CHAPTER 4

Quick Sort

- Quick Sort is a divide-and-conquer algorithm to sort an N element array in ascending order.
- It divides the array into two partitions and then sorting each partition recursively.
- It works recursively, by first selecting a random **pivot element** from the list array.
- Then it partitions the list into elements that are less than the pivot and greater than the pivot.
- The three steps of Quick sort are as follows:

- **Divide:** Partition (Rearrange) the elements and split the array into two subarrays and an element in between such that so that each element in the left subarray is less than or equal the middle element and each element in the right subarray is greater than the middle element.
- **Conquer:** Recursively sort the two subarrays.
- **Combine:** Since the subarrays are sorted in place, no work is needed to combine them.

```
Procedure Quicksort (a, p, r)
{
[Here a [] is the array, p is starting index, that is 0, and r is the
last index of array.]
 if (p < r) then
 {
  [Calling procedure partition]
   q = Partition (a, p, r);
  [First sublist]
   Quicksort (a, p, q);
  [Second sublist]
   Quicksort (a, q+1, r);
 }
}
```

```
Procedure Partition (a [], p, r)
{
[Initialization]
Set
    i)  pivot = a[p];
    ii)  i = p;
    iii)  j = r;
 while (i < j)
    {
 while(a[i] < = pivot && i < r)
                    set i = i + 1;
 while(a[j] > pivot && j > p)
                    set j =  j - 1;
 if(i < j) then
     {
  temp = a[i];
  a[i] = a[j];
  a[j] = temp;
      }
  }
Exchange a[p] ↔ a[j]
  return j;
 }
```

Performance of quick sort

Worst-case partitioning

- The worst-case behavior for quick sort occurs when the partitioning routine produces one subproblem with n - 1 elements and one with 0 elements.
- The partitioning costs $\Theta(n)$ time. Since the recursive call on an array of size 0 just returns, $T(0) = \Theta(1)$, and the recurrence for the running time is

$$T(n) = T(n - 1) + T(0) + \Theta(n)$$
$$= T(n - 1) + \Theta(n).$$

Prepared By
Sasmita Kumari Nayak

Intuitively, if we sum the costs incurred at each level of the recursion, we get an arithmetic Series, which evaluates to $\Theta(n^2)$ as shown in figure 3.6.

- Indeed, it is straightforward to use the substitution method to prove that the recurrence $T(n) = T(n - 1) + \Theta(n)$ has the solution $T(n) = \Theta(n^2)$.

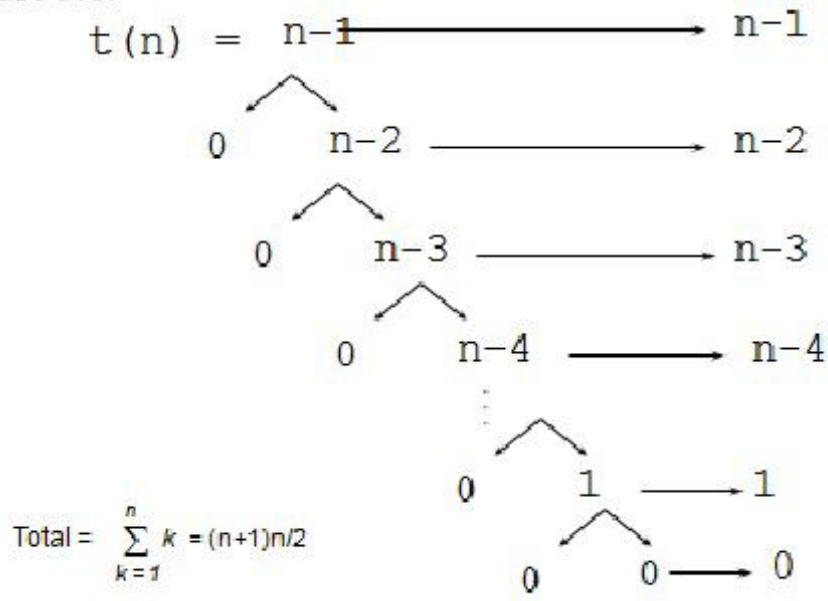


Figure 3.6

Best-case partitioning

- In best case analysis, the techniques divide and conquer works best if each array is divided into two equal subarrays of size n/2. This generates log n levels in the recursion tree.
- It is to be noted that partitioning is linear in the size i.e. on each level this is $O(n)$ as shown in figure 3.7. In this case, quicksort runs much faster. The recurrence for the running time is then,

