Machine Problem 1: A Simple Memory Allocator
Performance Analysis

Julian Tiu* | Dr. Bettati | CSCE 313 504
1 October 2017

Must Read:

The environment used to compose the program is Linux through the use of Mac OS and compiled with g++. Compilation is available just by entering "make" in the command line. This will generate some warnings, but the program should still build without any errors. An executable file, memtest, is created and ran by typing "./memtest" on the command line.

Analysis:

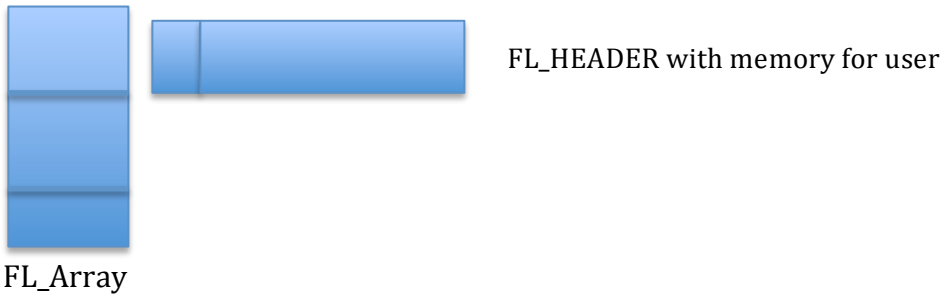
Milestone	a	b	Result	Seconds	Microseconds	Cycles
2	1	1	3	0	1659	4
3	1	1	3	0	1825	4
2	1	5	7	0	588	12
3	1	5	7	0	515	12
2	2	1	5	0	811	14
3	2	1	5	0	834	14
2	2	5	13	0	8994	90
3	2	5	13	0	10480	90
2	3	1	13	0	13199	106
3	3	1	13	0	13287	106

Performance-wise in some cases milestone 3 generated a faster time than milestone 2, but on average milestone 2 results were faster than that of milestone 3 results. The variation in times is due to the random length passed in to my_alloc for each trial.

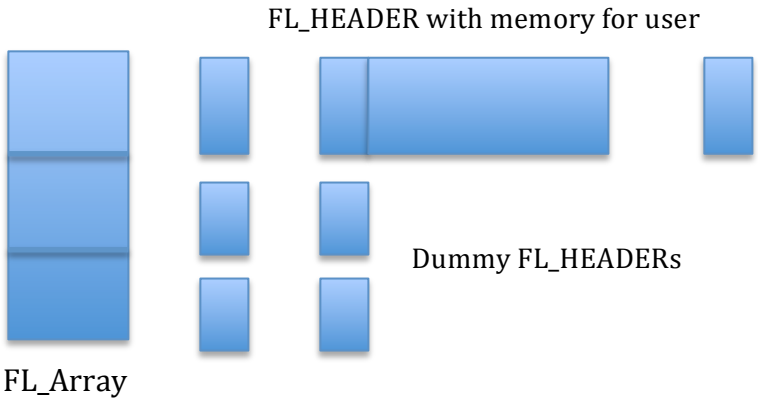
Milestone 2 generating a faster time than milestone 3 may be due to the differences in simplicity or complexity of the implementation for both milestones. In milestone 2, there was only one free_list to manage, while milestone 3 had *n* amount of free_lists to manage, and each were accessed through a for-loop, which is O(*n*) time.

The bottleneck of performance may be due to my specific choice of implementation as oppose to how the suggested implementation was presented. In the suggested implementation, each index of the array that manages different sizes of free_lists contains the actual FL_HEADER that contains the memory to be returned to the user. My implementation had two extra dummy FL_HEADERs each located at the index of the array and at the end of that index, and the actual FL_HEADER that contained the memory to be returned to the user was placed between the two dummy FL_HEADERs. Although in turn it made it easier to implement the FL_add the FL_remove functions, it was a waste of memory for each allocation of dummy FL_HEADERs and a waste of time to iterate through dummy FL_HEADERs.

Suggested Implementation:



My Implementation:



One way to improve the performance of the implementation is to sacrifice the simplicity of the FL_add and FL_remove function by getting rid of the dummy FL_HEADER at the end of each index. This would make iteration through each array index a bit faster and would allocate less memory from the heap for each dummy FL_HEADER.