

QUICKSORT

QUICKSORT, AN APPLICATION OF STACKS

Let A be a list of n data items. “Sorting A” refers to the operation of rearranging the elements of A so that they are in some logical order, such as numerically ordered when A contains numerical data, or alphabetically ordered when A contains character data. The subject of sorting, including various sorting algorithms, is treated mainly in Chap. 9. This section gives only one sorting algorithm, called *quicksort*, in order to illustrate an application of stacks.

Quicksort is an algorithm of the divide-and-conquer type. That is, the problem of sorting a set is reduced to the problem of sorting two smaller sets. We illustrate this “reduction step” by means of a specific example.

Suppose A is the following list of 12 numbers:

(44), 33, 11, 55, 77, 90, 40, 60, 99, 22, 88, (66)

The reduction step of the quicksort algorithm finds the final position of one of the numbers; in this illustration, we use the first number, 44. This is accomplished as follows. Beginning with the last number, 66, scan the list from right to left, comparing each number with 44 and stopping at the first number less than 44. The number is 22. Interchange 44 and 22 to obtain the list

(22), 33, 11, 55, 77, 90, 40, 60, 99, (44), 88, 66

(Observe that the numbers 88 and 66 to the right of 44 are each greater than 44.) Beginning with 22, next scan the list in the opposite direction, from left to right, comparing each number with 44 and stopping at the first number greater than 44. The number is 55. Interchange 44 and 55 to obtain the list

22, 33, 11, (44), 77, 90, 40, 60, 99, (55), 88, 66

(Observe that the numbers 22, 33 and 11 to the left of 44 are each less than 44.) Beginning this time with 55, now scan the list in the original direction, from right to left, until meeting the first number less than 44. It is 40. Interchange 44 and 40 to obtain the list

22, 33, 11, (40), 77, 90, (44), 60, 99, 55, 88, 66

(Again, the numbers to the right of 44 are each greater than 44.) Beginning with 40, scan the list from left to right. The first number greater than 44 is 77. Interchange 44 and 77 to obtain the list

22, 33, 11, 40, (44), 90, (77), 60, 99, 55, 88, 66

(Again, the numbers to the left of 44 are each less than 44.) Beginning with 77, scan the list from right to left seeking a number less than 44. We do not meet such a number before meeting 44. This means all numbers have been scanned and compared with 44. Furthermore, all numbers less than 44 now form the sublist of numbers to the left of 44, and all numbers greater than 44 now form the sublist of numbers to the right of 44, as shown below:

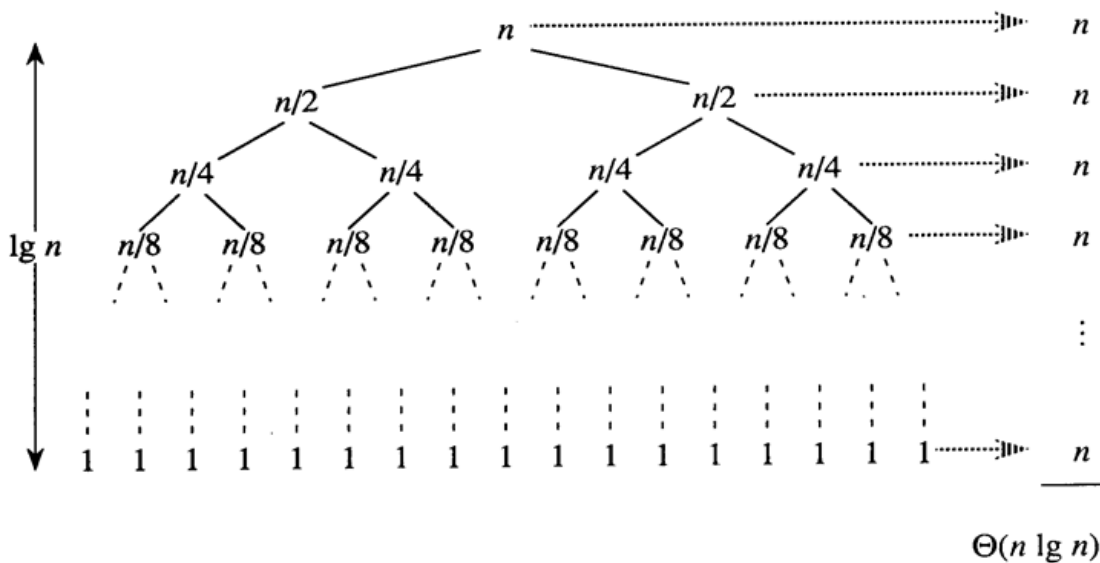
$\underbrace{22, \quad 33, \quad 11, \quad 40}_{\text{First sublist}}, \quad (44), \quad \underbrace{90, \quad 77, \quad 60, \quad 99, \quad 55, \quad 88, \quad 66}_{\text{Second sublist}}$

Thus 44 is correctly placed in its final position, and the task of sorting the original list A has now been reduced to the task of sorting each of the above sublists.

Best Case Partitioning

$$T(n) = T(n/2) + T(n/2) + n$$

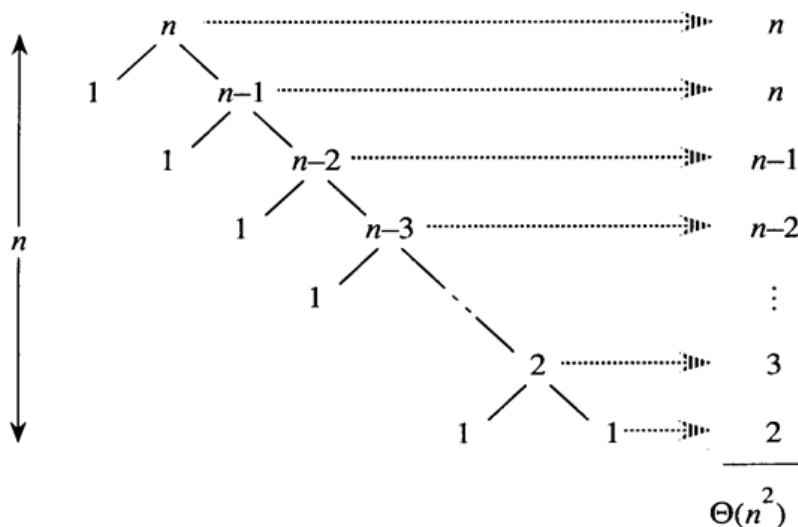
Why n ? Because the whole was scanned in a pass



A recursion tree for QUICKSORT in which PARTITION always balances the two sides of the partition equally (the best case). The resulting running time is $\Theta(n \lg n)$.

Worst Case Partitioning

$$T(n) = T(n-1) + n$$



A recursion tree for QUICKSORT in which the PARTITION procedure always puts only a single element on one side of the partition (the worst case). The resulting running time is $\Theta(n^2)$.

Average Case Partitioning

10% Case: $T(n) = T(n/10) + T(9n/10) + n$

