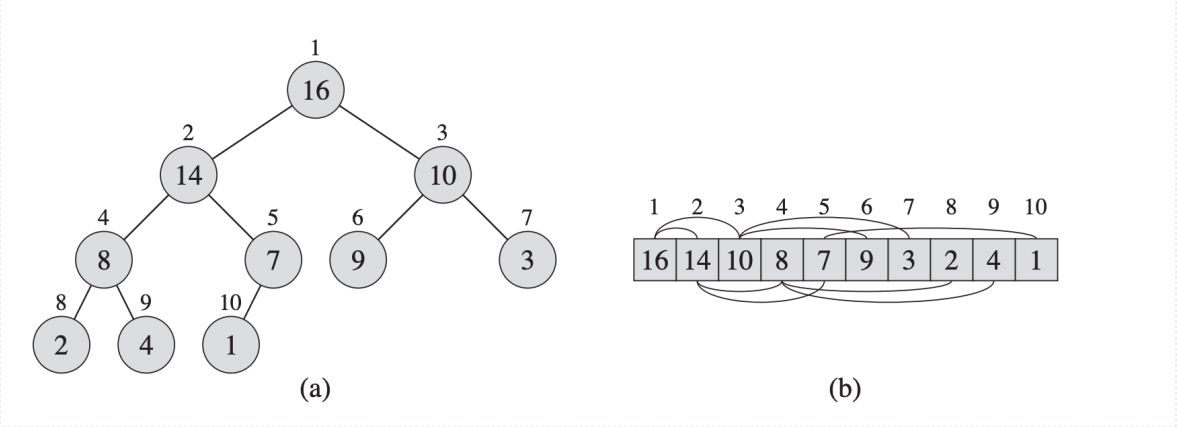
알고리즘 스터디

3주차: 트리(이진-트리, 탐색, 트라이)

트리: Tree

- **실세계의 나무(Tree)와 유사한 위상(Topology)을 가진 자료구조**
  - 뿌리(Root), 가지(Branch), 잎(Leaf)

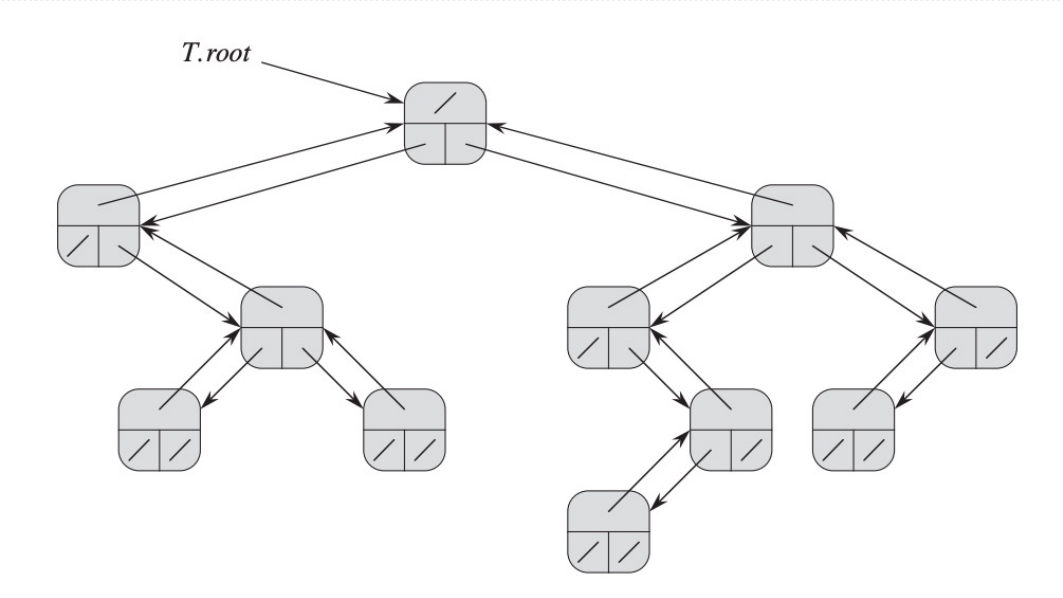


트리: Notation

- **차수(Degree)**
  - 어떤 노드의 자식 노드가 몇개인가
- **레벨(Level)**
  - 어떤 노드의 루트로부터의 거리
- **깊이(Depth)**
  - 트리에 속한 모든 노드의 최대 레벨

트리: How to?

