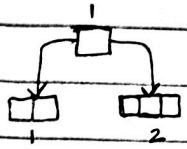
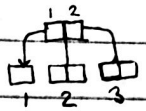
2-3-4 Trees

Basic Properties:

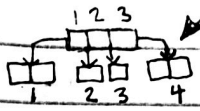
- Is a search tree → left subtree values ≤ current value < right subtree values



- All leaves must be on same level
- Tree grows/shrink from the root



2 nodes → 2 children nodes, current node has 1 item
3 nodes → 3 children nodes, current node has 2 items
4 nodes → 4 children nodes, current node has 3 items



- No gaps, so if your nodes don't meet the format above, something is wrong (Exception when only 1 node)

Why do we like it?

- Balanced Tree → Great for searching

TC:

- Search time is at most linear to height of tree

$$\log_4(N/3) \leq \text{height} \leq \log_2(N) \rightarrow \Theta(\log_2(N))$$

↑ if all nodes are 4-nodes

↑ if all nodes are 2-nodes

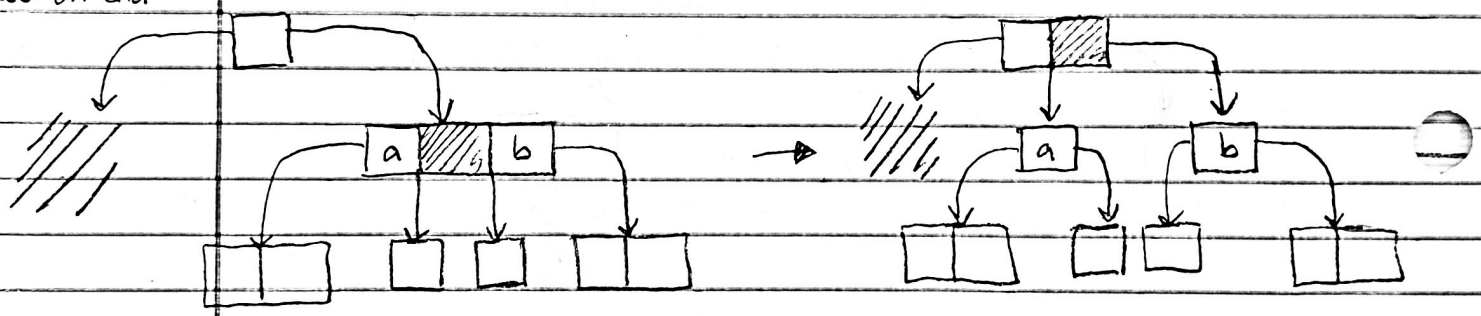
Inserting:

- We don't add new node to tree, but instead, we find a node that would satisfy the ordering and can fit another item into it

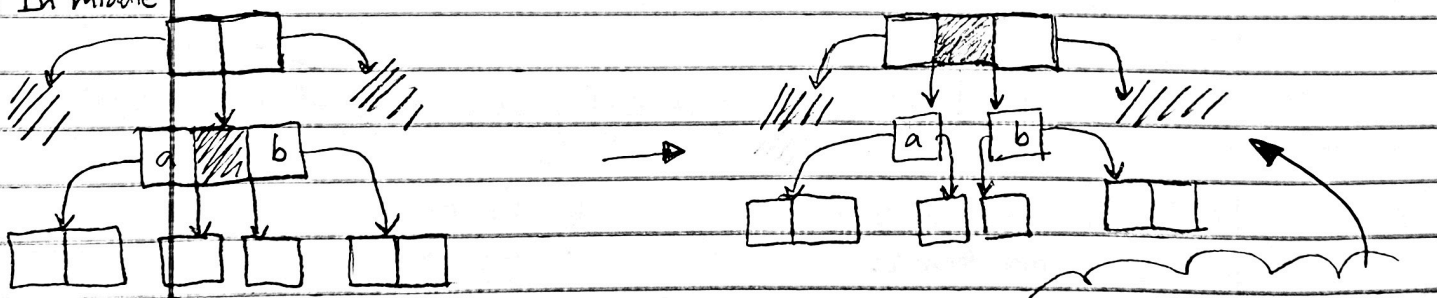
* Must watch out for case when node cannot fit anymore items ⇒ must split the node

Preemptive Split: fix (split) all 4-nodes reached when when searching through tree for correct node

