Data Structure & Algorithms

Sunbeam Infotech

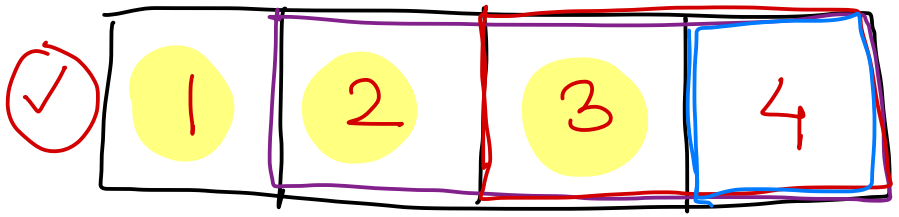
Agenda

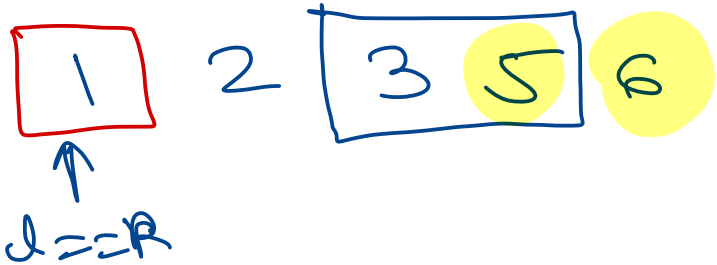
- Q & A
- Quick Sort
- Merge Sort
- Comparing Sorting Algorithms
- Stack & Queue
- Linear Queue
- Circular Queue

