자료구조론 CC343_2207

Reading assignment 10

경기대학교 컴퓨터공학부

201511837 이상민

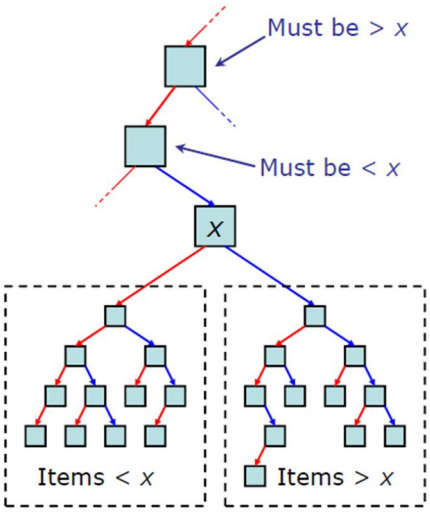
Review Questions

1. Explain the concept of binary search trees.

이진탐색트리란 이진탐색(binary search)과 연결리스트(linked list)를 결합한 자료구조의 일종입니다. 이진탐색의 효율적인 탐색 능력을 유지하면서도, 빈번한 자료 입력과 삭제를 가능하게끔 고안됐습니다.

예컨대 이진탐색의 경우 탐색에 소요되는 계산복잡성은 O(logn)으로 빠르지만 자료 입력, 삭제가 불가능합니다. 연결리스트의 경우 자료 입력, 삭제에 필요한 계산복잡성은 O(1)로 효율적이지만 탐색하는 데에는 O(n)의 계산복잡성이 발생합니다. 두 마리 토끼를 잡아보자는 것이 이진탐색트리의 목적입니다.

이진탐색트리는 다음과 같은 속성을 지니며 아래 그림과 같습니다.



- 각 노드의 왼쪽 서브트리에는 해당 노드의 값보다 작은 값을 지닌 노드들로 이루어져 있다.
- 각 노드의 오른쪽 서브트리에는 해당 노드의 값보다 큰 값을 지닌 노드들로 이루어져 있다.
- 중복된 노드가 없어야 한다.
- 왼쪽 서브트리, 오른쪽 서브트리 또한 이진탐색트리이다.

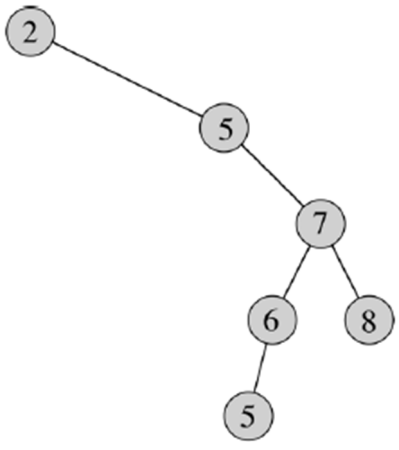
이진탐색트리를 순회할 때는 중위순회(inorder) 방식을 씁니다. (왼쪽 서브트리-노드-오른쪽 서브트리 순으로 순회) 이렇게 하면 이진탐색트리 내에 있는 모든 값들을 정렬된 순서대로 읽을 수 있습니다. 다음 예시와 같습니다.

2. Explain the operations on binary search trees.

이진탐색트리의 핵심 연산은 검색(retreive), 삽입(insert), 삭제(delete) 세 가지입니다. 이밖에 이진탐색트리 생성(create), 이진탐색트리 삭제(destroy), 해당 이진탐색트리가 비어 있는지 확인(isEmpty), 트리순회(tree traverse) 등의 연산이 있습니다.

3. How does the height of a binary search tree affect its performance?

이진탐색트리 핵심 연산인 탐색, 삽입, 삭제의 계산복잡성은 모두 O(h)입니다. 트리의 높이에 의해 수행시간이 결정되는 구조입니다. 그러나 트리가 다음과 같은 경우 문제가 됩니다.



위 그림의 경우 노드 수는 적은 편인데 높이가 4나 됩니다. 균형이 안 맞기 때문입니다. 극단적으로는 n개의 노드가 크기 순으로 일렬로 늘어뜨려져 높이 또한 n이 되는 경우도 이진트리탐색에 해당합니다. 결과적으로 이진탐색트리의 계산복잡성은 O(n)이라는 얘기입니다.

이래가지고서는 탐색 속도가 O(logn)으로 빠른 이진탐색을 계승했다고 보기 어렵습니다. 이 때문에 트리의 입력, 삭제 단계에 트리 전체의 균형을 맞추는 이진탐색트리의 일종인 AVL Tree가 제안되었습니다.

