1. **Purpose**

Implement the simplest version of the quicksort algorithm by choosing as pivot:

*1.* The last element.

*2.* The median (randomized select algorithm).

*3.* The median (deterministic select algorithm, median of medians).

Count the time and the number of comparisons for this version when sorting an array of random generated numbers and display the totals. Test it with large random generated arrays (10,000 - 100 million).

1. **Algorithm**
2. Choosing the last element as pivot for quicksort

Here is the pseudocode

Quicksort(A, p, r) {

if (p < r) {

q = partition(A, p, r)

Quicksort(A, p, q - 1)

Quicksort(A, q + 1, r)

}

}

partition(A, p, r) {

x = A[r]

i = p - 1

for j = p to r - 1

if A[j] <= 0

i++

swap(A, i, j)

swap(A, i + 1, r)

return i + 1

}

1. Choosing the median as pivot with randomized select algorithm.

RANDOMIZED-SELECT(A, p, r, i)

if p == r

return A[p]

q = RANDOMIZED-PARTITION(A, p, r )

k =q-p+1

if i == k // the pivot value is the answer

return A[q]

else if i < k

return RANDOMIZED-SELECT(A, p, q-1, i)

else return RANDOMIZED-SELECT(A, q+1, r, i- k)

1. Choosing the median as pivot with deterministic select algorithm (median of medians).

If (n > 1)

Divide n elements into groups of 5

Find median of each group

Use Select() recursively to find median x of the cell[𝑛/5] medians

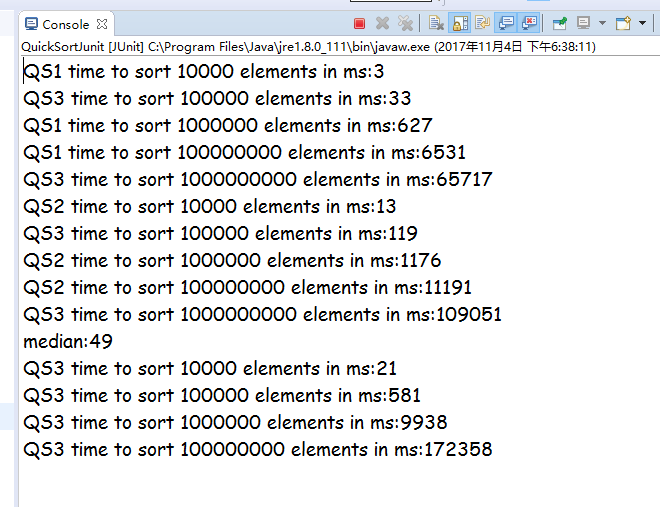
Use Select() recursively to find median x of the 𝑛/5 medians

if (i == k) then return x

if (i < k) then use Select() recursively to find i-th smallest element in first partition

else (i > k) use Select() recursively to find (i-k)-th smallest element in last partition

1. **Conclusion**



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Array Size | 10,000 | 100,000 | 1,000,000 | 10,000,000 | 100,000,000 |
| QS1 Running Time | 3 | 33 | 627 | 6531 | 65717 |
| QS2 Running Time | 13 | 119 | 1176 | 11191 | 109051 |
| Comparisons QS1 | 8742 | 110547 | 204587 | 305472 | 405924 |
| Comparisons QS2 | 297 | 3111 | 30141 | 325456 | 3080554 |

Choosing the median as pivot will improve the algorithm a lot. However, we do need to care the memory in case of stack overflow problem since it require a lot of recursion.