Mai Dinh 1001146634

CSE 3320 – Spring 2024

Executive summary:

The malloc program was designed to produce performance results of different algorithms in memory management. These results, specifically run time, are then compared to the system malloc() behavior in a test program to learn about the differences in performance of these algorithms.

Description of the algorithms implemented:

The four algorithms implemented were First Fit, Next Fit, Best Fit, and Worst Fit.

The First Fit algorithm simply searches from the beginning of the linked list each time, allocates memory to the first block available. The Next Fit algorithm also allocates memory to the first block available, but differing from First Fit in that it only searches from where it left off last time.

The Best Fit algorithm finds the block with the smallest leftover space possible to allocate memory. It iterates through the linked list to find the smallest block that fits the requested memory size. The Worst Fit algorithm, however, finds the block with the largest leftover space that fits the requested memory size.

Test implementation:

In my test implementation, I allocated memory for an array of 10000 blocks. Then I randomly chose blocks to allocate differing sizes, then allocated the remaining of the first 5000 a set size, and the remaining of the last 5000 another set size. Then I freed the blocks that I randomly chose, and allocated memory again to that array with random sizes. My goal was to have enough random blocks with varying sizes available so that the different algorithms would have some differences in splitting and would choose different address spaces.

Test results for all five candidates (malloc and your four algorithm implementations):

Heap Management Statistics					
Algorithm	First Fit	Next Fit	Best Fit	Worst Fit	System malloc()
grows	10069	10069	10069	10069	
splits	43	43	11	68	
coalesces	5323	5323	5323	5323	
blocks	4720	4720	4688	4745	
max heap	323996	323996	323996	323996	
run time (sec)	0.244851	0.244034	0.241272	0.245581	0.000511

Table 1. Heap Management Statistics results of running malloc.c with maiTestProgram.c

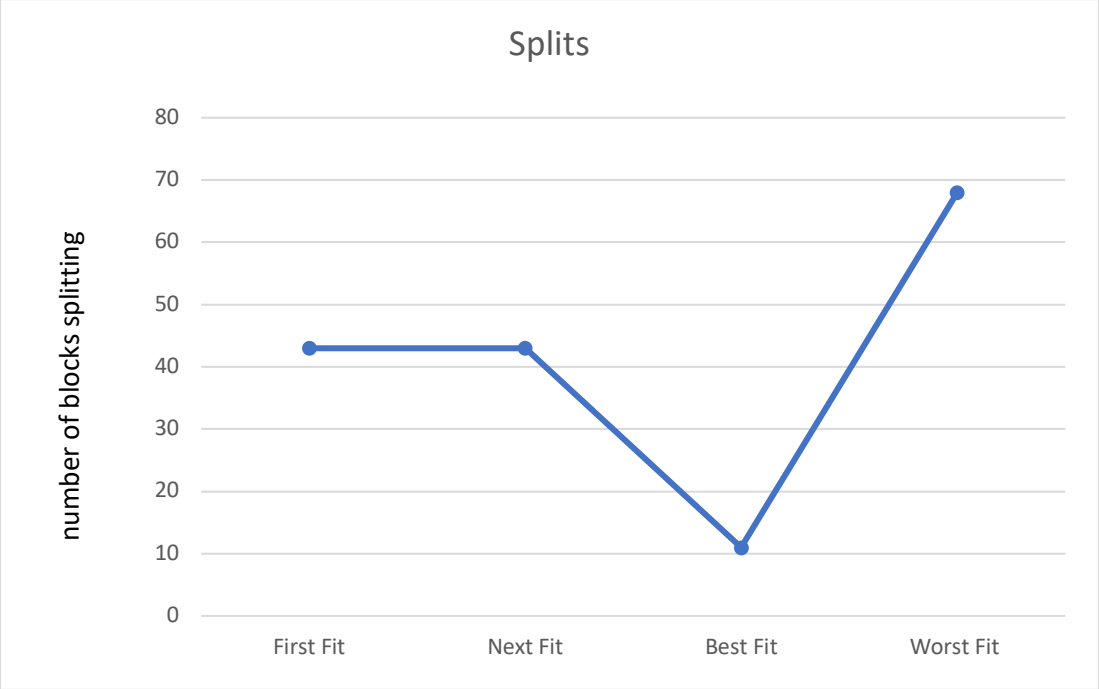

Figure 1. Comparison of the number of blocks that split between the four implemented algorithms.

