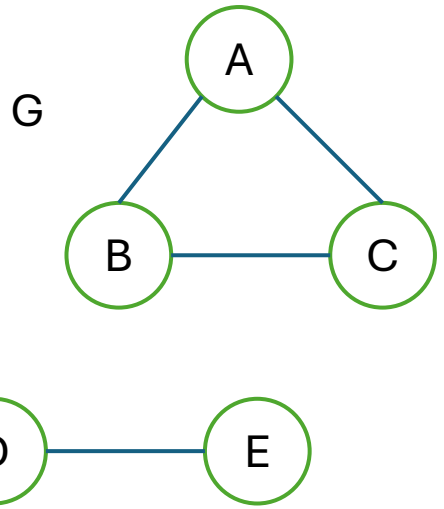
깊이 우선 방문 구현 예제 - 그룹핑

함수 GROUPING_CONNECTED(그래프 G):

- DFS_PREP(G)
- n = 0
- 반복: G의 모든 노드 u에 대해,
  만약: u->status가 방문전이면,
  n += 1
  DFS_VISIT(G, u)

깊이 우선 방문 구현 예제 - 그룹핑

- 다음 그래프에서 GROUPING_CONNECTED()의 실행 과정은?



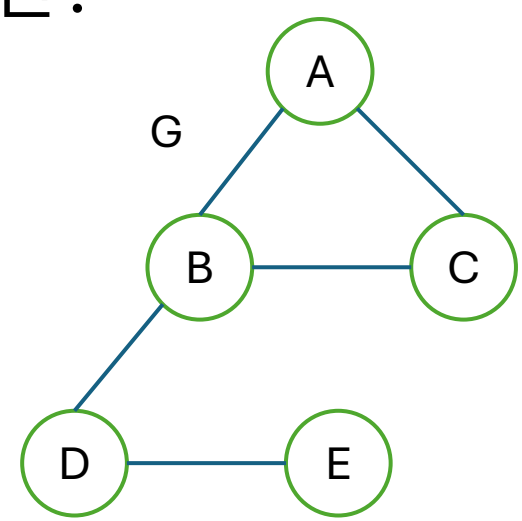
깊이 우선 방문시 연결 선의 분류

- 트리 선: 이웃이 방문전으로 DFS_VISIT이 실행된다
- 역방향 선: 이웃이 방문중이어서 건너뛰는 선

- 무방향 그래프에서는 이웃이 방문 후인 경우가 없다

깊이 우선 방문시 연결 선의 분류

- 다음 그래프의 선들을 A에서 깊이 우선 탐색시 트리 선과 역방향 선으로 구분하면?

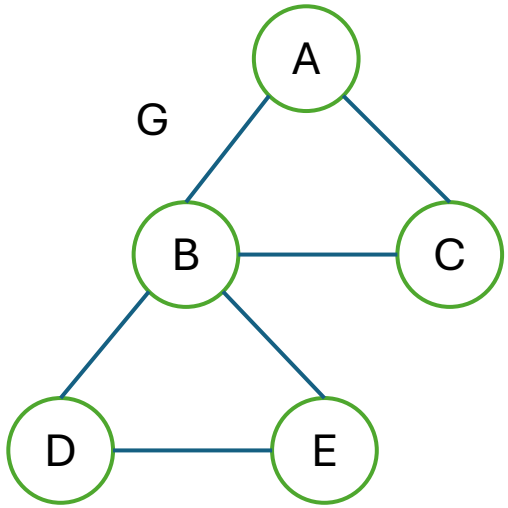


예제: 단절점 찾기

- 단절점: 제거하면 그래프가 둘로 나눠지는 노드
- 예시) 네트워크에서 모든 트래픽이 거쳐가는 서버

예제: 단절점 찾기

- 다음 그래프에서 단절점은?

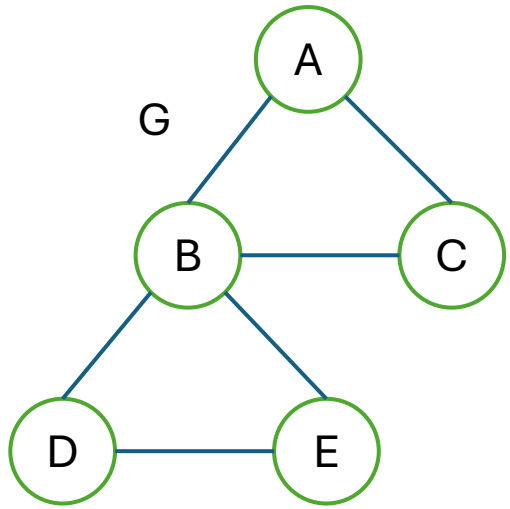


예제: 단절점 찾기

- 어떤 노드에서 깊이 우선 탐색을 시작했는데, 트리 선이 두개 이상이면 그 노드는 단절점이다

예제: 단절점 찾기

- 다음 그래프에서 깊이 우선 탐색을 시작했을 때 트리 선이 두개 이상인 노드는?

