A Note on Slowing Quicksort

R.Chaudhuri and A.C.Dempster
Department of Computer Science
Eastern Michigan University
Ypsilanti, Mich. 48197.

ABSTRACT:

We present a simple linear time algorithm for generating a worst case sequence for Quicksort when the pivot element is chosen as the middle element of the (sub)array in each pass.

1. INTRODUCTION:

Quicksort, introduced by C.A.R. Hoare in 1962, is a recursive method for sorting an array a[1 .. n] by partitioning it into two subarrays such that

1) some key x (called the 'pivot') is in its final position a[j] in the array if x happens to be the j-th smallest element
2) the elements a[1], a[2], ... , a[j-1] of the left subarray are less than or equal to x
3) the elements a[j+1], a[j+2], ... , a[n] of the right subarray are greater than or equal to x
4) the left and the right subarrays are then sorted recursively by Quicksort.

The performance of Quicksort depends on the choice of the pivot element in successive passes. If we always happen to select the median of the elements as the pivot, then the partitioning process splits the array into two halves resulting in the O(n log n) best performance of the sort. On the other hand, if in each pass we pick the largest or the smallest value as the pivot then the worst case performance of $O(n^2)$ happens. However, the efficiency of Quicksort is due to the fact that its average performance is also O(n log n).

In most texts, the pivot is selected as the first or the last element of each subarray. In such a case, a worst case sequence for Quicksort is the sorted array itself. We suspect that the ease of identifying and constructing this particular sequence is a main reason why most texts use such a pivot choice. In [1], however, Wirth chooses the pivot as the middle element of the subarray in each pass and asserts that the average performance improves slightly (by a constant factor) as a result of such a choice. In the following, we give a Pascal version of the Quicksort procedure where the pivot is always the middle element of the subarray to be sorted.

```
PROCEDURE  Quicksort ( VAR a : array ;left, right : index);
VAR   i, j : index;  x : item;
BEGIN
   i := left ;  j := right ; x :=  a [(left + right) DIV 2] ;
   REPEAT
      WHILE   a[i] < x   DO  i := i + 1;
      WHILE   x < a[j]   DO  j := j  - 1;
      IF  i  <= j  THEN
         BEGIN
            Swap (a[i] , a[j]) ; i := i + 1 ; j := j - 1
         END
   UNTIL  i > j ;
   IF  left < j THEN  Quicksort (a, left , j);
   IF  i  < right  THEN Quicksort (a, i , right)
END;
```

The worst case behavior of the above algorithm is exhibited when the pivot turns out to be the smallest or the largest element in each pass. But what is a worst case sequence for the above algorithm ? Does there exist an algorithm to generate a worst case sequence for every n ? We needed one to use in an in-class sort timing demonstration program, and we give it in the next section.

2. THE ALGORITHM:

In this section, we give a simple algorithm to generate a worst case sequence for Quicksort for any positive integer n (>1) assuming that pivot element in each pass is the middle element of the subarray to be sorted. We use an array a[1..n] to store the sequence.

Step 1. (Initialize)

```
FOR i := 1 TO n DO
    a[i] := i ;
```

Step 2. (Generate worst case sequence for each i until n is reached)

```
FOR i := 2 TO n DO
    Swap ( a[(1 + i) DIV 2] , a[i] ) ;
```

The above algorithm generates a worst case sequence that yields the pivotal sequence n,n-1, n-2, ... , 2, 1 if we apply the above version of Quicksort to it. A simple inductive argument shows that the algorithm is indeed correct. Clearly, for n=2 the sequence [2, 1] generated by the algorithm is a worst case sequence. Assuming (a[1], a[2], ... , a[k]) to be the worst case sequence of length k generated after k-1 iterations of the loop in step 2, note that the worst case sequence of length k+1 is obtained by appending the element k+1 to the right of the worst case sequence (a[1], a[2], ... , a[k]) of length k and swapping it with the middle element a[(1+k+1) div 2] of the subarray a[1 .. k+1] since applying Quicksort to the subarray a[1.. k+1] now yields back the sequence (a[1], a[2], ... , a[k], k+1) and the subarray to be sorted next is the worst case sequence of length k by our assumption. Replacing the loop in step 2 by the loop

```
FOR i := n-1 DOWNTO 1 DO
    Swap ( a[i] , a[(i+n) DIV 2] )
```

produces a worst case sequence for Quicksort whose pivotal sequence is 1, 2, ... , n. Clearly, our algorithm runs in linear time.

REFERENCE:

1) N.Wirth : " Algorithms + Data Structures = Programs", Prentice Hall, Englewood Cliffs, N.J , 1976.