Data structure [A09]

김종규, PhD

2017-05-01

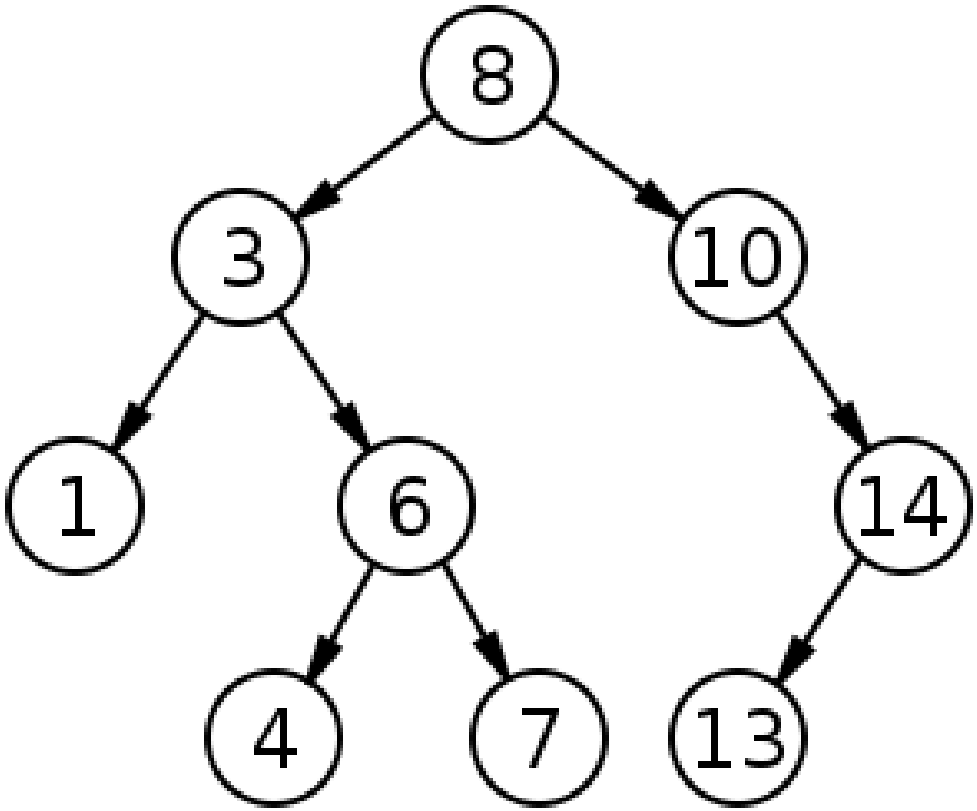
Reviews

- 연습문제
- 중간고사

Outline

- Review: Binary search tree
- Utility: print operation
- insert operation

Review: Binary search tree



Review: Binary tree

- Binary tree 는 linked list 에 비해서 pointer 를 하나 더 사용한다.
- Balance 되어 있다면 search 성능이 비약적으로 좋아진다
  - $O(n) \longrightarrow O(\log n)$
- Programming 언어로 어떻게 표현할 수 있을까?
- 저장된 binary tree 를 어떻게 효과적으로 출력 할 수 있을까?

$\longrightarrow$ Almost the only debugging mechanism

Binary tree 의 node 정의 (PL-C)

```
struct Node {
  int val;
  struct Node* left;
  struct Node* right;
};
typedef struct Node* NodePtr;
```

Binary tree 의 node 정의 (PL-Python)

```
class Node:
    def __init__(self):
        self.val = 0
        self.left = None
        self.right = None
```

Administrivia

- 이번 주 수 (5/3) 공휴일 휴강

Binary tree 의 node 할당 (PL-C)

```
NodePtr bst_alloc(int val) {
  NodePtr res
    = ( NodePtr )malloc(sizeof(struct Node));
  res->val = val;
  res->left = res->right = NULL;
  return res;
}
```

Binary tree 의 node 할당 (PL-Python)

```
def bst_alloc(val):
    n = Node()
    n.val = val
    return n
```

Binary tree 의 구성

```
NodePtr tree = bst_alloc(8);
tree->left = bst_alloc(3);
tree->right = bst_alloc(1);
```

Binary tree 의 print (PL-C)


```
      1
8
      3
```

Binary tree 의 print (PL-C)

```
void bst_print(NodePtr tree, int level) {
  if (tree->right != NULL)
    bst_print(tree->right, level + 1);
  for(int i = 0; i < level; i++)
    printf("    ");
  printf("%d\n", tree->val);
  if (tree->left != NULL)
    bst_print(tree->left, level + 1);
}
```

Binary tree 의 print (PL-Python)

```
def bst_print(tree, level):
    if (tree.right):
        bst_print(tree.right, level + 1)
    for i in range(level):
        print('    ', end = '')
    print(tree.val)
    if (tree.left):
        bst_print(tree.left, level + 1)
```

Binary search tree

- Binary search tree (BST) 는 binary tree 의 일종이다
- 루트의 값을 중심으로 왼쪽에는 작은 값이, 오른쪽에는 큰 값이 들어간다는 성질을 갖고 있다
- 이 성질이 왼쪽, 오른쪽 트리에도 재귀적으로 적용된다

$\longrightarrow$ 새로운 값을 넣는 경우에도 이 성질을 유지할 수 있을까?

