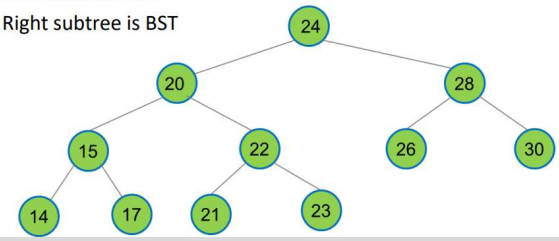
BST

Thursday, 2 June, 2022 23:46

Tree
And Binary Search Tree (BST)

MONASH University

- What is a binary search tree?
  - Empty tree is BST
  - All elements in left subtree < root
  - All elements in right subtree > root
  - What about ==? Key are unique!
  - Left subtree is BST
  - Right subtree is BST



Tree
And Binary Search Tree (BST)

MONASH University

- The main functions?
  - Search complexity? O(N)
    - Complexity is high when tree is imbalanced!

When balanced then it would be O(log n) as imbalanced tree can have all node on only one side which when recursive down it would take O(n) to find.

BST Studio11 Q8

Tuesday, 17 May, 2022 14:37

Problem 8. You are given a set of distinct keys $x_1, x_2, ..., x_n$.

(a) Design an algorithm that creates a binary search tree of minimal height containing those keys in $O(n \log(n))$ time.

not AVL

(b) Prove that your algorithm produces a BST of height at most $\log(n)$.

(c) Prove that the fastest algorithm for this problem in the comparison-based model has time complexity $\Theta(n \log(n))$.

keys = $[x_1, x_2, \ldots, x_n]$

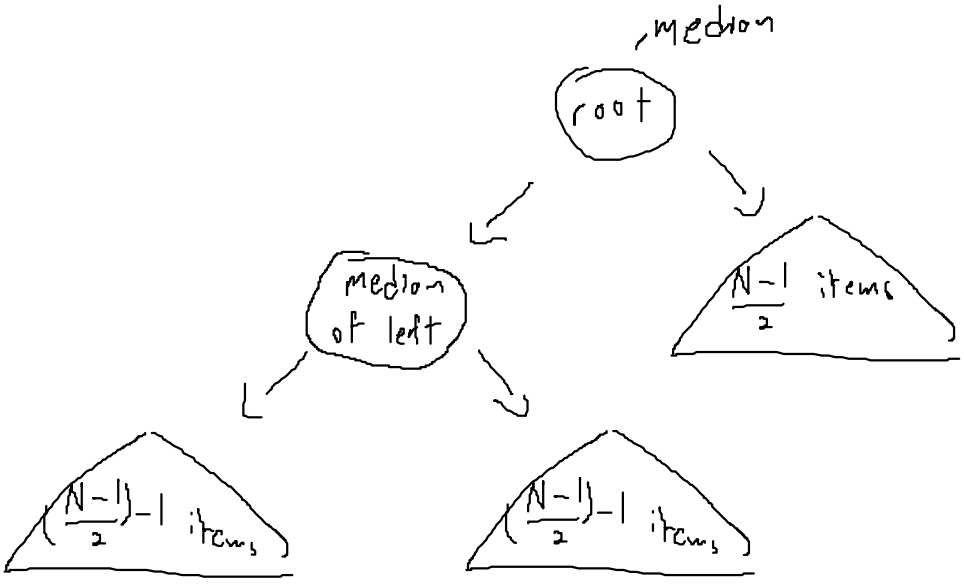
minimal height = log N

preprocessing

sort(keys) = O(N log N) X counting (N+M) M > N log N X

merge quicke heap

sorted_keys $(k_1, k_2, k_3 \ldots k_n)$



1) Sort the key list (with heap sort for N log N)
2) Insert median into tree O(1)
3) Repeat 2) for left side till no more item O(log N)
4) Repeat 2) for right side

Overall complexity is N log N

c) The lower bound complexity for sorting is N log N so it can't be faster than N log N

AVL Tree

Wednesday, 1 June, 2022 22:02

AVL Tree
The better BST!

