QUICKSORT

Quicksort or partition exchange sort works on divide and conquer method. Array $A[p..q]$ is partitioned into sub arrays, $A[p..q]$ and $A[q+1..r]$ s.t. each element in $A[p..q]$ is less than or equal to $A[q+1..r]$ index of q is found by partitioning method and each sub arrays are sorted through recursive calls. e.g.,

Let $x = A[0]$. Suppose that elements of A are partitioned so that x is a placed into position j and following condition hold.

(i) Each of the element in position 0 through $j-1$ is less than or equal to x.

(ii) Each of the elements in position $j+1$ through $n-1$ is greater than or equal to x.

The two pointers (down) *dn* and up are moved towards each other in following manner.

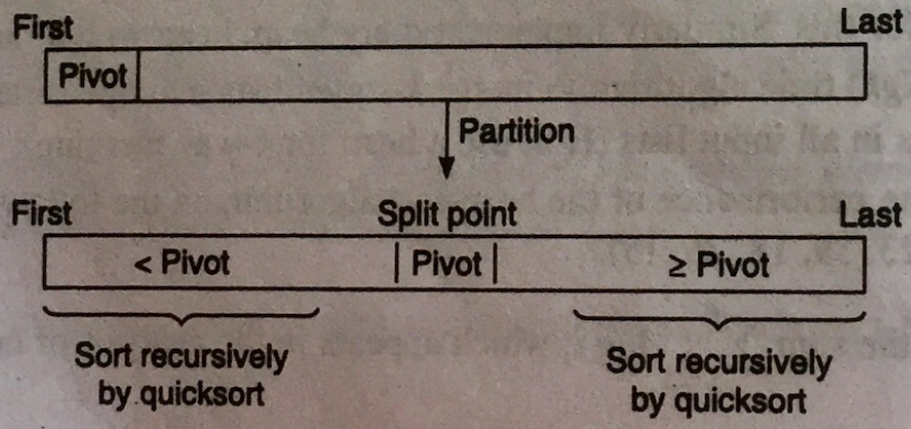
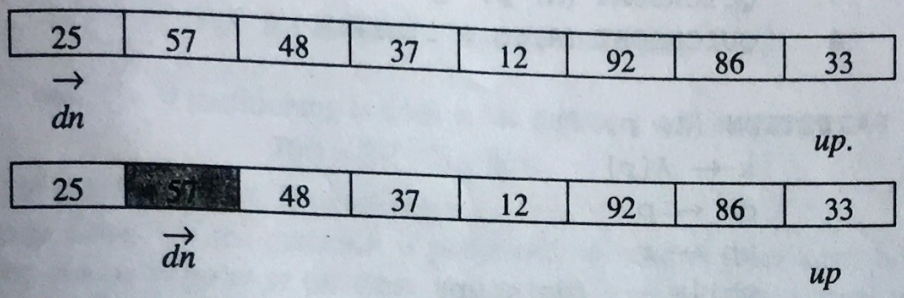


Fig. 8.1. Shows partitioning point.

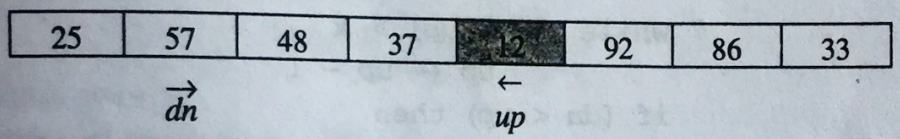
1. Repeatedly increase the pointer *dn* by one position until $A[dn] > x$ (till any element becomes greater than x)
2. Repeatedly decrease the pointer *up* by one position until $A[up] <= x$ (till any element becomes less than x)
3. If $up > dn$, interchange $A[dn]$ with $A[up]$, this, process is repeated till condition of step 3 fails (*i.e.*, $up <= dn$.) at which $A[up]$ is interchanged with x

Let $x = A[0] = 25$

Step 1

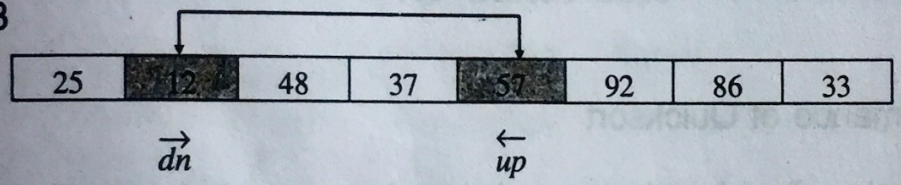


Step 2

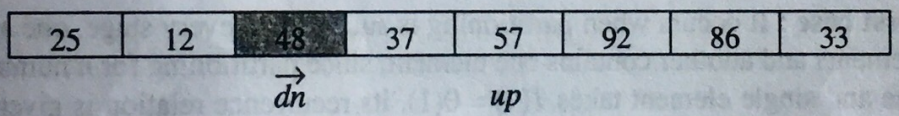


$up > dn$ interchange

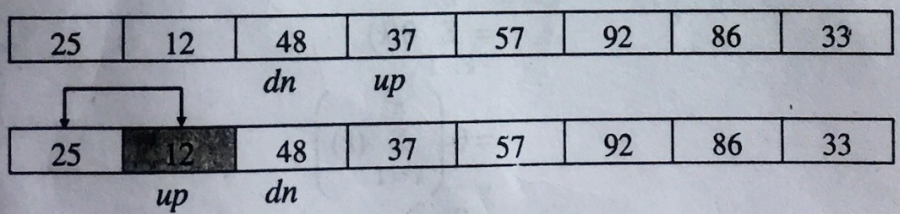
Step 3



Step 1



Step 2



$up < dn$, condition 3 fails, interchange $A[up]$ with x

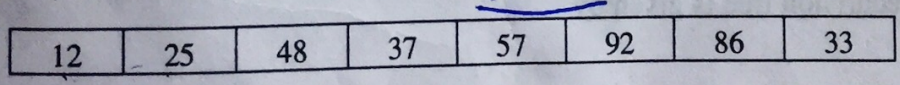


Fig. 8.2. Execution of quicksort on array [25, 57, 48, 37, 12, 92, 86, 33]

At this point 25 is in its proper position (pos 1) and every element to its left is-less than or equal to 25, and every element to its right is greater than or equal to 25. We could now proceed to sort the two subarrays (12) and (48 37 57 92 86 33) by applying the same method recursively,

The above procedure can be implemented using following procedure,

```
QUICKSORT (A, p, r)
  1.   if p < r
  2.   then q ← PARTITION (A, p, r)
  3.   QUICKSORT (A, p, q)
  4.   QUICKSORT (A, q + 1, r)

PARTITION (A, p, r)
        x ← A[p]
        dn ← p
        up ← r
        while     (dn < up)
             while     (A[dn] < = x)
                       dn ← dn + 1
             while     A[up] > x
                       up ← up - 1
             if (dn < up) then
                  exchange A[dn] ←→ A[up]
             else return up.
```