

Poojitha Konduparti (002103368)  
**PROGRAM STRUCTURES AND ALGORITHMS**  
**FALL 2021**  
**ASSIGNMENT NO. 4**

---

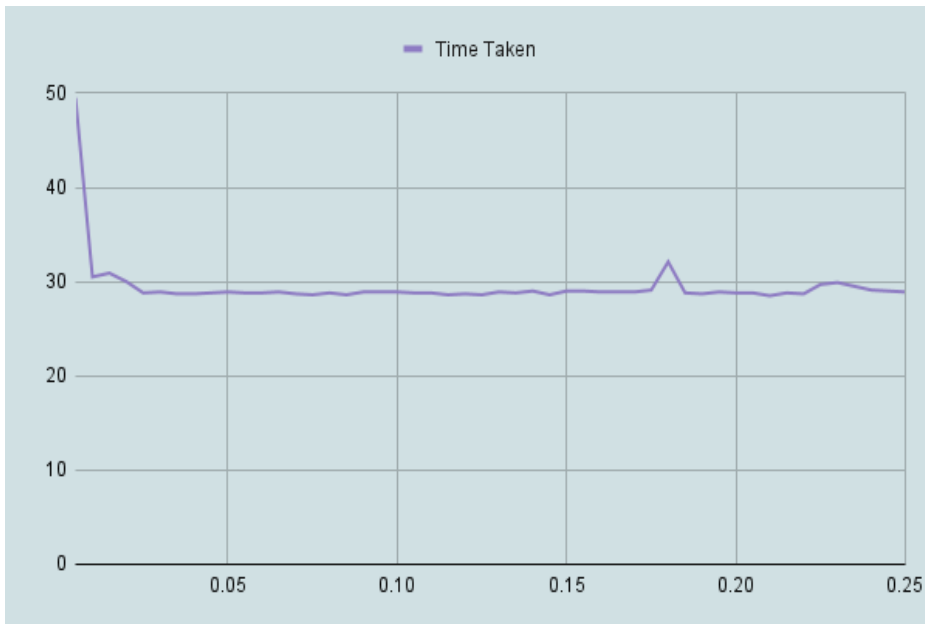
**Task**

- Implementing a parallel sort algorithm such that each partition of the array is sorted in parallel.
  - Update the value of the cutoff to find a good value for it. Use the system sort if the number of elements to sort are less than the cutoff.
  - Decide on an optimal number of threads in the powers of 2.
  - Use a combination of both of these
- 