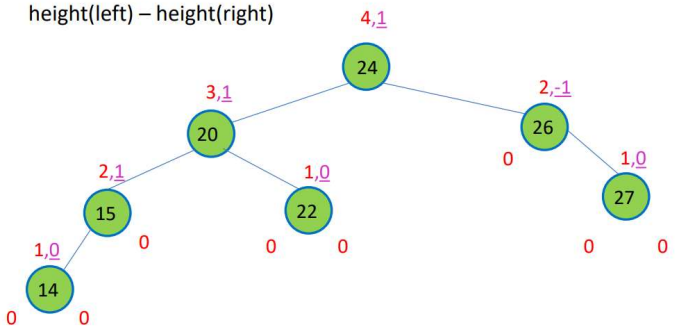
MONASH University

- AVL trees are self balancing…
- Height balanced
  - absolute(Height(left) – Height(right)) <= 1
  - Ensure a maximum of O(log N) height
  - True for each subtree as well

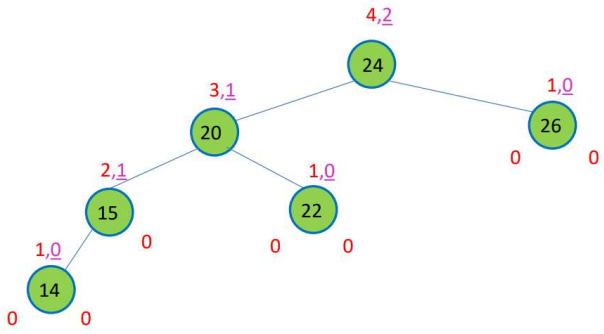
AVL Tree
The better BST!

MONASH University

- What is the height?
- Is the tree balanced? YES
  - Balance factor = {-1, 0, 1}
    height(left) – height(right)



- Let's try a new tree…
  - Is this balanced?



AVL Tree
Complexity?

MONASH University

- Everything is the same as BST
  - Insert
  - Delete
  - Search

  } Check balance after that

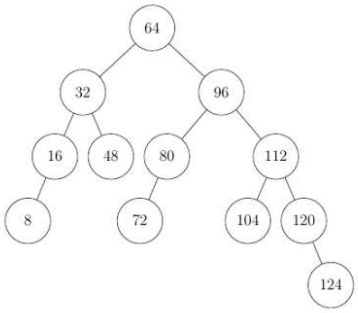
- So what is the balancing cost?
  - Check balance from the bottom
    - Done via recursion…
    - Note: This isn't O(N) because we would just update and not calculate from scratch as how we do by hand
  - If we find imbalance, we balance (at most 2 rotations – a constant)
  - So this would be O(log N)
  - Note: Analyzed further in tutorial supplementary question

