Lecture 2: Divide&Conquer Paradigm, Merge sort and Quicksort

Instructor: Saravanan Thirumuruganathan

Outline

1. Divide and Conquer
2. Merge sort
3. Quick sort

In-Class Quizzes

- **URL:** `http://m.socrative.com/`
- **Room Name: 4f2bb99e**

Divide And Conquer Paradigm

- D&C is a popular algorithmic technique
- Lots of applications
- Consists of three steps:
  1. **Divide** the problem into a number of sub-problems
  2. **Conquer** the sub-problems by solving them *recursively*
  3. **Combine** the solutions to sub-problems into solution for original problem

Divide And Conquer Paradigm

When can you use it?

- The sub-problems are easier to solve than original problem
- The number of sub-problems is **small**
- Solution to original problem can be obtained easily, once the sub-problems are solved

Recursion and Recurrences

- Typically, D&C algorithms use **recursion** as it makes coding simpler
- Non-recursive variants can be designed, but are often slower
- If all sub-problems are of equal size, can be analyzed by the recurrence equation

$$T(n) = aT(\frac{n}{b}) + D(n) + C(n)$$

- a: number of sub-problems to solve
- b: how fast the problem size shrinks
- $D(n)$: time complexity for the divide step
- $C(n)$: time complexity for the combine step

D&C Approach to Sorting

How to use D&C in Sorting?

- Partition the array into sub-groups
- Sort each sub-group recursively
- Combine sorted sub-groups if needed

Why study Merge Sort?

- One of the simplest and efficient sorting algorithms
- Time complexity is $\Theta(n \log n)$ (vast improvement over Bubble, Selection and Insertion sorts)
- Transparent application of D&C paradigm
- Good showcase for time complexity analysis

Merge Sort

High Level Idea :

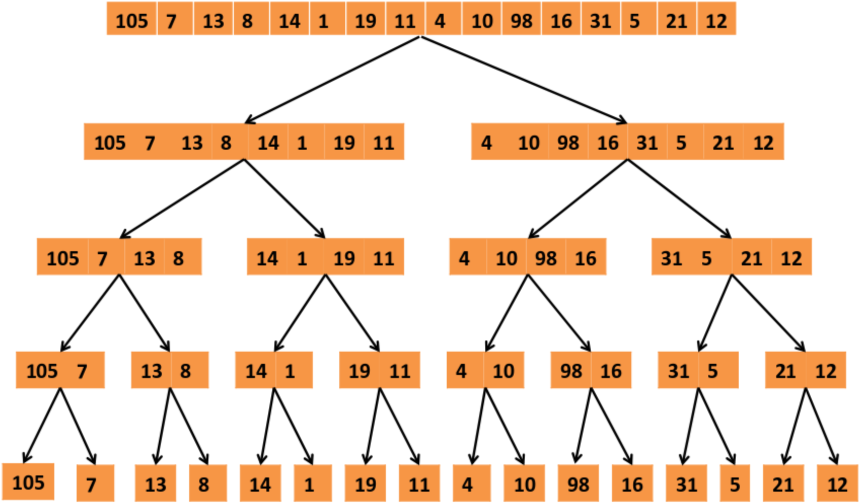
- Divide the array into two **equal** partitions - left and right (ignore edge cases for now)
- Sort left partition recursively
- Sort right partition recursively
- Merge the two sorted partitions into the output array

Merge Sort Pseudocode

Pseudocode:

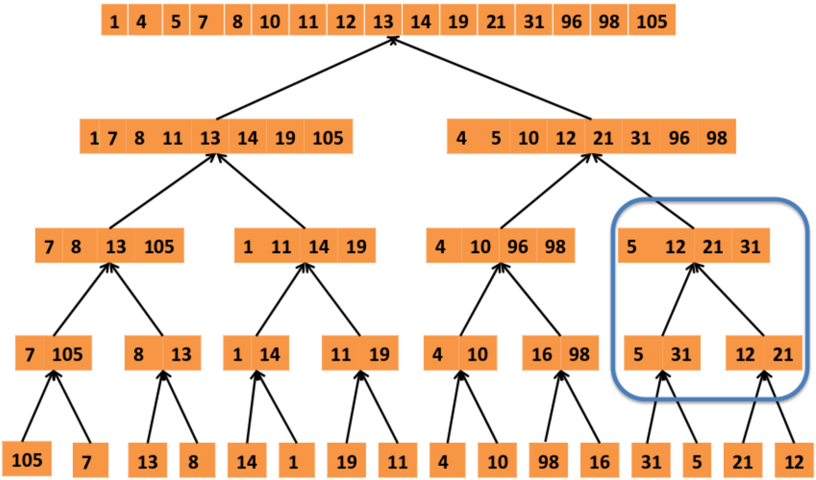
```
MergeSort(A, p, r):
    if p < r:
        q = (p+r)/2
        Mergesort(A, p , q)
        Mergesort(A, q+1, r)
        Merge(A, p, q, r)
```

