QUICKSORT

This was invented by Turing laureate C.A.R. Hoare in 1961. This method, though has a worst-case complexity of $O(n^2)$, is popular because it perhaps has the best average case performance among all sorting methods.

The method is a classic example of divide-and-conquer paradigm and can be implemented in a natural way as a recursive program.

The method crucially depends on a procedure for partitioning an array of elements, say, $a[i], \ldots, a[j]$ with respect to a "pivot" element such that the output array $a[i] .. a[j]$ satisfies the following property:

$$a[i], \ldots, a[k-1] < \text{pivot}$$
$$a[k], \ldots, a[j] \geq \text{pivot}$$

for some k such that $i \leq k \leq j+1$

Let us call the above procedure as

partition (i, j, pivot)

QUICK 2

An Example for partition (i, j, pivot):
choose the following array.

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

Suppose pivot = 3. We choose two pointers left and right. Initially

left = i = 0 (in this case)
right = j = 9 (in this case)

We keep incrementing left until A[left] ≥ pivot and we keep decrementing right until A[right] < pivot. Then we swap A[left] and A[right].

In the above case, A[0] = 3 and A[9] = 3.
We don't increment left because A[left] ≥ pivot.
We keep decrementing right until right becomes 6 because

A[9] ≥ pivot; A[8] ≥ pivot;
A[7] ≥ pivot; A[6] < pivot

Now we swap A[0] and A[6] to obtain

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

We continue to increment left and decrement right and do appropriate swaps until the two pointers cross each other. We then get a partition

2	1	1		4	5	9	3	6	5	3

It can be shown that the worst case complexity of the partition algorithm on an array with n elements is $O(n)$.

choosing a pivot is an important step and there are many ways which have been explored. Let us go for the following choice:

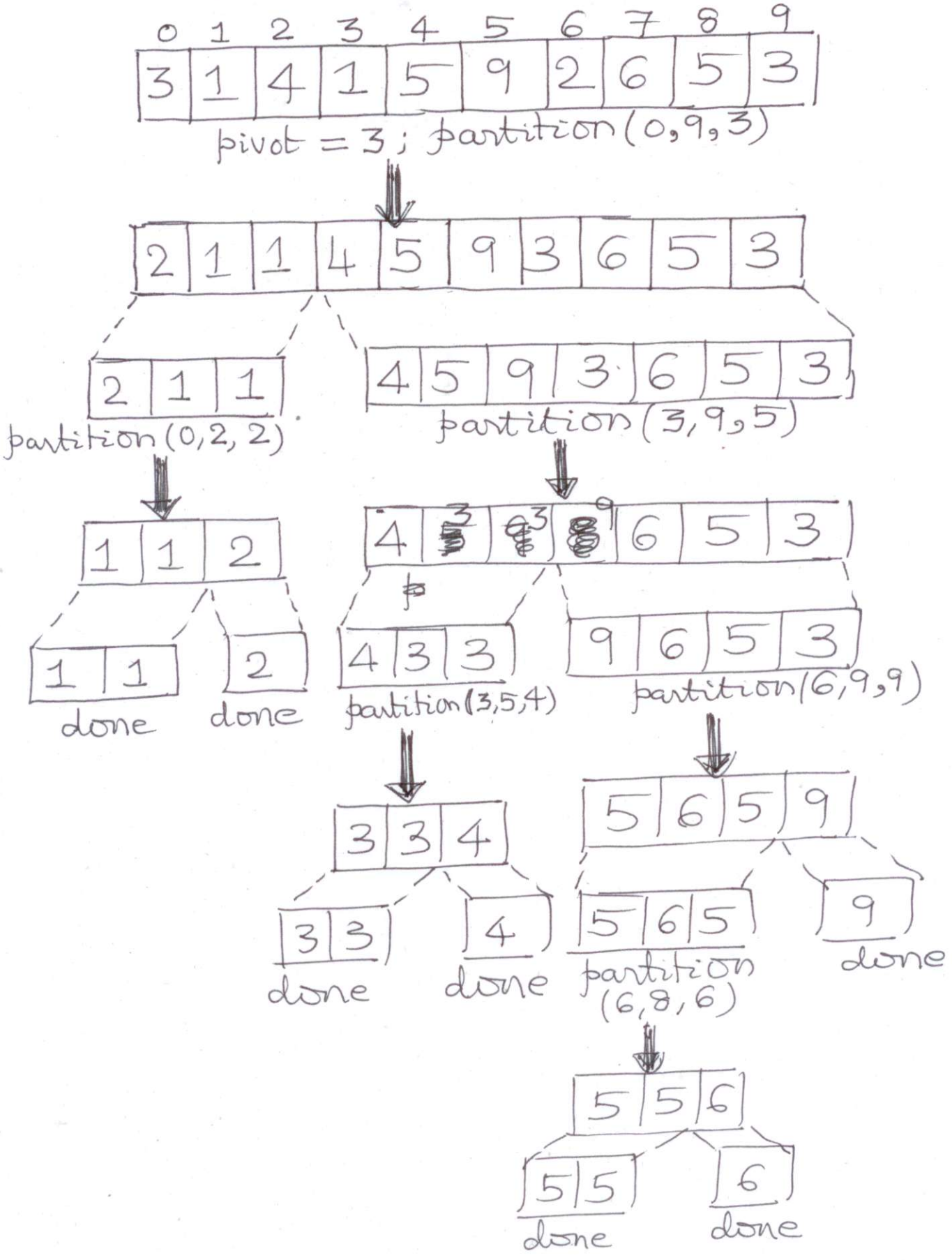
pivot = larger of the first two distinct elements in the array a[i] ... a[j].

With this choice of pivot, the quicksort(i, j) algorithm for sorting the elements A[i], A[i+1], ..., A[j] would be:

```
{ if (A[i], A[i+1], ..., A[j] contain
      at least two distinct elements)
  { let pivot = larger of the first two
                distinct elements;
    partition (i, j, pivot);
    let k be the position at which
          the partition occurs;
    quicksort (i, k - 1);
    quicksort (k, j);
  }}
```

Notice the natural recursive structure of the quicksort algorithm.

An Example of Quicksort



Notice that there are 4 levels in the above execution tree.

Some observations

- Quicksort with the above choice of the pivot exhibits its worst-case performance when the input sequence is already sorted. The execution tree of the algorithm will then have $(n-1)$ levels leading to a worst case complexity of

$$O(n^2)$$

- The best case is of course obtained when partitioning at each level results in exactly equal parts. The choice of the pivot is therefore very important.

- Quicksort has perhaps the best average case complexity among sorting algorithms. It can be shown that the ACC is

$$O(n \log n)$$

and the constant of proportionality involved is ~~among~~ the smallest among known sorting algorithms.