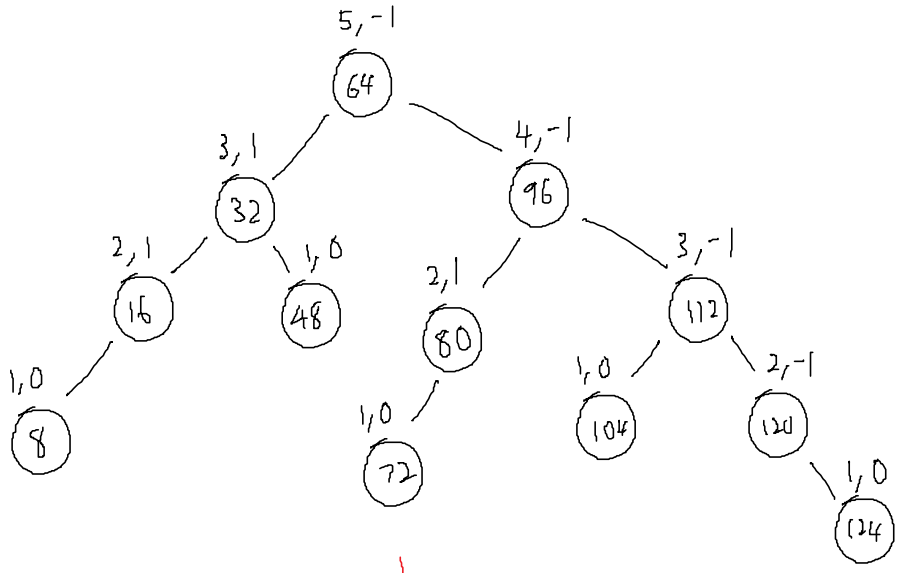
Self-Balancing Trees

Question 11

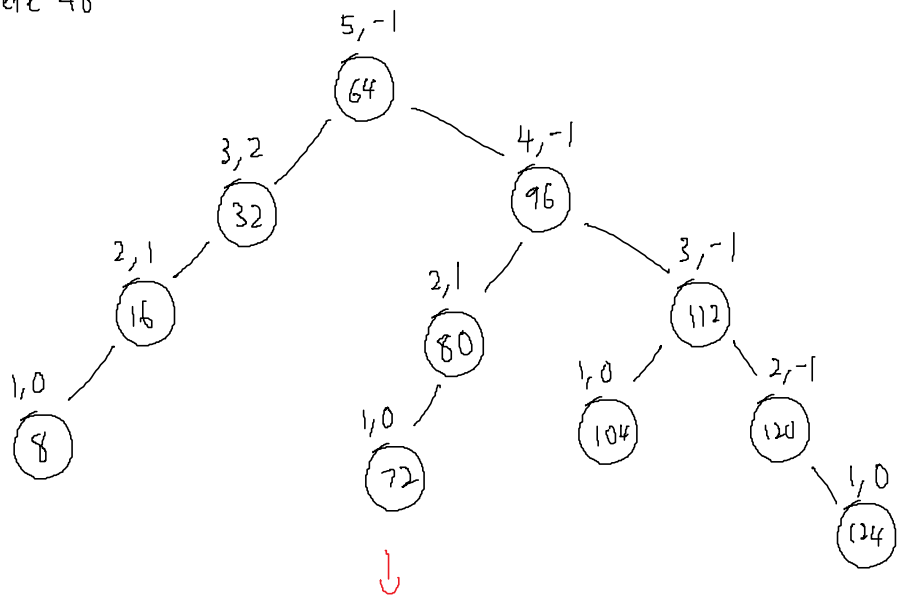
Which of the following would be the result of deleting 48 from the following AVL tree?