Merge Sort - Divide[1]



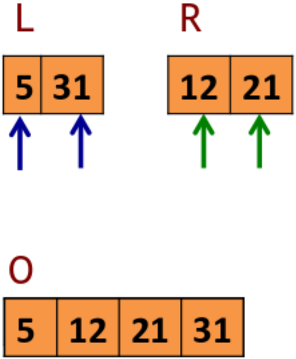
[1] http://web.stanford.edu/class/cs161/slides/0623_mergesort.pdf

Merge Sort - Combine[2]



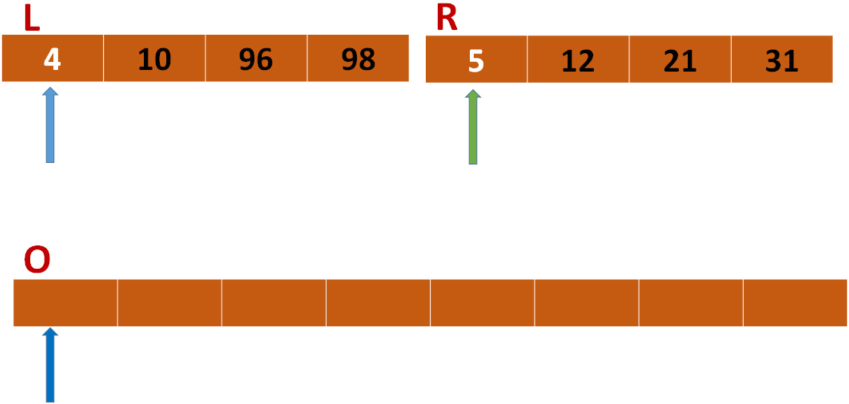
[2] http://web.stanford.edu/class/cs161/slides/0623_mergesort.pdf

Merging Two Sorted Lists[3]