Case: On end

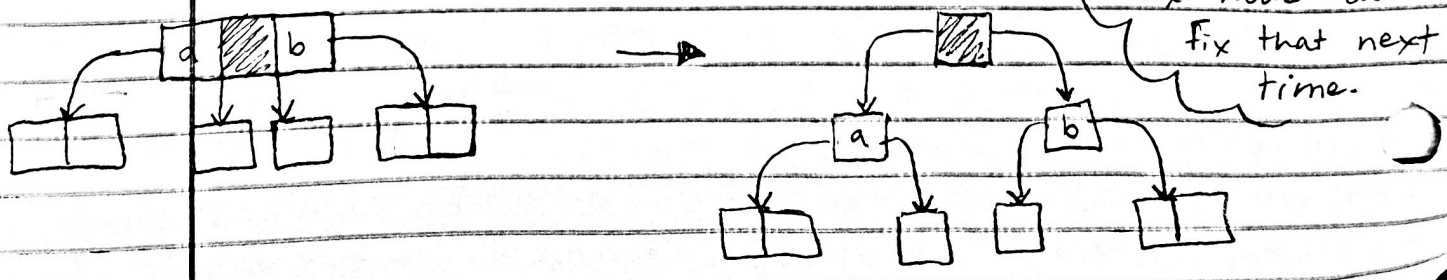


Case: In middle



* Note after fixed, do not go back to fix node above. Fix that next time.

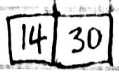
Case: root (this is how the tree grows)



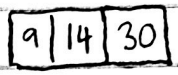
Ex: Insert 30, 14, 9, 33, 24, 15, 17, 20, 42

30:

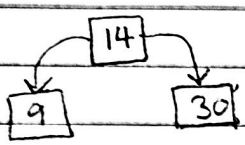
14:



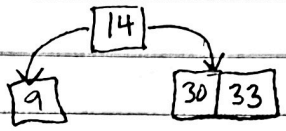
9:



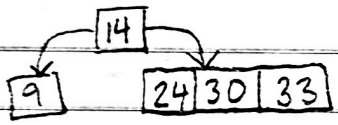
33: Found 4-node
⇒ preemptive split



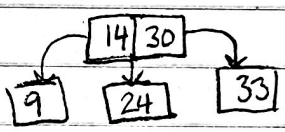
Add in 33 to correct node



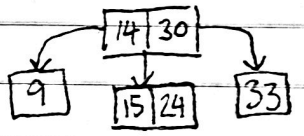
24:



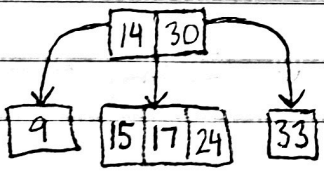
15: Found 4-node
⇒ preemptive split



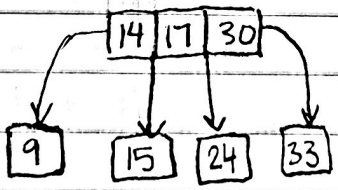
Add in 15 to correct node



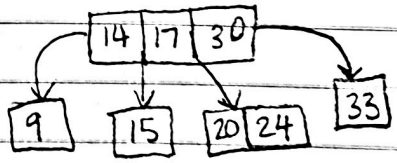
17:



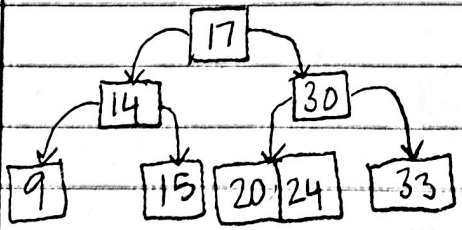
20: Found 4-node
⇒ preemptive split



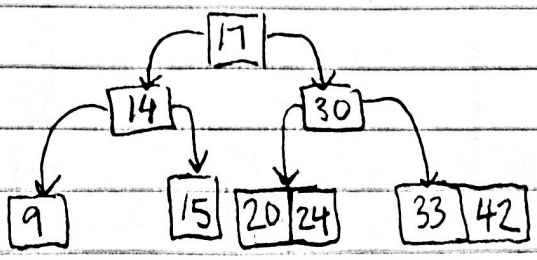
Add in 20 to correct node



42: Found 4-node
⇒ preemptive split



Add in 42 to correct node



Hash Tables:

Load factor: $\alpha = N/M$ ← should be a value b/w 0 & 1 for open addressing

N = # items in table M = table size

-Used to determine when to resize table (If > than specific threshold when inserting)

If threshold too large → Increased chances of collisions

If threshold too small → A lot of processing power spent on creating new table and rehashing (Increased amount of space used)

2 Collision Methods:

Separate Chaining: Same index → add to linked list

Major Concerns: if same idx over & over, end up searching through length of linked list

Open Addressing: store item in another available spot

Major Concerns: if same area → chances of data clustering Also deletion becomes harder

linear probing ⇒ primary clustering: longer the chain of next address filled → higher chance of another collision and increase of chain (positive-feedback loop)