1 Mark

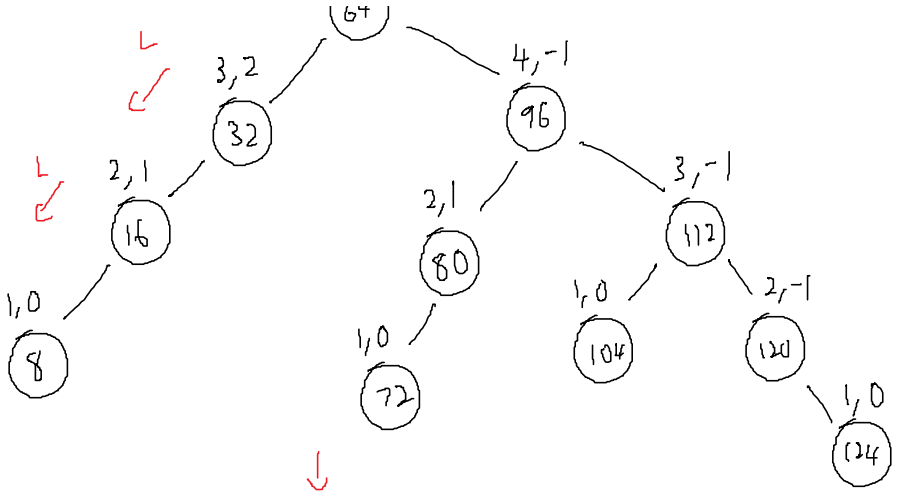




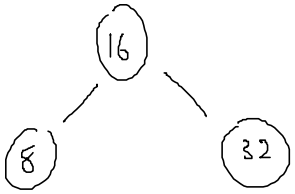
delete 48

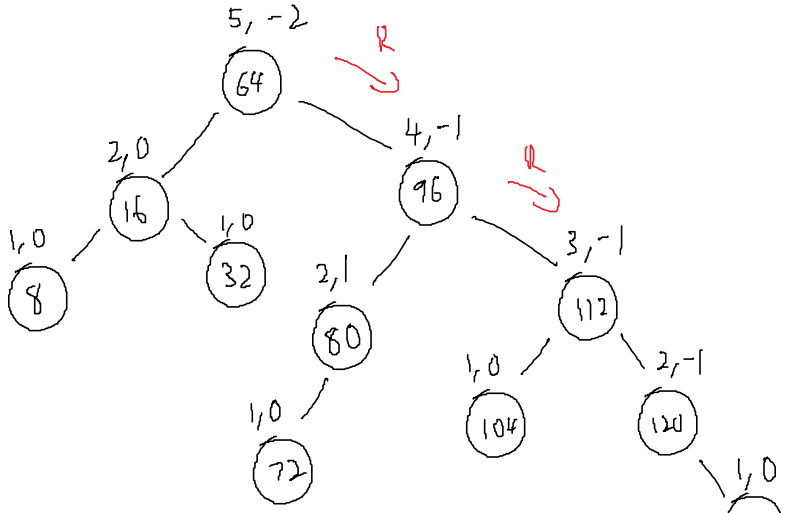


8, 16, 32

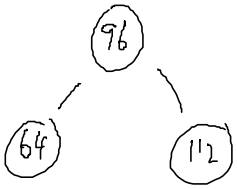


