Spencer Whitehead
1001837401

## Executive Summary

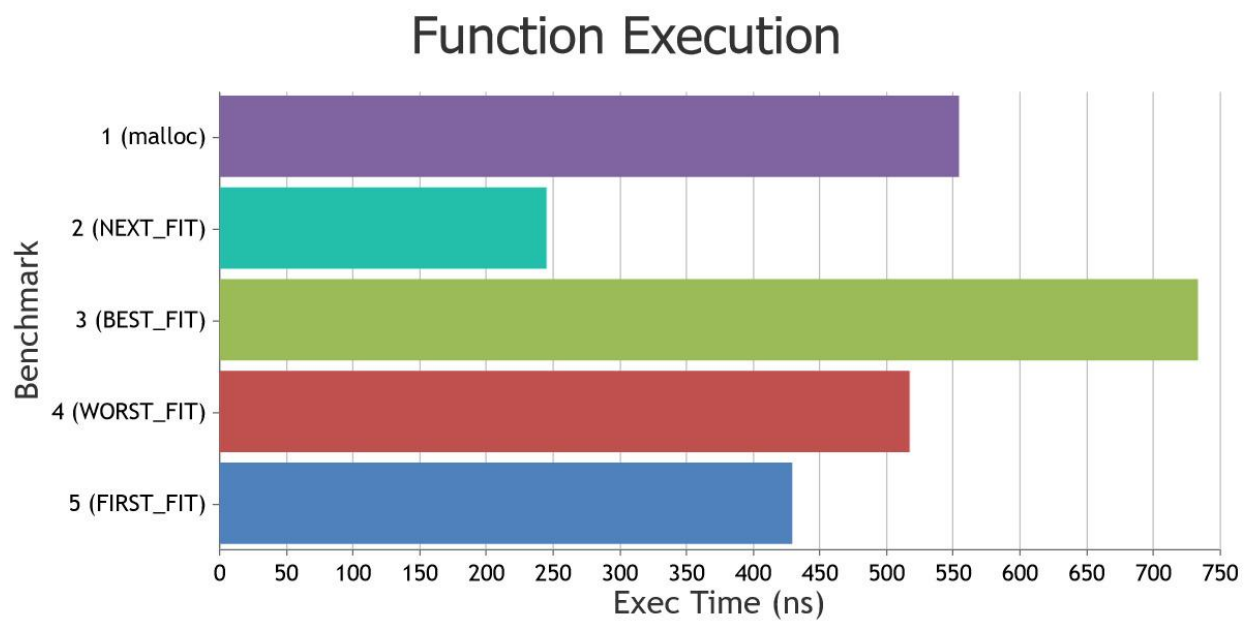
In this project, a program named mavalloc was created to serve as a replacement to the malloc function, by creating an arena of memory that could be managed in a more efficient manner than malloc's calls to the system. The program can allocate processes using four different algorithms: next fit, best fit, worst fit, and first fit. Each of these different algorithms were tested in their execution time against malloc in a series of benchmark programs, the results of which will be detailed in this report along with concluding which algorithm works the best.

## Benchmarks

The five benchmark programs put both the malloc and mavalloc program (with each algorithm) through a series of allocations and times them, allowing their efficiency to be measured.

The process for each benchmark starts by opening a file for recording data, and starting a timer. For each benchmark aside from malloc, a total of 256,000 bytes are initialized. The code then goes through a series of allocations and frees that were copied from the twenty tests in main.c, which allocate and free several processes of different sizes. After this an array is created, and each index in the array is allocated with a random size, having a maximum size of 1/(array size). After the array has been fully initialized, it is then shuffled, by swapping each index with another random index. The first half of the shuffled array is then freed, leaving a series of random holes throughout the arena in the mavalloc benchmarks. The first half of the array is then reallocated with a new random size, allowing the mavalloc benchmarks to attempt to fill the random holes in a different order. This array process is done three times, with arrays of size 64, 256, and 1024 to allow for testing numerous different amounts of allocations and various sizes of processes. Once this has finished, the mavalloc functions are destroyed, the timer is stopped, and the time is recorded.

Each benchmark's execution time was recorded fifty times, providing enough data to be able to see a clear average. The following graph and table show the average execution time for each benchmark, with the table also showcasing the standard deviation:



| **Benchmark** | **Avg Execution Time** | **Standard Deviation** |
|---|---|---|
| 1 (malloc) | 554 | 105 |
| 2 (NEXT_FIT) | 245 | 63 |
| 3 (BEST_FIT) | 733 | 99 |
| 4 (WORST_FIT) | 517 | 107 |
| 5 (FIRST_FIT) | 429 | 80 |

From this table, it can be seen that benchmark 2, using the next fit algorithm, provided the fastest time by far. First fit provided the second best time, followed by worst fit and then malloc. Best fit took the longest, nearly three times longer on average than next fit. However, it should be noted that best fit had the lowest ratio of average to standard deviation, meaning that while it took the longest, it was the most consistent in its time.

If there were any prior expectations to which algorithm would be most efficient, next fit being the fastest may be a bit of a surprise. However, if the process in the benchmark is analyzed, it may provide an explanation for this. Next fit is the only algorithm that doesn't start at the base address of the memory when finding space for a new process, instead starting from the index of the previous allocation. When an array of numerous processes are allocated, next fit will start right after the previous allocation. When most of the memory is still a singular large hole, it will most likely be able to create an allocation immediately after the previous one. Each of the other algorithms in this context would still have to iterate through each of the previously allocated processes, which could add a rather substantial amount of time when making over 1,000 allocations at once.

## Conclusion

After presenting and analyzing the data provided from the benchmarks, it can be concluded that the next fit algorithm performed more efficiently in the benchmarks than any other algorithm, with the lowest average execution time by far. Even if part of this efficiency could be from the process used in the benchmark, it still showcases a notable advantage that next fit has over the other algorithms by tracking the previous allocation.