TREE

알고리즘 스터디

최아현

목차

1. Tree

- 트리 특징
- 그래프와 트리의 차이
- 트리 용어

2. Tree 종류

3. 트리 구현 방법

4. 문제

1.1. TREE

- 노드들이 연결된 비선형 계층적 자료구조
- 트리내에 또 다른 트리가 있는 재귀적 자료구조
- DAG(Directed Acyclic Graphs, 방향성이 있는 비순환 그래프)의 한 종류이다. '최소 연결 트리' 라고도 불린다.

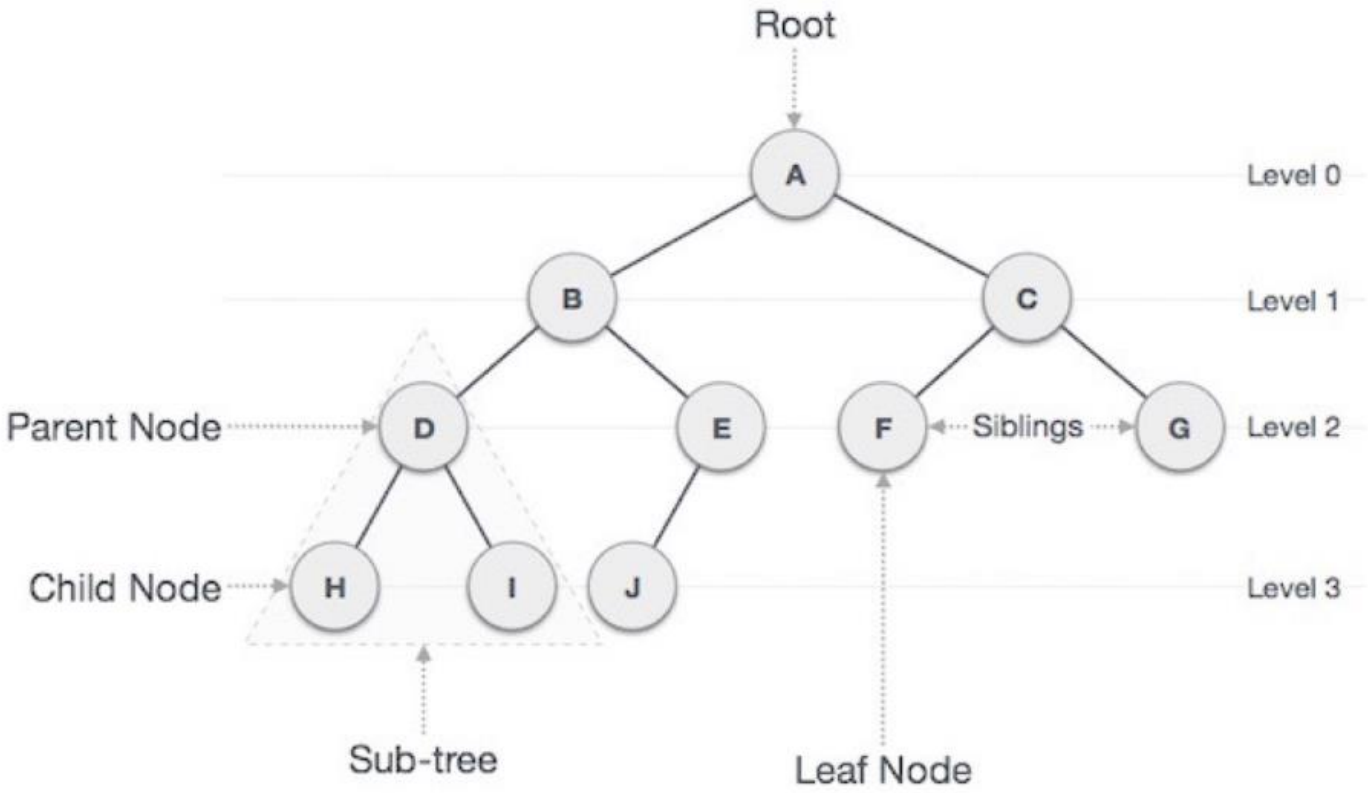
- 노드들과 노드들을 연결하는 간선들로 구성
- 노드가 N개인 트리는 항상 N-1개의 간선(edge)을 가진다
- 각 노드는 어떤 자료형으로도 표현 가능
- loop, circuit 등 사이클(cycle)이 존재 X

