Christopher Simon
CSC 139
Assignment 2
10/22/2017

System Specs

```
Architecture:            x86_64
CPU op-mode(s):          32-bit, 64-bit
Byte Order:              Little Endian
CPU(s):                  8
On-line CPU(s) list:     0-7
Thread(s) per core:      2
Core(s) per socket:      4
Socket(s):               1
NUMA node(s):            1
Vendor ID:               GenuineIntel
CPU family:              6
Model:                   60
Model name:              Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz
Stepping:                3
CPU MHz:                 800.000
BogoMIPS:                7183.48
Virtualization:          VT-x
L1d cache:               32K
L1i cache:               32K
L2 cache:                256K
L3 cache:                8192K
NUMA node0 CPU(s):       0-7
```

Dedicated Machine (2 runs each)

```
1.  Array size = 100M, T=2, index for zero = 50M+1
Run 1:
Sequential search completed in 26 ms. Min = 0
Threaded FindMin with parent waiting for all children completed in 30 ms. Min = 0
Threaded FindMin with parent continually checking on children completed in 0 ms. Min = 0
Threaded FindMin with parent waiting on a semaphore completed in 0 ms. Min = 0
Run 2:
Sequential search completed in 25 ms. Min = 0
Threaded FindMin with parent waiting for all children completed in 31 ms. Min = 0
Threaded FindMin with parent continually checking on children completed in 0 ms. Min = 0
Threaded FindMin with parent waiting on a semaphore completed in 0 ms. Min = 0

2.  Array size = 100M, T=4, index for zero = 75M+1
Run 1:
Sequential search completed in 39 ms. Min = 0
Threaded FindMin with parent waiting for all children completed in 17 ms. Min = 0
Threaded FindMin with parent continually checking on children completed in 0 ms. Min = 0
Threaded FindMin with parent waiting on a semaphore completed in 0 ms. Min = 0
Run 2:
Sequential search completed in 40 ms. Min = 0
Threaded FindMin with parent waiting for all children completed in 17 ms. Min = 0
Threaded FindMin with parent continually checking on children completed in 0 ms. Min = 0
Threaded FindMin with parent waiting on a semaphore completed in 0 ms. Min = 0

3.  Array size = 100M, T=8, index for zero = 88M
Run 1:
Sequential search completed in 45 ms. Min = 0
Threaded FindMin with parent waiting for all children completed in 15 ms. Min = 0
Threaded FindMin with parent continually checking on children completed in 15 ms. Min = 0
Threaded FindMin with parent waiting on a semaphore completed in 3 ms. Min = 0
Run 2:
Sequential search completed in 45 ms. Min = 0
Threaded FindMin with parent waiting for all children completed in 15 ms. Min = 0
```

Christopher Simon
CSC 139
Assignment 2
10/22/2017

```
Threaded FindMin with parent continually checking on children completed in 15 ms. Min = 0
Threaded FindMin with parent waiting on a semaphore completed in 1 ms. Min = 0

4.  Array size = 100M, T=2, index for zero = -1 (no zero)
Run 1:
Sequential search completed in 51 ms. Min = 1
Threaded FindMin with parent waiting for all children completed in 31 ms. Min = 1
Threaded FindMin with parent continually checking on children completed in 31 ms. Min = 1
Threaded FindMin with parent waiting on a semaphore completed in 30 ms. Min = 1
Run 2:
Sequential search completed in 52 ms. Min = 1
Threaded FindMin with parent waiting for all children completed in 30 ms. Min = 1
Threaded FindMin with parent continually checking on children completed in 30 ms. Min = 1
Threaded FindMin with parent waiting on a semaphore completed in 29 ms. Min = 1

5.  Array size = 100M, T=4, index for zero = -1 (no zero)
Run 1:
Sequential search completed in 52 ms. Min = 1
Threaded FindMin with parent waiting for all children completed in 17 ms. Min = 1
Threaded FindMin with parent continually checking on children completed in 23 ms. Min = 1
Threaded FindMin with parent waiting on a semaphore completed in 17 ms. Min = 1
Run 2:
Sequential search completed in 51 ms. Min = 1
Threaded FindMin with parent waiting for all children completed in 17 ms. Min = 1
Threaded FindMin with parent continually checking on children completed in 22 ms. Min = 1
Threaded FindMin with parent waiting on a semaphore completed in 17 ms. Min = 1

6. Array size = 100M, T=8, index for zero = -1 (no zero)
Run 1:
Sequential search completed in 52 ms. Min = 1
Threaded FindMin with parent waiting for all children completed in 15 ms. Min = 1
Threaded FindMin with parent continually checking on children completed in 24 ms. Min = 1
Threaded FindMin with parent waiting on a semaphore completed in 14 ms. Min = 1
Run 2:
Sequential search completed in 52 ms. Min = 1
Threaded FindMin with parent waiting for all children completed in 15 ms. Min = 1
Threaded FindMin with parent continually checking on children completed in 23 ms. Min = 1
Threaded FindMin with parent waiting on a semaphore completed in 15 ms. Min = 1
```

Sequential search is definitely the slowest in all cases because it is a single threaded linear search. Upon setting the index to 0, the next slowest should be when the Parent that waits for all it's children, since it goes into a busy loop until it is done. If the zero is set to -1, Semaphores lose some speed because it has to lock, then wait for the process to complete for every thread that searches, only to find no 0. The third longest search method when there is no zero, is when the parent continually checks on the children to finish. The quickest method is always semaphores, even if by a little bit when there is no zero.

In sequential searching, we have to iterate through each element until a zero is found. Then it ends. The second one which waits for children stops when all the threads finish, even if the zero has already been found. The third where the parent checks the children means that the parent slows down the processing, by going back and checking to see if 0 was found, though it is still faser than waiting for the children to return because it's proactively checking for 0 as opposed to being in a busy loop. The last search with semaphores has the parent placed in the waiting queue by using semaphore wait commands, so that it frees up resources that can be used on other threads once it finds 0 and is canceled. If there is no 0, the threads will have to wait since the critical section is locked via the semaphores, which prevents us the program from checking when the thread count has reached the desired value.