$$T(n) \leq 2T(n/2) + O(n),$$

- The recurrence tree for the above recurrence is as shown in figure 3.7.
- By using case 2 of the master theorem has the solution $T(n) = O(n \lg n)$. Thus, the equal balancing of the two sides of the partition at every level of the recursion produces an asymptotically faster algorithm.

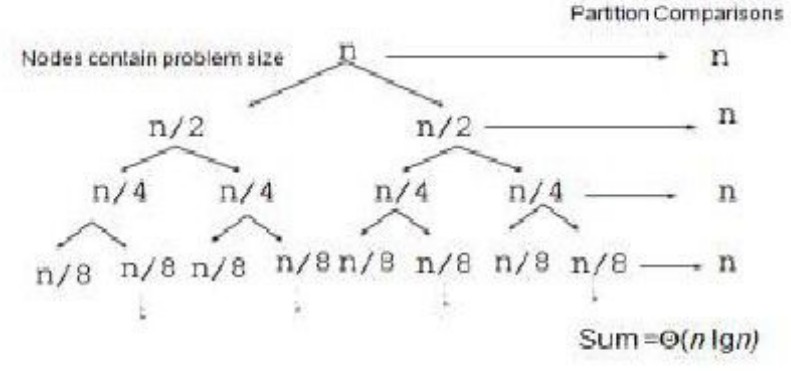


Figure 3.7

Average case analysis (Balanced partitioning)

- The average-case running time of quicksort is much closer to the best case than to the worst case.
- The recurrence equation is: $T(n) \leq T(9n/10) + T(n/10) + cn$
- Figure 3.8 shows the recursion tree for this recurrence. Every level of the tree has cost cn, until a boundary condition is reached at depth $\log 10n = \Theta(\lg n)$, and then the levels have cost at most cn. The recursion terminates at depth $\log 10/9n = \Theta(\lg n)$. The total cost of quicksort is therefore $O(n \lg n)$.
- Thus, with a 9-to-1 proportional split at every level of recursion, which intuitively seems quite unbalanced, quicksort runs in $O(n \lg n)$ time asymptotically the same as if the split were right down the middle.
- In fact, even a 99-to-1 split yields an $O(n \lg n)$ running time.

Prepared By
Sasmita Kumari Nayak

- The reason is that any split of *constant* proportionality yields a recursion tree of depth $\Theta(\lg n)$, where the cost at each level is $O(n)$.
- The running time is therefore $O(n \lg n)$ whenever the split has constant proportionality.

