Buddy Allocator Analysis

The buddy allocator is a memory management system wherein the memory space can be split up into smaller and smaller pieces known as buddies. Each memory split can only be done to one chunk of memory at a time and only create two equal chunks of memory. The idea behind this system is that it allows efficient rearrangement of memory by allowing chunks to be split and merged as needed.

While analyzing my implementation of the buddy allocator, I noticed an interesting trend. The amount of time needed to complete a series of allocate/free operations increased linearly for any number of allocate/free operations. This was expected because the runtime of the recursive split and merge functions (which form the core of the buddy allocator) all scaled relative only to the total size of the memory space, not how much has been allocated. Therefore, each allocation/free should take approximately the same amount of time, and the total should scale linearly. As can be seen in figure 1, when a linear equation was fitted to the graph of the data the R2 value was 1.

Figure : Time taken vs number of allocate/free cycles

Thinking about the implementation of this buddy allocator, I can identify a potential bottleneck affecting the amount of internal fragmentation present in the system. The main drawback of the buddy allocator is that smaller units of memory may exist because of earlier splits but may not be big enough to fit any further data. This can lead to systems where no more memory above a certain size can be allocated, yet the total amount of unallocated memory is even larger. I believe this can be reduced by prioritizing the usage of buddies that already have