- **링크드-리스트 구현과 유사한 형태로 구현한다**
  - 자식-노드를 가리키는 Pointer, 부모-노드를 가리키는 Pointer



트리: How to?

- **링크드-리스트 구현과 유사한 형태로 구현한다**
  - C언어로 구현하려면 아래와 같은 구조체를 이용

```
struct
{
    int data;
    struct node *parent;
    struct node *left;
    struct node *right;
} node;
```

트리: How to?

- **링크드-리스트 구현과 유사한 형태로 구현한다**
  - 배열인덱스 기반으로 구현해볼 수 있다

1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1

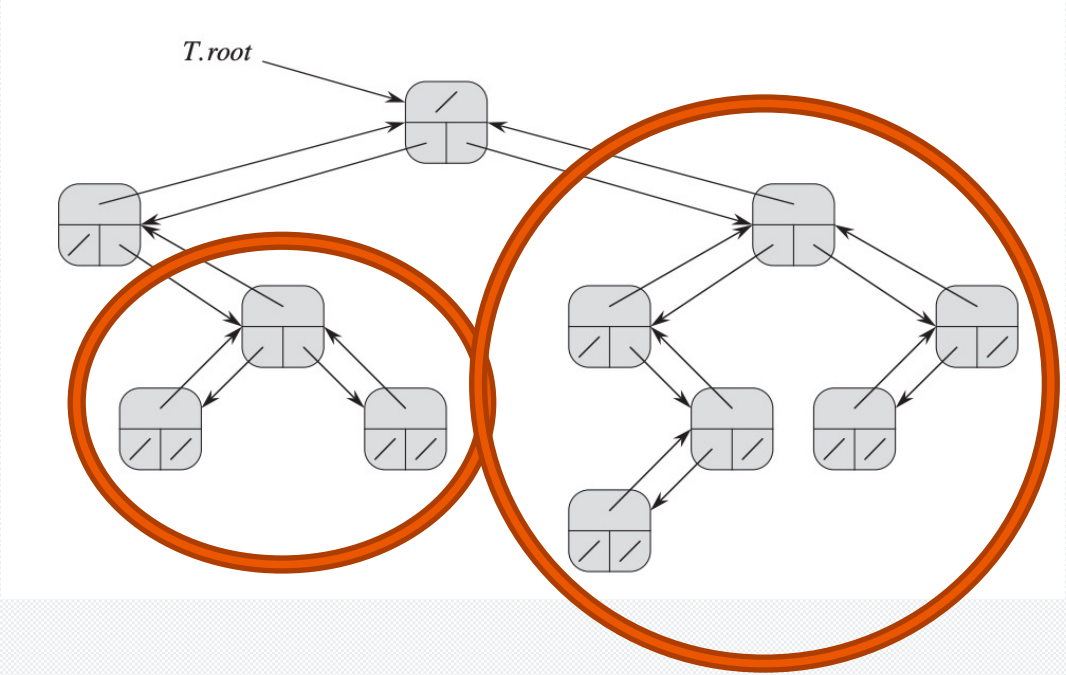
트리: How to?

- **링크드-리스트 구현과 유사한 형태로 구현한다**
  - 이때, 필요한 배열의 길이 $S = \frac{r^n - 1}{r - 1}$ $(r: degree, n: depth)$
- **i번째 node에서 (좌로부터)k번째 child의 index를 구하려면**
  - $i * r + k$