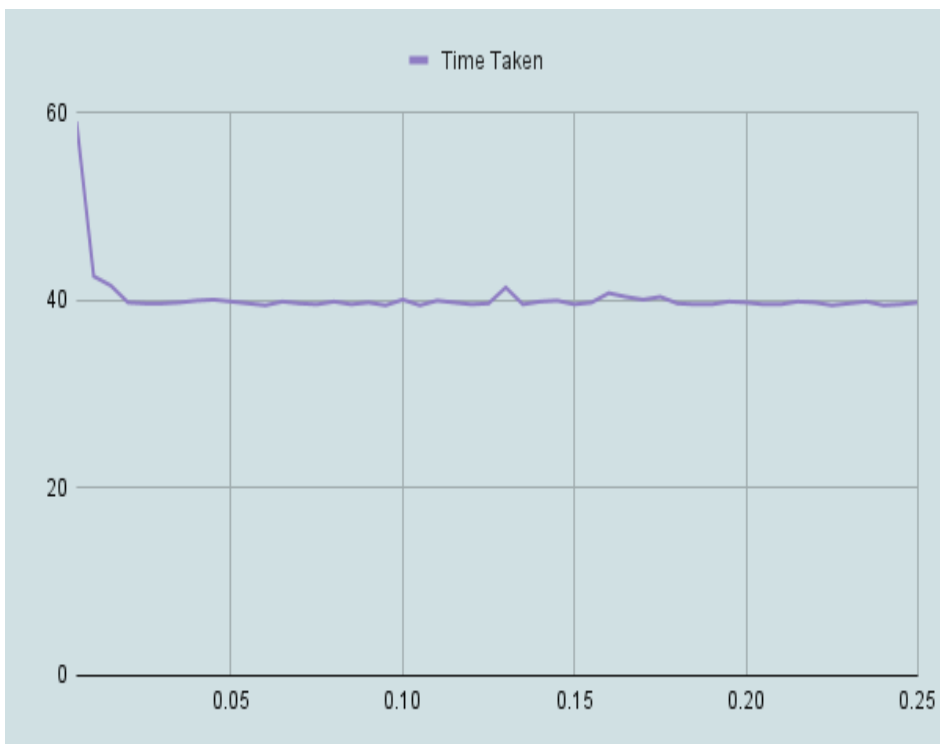
**Relationship Conclusion**

1. For varying cutoffs with a constant number of threads, the time taken to execute is high in the beginning when the cutoff is low and gradually keeps on decreasing with a small spike in the middle and reaches more or less a constant value with very minor difference in time. This holds true for multiple sizes of the array(N). For my system, a good cutoff value is 640000.
  2. When the cutoff is fixed and the number of threads are increasing in powers of 2, we can observe that the time taken to execute when the thread count is 2 is high, but as the recursion depth increases, the time taken is reduced until a certain point(512 threads) after which there is a sharp increase in the time taken.
  3. As a combination of both, I have fixed the number of threads for each run and incremented the cutoff, and it can be said that when the cutoff is lowest, the time taken is the highest and gradually decreases as the cutoff value increases.
-

