cphbusiness

COPENHAGEN BUSINESS ACADEMY

Algorithms and Data Structure-Day3

Today

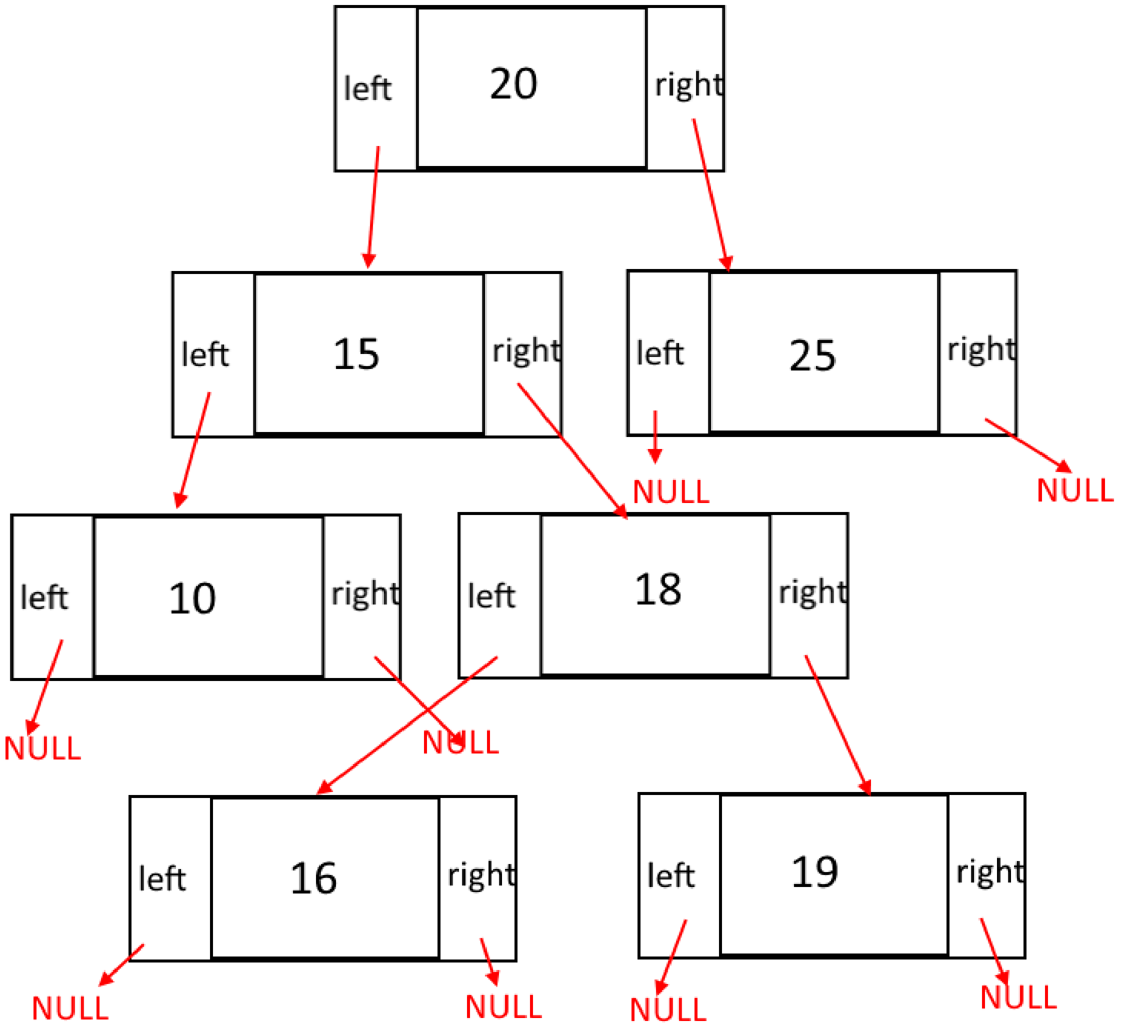
- Classic Algorithms
  - Recursion
  - Merge sort
  - Quick sort

- Data Structures
  - Hash Table/Hash map

- Efficiency of algorithms
  - Big O

Binary Search Tree (BST)

