//*REPORT-CONCURRENT QUICKSORT*//

- Implemented a variation of Concurrent version Quicksort algorithm which uses threads and processes and compared their time with normal implementation of Quicksort.

Implementation :

The algorithm first picks an element called a pivot (takes the median of three random numbers to avoid worst case complexity of O(n^2)). Then partition the array around this pivot. The function is then recursively applied to the left and right subarrays in child of parent process (in case of concurrent quicksort) and in different threads in case of threaded quicksort.

Tests :

Input files :

Tests were run with 3 input files of 10^3, 10^4, and 10^5 inputs respectively.
They were generated with the commands `echo <number of test cases> > filename`
and `seq 1 <number of test cases> | sort -r >> filename`

Results :

All times reported are an average of three tests each, in seconds

Program	Time on 10^3	Time on 10^4	Time on 10^5	Time on 10^6
Normal	0.0001	0.0009	0.02233	9.25
Process	0.0261	0.3012	3.6643	0.17
Threaded	0.0142	0.1263	(Segfault)	(Segfault)

General Observation:

The normal merge sort ran the fastest, while the concurrent merge sort based on processes ran the slowest.

The process based program ran the slowest because of,

1. ***Constant context switches***
2. ***Generating new processes is a time consuming task***