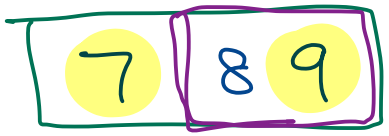
Quick Sort

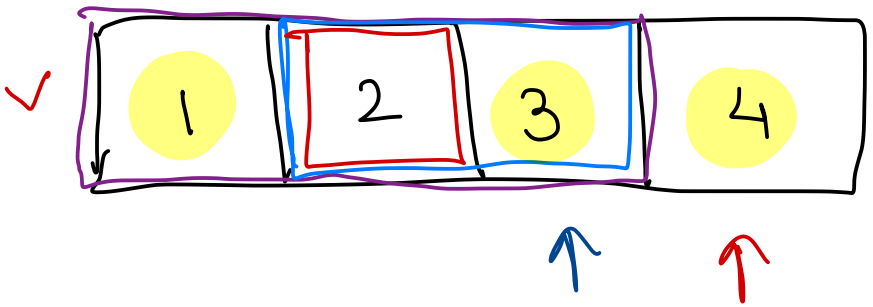
```
if(left >= right)
    return;
```

0	1	2	3	4	5	6	7	8
2	3	6	1	5	6	7	9	8



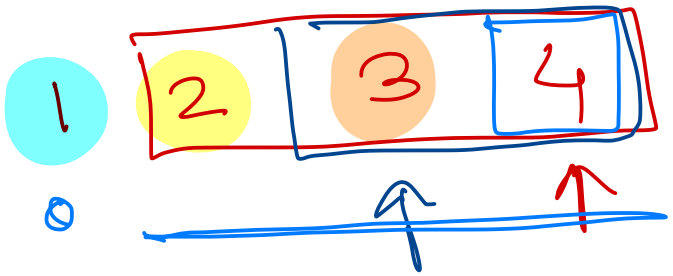


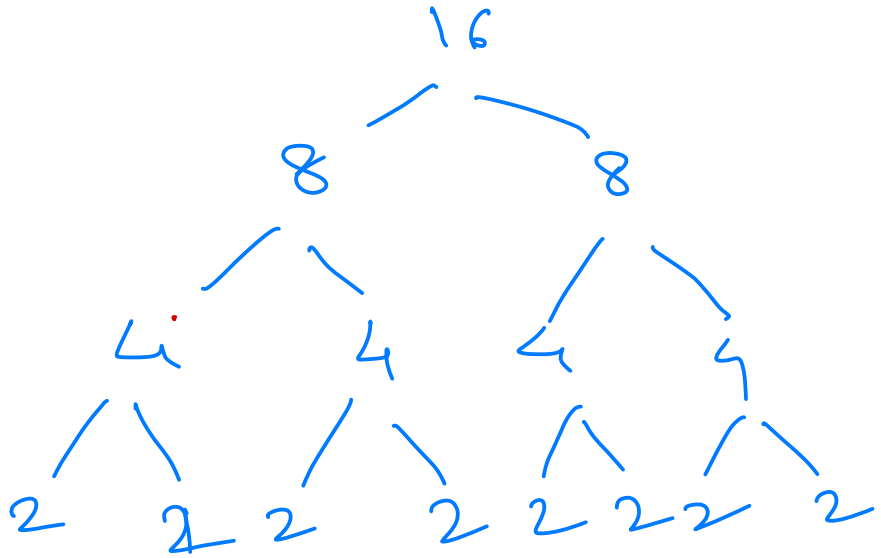


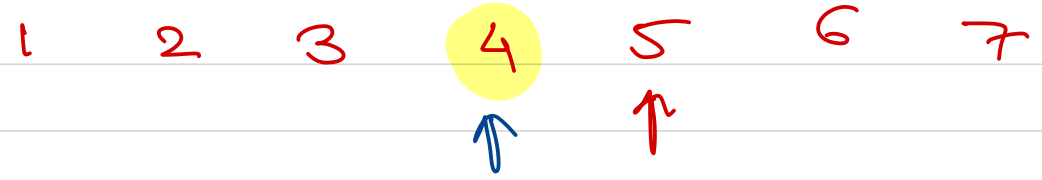


Quick Sort – Time complexity

- Quick sort pivot element can be
  - First element or Last element
  - Random element
  - Median of the array
- Quick sort time
  - Time to partition as per pivot – T(n)
  - Time to sort left partition – T(k)
  - Time to sort ~~left~~ right partition – T(n-k-1)
- Worst case
  - $T(n) = T(0) + T(n-1) + O(n) \Rightarrow O(n^2)$
- Best case
  - $T(n) = T(n/2) + T(n/2) + O(n) \Rightarrow O(n \log n)$
- Average case
  - $T(n) = T(n/9) + T(9n/10) + O(n) \Rightarrow O(n \log n)$





