<u>**Comparing the performance of four custom implementations of allocators against standard system malloc for dynamic memory allocation**</u>

Executive Summary

In this assignment, four different heap management schemes were implemented as alternatives to the traditional system malloc. Test programs were then created to benchmark these tests against the system malloc on performance, relative comparison of number of splits and heap growth, heap fragmentation, and max heap size. The tests involved allocating a large number of memory blocks, subsequently freeing all of them except the last block and then allocating one more block. The aim was to stress the algorithms enough to show results that can help in distinguishing their results from one another. The results showed that all four of the custom algorithms were considerably slower than the system allocator. Among the four algorithms, next fit showed the best performance in terms of speed, followed by best fit, first fit and finally, worst fit.

Description of the Algorithms Implemented

<u>First Fit:</u> Allocates the first available memory block that is large enough to accommodate the data, scanning from the beginning of memory. Efficient in time but can lead to fragmentation.

<u>Next Fit</u><u>:</u> Similar to first fit, but starts searching from the end of the last allocated space, not from the beginning.

<u>Best Fit:</u> Finds the smallest memory block that is big enough to hold the data, aiming to minimize wasted space but can he slow.

<u>Worst Fit:</u> Allocates the largest available memory block, with the idea of leaving as large a block as possible after the allocation, which can be slow and may also lead to significant fragmentation.

Test implementation

The benchmark test aimed to evaluate the performance of the aforementioned custom memory allocation strategies against the standard malloc function by measuring the time taken to for each of the five algorithms to implement. It measures the time taken to allocate 10000 blocks of size 1000 bytes each and the time to allocate one more block after freeing all except the last block. This helps in providing insights into the efficiency of the allocation and deallocation process and the impact of heap fragmentation on allocation time. By comparing these metrics against the standard malloc, we can effectively measure the effectiveness of custom memory allocation algorithms in terms of speed and resource utilization.

Test Results

The benchmarking results clearly showed that the system malloc heavily outperformed the four custom memory allocation algorithms we tested. showed far better performance than the four custom algorithms. This was anticipated, given the extensive optimization efforts that have gone into the development of the system's memory allocation mechanisms over the years. Among the custom algorithms, next fit emerged as the leader in performance speed, which was a notable finding. It managed to allocate memory with minimal overhead, making it the fastest. Following closely behind was the best fit algorithm, which, despite its slightly slower speed, excelled in minimizing wasted space. The first fit algorithm ranked third, offering a balance between speed and space efficiency. At last was the worst fit algorithm, which, as its name might suggest, performed the poorest in terms of speed.

Time difference between custom implementations and standard

	Test 1 (in seconds)	**Test 2 (in seconds)**
System Malloc	0.006384	0.000002
First Fit	1.984677	0.000553
Next Fit	1.881907	0.000502
Best Fit	1.992705	0.000509
Worst Fit	2.078936	0.000502

Test 1 = Allocating 10000 blocks of 1000 bytes each and then freeing all except one block
Test 2 = Allocating 1 one more block of 1000 bytes

Conclusion

In conclusion, the comparative analysis of four custom memory allocation algorithms against the standard system malloc revealed that the system malloc significantly outperforms the custom implementations in terms of speed. Among the custom algorithms, next fit demonstrated the best performance, closely followed by best fit, first fit, and worst fit, in that order. The study highlights the efficiency of system malloc due to years of optimization and sheds light on the limitations of custom memory allocation strategies in managing heap memory effectively.