- All nodes smaller than root is on the left
- All nodes greater than root is on the right
- Olog(n) for a search



Heapsort

- Ordered binary tree
- Max heap – parent > child
- Array of values
- Sort the array in location
- Left child = 2*i +1
- Right child = 2* + 2

Binary Search Tree

- Balances and not Balanced binary tree

- Complete binary tree if all levels except possibly the last are completely filled with exception to the last level

- Divide and conquer

- Balanced binary tree makes the search quicker

Recursive Sorting

- Divide and Conquor
- Merge Sort
- Quick Sort

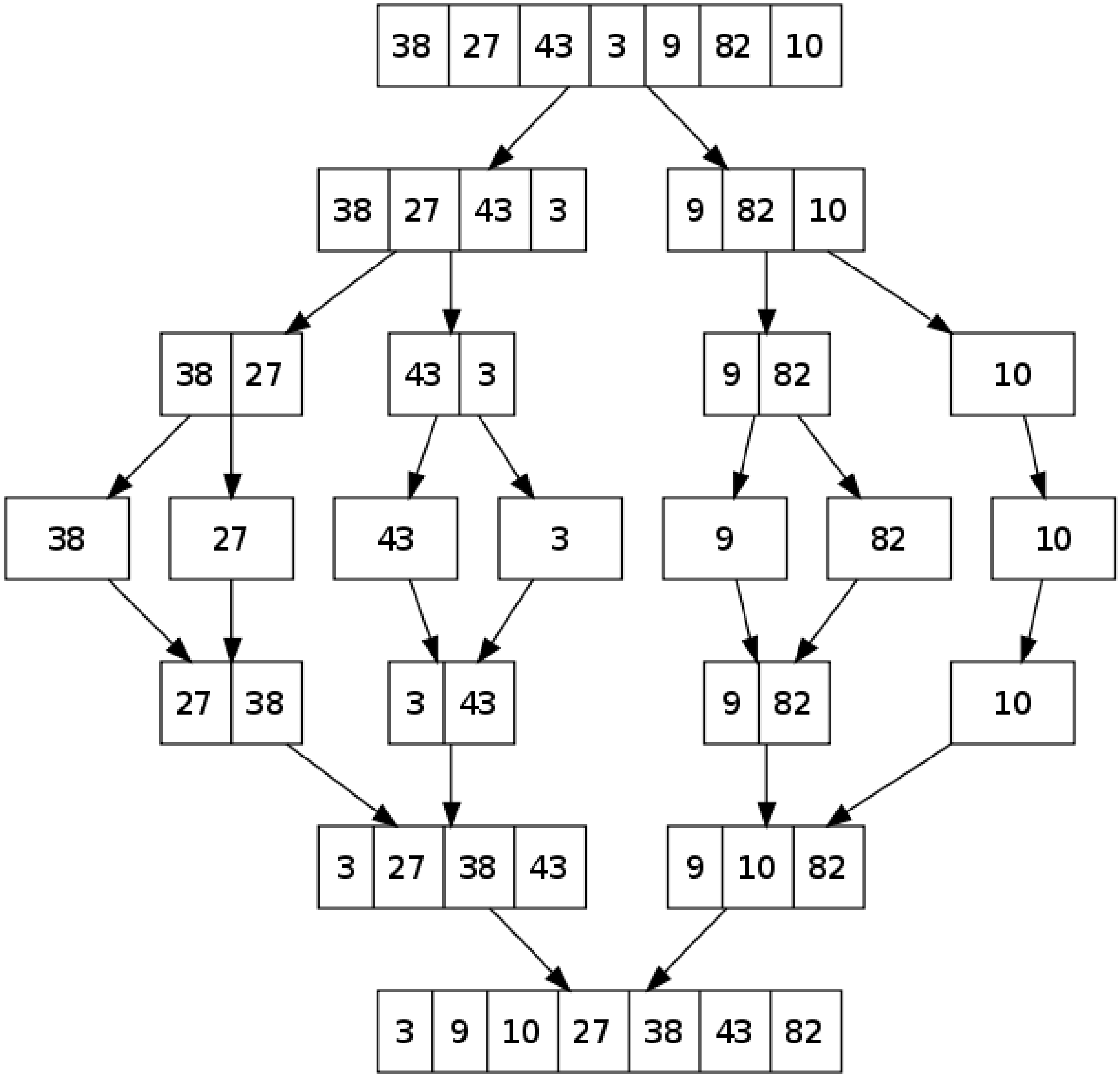
Divide and Conquor

- Divide and break
  - Break the problem in to smalle sub-problem recursively
  - Sub-problem should represent a part of the original problem
  - Keep on dividing until no more division is possible