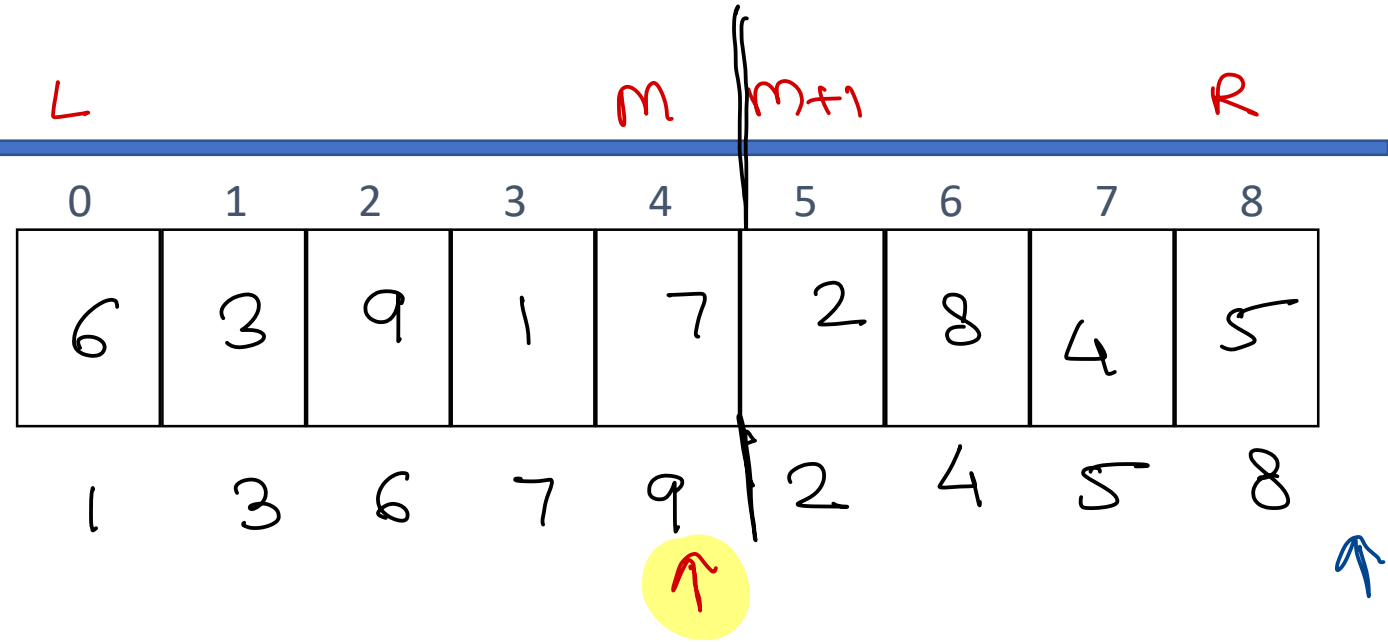


a[i] < a[left]

a[j] >= a[right]

Merge Sort

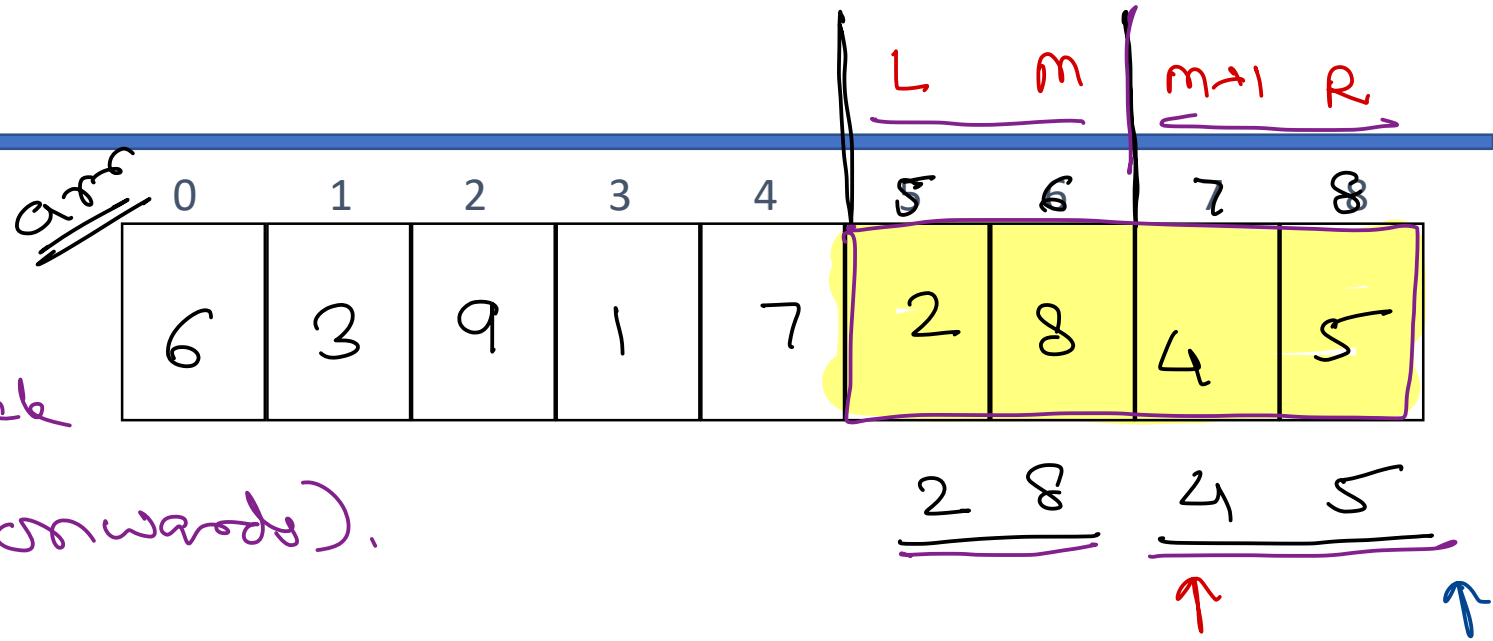
1. Divide array in two equal partitions.
2. sort both partitions individually (how ?).
3. merge these sorted partitions into a temp array
4. overwrite temp array back to original array.




```
(3) i=L, j=m+1, k=0;
while(i<=m && j<=R)
{
    if(a[i]<a[j]){
        t[k]=a[i];
        i++;
        k++;
    }else{
        t[k]=a[j];
        j++;
        k++;
    }
}
```

```
while(i<=m){
    t[k]=a[i];
    i++;
    k++;
}
while(j<=R){
    t[k]=a[j];
    j++;
    k++;
}
```