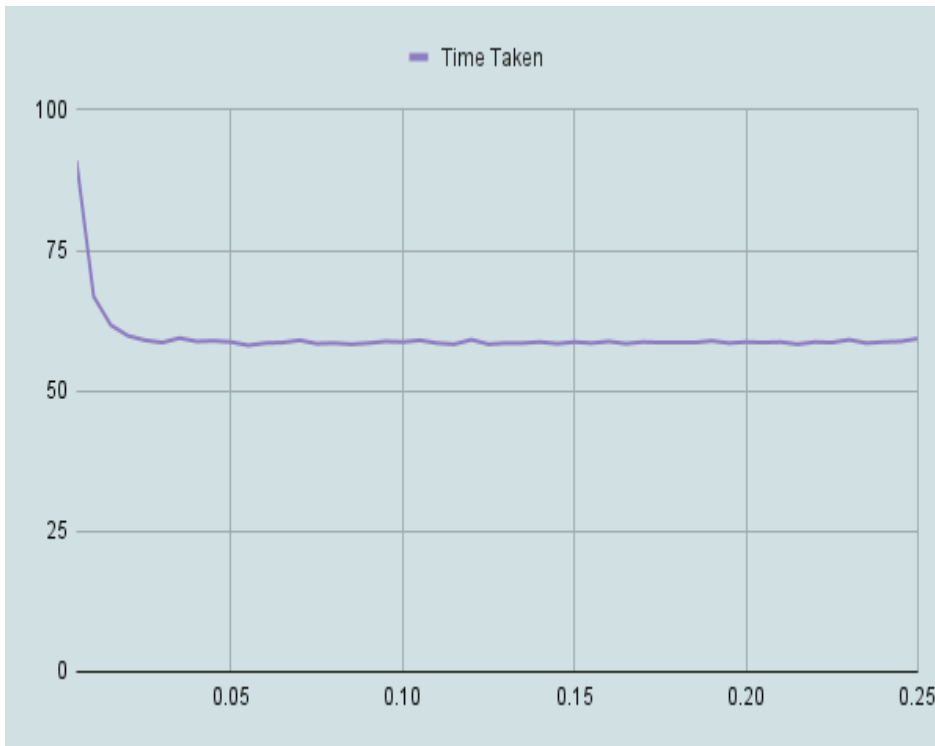
## Graphs and Evidence



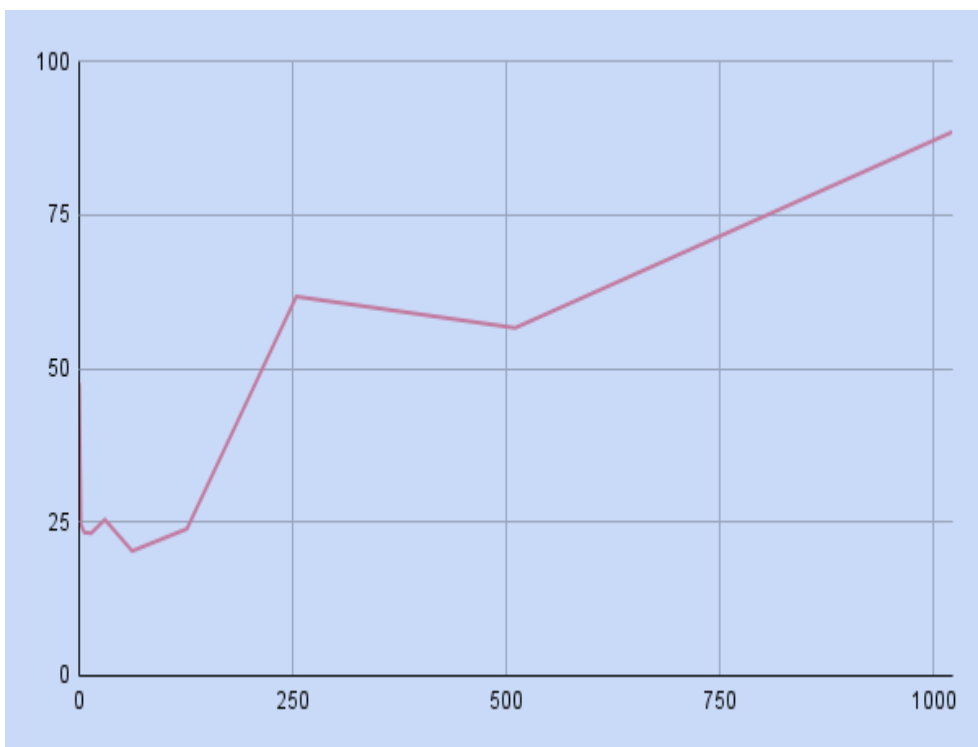
*Graph for increasing cutoffs and the Time Taken for Array Size of 500,000*



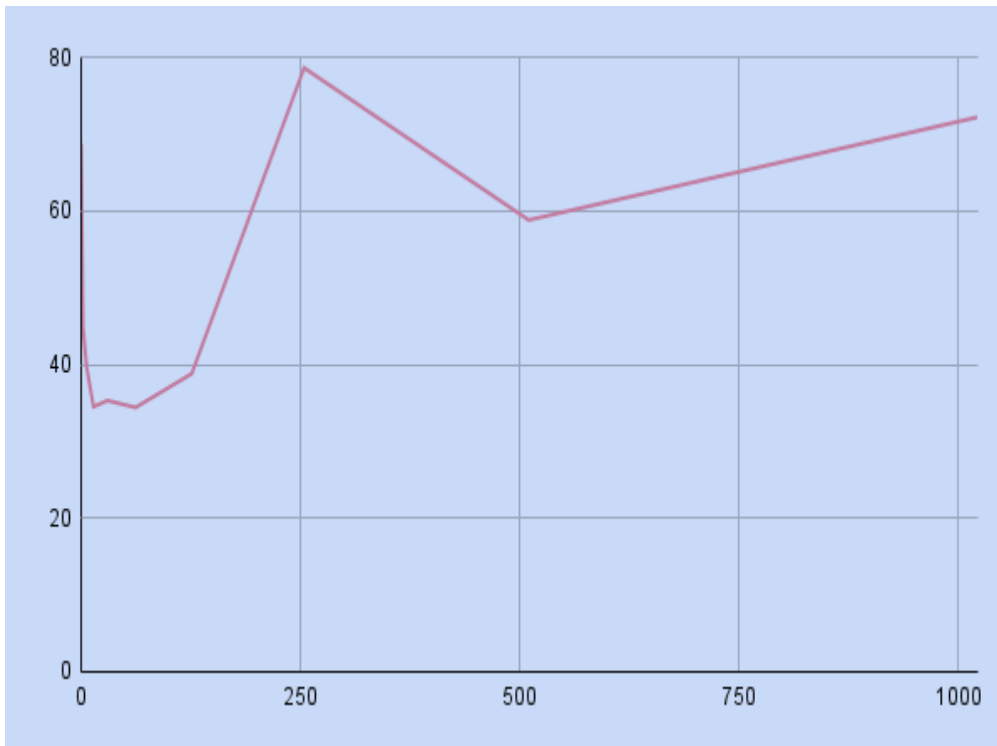
*Graph for increasing cutoffs and the Time Taken for Array Size of 1,000,000*



