Quicksort Project

The project is to explore Quicksort and some variants. Instead of the actual time you will explore the number of comparisons ($COMP$ below) that your program makes.

Start with $A[1], \ldots, A[n]$ in random order. To do this start with $A[i] = i$. If you have a shuffle routine in your toolbox feel free to use it. Otherwise, with $RAN[j]$ denoting a random integer from 1 to j: you can use the following:

```
FOR i = n DOWN TO 2
    k = RAND[i]
    IF k ≠ i *else leave where is*
        A[i] ↔ A[k]
    ENDIF
ENDFOR
```

You will *start* with this random order. With this start you do the regular QUICKSORT, simply taking the last of an array as the pivot.

Write QUICKSORT (no package, please!). Run it several times for various n. Take $n = 100$ and then keep doubling it until, say, you reach a million. (If its taking too long for large n you can stop sooner.) Let $COMP$ denote the number of times you compare some $A[i]$ with some $A[j]$. (e.g.: with the pivot on an array of size w you will make $w - 1$ comparisons.) In your program start $COMP = 0$ and increment $COMP$ every time you make a comparison. The output for a run will be the final value of $COMP$.

`The variant:` Call this $QUICKSORT{-}3$. When recursive $QUICKSORT[p, r]$ has $r - p > 2$: Take the last three values $A[r], A[r-1], A[r-2]$ of the array. Find their median. (You have to write this – be sure that the comparisons you make are counted in $COMP$.) Use this median as the pivot. (Small savings: You've already compared this pivot with the other two so you needn't do that again.) Then use recursion. (When $r - p \leq 2$ just use standard $QUICKSORT$.

`What to examine:` Compare the results for $QUICKSORT$ and $QUICKSORT-3$. $QUICKSORT - 3$ has the advantage that the pivot is more likely to be near the middle. It has the disadvantage that finding the pivot is taking comparisons. Looking at the data discuss which one seems better for which n and what their asymptotic behavior looks like. (It is probably handy to look at $COMP$ divided by $n \ln n$ when doing this.)

`Further variants:` $QUICKSORT{-}5$,$QUICKSORT{-}7$,$QUICKSORT{-}9$,$QUICKSORT - 11$. With $QUICKSORT - (2t + 1)$ you take the last $2t + 1$ elements of the array and find their median and use it as a pivot.

When the number of elements of the array falls to $\leq 2t+1$ then revert to standard $QUICKSORT$.

`What further to examine:` Look at the data for these other variants. When doing, say $QUICKSORT - 11$ your pivot is very likely very close to halfway in the array which is good. But you are spending a lot of comparisons finding the median of the 11 which is bad. Does it wind up better or worse, or does it vary depending on which n you look at? Can you make an educated guess at the asymptotics of $COMP$ for these variants.

Most of all, have fun – explore – take to heart the words of the founder of Theoretical Computer Science, Don Knuth:

> ...pleasure has probably been the main goal all along. But I hesitate to admit it, because computer scientists want to mantain their image as hard-working individuals who deserve high salaries. Sooner or later society will realise that certain kinds of hard work are in fact admirable even though they are more fun than just about anything else.