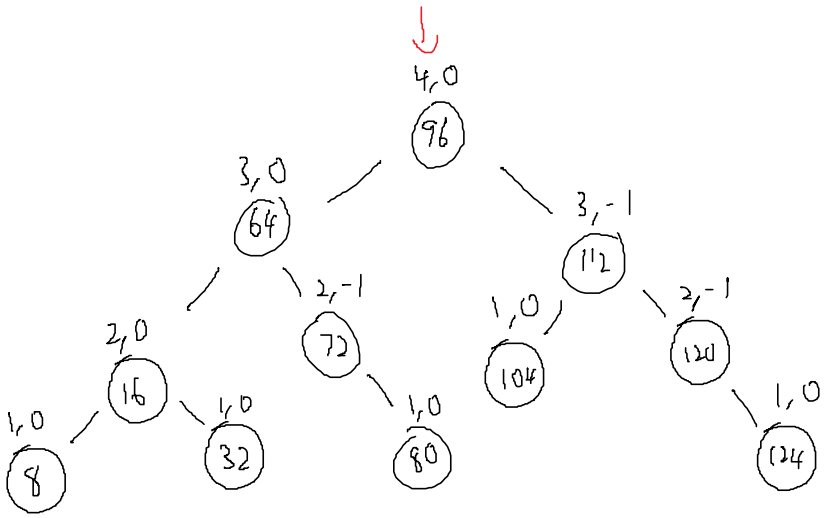
8, 16, 32





64, 96, 112

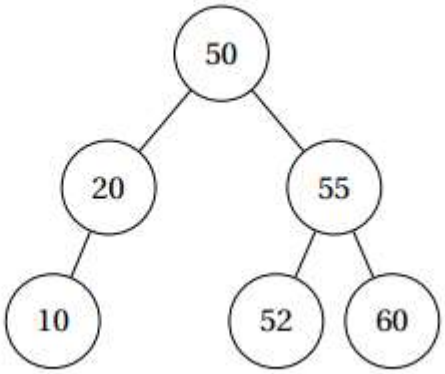


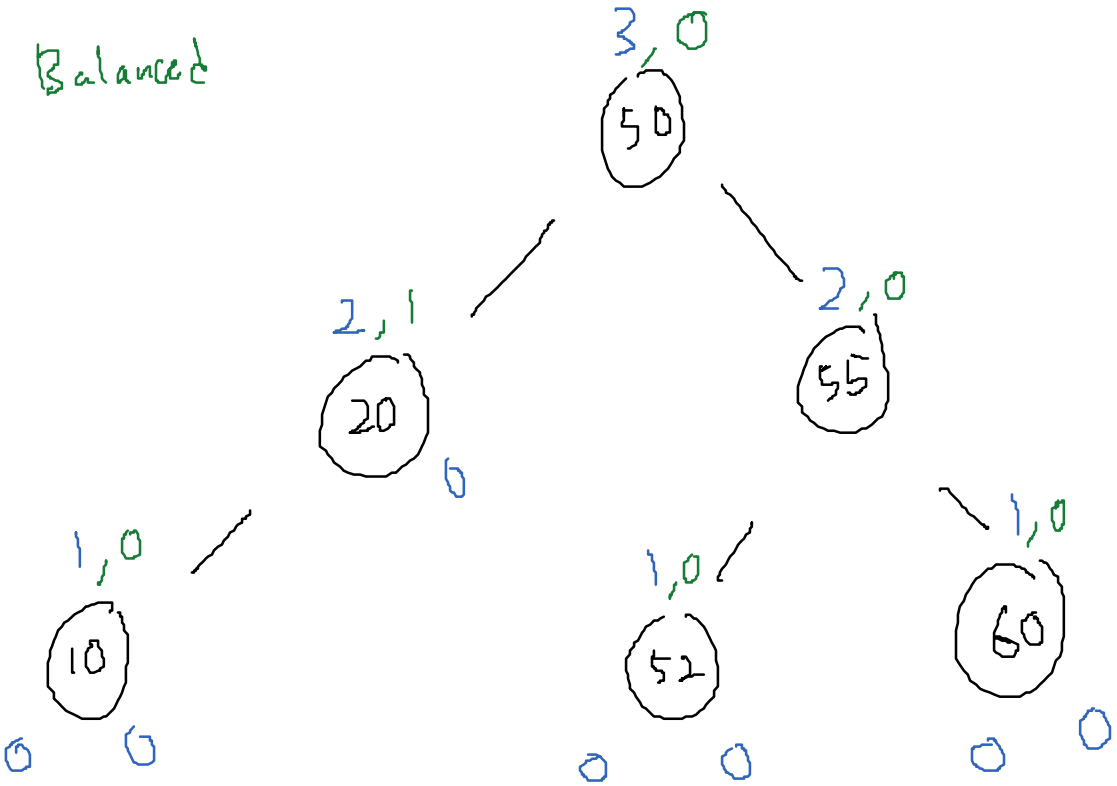


AVL Tree Studio11 Q2