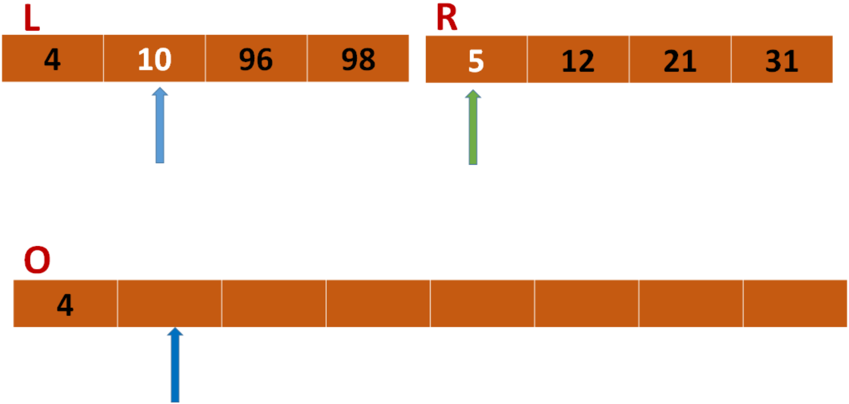


[3] http://web.stanford.edu/class/cs161/slides/0623_mergesort.pdf

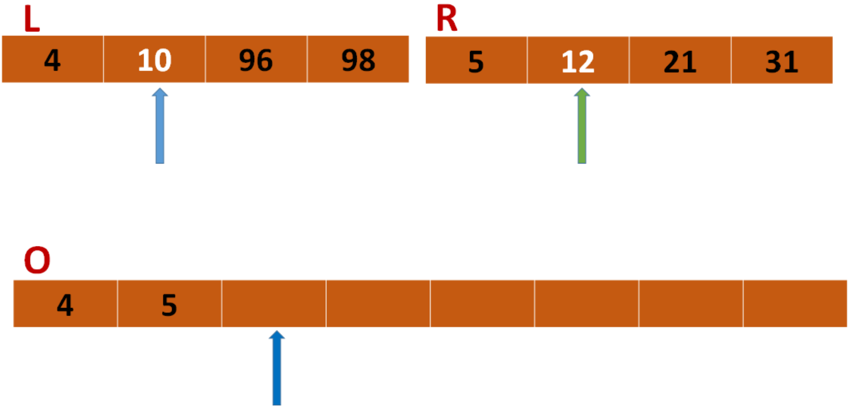
Merging Two Sorted Lists



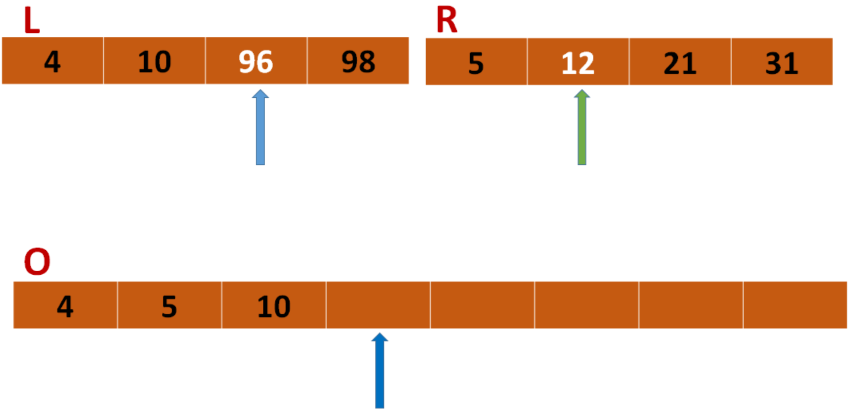
Merging Two Sorted Lists



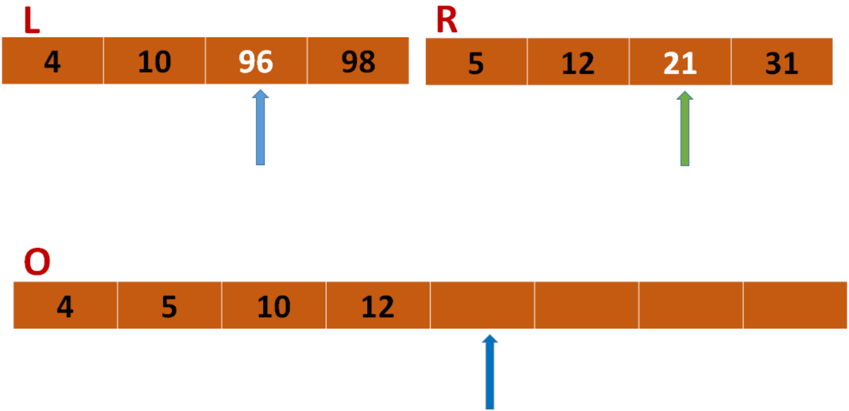
Merging Two Sorted Lists



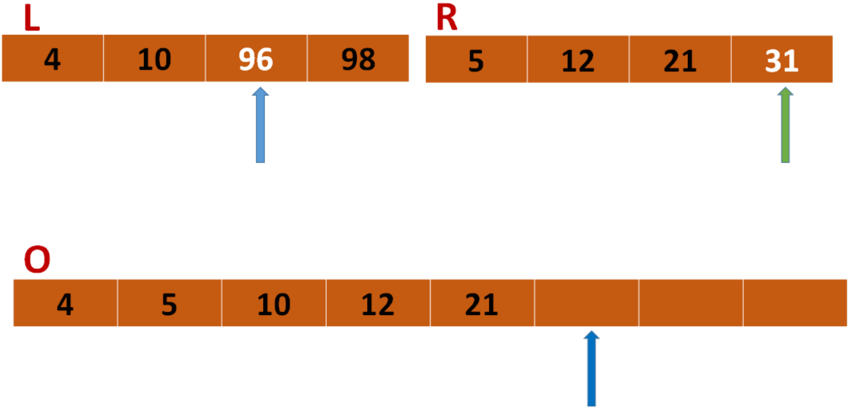
Merging Two Sorted Lists



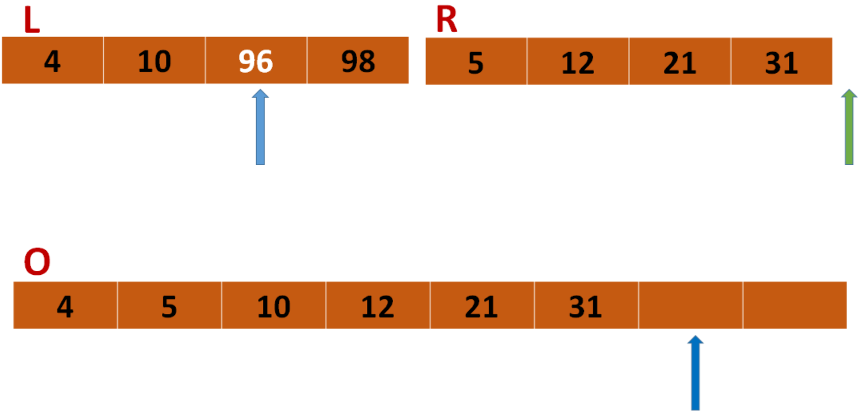
Merging Two Sorted Lists



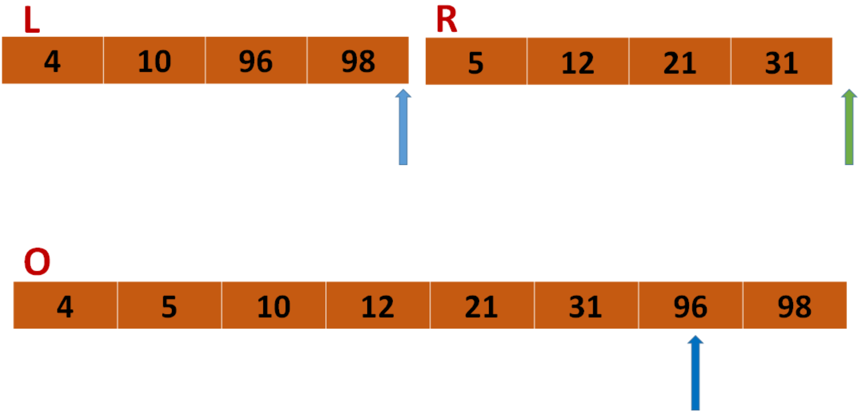
Merging Two Sorted Lists



Merging Two Sorted Lists



Merging Two Sorted Lists



Merging Two Sorted Lists

Merge Pseudocode:

```
Merge(A,B,C):
    i = j = 1
    for k = 1 to n:
        if A[i] < B[j]:
            C[k] = A[i]
            i = i + 1
        else: (A[i] > B[j])
            C[k] = B[j]
            j = j + 1
```

Analyzing Merge Sort: Master Method

Quiz!

- General recurrence formula for D&C is $T(n) = aT(\frac{n}{b}) + D(n) + C(n)$
- What is a?
- What is b?