Merge Sort



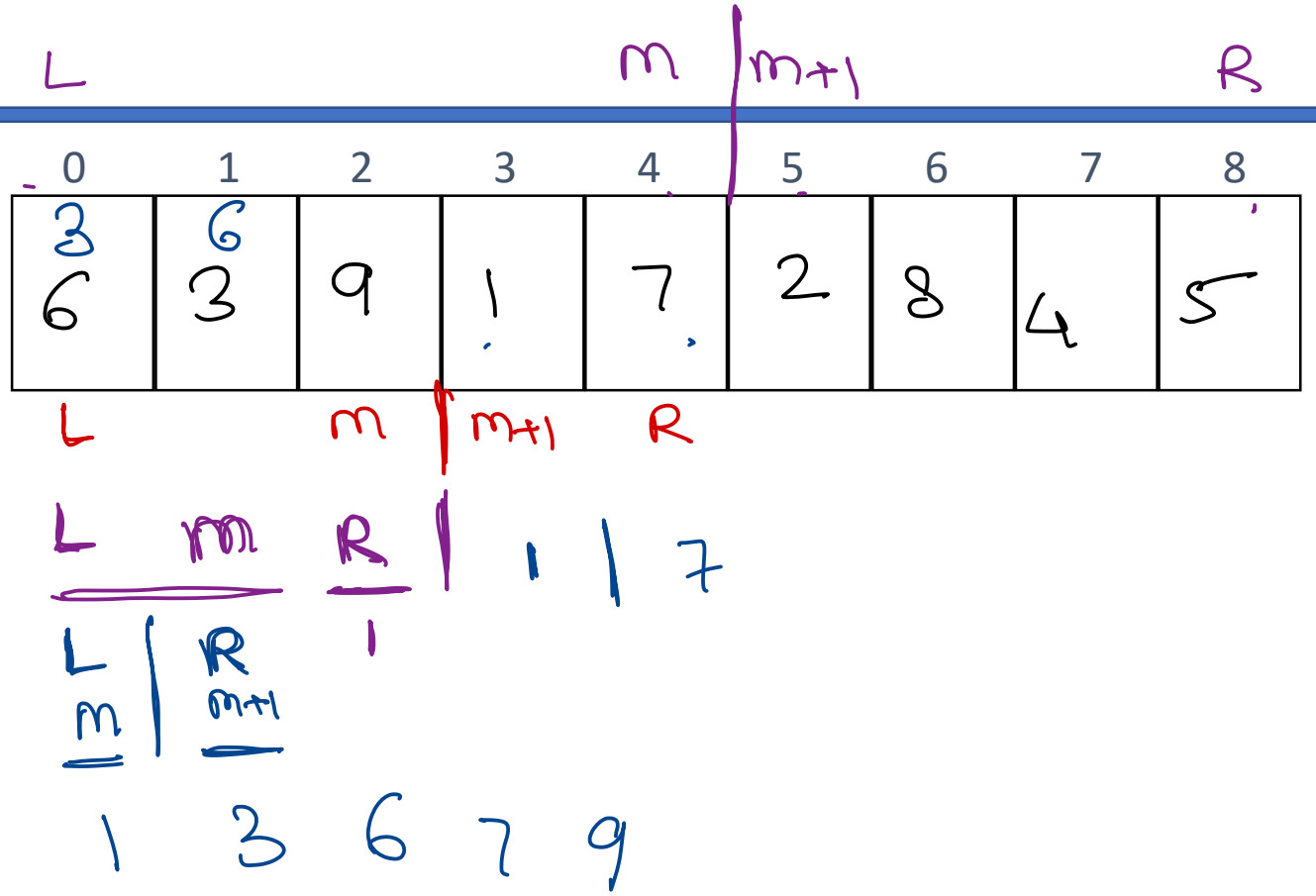
④ overwrite temp array back to original array (left onwards).