- 한 개의 루트 노드만이 존재하며 모든 자식 노드는 한 개의 부모 노드만을 가진다.
- 부모-자식 관계이므로 흐름은 top-bottom 아니면 bottom-top으로 이루어짐
- 순회는 전위, 중위, 후위 순회로 이루어진다. 이 3가지 모두 DFS/BFS 안에 있다.
- 트리는 이진 트리, 이진 탐색 트리, 균형 트리(AVL 트리, red-black 트리), 이진 힙(최대힙, 최소힙) 등이 있다.

1.2. 그래프와 TREE의 차이

	그래프	트리
정의	노드(node)와 그 노드를 연결하는 간선(edge)을 하나로 모아 놓은 자료 구조	그래프의 한 종류 DAG(Directed Acyclic Graph, 방향성이 있는 비순환 그래프)의 한 종류
방향성	방향 그래프(Directed), 무방향 그래프(Undirected) 모두 존재	방향 그래프(Directed Graph)
사이클	사이클(Cycle) 가능, 자체 간선(self-loop)도 가능, 순환 그래프(Cyclic), 비순환 그래프(Acyclic) 모두 존재	사이클(Cycle) 불가능, 자체 간선(self-loop)도 불가능, 비순환 그래프(Acyclic Graph)
루트 노드	루트 노드의 개념이 없음	한 개의 루트 노드만이 존재, 모든 자식 노드는 한 개의 부모 노드 만을 가짐
부모-자식	부모-자식의 개념이 없음	부모-자식 관계 top-bottom 또는 bottom-top으로 이루어짐
모델	네트워크 모델	계층 모델
순회	DFS, BFS	DFS, BFS안의 Pre-, In-, Post-order
간선의 수	그래프에 따라 간선의 수가 다름, 간선이 없을 수도 있음	노드가 N인 트리는 항상 N-1의 간선을 가짐
경로	-	임의의 두 노드 간의 경로는 유일
예시 및 종류	지도, 지하철 노선도의 최단 경로, 전기 회로의 소자들, 도로(교차점과 일방 통행길), 선수 과목	이진 트리, 이진 탐색 트리, 균형 트리(AVL 트리, red-black 트리), 이진 힙(최대힙, 최소힙) 등

1.3. TREE와 관련된 용어



1.3. TREE와 관련된 용어

용어	의미
루트 노드(root node)	부모가 없는 노드, 트리는 하나의 루트 노드만을 가짐
단말 노드(leaf node)	자식이 없는 노드, '말단 노드' 또는 '잎 노드'라고도 부름
내부(internal) 노드	단말 노드가 아닌 노드
간선(edge)	노드를 연결하는 선 (link, branch 라고도 부름)
형제(sibling)	같은 부모를 가지는 노드
노드의 크기(size)	자신을 포함한 모든 자손 노드의 개수
노드의 깊이(depth)	루트에서 어떤 노드에 도달하기 위해 거쳐야 하는 간선의 수
노드의 레벨(level)	트리의 특정 깊이를 가지는 노드의 집합
노드의 차수(degree)	하위 트리 개수 / 간선 수 (degree) = 각 노드가 지닌 가지의 수
트리의 차수(degree of tree)	트리의 최대 차수
트리의 높이(height):	루트 노드에서 가장 깊숙히 있는 노드의 깊이