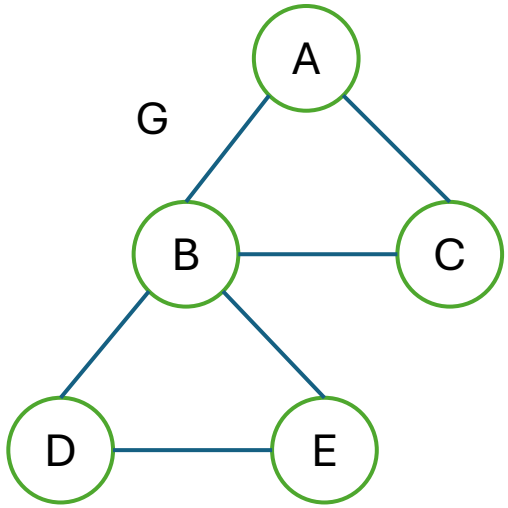


예제: 단절점 찾기

- 깊이 우선 탐색 중에, 노드 x의 이웃 노드가 주변에 x의 조상으로 가는 역방향 선이 없으면 x는 단절점이다

예제: 단절점 찾기

- 다음 그래프에서 깊이 우선 탐색 중에, 자신의 조상으로 가는 역방향 선이 하나도 없는 이웃 노드가 존재하는 노드는?



예제: 단절점 찾기

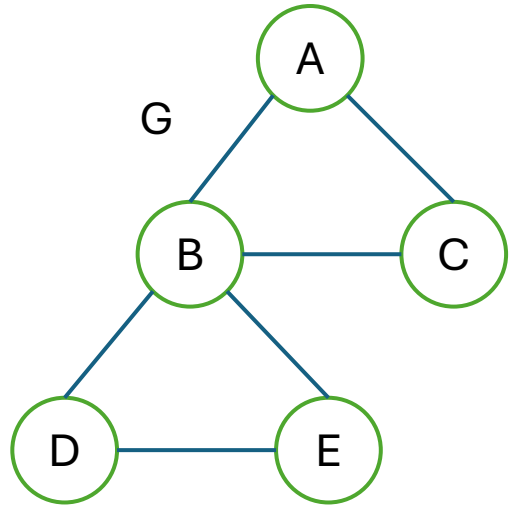
- 그 외의 노드들은 모두 단절점이 아니다

예제: 단절점 찾기

- 그래프 노드 타입에 추가할 항목:
  - is_ap: 단절점인지 여부 (초기값 0)
  - visited_time: 방문된 순서 (초기값 0)
  - lowest_back: 트리 선 + 한개의 역방향 선으로 연결된 가장 먼 조상
- 전역 변수:
  - time: 방문 순서를 위한 전역 시간 (초기값 0)

예제: 단절점 찾기

- 다음 그래프에서 A에서 깊이 우선 탐색을 시작할 때, 각 노드의 is_ap, visited_time, lowest_back은?



예제: 단절점 찾기

함수 FIND_AP(그래프 G):

- time = 0
- 반복: G의 모든 노드 u에 대해,

    u->is_ap = 0

    u->visited_time = 0

    u->lowest_back = 무한대
- 반복: G의 모든 노드 u에 대해,

    만약: u->visited_time이 0이면,

        DFS_VISIT_ROOT(u)

예제: 단절점 찾기

함수 DFS_VISIT_ROOT(시작 노드 u):

- time += 1
- u->visited_time = u->lowest_back = time
- root_children_count = 0
- 반복: u->adj의 모든 노드들 v에 대해,
  - 만약: v->visited_time이 0이면,
    - root_children_count += 1
    - DFS_VISIT_NONROOT(v, u)
- 만약: root_children_count > 1이면,
  - u->is_ap = 1

예제: 단절점 찾기

함수 DFS_VISIT_NONROOT(방문 노드 u, 부모 노드 p):

- time += 1
- u->visited_time = time
- u->lowest_back = time

- 반복: u->adj의 모든 노드들 v에 대해,
  만약: v == p면,
  다음 반복으로

예제: 단절점 찾기

만약: v->visited_time이 0이 아니면,

    만약: v->visited_time < u->lowest_back

        u->lowest_back = v->visited_time

만약: v->visited_time이 0이면,

    DFS_VISIT_NONROOT(v, u)

    만약: v->lowest_back < u->lowest_back

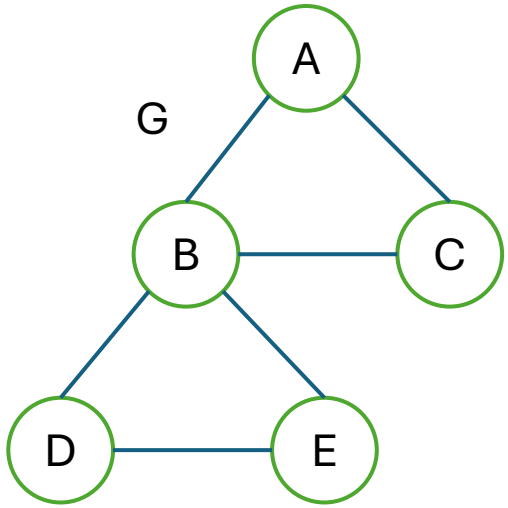
        u->lowest_back = v->lowest_back

    만약: v->lowest_back >= u->visited_time

        u->is_ap = 1

예제: 단절점 찾기

- 다음 그래프에서 FIND_AP의 진행과정은?



요약

- 그래프: 노드들이 선으로 연결된 구조
- 깊이 우선 탐색: 한 이웃의 이웃들을 먼저 탐색하고, 다른 이웃에 대해 반복
- 단절점: 제거하면 그래프가 둘로 나눠지는 노드

2021년 기말고사 문제

V= {1,2,3,4,5,6,7,8},

E= {(1,2), (1,6), (2,3), (2,4), (4,5), (4,7), (5,7), (5,8)}

1) 그래프 G를 그림으로 나타내시오.

2) dfs(1)를 수행하는 과정에서 edge들로 만들어지는 tree를 그림으로 나타내시오

3) 각 노드의 dfn과 low 값을 적으시오.

4) Articulation point에 해당하는 vertex가 어떤 것인지 적으시오.