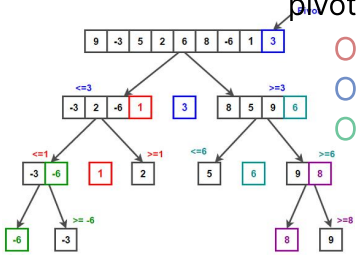
Sorting

Worst Case
Average Case
Best Case

Quick Sort

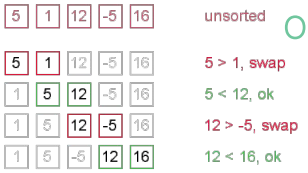
choose any pivot



O(n^2)
O(nlogn)
O(nlogn)

Bubble Sort

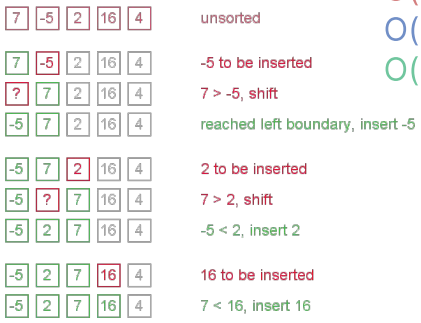
O(n^2)
O(n^2)
O(n)



repeat till sorted

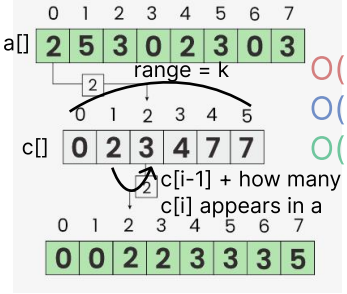
Insertion Sort

O(n^2)
O(n^2)
O(n)



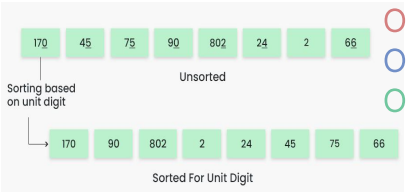
insert i+1 till sorted

Counting Sort



O(n + k)
O(n + k)
O(n + k)

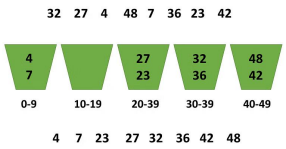
Radix Sort



O(n * digits)
O(n * digits)
O(n * digits)

Use any O(n) alg to sort for each digit

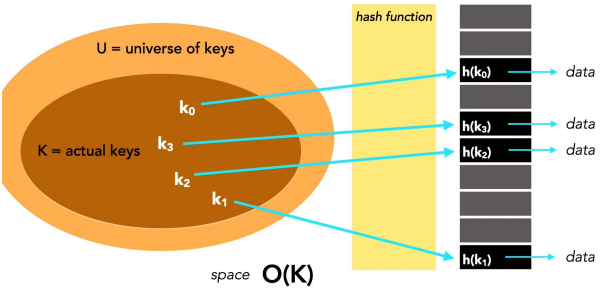
Bucket Sort



O(n^2)
O(n logn)
O(n + k)

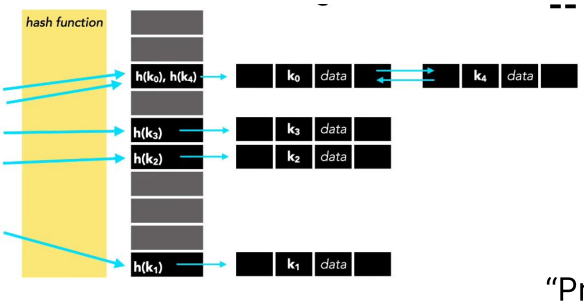
Put each element type into it's own bucket
then insertion sort each bucket

Hash Tables

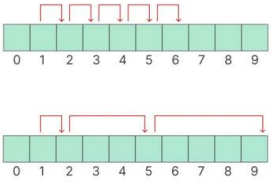


Collisions occur when 2 keys have the same hash

Method 1: Chaining



Method 2: Open Addressing -- Probing

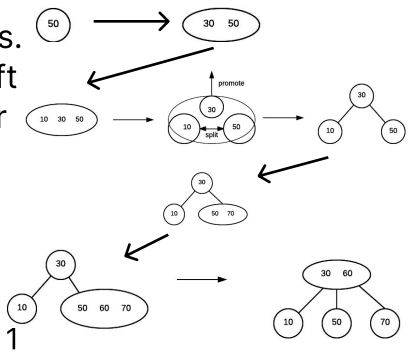


"Probe" until next empty slot if any collisions

B-Trees

Special, balanced trees. Sorted, searchable (left < right). m children per node

insertion



2-3 Trees

A specific form of B-tree

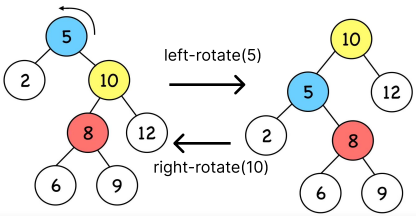
1. each node has either 1 value or 2 values
2. 1 value → 2 nodes
3. 2 values → 3 nodes
4. all leaf nodes are at the same level of the tree

for deletes, do the insertion in reverse: imagine the node to delete was just inserted.

Red-black Tree

Special type of BST.

1. every node be either red or black.
2. Root must be black
3. Red node → black children
4. Null nodes are black.
5. Every path from root to null must have exactly the same number of black nodes.



insertion

Null Nodes

Always insert as red, then rotate/colorflip as needed.