데이터 구조 assn5

20180551 이준석

Problem 1 (R-7.5)

R-7.5 What are the minimum and maximum number of internal and external nodes in an improper binary tree with n nodes?

h denotes the height of T

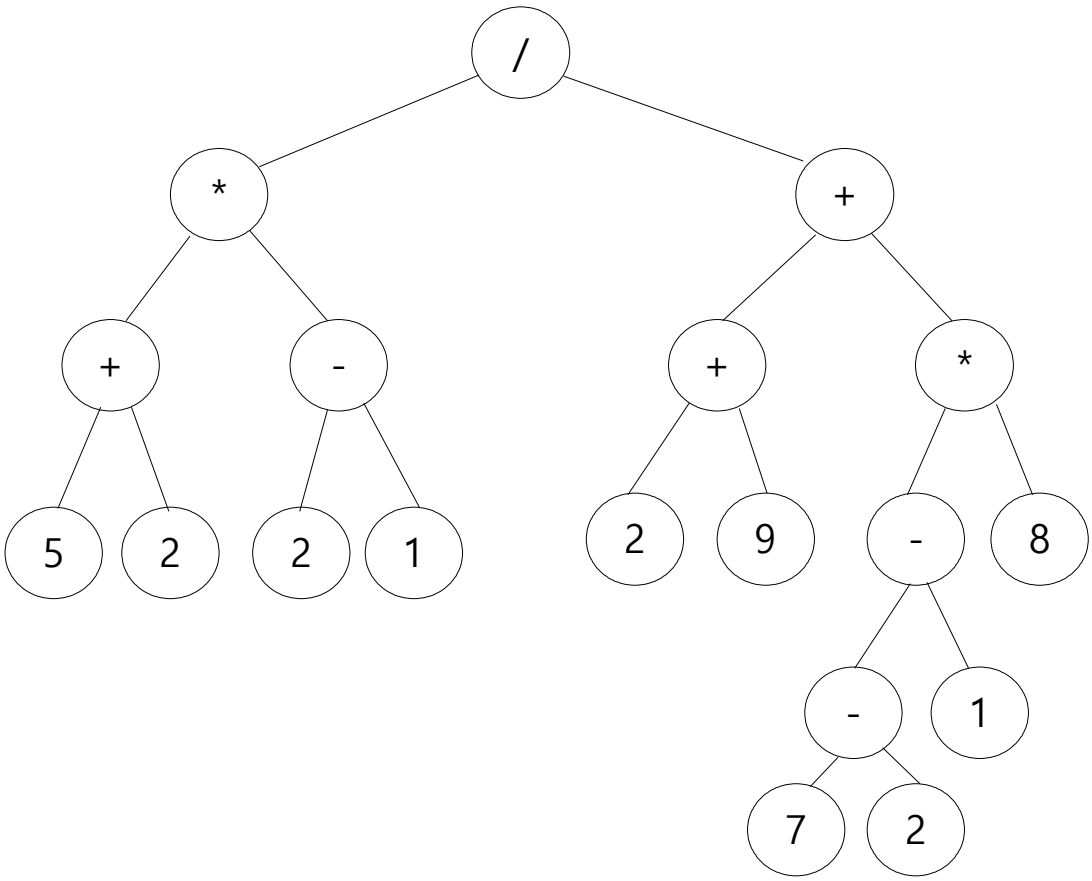
	minimum	maximum
internal nodes	h	2^h - 1
external nodes	1	2^h

log(n+1) - 1 ≤ *h* ≤ n – 1

이므로 n에 대해서 표현하면 다음과 같게 된다.

	minimum	maximum
internal nodes	log(n+1) - 1	2^(n-1) - 1
external nodes	1	2^(n-1)

Problem 2 (R-7.25)



((5+2) * (2-1)) / ((2+9)+((7-2)-1)*8)

Problem 3 (C-7.6)

BFS를 이용한다. BFS에서 같은 depth를 가지는 node들의 집합 단위로 수행된다. 예를 들어 Problem2의 트리를 보면, BFS를 했을 때 { / }, (*, + }, { +, -, +, * }, { 5, 2, 2, 1, 2, 9, -, 8 }, { -, 1 }, { 7, 2 } 단위로 실행된다. 그러므로 나열한 순서대로 queue를 이용하여 BFS를 하면서 집합 단위로 나누어서 root일 때 depth = 0을 반환하고 depth를 1씩 증가시키며 반환한다면 모든 node의 depth를 구할 수 있다. queue를 통해 구현된 BFS이기 때문에 node가 총 n개 있다고 하면 n번의 enqueue와 dequeue 과정이 있을 것이다. 그러므로 O(n)이 된다.

Problem 4 (C-7.33)

C-7.33 Describe, in pseudo-code, a nonrecursive method for performing an inorder traversal of a binary tree in linear time. (Hint: Use a stack.)

```
inorder traversal (T, p) :

while (p is an internal node)
{
        while (p.left() is an internal node)
        {
                push p to a stack;
                p = p.left;

        }
        top from the stack;
        pop from the stack;
        push p to the stack;
        p = p.right();
}
```

Problem 5 (C-7.35)

C-7.35 The path length of a tree T is the sum of the depths of all the nodes in T. Describe a linear-time method for computing the path length of a tree T (which is not necessarily binary)

bfs를 이용한다. bfs는 같은 level에 있는 것들을 먼저 탐색한다. 그러고 나서 다음 level의 node를 방문한다. 그렇기 때문에 bfs를 이용하면서 같은 level에 있는 node들의 depth들을 순차적으로 저장하면서 탐색을 한다면 모든 node들의 depth의 합을 구할 수 있다.

int pathLength = 0;
int depth = 0;
를 선언해놓고 level이 바뀔 때마다
depth++;
를 통해 depth의 크기를 증가시킨다. bfs는 queue를 이용하므로 dequeue(queue에서 먼저 들어왔던 것을 빼낸다)가 일어날 때마다
pathLength = pathLength + depth;
를 해주면 bfs가 완료되었을 때, 모든 node들의 depth의 합이 구해지게 된다.