1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1

트리: Sub Tree

- 트리는 자기자신을 구성하는 더 작은 트리로 구성되어 있다



트리: 의문점

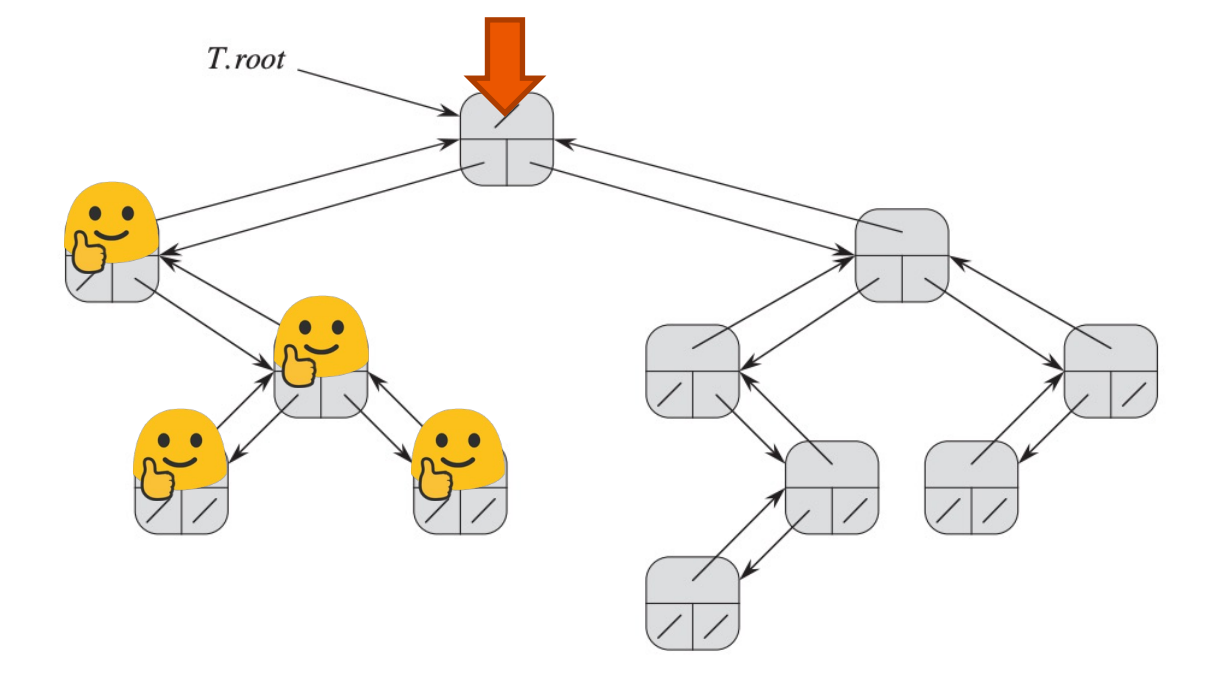
- **왜 저런 모양으로 저장해야 하죠?**
  - 원하는 데이터를 빠르게 찾고 싶으니까
- **어떻게 원하는 데이터를 빠르게 찾죠? - Write**
  - eg. 자신보다 작은 데이터는 left child node, 큰 데이터는 right child node
  - eg. 특정 집합에 속한 데이터를 대표node 밑으로
- **어떻게 원하는 데이터를 빠르게 찾죠? - Read**
  - 트리순회(tree traversal)를 통해 목표하는 값을 찾아간다
  - pre-order / in-order / post-order