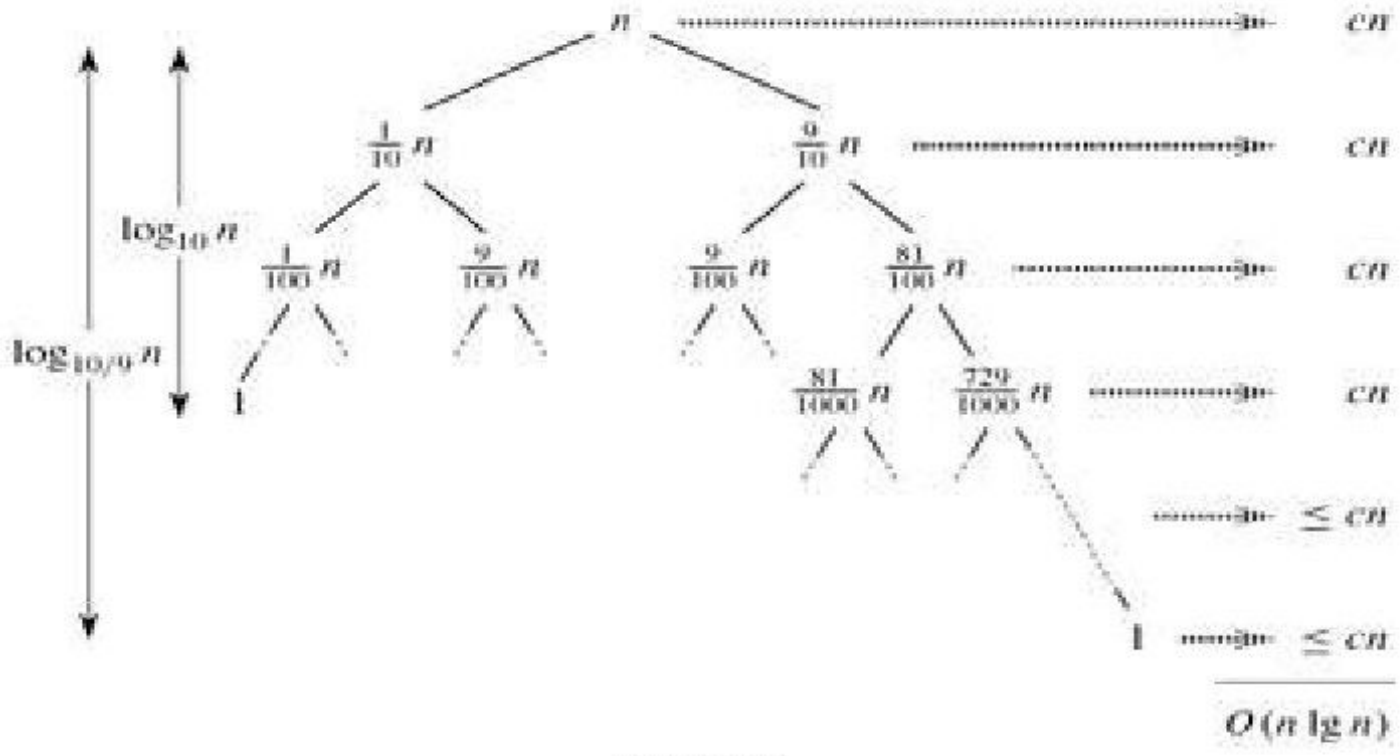
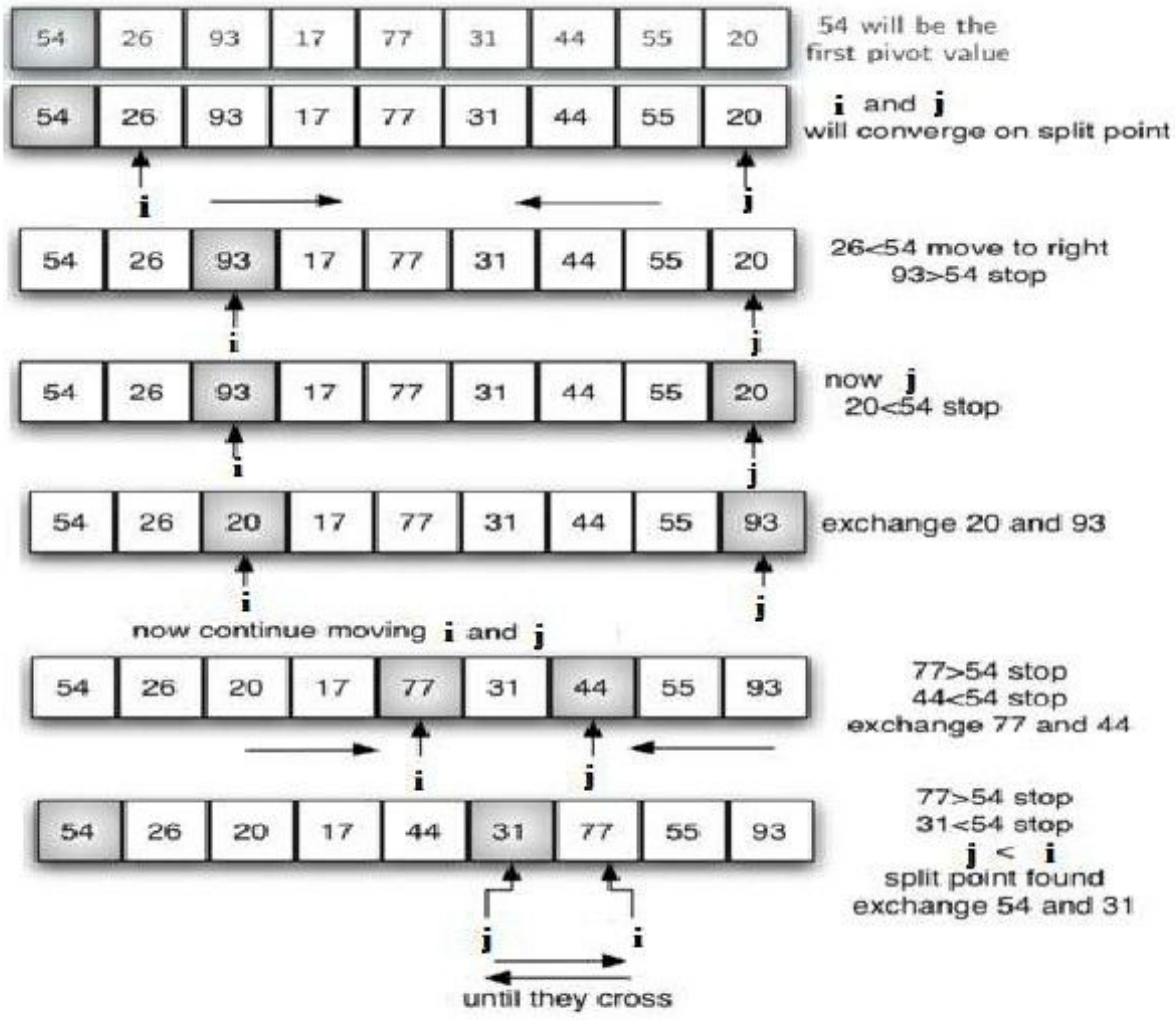