2. TREE 종류

- 이진 트리
- 완전 이진 트리
- 이진탐색트리
- 균형트리
- 이진 힙
- 최소 신장 트리

+사용 사례

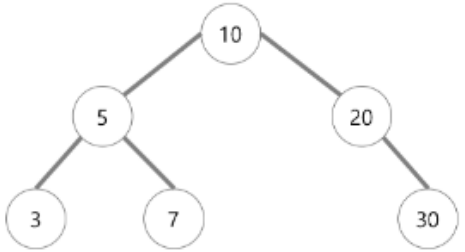
2.1. 이진 트리

- 각 노드가 최대 두 개의 자식을 갖는 트리
- 모든 트리가 이진트리는 x
- 이진트리순회

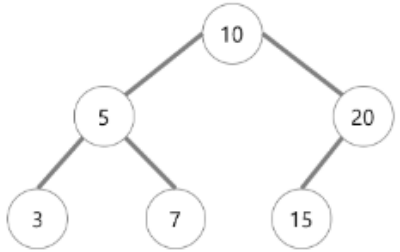
- 중위순회 : 왼쪽가지 -> 현재노드 -> 오른쪽가지
- 전위순회 : 현재노드 -> 왼쪽가지 -> 오른쪽가지
- 후위순회 : 왼쪽가지 -> 오른쪽가지 -> 현재노드

2.2. 완전 이진 트리(BINARY SEARCH TREE)

- 트리의 모든 높이에서 노드가 꽉 차있는 이진 트리
- 마지막 레벨을 제외하고 모든 레벨이 완전히 채워짐
- 마지막 레벨은 꽉 차 있지 않아도 되자만 노드가 왼쪽에서 오른쪽으로 채워져야 한다.마지막 레벨 h에서 (1 ~ 2h-1)개의 노드를 가질 수 있다.
- Tree 자료구조 중 가장 많이 사용되는 자료구조



완전 이진 트리가 아님



완전 이진 트리가 맞음

2.3. 이진 탐색 트리

- 순서화된 이진 트리
- 노드의 왼쪽 자식은 부모의 값보다 작은 값을 가져야 하며 노드의 오른쪽 자식은 부모의 값보다 큰 값을 가져야 함.

모든 왼쪽 자식들 <= n < 모든 오른쪽 자식들 (모든 노드 n에 대해서 반드시 참)

2.4. 균형 트리

- O(logN) 시간에 insert와 find를 할 수 있을 정도로 균형이 잘 잡혀 있는 경우
- Ex) 레드-블랙 트리, AVL 트리

2.5. 이진 힙(최소 힙과 최대 힙)

- 최소힙

- 트리의 마지막 단계에서 오른쪽 부분을 뺀 나머지 부분이 가득 채워져 있는 완전 이진 트리이며, 각 노드의 원소가 자식들의 원소보다 작다.

- N개의 힙에 들어가 있으면 높이는 log(N)

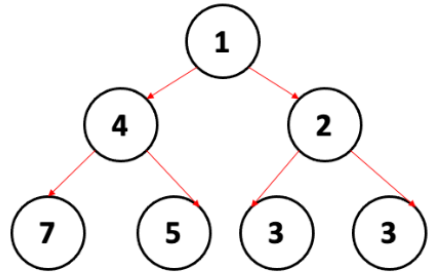


그림 5) 최소 힙의 예제

- 최대힙

- 원소가 내림차순으로 정렬되어 있다는 점에서만 최소힙과 다름
- 각 노드의 원소가 자식들의 원소보다 큼

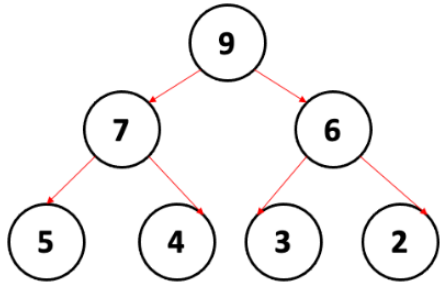


그림 4) 최대힙의 예제

2.6. 최소 신장 트리(MINIMUM SPANNING TREE)

- 신장트리를 의미하는 Spanning Tree는 그래프 내의 모든 장점을 포함하는 트리를 의미

- 이 중 가중치가 최소가 되는 트리가 최소 신장 트리 MST 이다.