트리: Traversal

post-order traversal

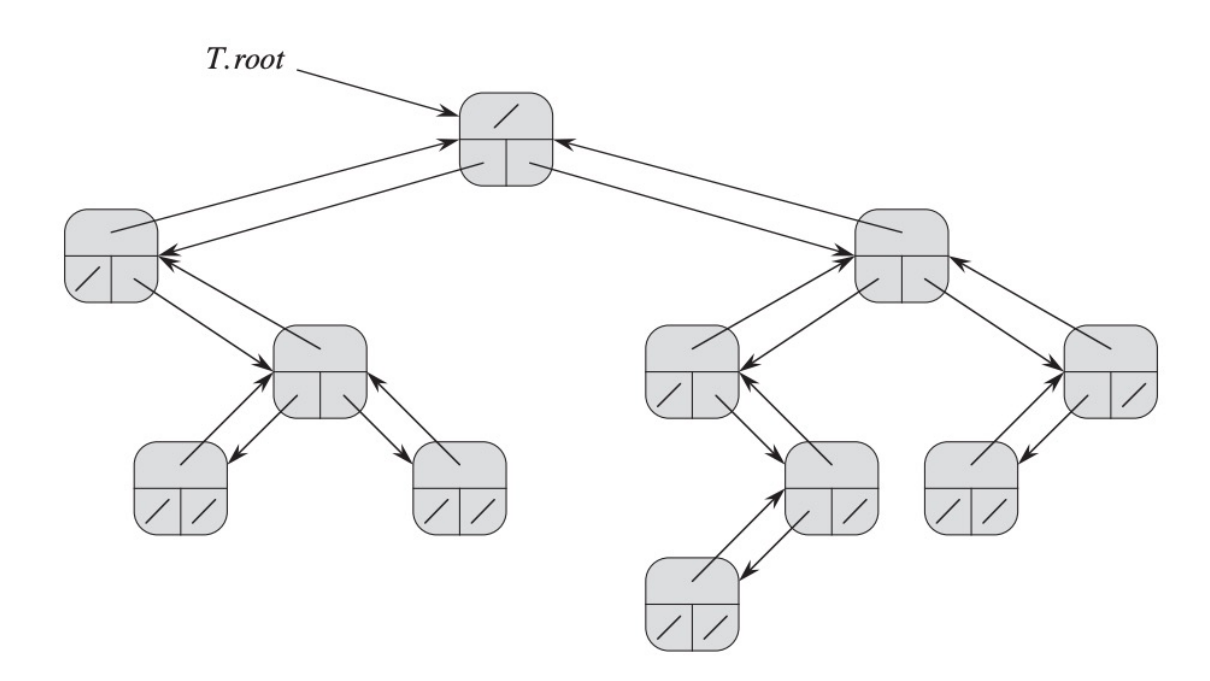
- Root노드에서 출발
- current노드를 root로 갖는 sub-tree를 생각해보자
- 모든 노드에 대해서 아래 순서대로 접근
  - left sub-tree -> right sub-tree -> root node

트리: Traversal



트리: Binary Tree

- Degree가 2인 트리 -> 이진트리



트리: Traversal

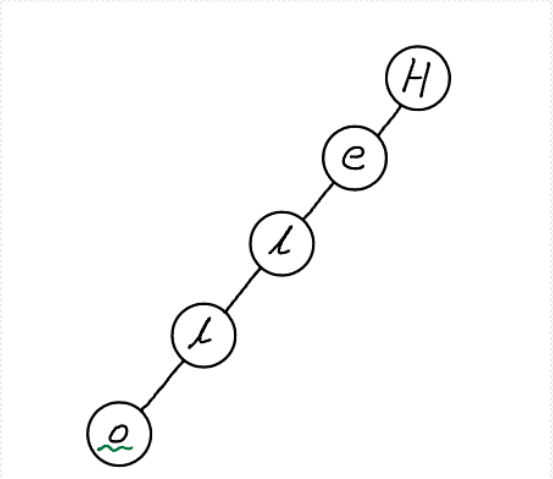
- **방식은 알겠는데 어떻게 구현할까**
  - 방법 1. 재귀함수를 통한 방법 (근본적으로 스택이 아닌가?)
  - 방법 2. Stack자료구조를 통한 방법

```
void postOrderTraversal(Node* node) {
    if (node->value == NULL) {
        return;
    }

    postOrderTraversal(node->left);
    postOrderTraversal(node->right);
    cout << node->value << endl;
}
```

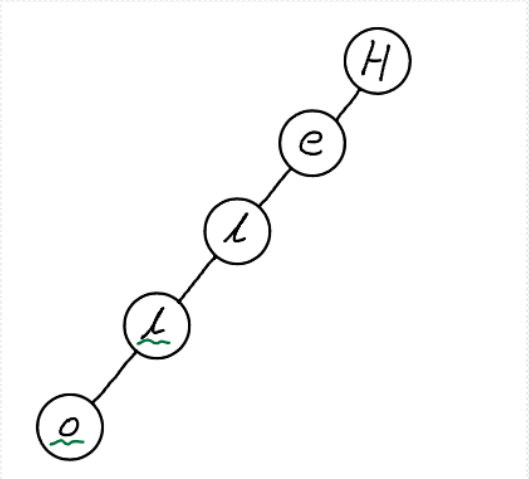
트리: Trie

- **여러 문자열을 트리 형태로 저장한 것**
  - 각 노드는 문자열의 각 문자를 표현
- **아래의 문자열을 가정**
  - “Hello”, “Hell”, “Hi”, “Hey”