Insert operation for a BST

- 어떤 값 val 이 있는지 확인했는데 그 값이 없었다면?
- 두 가지 경우
  - 어떤 node 에 방문했는데 그 값보다 작은 값이고 left 가 null 인 경우
  - 어떤 node 에 방문했는데 그 값보다 큰 값이고 right 가 null 인 경우
- 여기에 insert 한다면? $\longrightarrow$ binary search tree 의 성질이 유지된다.

Binary search tree

- A tree with a single node is a binary search tree
- Given a binary tree n, a value x should be inserted on the left side of tree n if $x < n$
- It should be inserted on the right side of tree n otherwise

Binary search tree 의 insert (PL-C)

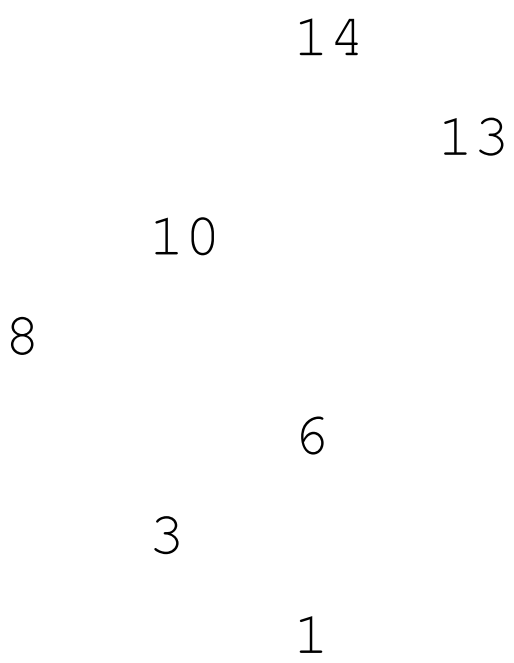
- Insert 시에 binary search tree 의 성질을 유지하면 된다

```
NodePtr bst_insert(NodePtr tree, NodePtr n) {
  if (tree == NULL) tree = n;
  else if (n->val < tree->val)
    if (tree->left == NULL)
      tree->left = n;
    else bst_insert(tree->left, n);
  else if (n->val > tree->val)
    if (tree->right == NULL)
      tree->right = n;
    else bst_insert(tree->right, n);
  return tree;
}
```

Binary search tree 의 insert (PL-C)

```
tree = bst_insert(tree, bst_alloc(8));
tree = bst_insert(tree, bst_alloc(3));
tree = bst_insert(tree, bst_alloc(1));
tree = bst_insert(tree, bst_alloc(6));
tree = bst_insert(tree, bst_alloc(10));
tree = bst_insert(tree, bst_alloc(14));
tree = bst_insert(tree, bst_alloc(13));
```

Binary search tree 의 insert (PL-C)



Binary search tree

- print 된 결과를 왼쪽에서 오른쪽으로 읽어보면?

  $\longrightarrow$ sorted list
- 왼쪽에서 오른쪽으로 순서대로 읽는 방법?

  $\longrightarrow$ infix traversal

Left to right print

- 주어진 tree 의 왼쪽을 모두 print 한다
- root node 를 print 한다
- 주어진 tree 의 오른쪽을 모두 print 한다
- 이 과정을 재귀적으로 적용한다

$\longrightarrow$ infix traversal

Infix traversal (PL-C)

```
void bst_infix(NodePtr tree) {
  if (tree->left != NULL)
    bst_infix(tree->left);
  printf(" %d", tree->val);
  if (tree->right != NULL)
    bst_infix(tree->right);
}
```

Infix traversal

```
1 3 6 8 10 13 14
```

$\longrightarrow$ Sorting: Complexity? $O(n \log n)$ if balanced

Wrap-up

- Binary search tree is a binary tree with a special property
- Searching a value in a balanced binary search tree takes $O(\log n)$
- Insering a value in a balanced binary search tree takes $O(\log n)$
- Infix traversal of a binary search tree produces a sorted list
- We will learn how to make binary search tree balanced in a few weeks
- Before that, we will learn more abstract concept called graph