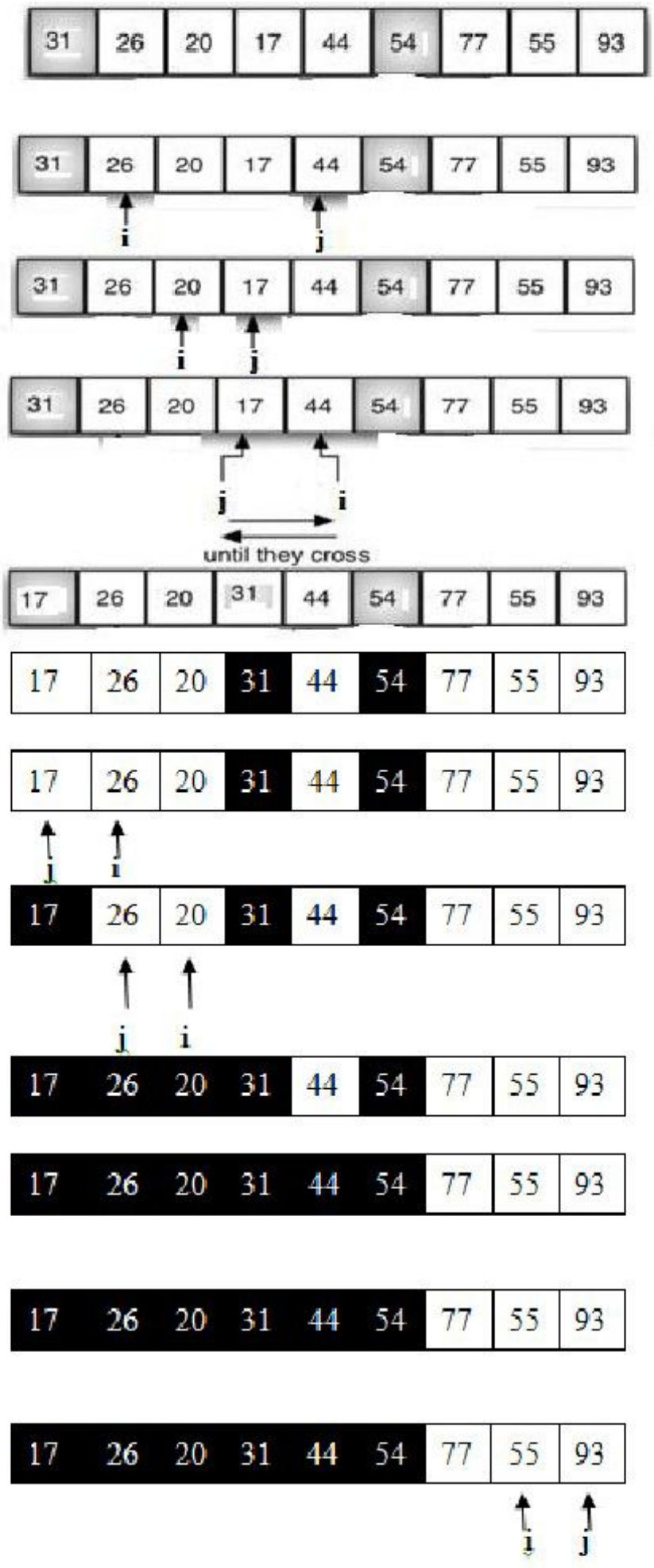