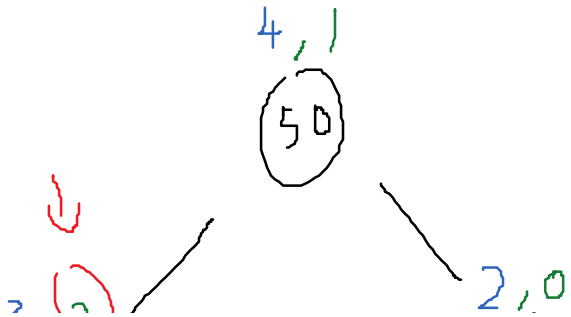
Tuesday, 17 May, 2022 13:52

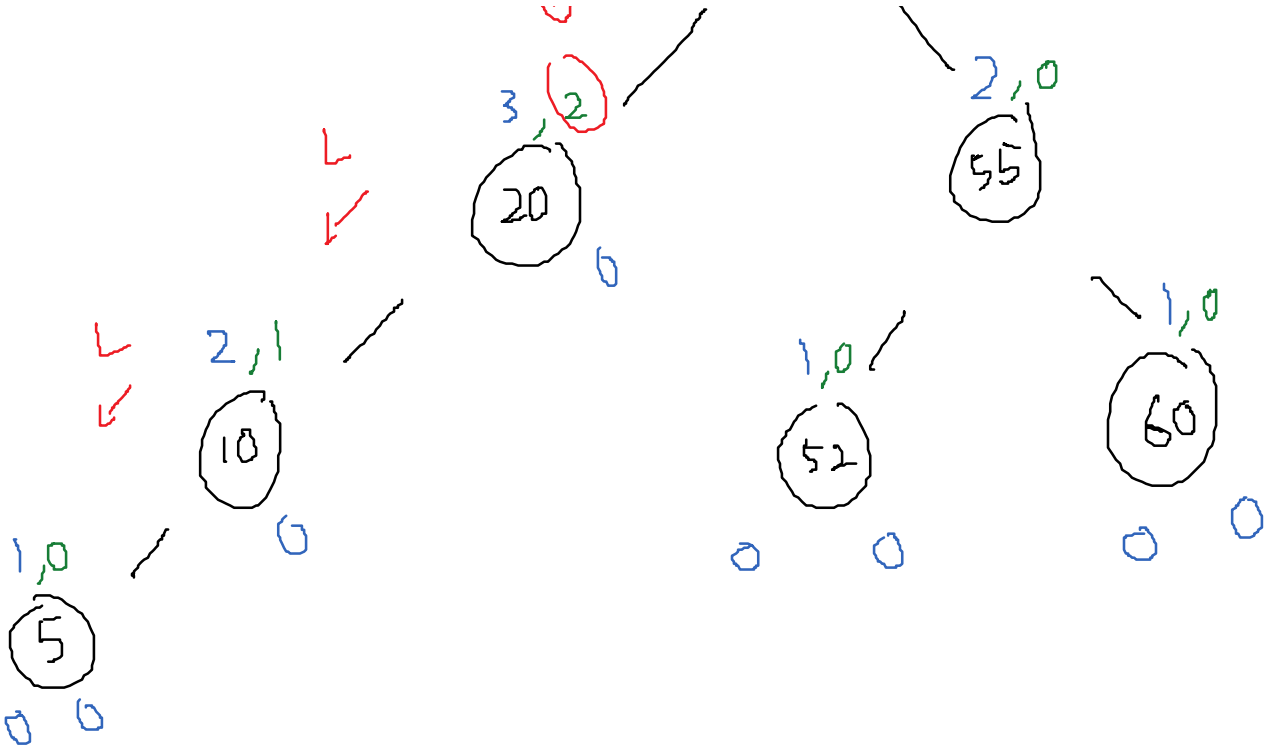
Problem 2. Insert 5 into the following AVL tree and show the rebalancing procedure step by step.





