CSE 3320 Malloc Assignment

Ikechukwu Ofili

1001862556

April 14, 2023

1 Executive summary

This report documents the performance of four memory allocation algorithms (first-fit, next-fit, best-fit, and worst-fit). These algorithms are tested by writing implementations of malloc, calloc, and realloc that use the algorithms. These functions are used instead of their *stdlib* counterparts of the same name in programs that we then use to compare the performance of each allocator with each other as well as with the system malloc, calloc, etc.

Obviously, the system imple will be better than any of the algorithms used not particularly because it uses a much better allocator algorithm but mostly because of better optimization choices like growing the heap at huge increments at a time instead of every time there is no free node in the list, etc.

2 Description of the Block-finding Algorithms

There are four algorithms that this report talks about. They are:

- First fit
- Next fit
- Best fit
- Worst fit

2.1 First fit

This algorithm is probably the simplest of all the allocator algorithms. The goal is to loop through the list of memory chunks and return the first **free** chunk that is big enough for the requested size.

2.2 Next fit

Next fit is exactly like first fit but instead of searching the list of memory chunks from the beginning every time, we remember where we stopped the last time we searched for a block and we start from there. That way, we do not end up picking the same block when a supposedly "better" block could be somehere farther in the list

2.3 Best fit

This algorithm picks the block that best fits the size that was requested. In other words, it picks the smallest block in the list that has a size greater than or equal to the size requested. This reduces the number of block splits that we need to do but it also is prone to heap fragmentation since there may be various small free blocks split from the allocated blocks

2.4 Worst fit

This is the opposite of best fit. For this algorithm, we pick the biggest block in the list, as long as the size is greater than or equal to the one requested. This algorithm tries to fix the heap fragmentation problems that come with best fit with the expense of multiple block splitting

I expect best fit and worst fit to do worse time-wise since they always have to traverse the entire list each time a memory allocation takes place. However, they are better than first and worst fit in managing heap fragmentation and reducing heap size

3 Benchmarking

4 Conclusion