Figure 3.8

Example of Quick sort: Sort the following elements using quick sort.



Prepared By
Sasmita Kumari Nayak



54 is sorted and break into two list. One list is from 31 to 44 and another is 77 to 93

31 is sorted and break into two list. One list is from 17 to 20 and another is 44

17 is sorted and two list are 17 and another is 26 to 20

26 will be the pivot value

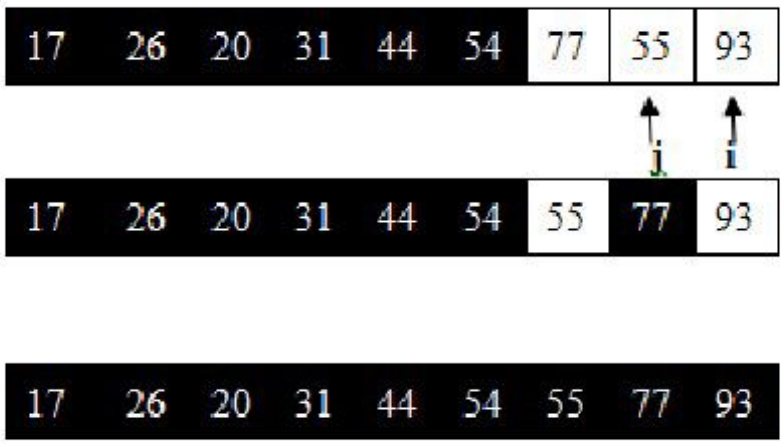
Exchange 26 and 20

44 containing one element in the list so already sorted

Consider the list 77 to 93 and 77 will be the pivot element

Consider the list 77 to 93 and 77 will be the pivot elemnt

Prepared By
Sasmita Kumari Nayak



Exchange 77 and 55

77 is sorted. Two list containing only one one element i.e. 55 and 93

Now all elements are sorted.

Prepared By
Sasmita Kumari Nayak