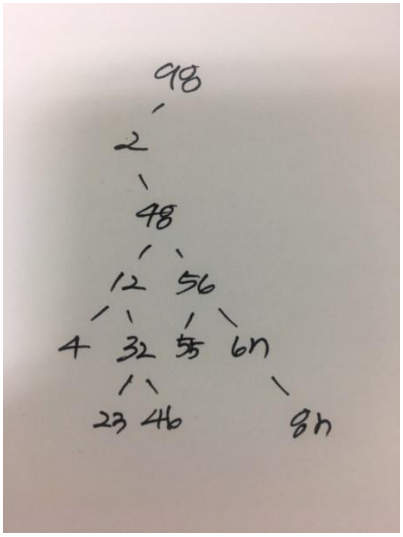
4. How many nodes will a complete binary tree with 27 nodes have in the last level? What will be the height of the tree?

complete binary tree with 27 nodes have in the last level : 12개

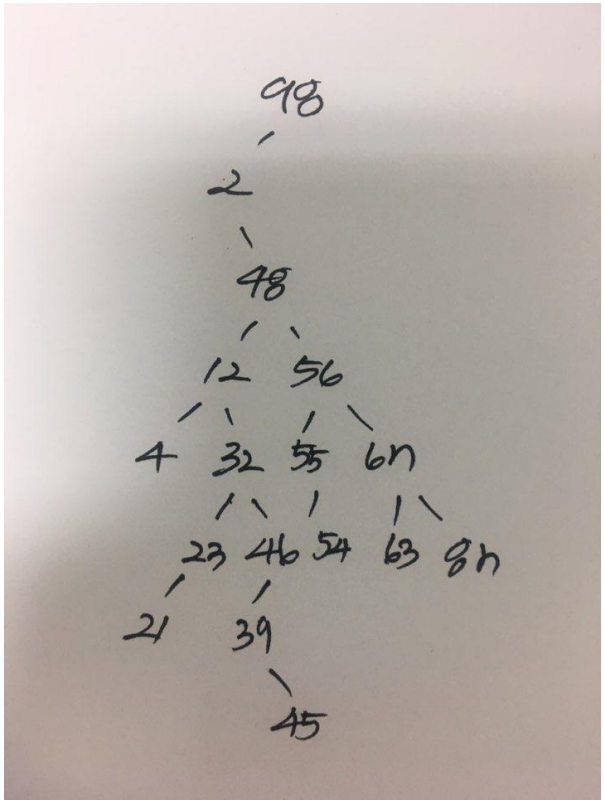
height of the tree : level 4

11. Create a binary search tree with the input given below:

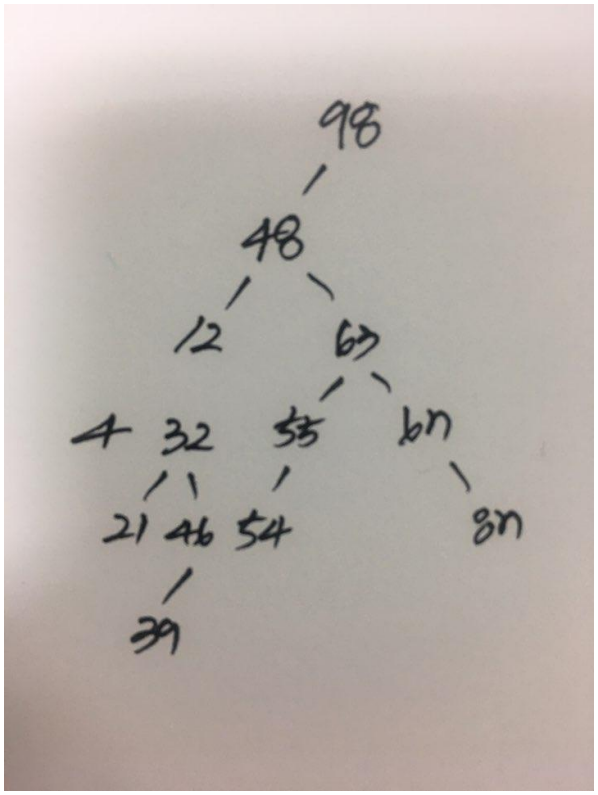
98, 2, 48, 12, 56, 32, 4, 67, 23, 87, 23, 55, 46



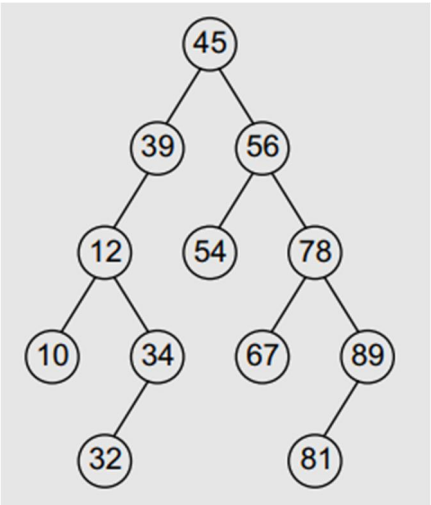
(a) Insert 21, 39, 45, 54, and 63 into the tree



