CSCA48 Winter 2018 Week 11:Searching & the Running Time

Marzieh Ahmadzadeh, Nick Cheng

University of Toronto Scarborough

Searching

- A `search` operation is as important as insertion and removal operations in most of the data structures.

- For which of the followings, searching operation is important?
  - A stack of function calls
  - A BST of information of all 400 students who took CSCA48
  - A queue of stand-by flyers
  - A list of all UTSC's employees
  - A binary tree of term test 2 marks
  - A list of $6x10^9$ nucleotides (A, C, T, G) that form 23 pairs of chromosomes

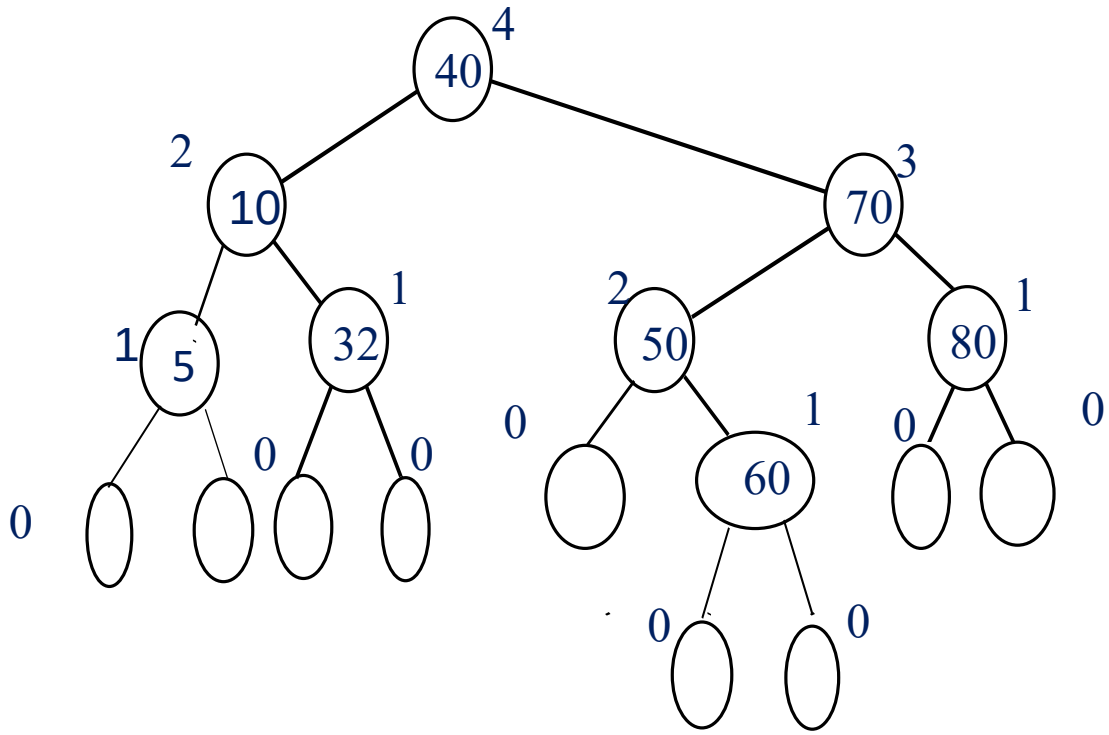
- So efficient searching has been a food for thought for computer scientists.

Searching Methods

- Linear
  - Time complexity, O(n), is not satisfactory if n is large.
- Binary Search
  - List-based binary search (remember that you should have a sorted list)
  - BST
    - Time complexity O(n)
  - AVL BST
    - Will be covered in more detail in CSCB63
  - And a few more
- Hashing Technique
  - Need a proper hash table (an array of size N + a hash function)
  - It will be covered in CSCB63