t: 2 4 5 8

0 1 2 3

④
```
for(i=0; i<k; i++)
    a[L+i] = t[i];
```

Merge Sort

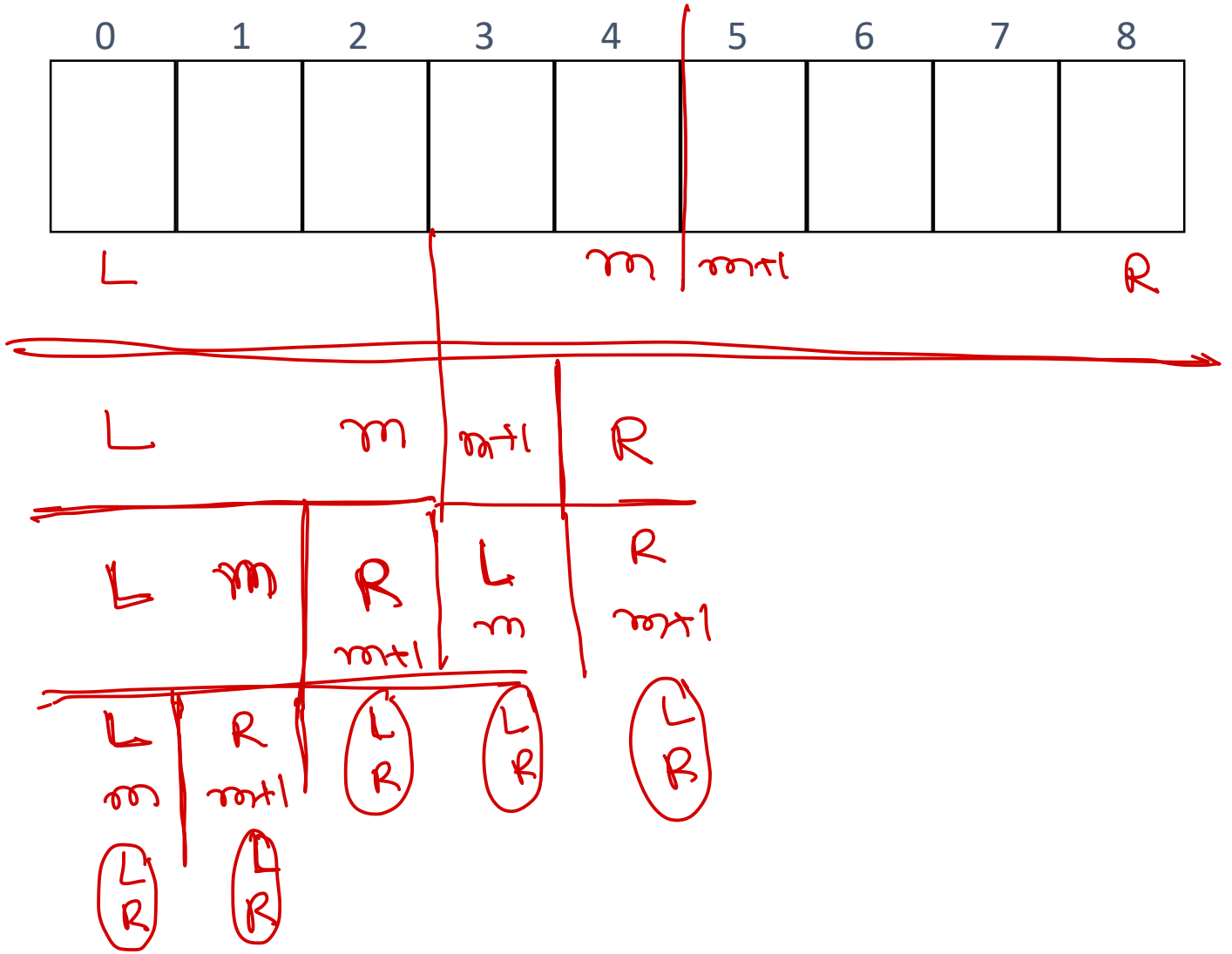


① `m = (L+R)/2;`

② `merge_sort(arr, L, m);`
`merge_sort(arr, m+1, R);`

```
if(left >= right)
    return;
```