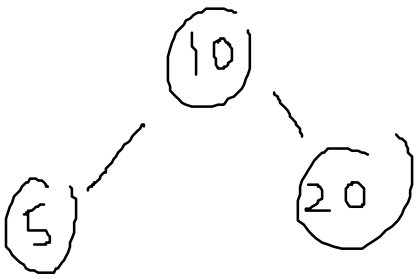
1) insert 5



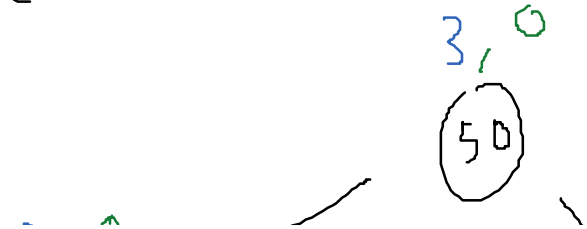


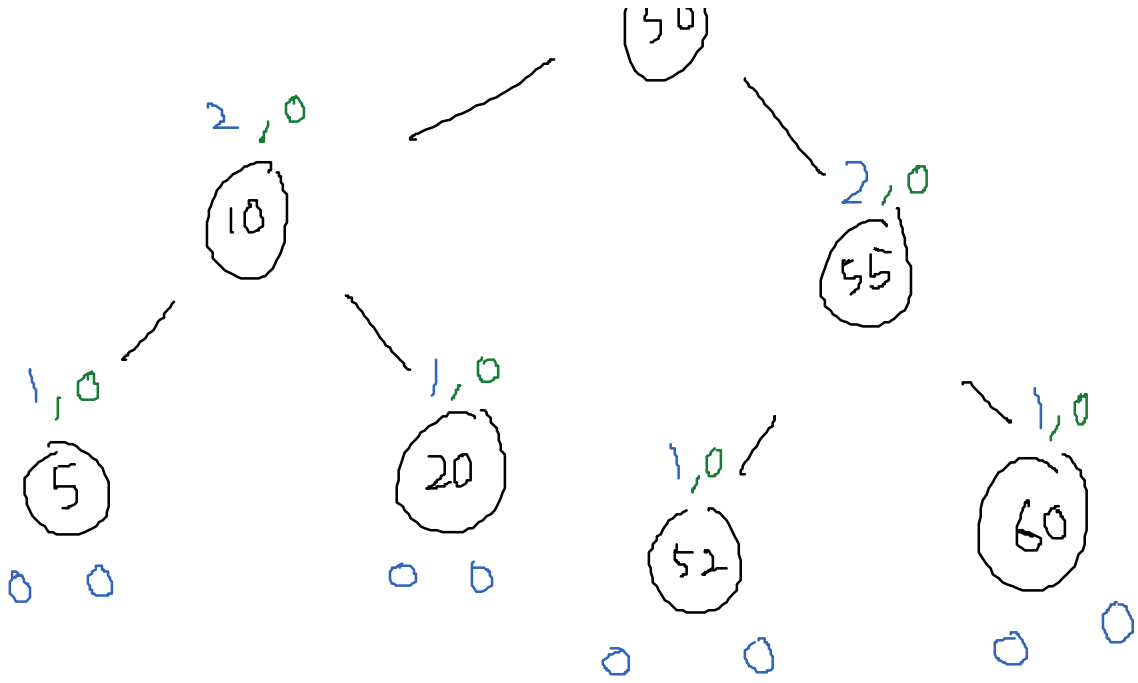
LL imbalance

2) 5, 10, 20



3) Balanced

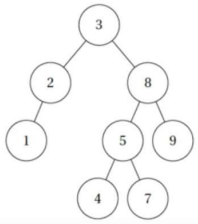



50
2,0
10
2,0
55
1,0
5
1,0
20
1,0
52
1,0
60

AVL Tree Studio11 Q3

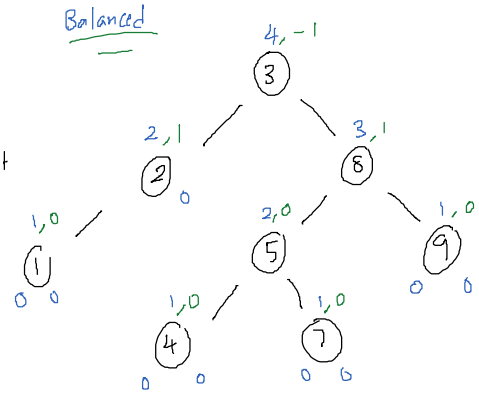
Tuesday, 17 May, 2022 13:52

Problem 3. Insert 6 into the following AVL tree and show the rebalancing procedure step by step.



Balanced

AVL
↓
balanced height
BST



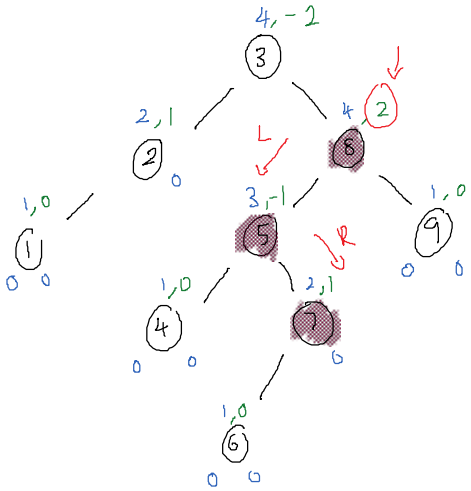
① is tree balanced ✓

② what is the complexity to calculate balance factor

O(N) insert

O(log N) update

1)



② insert 6 O(log N)

③ update balance O(log N)

④ identify imbalance

LR - imbalance

⑤ rotate left

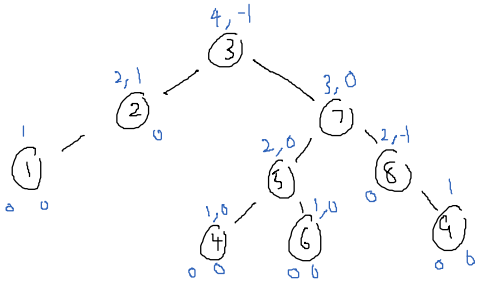
LL - imbalance

⑥ rotate right

RR - imbalance

combine into one way

2) 8, 5, 7



Hashtable

Thursday, 2 June, 2022 23:40

Data Lookup
An sorted array/ ~~list~~

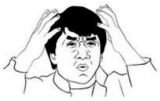
MONASH University

- If the data is sorted… Then we can use binary search!
  - Best case O(1)
  - Worst case O(log N)
- But we need to insert in order…
  - Giving us O(N) complexity still…
- Likewise for delete, we need to shift
  - Giving us O(N) complexity still…

Hash Table
Collision resolution

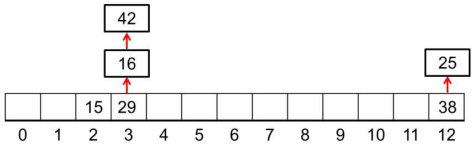
MONASH University

- Open addressing aka closed hashing
  - Linear probe
  - Quadratic probe
  - Double hashing
  - Cuckoo hashing
- Separate chaining aka closed addressing



- Separate chaining
  - Straight forward
  - Complexity to travel through the chain for operations
  - The chain can be anything!
    - List
    - Another hash table
    - Tree

And many more!