- What is $D(n)$?
- What is $C(n)$?

Quiz!

- General recurrence formula for D&C is

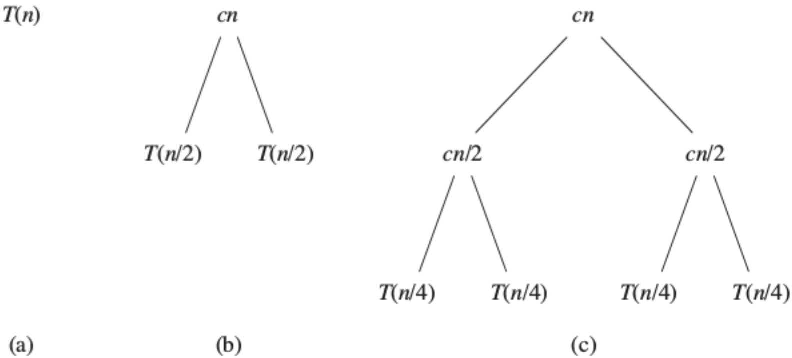
$$T(n) = aT(\frac{n}{b}) + D(n) + C(n)$$

- $a = 2,\ b = 2$
- $D(n) = O(1)$
- $C(n) = O(n)$
- Combining, we get:

$$T(n) = 2T(\frac{n}{2}) + O(n)$$

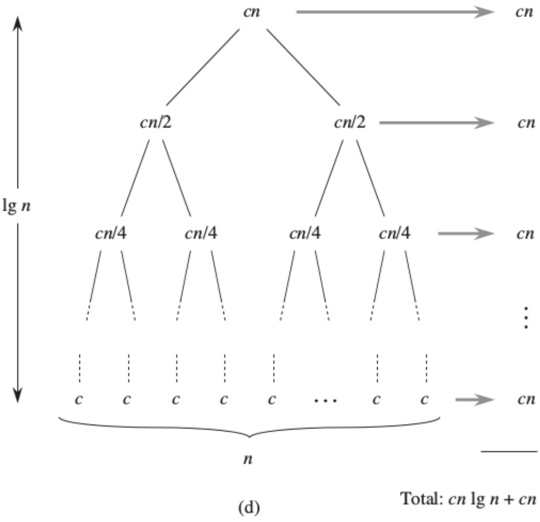
- Using Master method, we get $T(n) = O(n \log n)$
- If you are picky, $T(n) = T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + O(n)$

Analyzing Merge Sort: Recursion Tree[4]



[4] CLRS Book

Analyzing Merge Sort: Recursion Tree[5]



[5]CLRS Book

Merge Sort Vs Insertion Sort

- Merge Sort is very fast in general ($O(n \log n)$) than Insertion sort ($O(n^2)$)
- For "nearly" sorted arrays, Insertion sort is faster
- Merge sort has $\Theta(n \log n)$ (i.e. both best and worst case complexity is $n \log n$
- Overhead: recursive calls and extra space for copying
- Insertion sort is in-place and adaptive
- Merge sort is easily parallizable