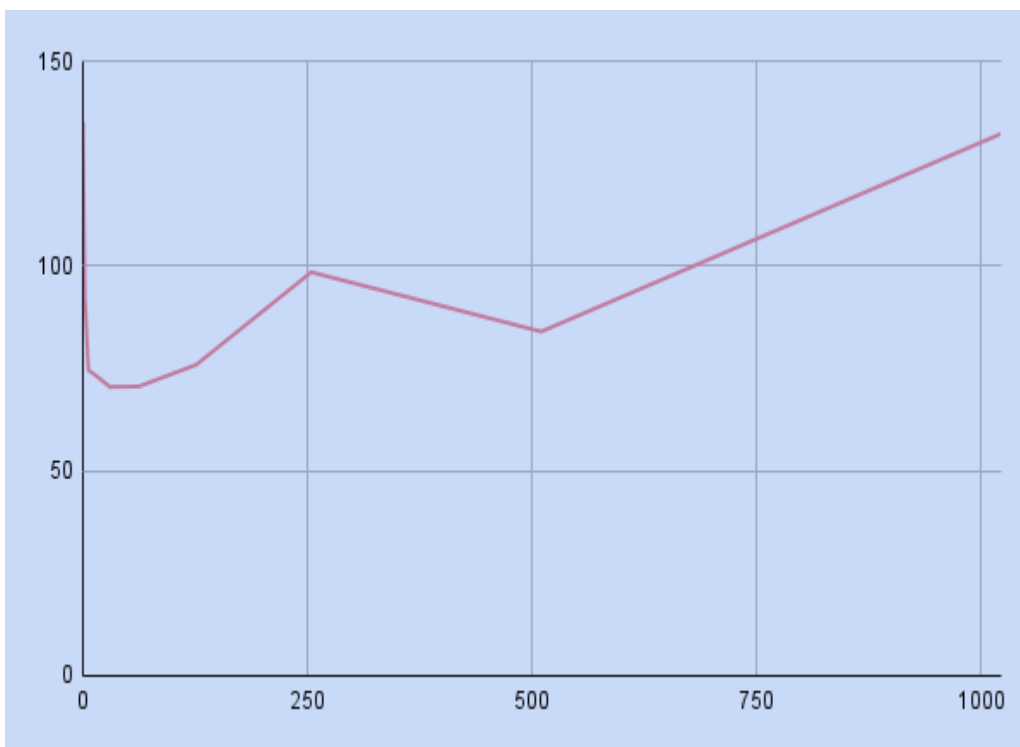
*Graph for increasing cutoffs and the Time Taken for Array Size of 2,000,000*