Past Year 2021

Hash Tables and Dictionary

Question 10

What is the best and worst case time complexity for looking up an item into a separate chaining hash table, where the chains are implemented using Binary Search Trees (BST)?

2 Marks

- M is the size of the table (i.e. the number of BST)
- N is the number of items currently in the table

What is the best case complexity?

- O(N + M) · O(M) · O(N) · O(1) · O(log N) · O(log M)

What is the worst case complexity?

- O(N + M) · O(M) · O(N) · O(1) · O(log N) · O(log M)

Past Year 2020 Sem2

Hashtable and Dictionary

Question 9

The objective of a good hash function is to distribute keys more or less randomly in the table.

Why then is it a bad idea to hash items to random positions in the table?

1 Mark

It is bad idea since random position is not deterministic at all. A good hashtable needs to be deterministic. We can't search for the items we need since it is randomly placed.

Question 10

What is the best and worst case time complexity for looking up an item into a separate chaining hash table, where the chains are implemented using AVL trees?

2 Marks

- M is the size of the table (i.e. the number of AVL trees)
- N is the number of items currently in the table

What is the best case complexity?

- O(N) · O(1) · O(M) · O(log(N)) · O(log(M))

What is the worst case complexity?

- O(N) · O(1) · O(M) · O(log(N)) · O(log(M))

Hashtable Studio11 Q1

Tuesday, 17 May, 2022 14:41

Problem 1. A hashtable can be used to store different types of elements such as integers, strings, tuples, or even arrays. Assume we want to use a hashtable to store arrays of integers (i.e., each element is an array of integers). Consider the following potential hash functions for hashing the arrays of positive integers. Rank them in terms of quality and give a brief explanation of the problems with each of them. Assume that they are all taken mod m, where m is the table size.

- Return the first number in the array (e.g., if `array=[10,5,7,2]`, hash index will be `10%m`)
- Return a random number in the array (e.g., if `array=[10,5,7,2]`, hash index will be a randomly chosen element from the array mod m)
- Return the sum of the numbers in the array (e.g., if `array=[10,5,7,2]`, hash index will be `24%m`)

Give a better hash function than these three and explain why it is an improvement.

If we were to rank the hash functions of the 1st, 2nd and 3rd. We can say that the random number (2nd) is by far the worst one, using random number for a hash index can't even be considered as a hash function as it isn't deterministic.

We can say that the first number in array and the sum of the numbers in the array are equally bad. Using the first number in the array is called using positional to determine the hash index. Sum of the numbers we can say that it uses equations to determine the hash index.

(1st)
The reason why using only positional to determine hash index, is bad because given an example of [2,1,3] and [2,5,1] it uses 2%m for its' hash index and that can cause duplicates.

(3rd)
Using equation as well, it isn't that good given an example of [1,2,3] and [3,1,2] it will produce 6%m.

A better hash function would be using both positional and equation to determine its' hash index. For example:

$2,1,3 \quad : \quad 2a + 1a^2 + 3a^3$

$3,1,2 \quad : \quad 3a + a^2 + 2a^3$

It shows that no matter the different position or provided that it has the same sum for both array. Its' hash index can be different.

Hashtable Studio11 Q6

Tuesday, 17 May, 2022 14:36

Problem 6. Consider the following potential hash functions for hashing integers. Rank them in terms of quality and give a brief explanation. Assume that they are all taken mod m, where m is the table size.

- Return $x \mod 2^p$ for some prime p.
- Return $(ax + b)$ for some positive, and randomly chosen, a, b.
- Return a random integer.

key = integers : hash(key) % size

value

a) $x \mod 2^p$

b) $ax+b$

c) ~~random~~

b) better than (a)

why?

$x \mod 2^p$, whatever I'm dividing is an % even

remainder is either 0 or 1

b > a > c