Etan Soclof

SecondSmallestElement

1. Number of Cores

Hardware Overview:

Model Name:	MacBook Pro
Model Identifier:	MacBookPro12,1
Processor Name:	Dual-Core Intel Core i5
Processor Speed:	2.7 GHz
Number of Processors:	1
Total Number of Cores:	2 ←
L2 Cache (per Core):	256 KB
L3 Cache:	3 MB
Hyper-Threading Technology:	Enabled
Memory:	8 GB
Boot ROM Version:	191.0.0.0.0
SMC Version (system):	2.28f7
Serial Number (system):	C02S3SDFFVH5
Hardware UUID:	66B073DC-3B59-5F4B-BC90-0F3A88BA5232

My computer has 4 **logical processors** as determined this number by running the following code:

```
int processors = Runtime.getRuntime().availableProcessors();
System.out.println("CPU cores: " + processors);
```

However, I do not expect the logical processors to increase performance significantly as per the following:

> "Hyperthreading makes a Physical Processor to behave like it has two Physical Processors, which are called Logical Processor. why?
> While hyperthreading does not double the performance of a system, it can increase performance by better utilizing idle resources leading to greater throughput for certain important workload types. An application running on one logical processor of a busy core can expect slightly more than half of the throughput that it obtains while running alone on a non-hyperthreaded processor." (https://unix.stackexchange.com/questions/88283/so-what-are-logical-cpu-cores-as-opposed-to-physical-cpu-cores#:~:text=Physical%20cores%20are%20number%20of,I%20have%208%20logical%20processors)

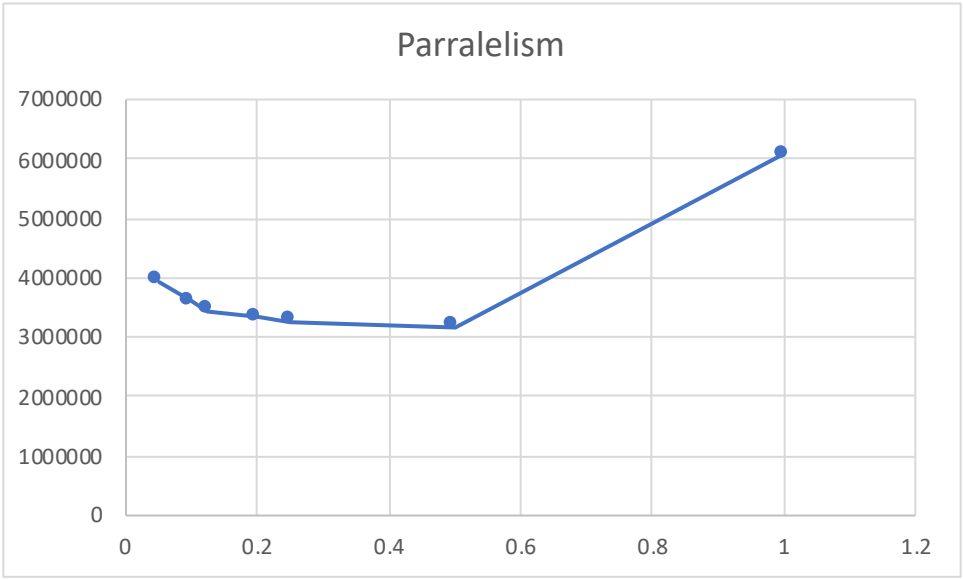
Therefore, I expect my performance to improve by a factor of 2 when using fork join, proportional to the amount of physical cores on my computer.

2. O(n) Runtime

	Average Runtime (Nano-seconds)	Doubling Method
n = 1000000	755813.898	

	Average Runtime (Nano-seconds)	Doubling Method
n = 2000000	1510103.41	1.99798312
n = 4000000	2815002.43	1.86411236
n = 10000000	6474518.58	2.15416429

3. Running in Parallel



Fraction to Apply to Sequential Cutoff	Average Runtime (Nano-seconds)
0.5	3171595.67
0.25	3252586.67
0.2	3323456.54
0.125	3441498.05
0.1	3604108.09
0.05	3943234.34
0	922859271

4. The Fork Join implementation achieves the best runtime when the Fraction to Apply to Sequential Cutoff is set to 0.5. This is because in this situation, the array is split into two and each half of the array is processed by a different and independent physical core, achieving nearly twice the efficiency of the fully sequential implementation.
Because there are only 2 cores on the computer, increasing the amount of threads beyond two will not result in an increase in efficiency - there are no more cores for the threads to run on. For this reason, when the Fraction to Apply to Sequential Cutoff was set to less than .5, the runtime was slightly worse. The gradual decrease in efficiency as the Fraction to Apply to Sequential Cutoff approaches zero can likely be understood as a consequence of the overhead of creating new threads. As more threads must be created, more time must be spent creating the threads.