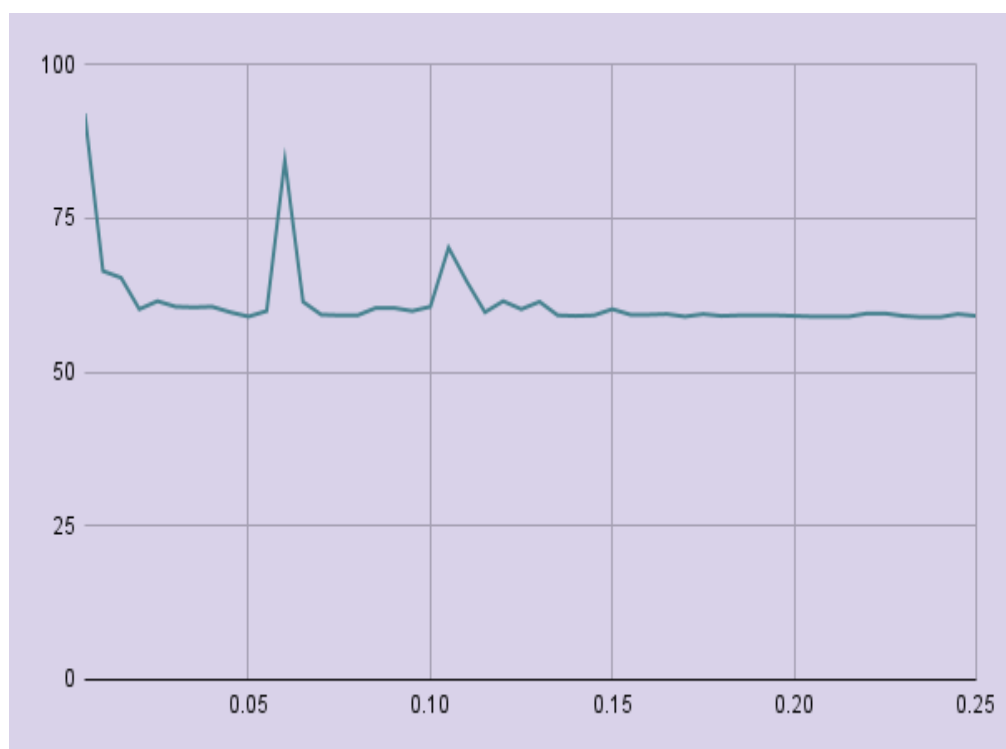
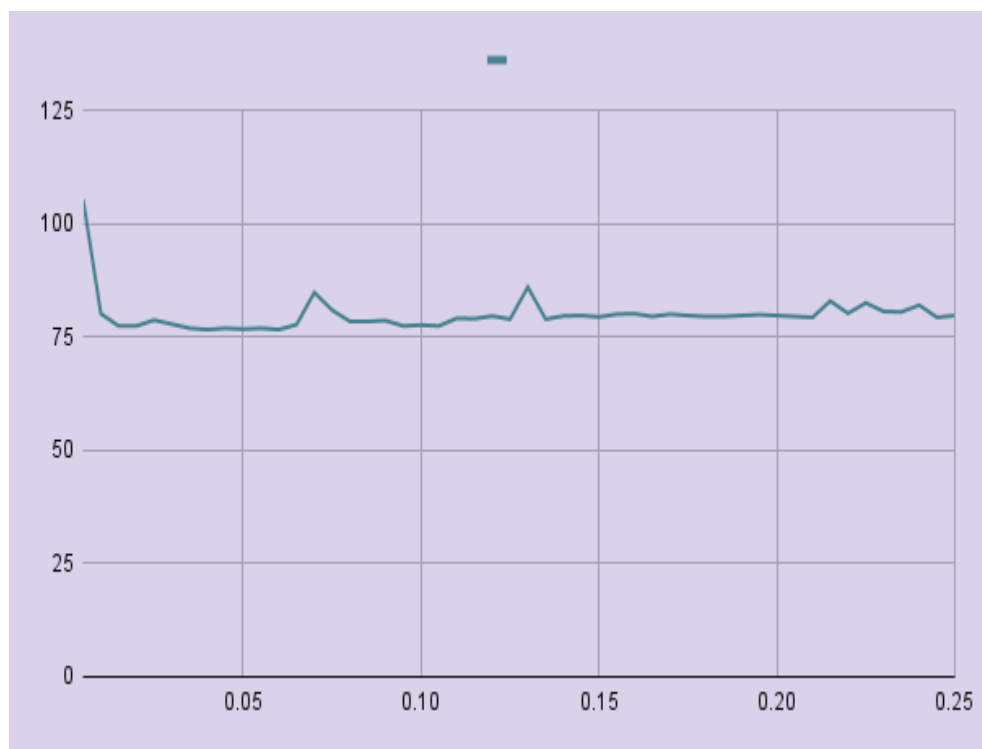
*Graph for doubling values of threads and the Time Taken for Array Size of 500,000*