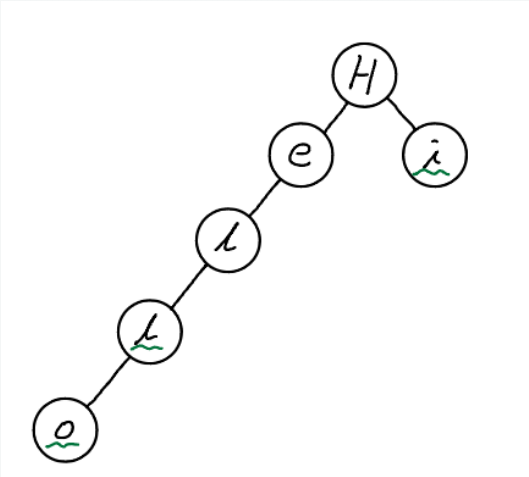
트리: Trie

- **여러 문자열을 트리 형태로 저장한 것**
  - 각 노드는 문자열의 각 문자를 표현
- **아래의 문자열을 가정**
  - “Hello”, “Hell”, “Hi”, “Hey”



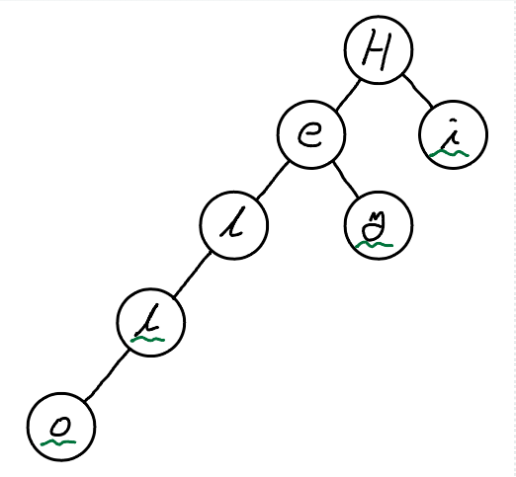
트리: Trie

- **여러 문자열을 트리 형태로 저장한 것**
  - 각 노드는 문자열의 각 문자를 표현
- **아래의 문자열을 가정**
  - “Hello”, “Hell”, “Hi”, “Hey”



트리: Trie

- **여러 문자열을 트리 형태로 저장한 것**
  - 각 노드는 문자열의 각 문자를 표현
- **아래의 문자열을 가정**
  - “Hello”, “Hell”, “Hi”, “Hey”



트리: 연습문제

https://www.acmicpc.net/problem/14725

개미굴 성공

☆

골드 III

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	256 MB	5003	3220	2551	66.157%

문제

개미는(뚠뚠) 오늘도(뚠뚠) 열심히(뚠뚠) 일을 하네.

개미는 아무말도 하지 않지만 땀을 뻘뻘 흘리면서 매일 매일을 살길 위해서 열심히 일을 하네.

한 치 앞도(뚠뚠) 모르는(뚠뚠) 험한 이 세상(뚠뚠) 그렇지만(뚠뚠) 오늘도 행복한 개미들!

우리의 천재 공학자 윤수는 이 개미들이 왜 행복한지 궁금해졌다.

행복의 비결이 개미가 사는 개미굴에 있다고 생각한 윤수는 개미굴의 구조를 알아보기 위해 로봇 개미를 만들었다.

로봇 개미는 센서가 있어 개미굴의 각 층에 먹이가 있는 방을 따라 내려가다 더 이상 내려갈 수 없으면 그 자리에서 움직이지 않고 신호를 보낸다.

이 신호로 로봇 개미는 개미굴 각 층을 따라 내려오면서 알게 된 각 방에 저장된 먹이 정보를 윤수한테 알려줄 수 있다.