(b) Delete values 23, 56, 2, and 45 from the tree



12. Consider the binary search tree given below. Now do the following operations:



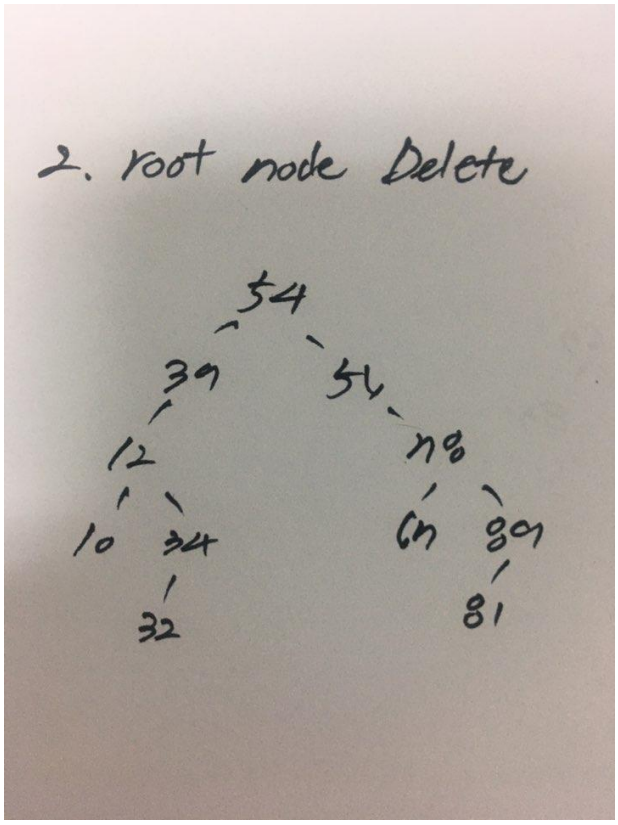
- Find the result of in-order, pre-order, and post-order traversals.

In-order : 10, 12, 32, 34, 39, 45, 54, 56, 67, 78, 81, 89

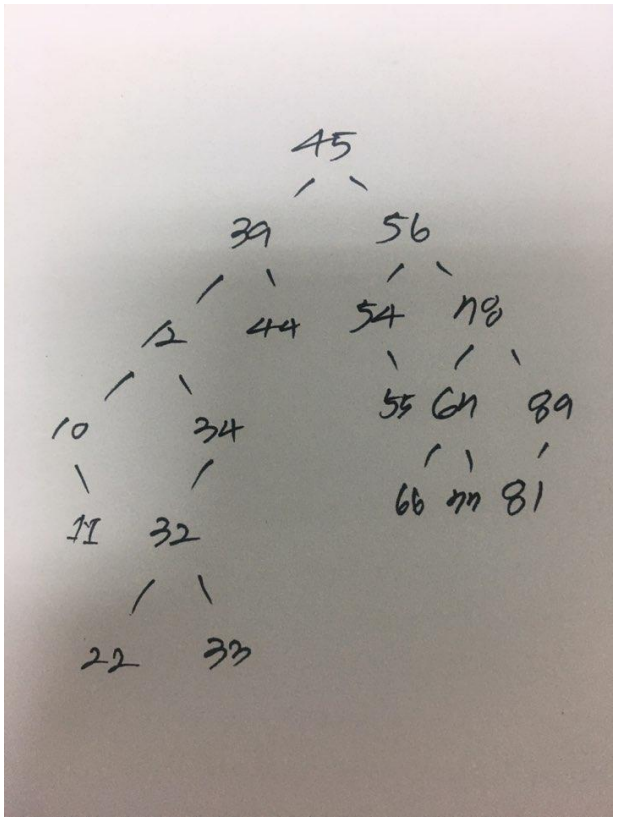
Pre-order : 45, 39, 12, 10, 34, 32, 56, 54, 78, 67, 89, 81

Post-order : 10, 32, 34, 12, 39, 54, 67, 81, 89, 78, 56, 45

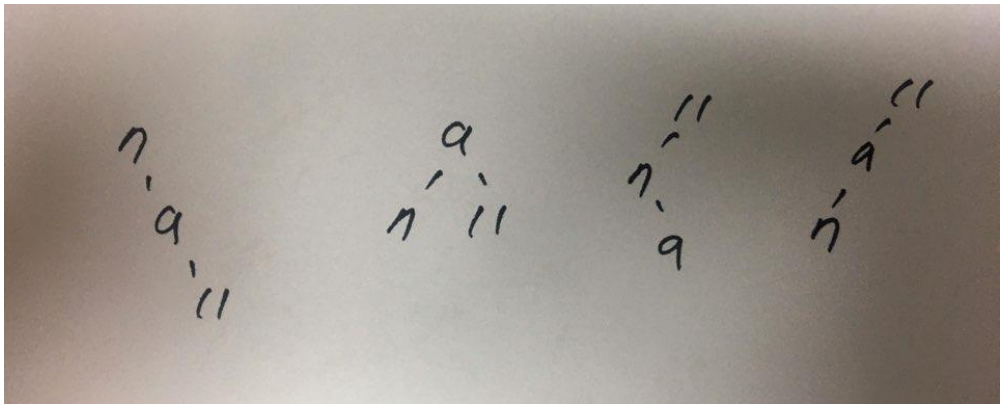
- Show the deletion of the root node



- Insert 11, 22, 33, 44, 55, 66, and 77 in the tree



19. Draw all possible binary search trees of 7, 9, and 11.



Multiple-choice Questions

1. a

True or False

1. True

2. False

3. False

4. True