AVL BST

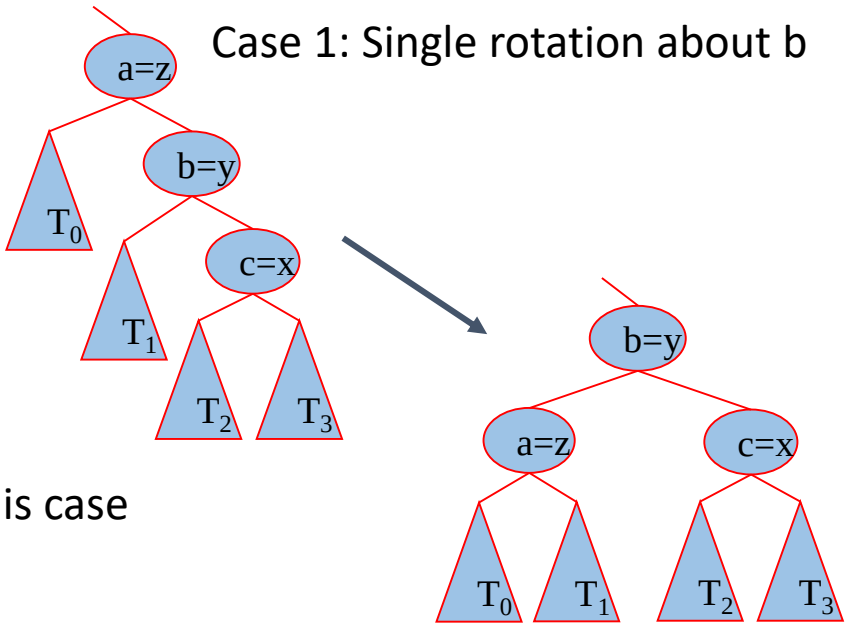
- AVL tree is a BST, such that for each node in the tree, the height of its right and left children differs at most by 1.
- Assume that we only store data in internal nodes and all the external nodes contain `None`.
- The height of a AVL tree is log(n)
  - search() and find() running time is O(log(n))



Insertion and removal in AVLs(1)

- may lead to an unbalanced tree.
- `Trinode restructuring` transforms the tree to a balanced tree.
  - Go up the tree from the newly added node (or parent of deleted node) to find the first node that does not maintain AVL property. Call this node `z`
  - `y = Highest_child(z)`
  - `x = Highest_child(y)`
  - Rename `xyz` to `abc` such that in-order traversal generates `abc`
  - Depending on the tree structure either rotate once or twice

Case 1: Single rotation about b

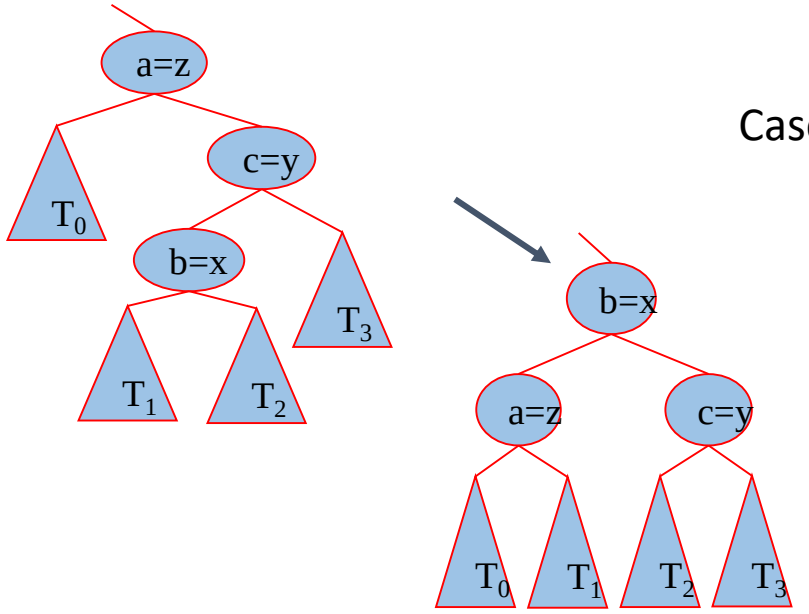


There is a symmetrical case to this case

Pictures from Data Structures and algorithm by Goodrich et. al. (2013)

Insertion and removal in AVLs(2)

- Follow the restructuring algorithm to create a balanced tree.



Case 2: double rotation about b

There is a symmetrical case to this case

Pictures from Data Structures and algorithm by Goodrich et. al. (2013)

Hashing Technique

- A hash function ***h*** maps the input data to integers in a fixed interval [0, ***N*** - 1]
  - e.g. ***h***(***k***) = ***k*** mod ***N***
  - h(k) is called the hash value of k
  - we store k at index h(k)

- A hash table consists of
  - hash function ***h***
  - array of size ***N***

Example

- Let's have fun comparing the running time of insertion, removal and searching operations in a BST, AVL and hash table with hash function `K mod 10`.

Insert 1, 2, 3, 4, 5, 6, 10, 7, 8, 9

Find(9)

Remove(9)

Running Time for Searching

Searching Technique	Time complexity
Linear search	O(n)
In-Place Binary Search	O(log(n))
BST	O(n)
AVL	O(log(n))
Hashing	O(1)