- Conquer/Solve
  - Smalles sub-problem are solved
  - Solutions of all the sub-problems are merged

- Merge/Combine
  - Recursively combines small solutions to the big solutions

Merge Sort – top down



38 27 43 3 9 82 10

38 27 43 3 | 9 82 10

38 27 | 43 3 | 9 82 | 10

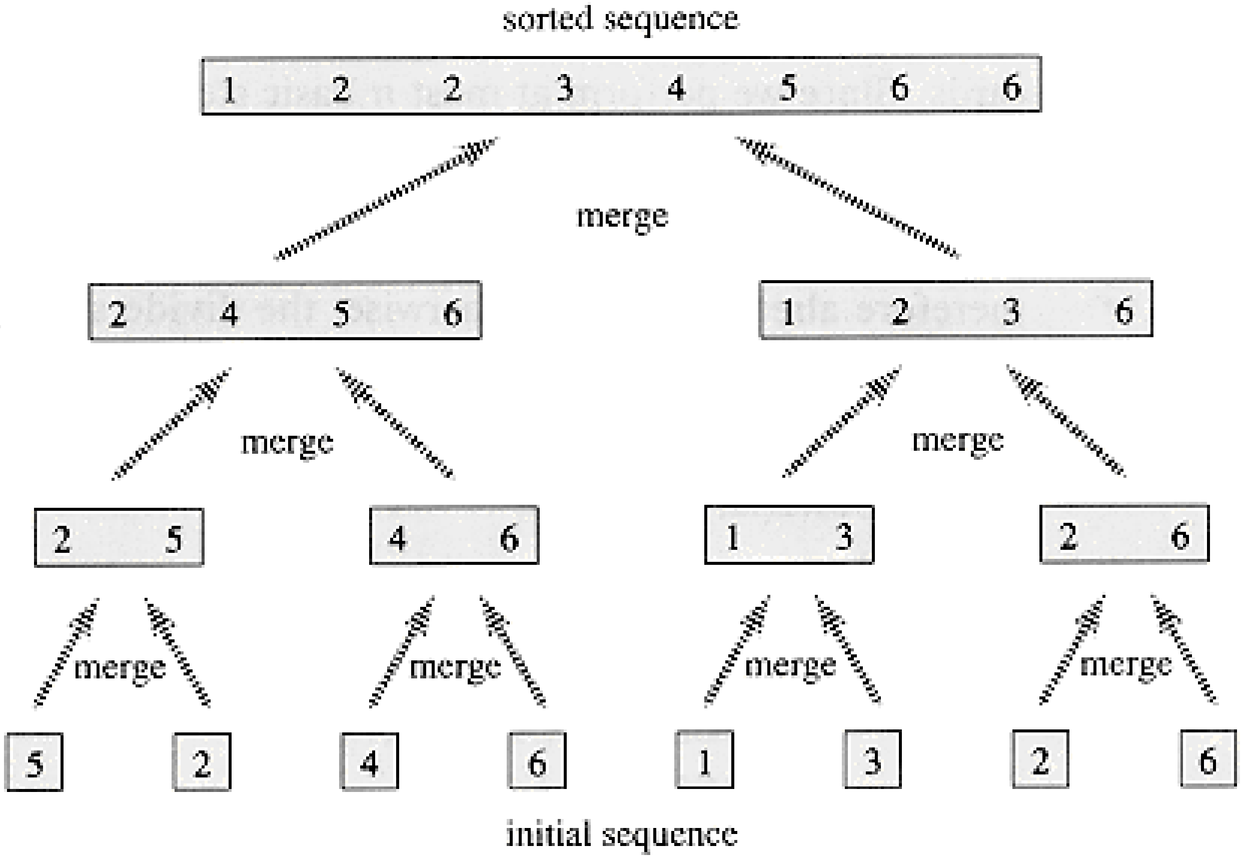
38 | 27 | 43 | 3 | 9 | 82 | 10

27 38 | 3 43 | 9 82 | 10

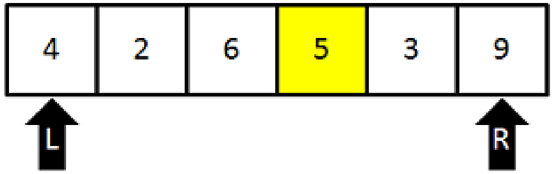
3 27 38 43 | 9 10 82

3 9 10 27 38 43 82

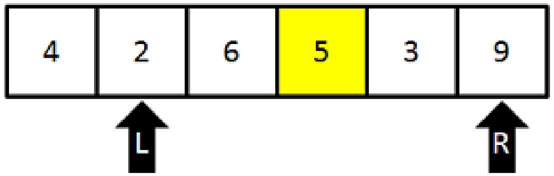
Merge Sort – Bottom up



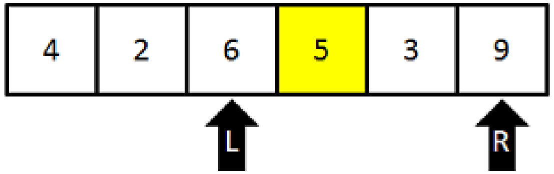
Step 2
Start pointers at left and right



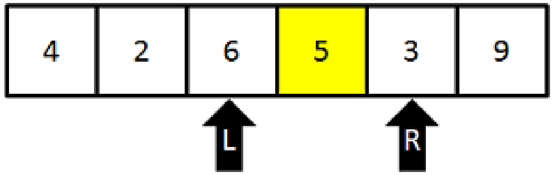
Step 3
Since 4 < 5, shift left pointer



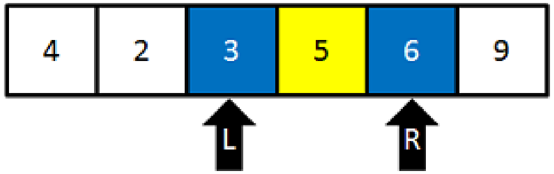
Step 4