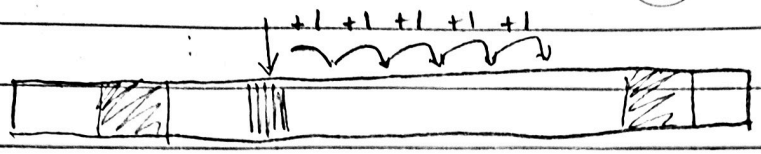
quadratic probing ⇒ secondary clustering

Handling deletion:

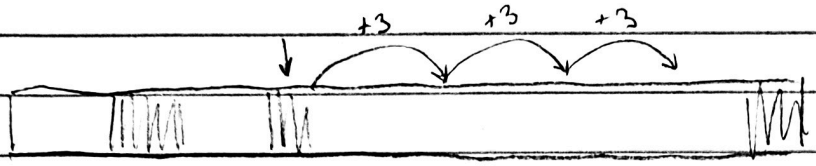
-When deleting, do not leave deleted cells as empty, but mark as deleted

- Will allow for continued searching after a deletion has occured

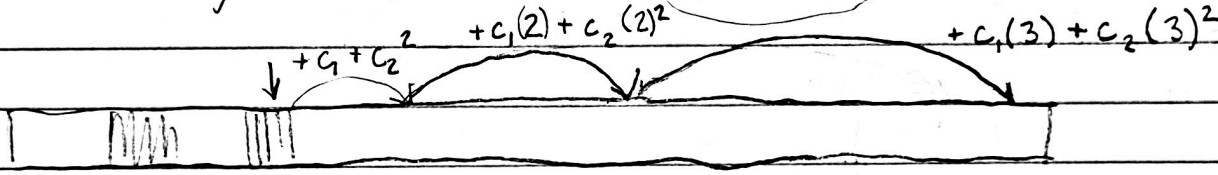
Linear Probing: $h(k, f, M) = (h_1(k) + f) \% M$



3 rehash (linear): $h(k, f, M) = (h_1(k, M) + 3f) \% M$

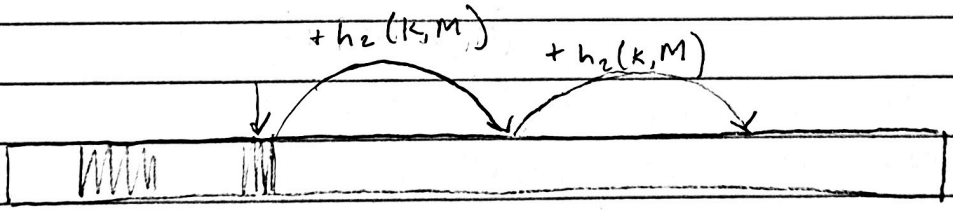


Quadratic Probing = $h(k, f, M) = (h_1(k) + c_1 f + c_2 f^2) \% M$



Double hashing: $h(k, f, M) = (h_1(k, M) + f * h_2(k, M)) \% M$

should not equal 0



f = # of failed probes

% = modulus (remainder)

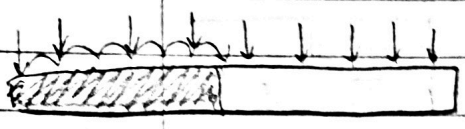
Ex: $5 \div 3 = 3$ R2

$5 \% 3 = 2$

Ex: Half-filled M sized table using linear probing

Average Insertion Time:

Occupied Cells Consecutive:



TC for updating cell with inserted value:
$= O(1)$

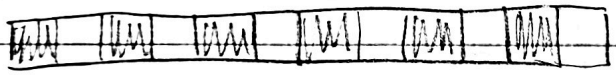
Needs to be done for any index chosen
$= M * O(1) = M$

If non empty cell, addition TC for iterating through cells until empty one found = # iterations can be from 1 to $\frac{M}{2}$

$$\frac{\left(M + \sum_{n=0}^{M/2} k\right)}{M} = \frac{M + \frac{M^2+2M}{8}}{M} = 1 + \frac{M+2}{8} = \boxed{O(M)}$$

# of all possible indexes we might get → M

Occupied Cells Alternate:



TC of landing on an available cell: $O(1)$
Chances of this happening = $M/2$

TC of landing on an occupied cell: $O(2)$
- $O(1)$ for updating cell with inserted value
- $O(1)$ for shifting down 1 to next empty cell

Chances of this happening: $M/2$

$$\frac{\left(1 * \frac{M}{2}\right) + \left(2 * \frac{M}{2}\right)}{M} = \frac{\frac{3M}{2}}{M} = \frac{3}{2} = \boxed{O(1)}$$

# of possible indexes we might get → M