- 구현하는 방법 : Prim, Kruskal 알고리즘 사용

2.7. 사용 사례

- 계층 적 데이터 저장 : 트리는 데이터를 계층 구조로 저장하는 데 사용됩니다.

Ex) 파일 및 폴더는 계층적 트리 형태로 저장됨

- 효율적인 검색 속도 : 효율적인 삽입, 삭제 및 검색을 위해 트리 구조를 사용합니다.
- 힙(Heap) : 힙도 트리로 된 자료 구조입니다.
- 데이터베이스 인덱싱을 구현하는데 트리를 사용. Ex) B-Tree, B+Tree, AVL-Tree..
- Trie : 사전을 저장하는 데 사용되는 특별한 종류의 트리입니다.

3. 트리의 구현 방법

기본적으로 트리는 그래프의 한 종류이므로 그래프의 구현 방법(인접 리스트 또는 인접 배열)으로 구현할 수 있다.

- 인접 배열 이용

1. 1차원 배열에 자신의 부모 노드만 저장하는 방법

트리는 부모 노드를 0개 또는 1개를 가지기 때문

부모 노드를 0개: 루트 노드

2. 이진 트리의 경우, 2차원 배열에 자식 노드를 저장하는 방법

이진 트리는 각 노드가 최대 두 개의 자식을 갖는 트리이기 때문

Ex) A[i][0]: 왼쪽 자식 노드, A[i][1]: 오른쪽 자식 노드

- 인접 리스트 이용

1. 가중치가 없는 트리의 경우 : ArrayList< ArrayList > list = new ArrayList<>();

2. 가중치가 있는 트리의 경우 :

1) class Node { int num, dist; // 노드 번호, 거리 } 정의

2) ArrayList[] list = new ArrayList[정점의 수 + 1];

3. 트리의 구현 방법

- **1) n-링크 표현법**

 - 노드에 n 개의 링크를 두고 자식의 개수만큼 링크에 저장한다. 모든 노드 는 자식 노드 수에 관계없이 최대 n개의 링크를 갖는다. 각 링크는 부속 트리 가 저장된 곳을 링크한다.

- 차수가 n인 경우 표현된 노드 data link 1 link 2 link n …

참고 : 트리를 리스트 형태로 표현하는 방법 – 괄호를 사용하여 같은 레벨에 있는 노드들을 같은 괄호로 묶는다.

예) (A(B(E(K,L),F),C(G),D(H(M),I,J)))