Since 2 < 5, shift left pointer
Since 6 > 5, stop



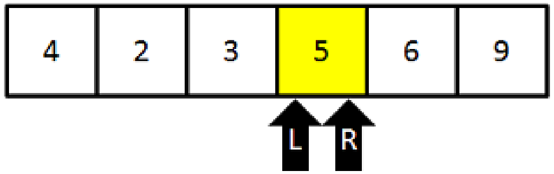
Step 5
Since 9 > 5, shift right pointer
Since 3 < 5, stop



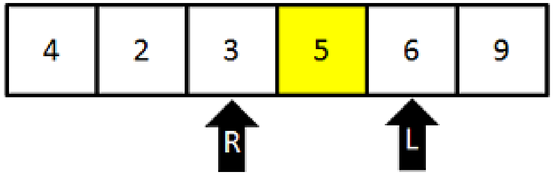
Step 6
Swap values at pointers



Step 7
Move pointers one more step



Step 8
Since 5 == 5,
move pointers one more step
Stop



Choice of data structure

	Access	Search	Insert	Delete
Array	O(1)	O(n)	O(n)	O(n)
Linked List	O(n)	O(n)	O(1)	O(1)

Binary Search Tree	O(n)
	O(log(n))
		O(n)
		O(log(n))
			O(n)
			O(log(n))
				O(n)
				O(log(n))

HashTable	??	??	??	??

Efficiency of algorithms

Algorithm	Time Complexity
Quicksort	O(n^2)
Merge sort	O(n log(n))
Balanced binary tree	O(log *n*)
Heapsort	O(n log(n))
Bubble sort	O(n^2)
Insertion sort	O(n^2)
Selectio sort	O(n^2)