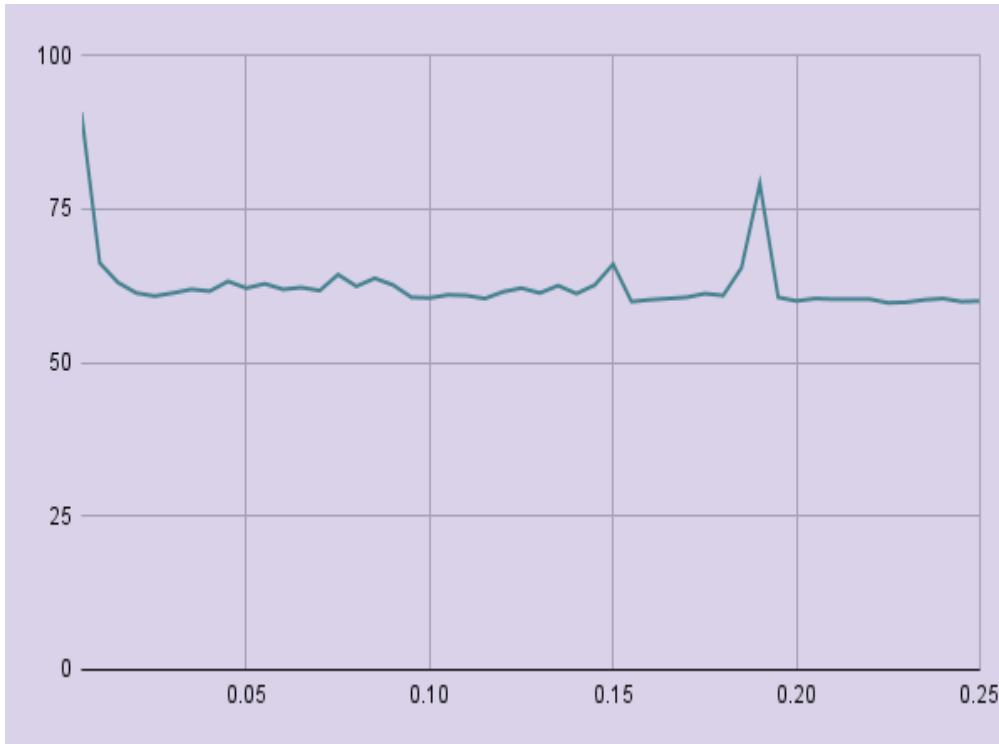


*Graph for doubling values of threads and the Time Taken for Array Size of 1,000,000*



*Graph for doubling values of threads and the Time Taken for Array Size of 2,000,000*





## Output

```
Run: Main x
/Users/poojithakonduparti/Library/Java/JavaVirtualMachines/openjdk-16.0.2/Contents/Home/bin/java ...
cutoff: 510000 10times Time:1055ms
cutoff: 520000 10times Time:802ms
cutoff: 530000 10times Time:775ms
cutoff: 540000 10times Time:775ms
cutoff: 550000 10times Time:788ms
cutoff: 560000 10times Time:779ms
cutoff: 570000 10times Time:770ms
cutoff: 580000 10times Time:767ms
cutoff: 590000 10times Time:770ms
cutoff: 600000 10times Time:768ms
cutoff: 610000 10times Time:770ms
cutoff: 620000 10times Time:767ms
cutoff: 630000 10times Time:778ms
```

Externally added files can be added to Git  
[View Files](#) [Always Add](#) [Don't Ask Again](#)

Build completed successfully in 3 sec, 496 ms (46 minutes ago)

## Screenshot of varying cutoffs

```
Run: Main x
/Users/poojithakonduparti/Library/Java/JavaVirtualMachines/openjdk-16.0.2/Contents/Home/bin/java ...
Degree of parallelism: 7
Thread: 2 10times Time:1319ms
Thread: 4 10times Time:881ms
Thread: 8 10times Time:799ms
Thread: 16 10times Time:730ms
Thread: 32 10times Time:666ms
Thread: 64 10times Time:716ms
Thread: 128 10times Time:704ms
Thread: 256 10times Time:990ms
Thread: 512 10times Time:1250ms
Thread: 1024 10times Time:1281ms

Process finished with exit code 0
```

Build completed successfully in 4 sec, 921 ms (a minute ago)

## Screenshot of increasing thread count