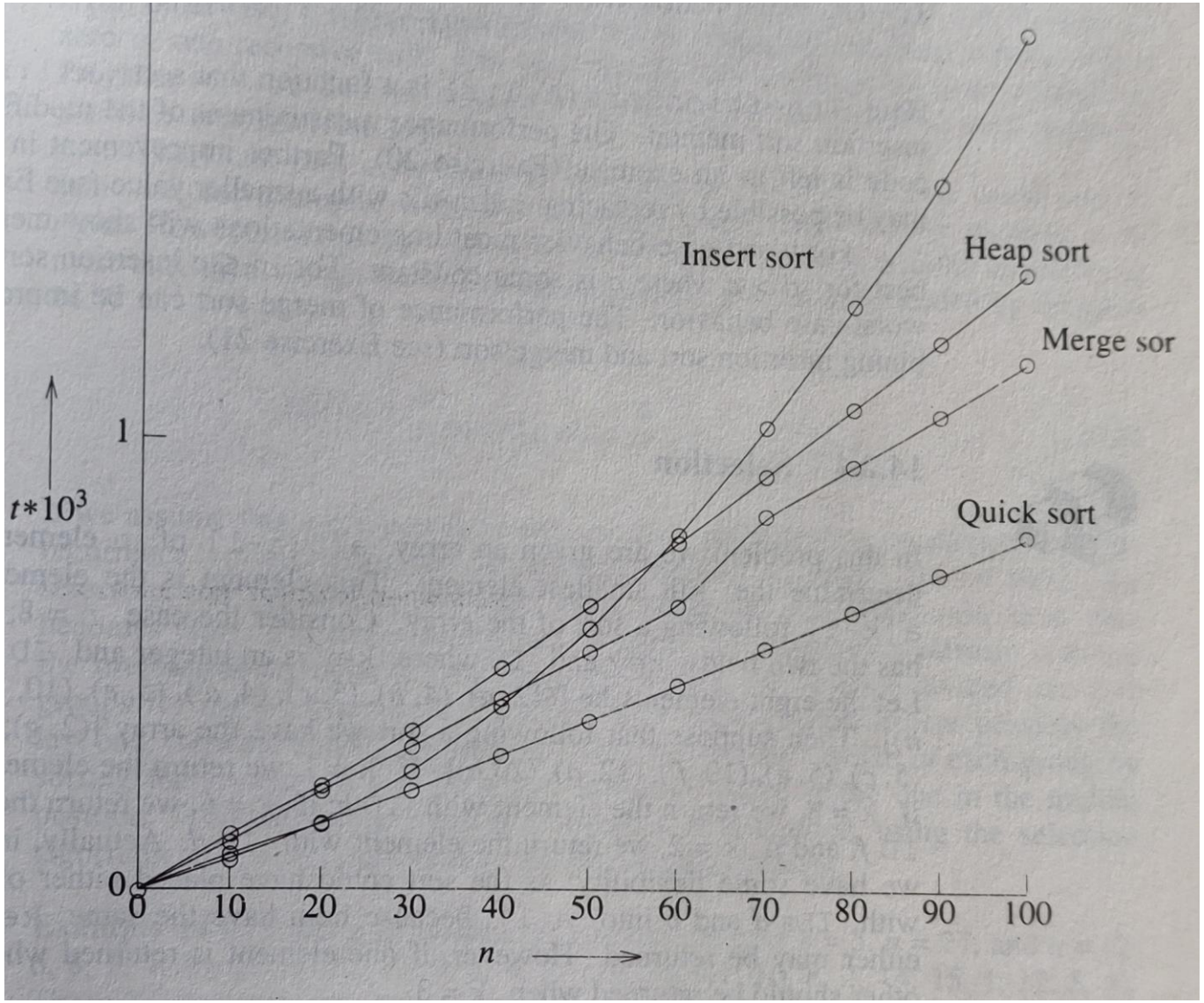
Merge Sort

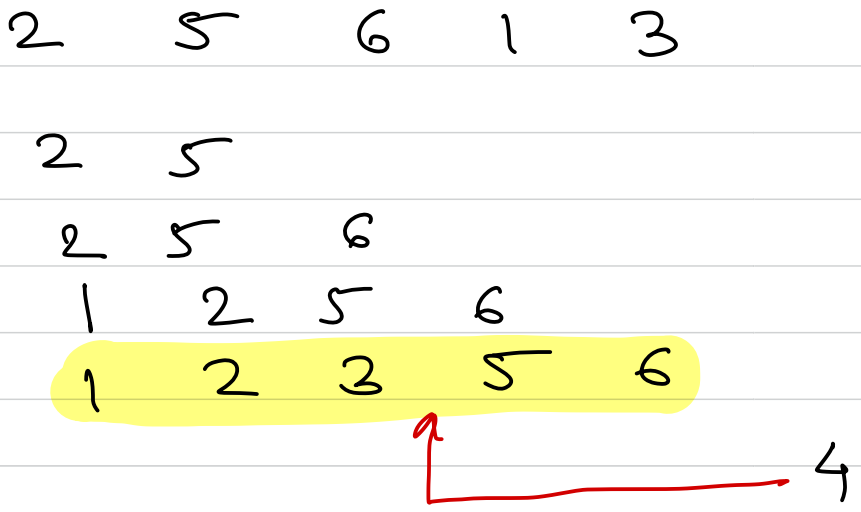


Sorting Algorithm Comparison

qsort() bsearch()

- Selection sort algorithm is too simple, but performs poor and no optimization possible.
- Bubble sort can be improved to reduce number of iterations.
- Insertion sort performs well if number of elements are too less. Good if adding elements and resorting.
- Quick sort is stable if number of elements increase. However worst case performance is poor.
- Merge sort also perform good, but need extra auxiliary space.





$O(n)$

$O(n)$

$O(n^2)$.

$O(n)$

$O(\log n)$

$O(n \log n)$

Thank you!

Nilesh Ghule <nilesh@sunbeaminfo.com>