3. 트리의 구현 방법

- **2) 왼쪽자식노드-오른쪽형제노드 표현 방법**

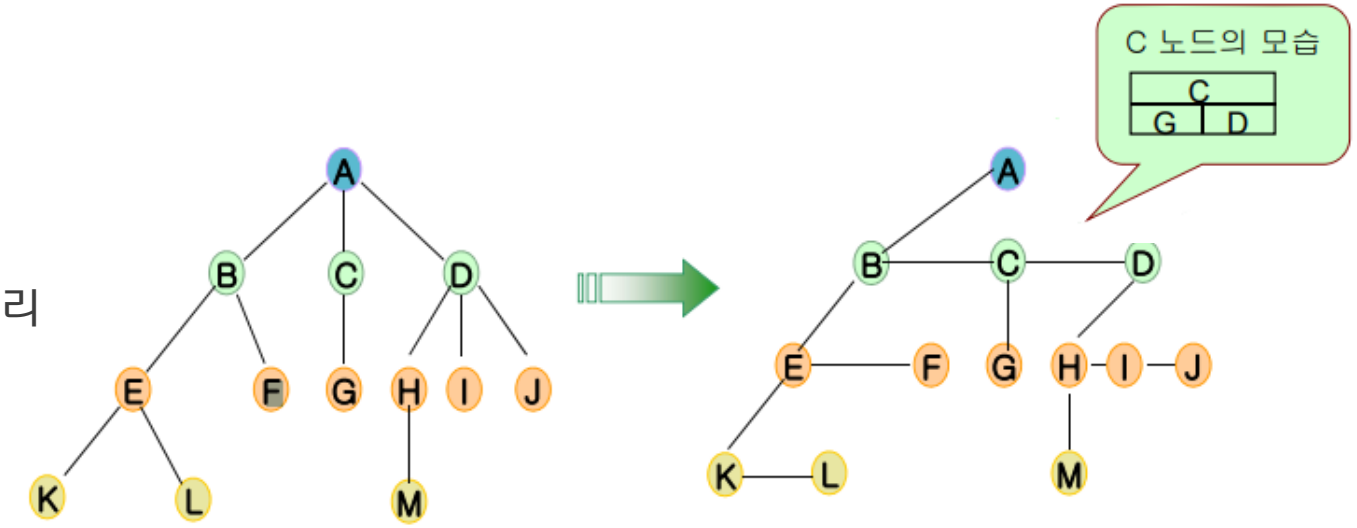
- 모든 노드에 링크를 2개를 둔다.

첫째 링크는 첫번째 자식노드를 표현하고 둘째 링크는 자신의 오른쪽 형제 노드를 표현한다.

- 노드의 길이가 2개로 고정되기 때문에 n-링크 방법보다 간편하다.

- 왼쪽자식노드-오른쪽형제노드 표현법

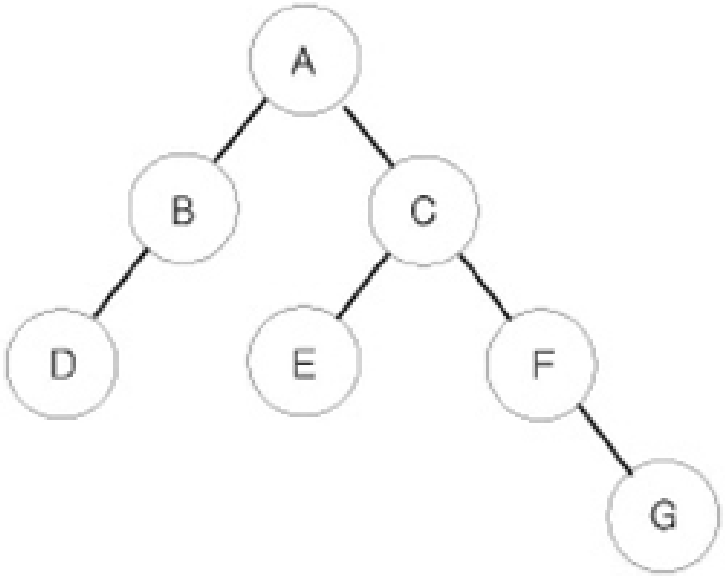
- 왼쪽자식노드-오른쪽형제노드 방법으로 저장된 트리



4. 문제 백준 1991번 – 트리 순회

- 문제 (실버1)

이진 트리를 입력받아 전위 순회(preorder traversal), 중위 순회(inorder traversal), 후위 순회(postorder traversal)한 결과를 출력하는 프로그램을 작성하시오.



예를 들어 위와 같은 이진 트리가 입력되면,

- 전위 순회한 결과 : ABDCEFG // (루트) (왼쪽 자식) (오른쪽 자식)
- 중위 순회한 결과 : DBAECFG // (왼쪽 자식) (루트) (오른쪽 자식)
- 후위 순회한 결과 : DBEGFCA // (왼쪽 자식) (오른쪽 자식) (루트)

가 된다.

참고

- https://gmlwjd9405.github.io/2018/08/12/data-structure-tree.html
- https://yoongrammer.tistory.com/68
- http://dblab.duksung.ac.kr/ds/pdf/Chap08.pdf