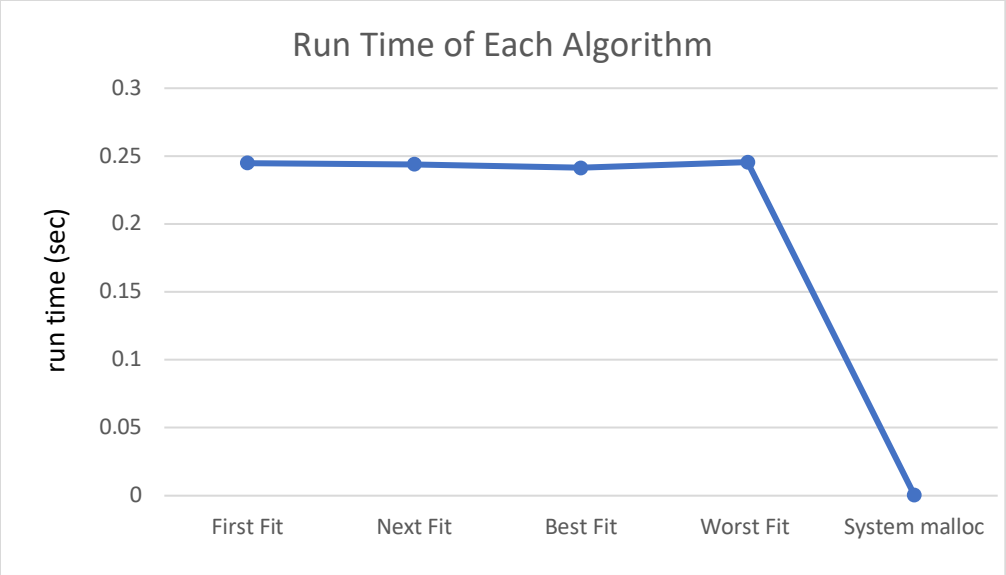

Figure 2. Run time between the implemented algorithms and system malloc()

Explanation and interpretation of the results including any anomalies in the test results:

Firstly, the statistics that are the same across the four implemented algorithms includes heap growth, coalesces, and max heap size. These are the same because I did not allocate enough pointers after freeing the blocks so there weren't enough opportunities to differ in these statistics. However, the number of splits in best fit and worst fit differed from the other two and from each other. The best fit had the least number of splits since it finds the smallest space possible it can fit. This reduces the need for splitting when it's able to fit more blocks where the requested size and available block size are equal. In contrast, the worst fit algorithm had the most number of splits since it finds the largest space it can fit so there's bound to be space leftover. The number of splits with first fit and next fit are the same because they are very similar and there was more a lot of free space than requested memory so it's likely that the first fit and next fit used the same blocks or blocks with similar amounts of space for most allocations and therefore split the same number of times. The number of blocks followed the same pattern of splits because they are directly proportional to each other. More splits create more blocks, and less splits create less blocks.

Run time results varied very little between the four implemented algorithms but system malloc() is significantly faster than those algorithms. The insignificant difference between runtime in the implemented algorithms could be due to all of the algorithms having both cons and pros and they even each other out. It could also be due to the test program not being tailored enough for at least one algorithm differ significantly from the others. The runtime for the system malloc is

significantly faster because it has more efficient algorithms than just the simple algorithms implemented.

Conclusion:

In conclusion, the test program did well in highlighting the number of splits and blocks between the four implemented algorithms. It did not do well in tailoring the program enough to see a significant difference in runtimes between those algorithms. However, it did show that the system malloc() is much more efficient than any of the other algorithms alone.