Alan Achtenberg

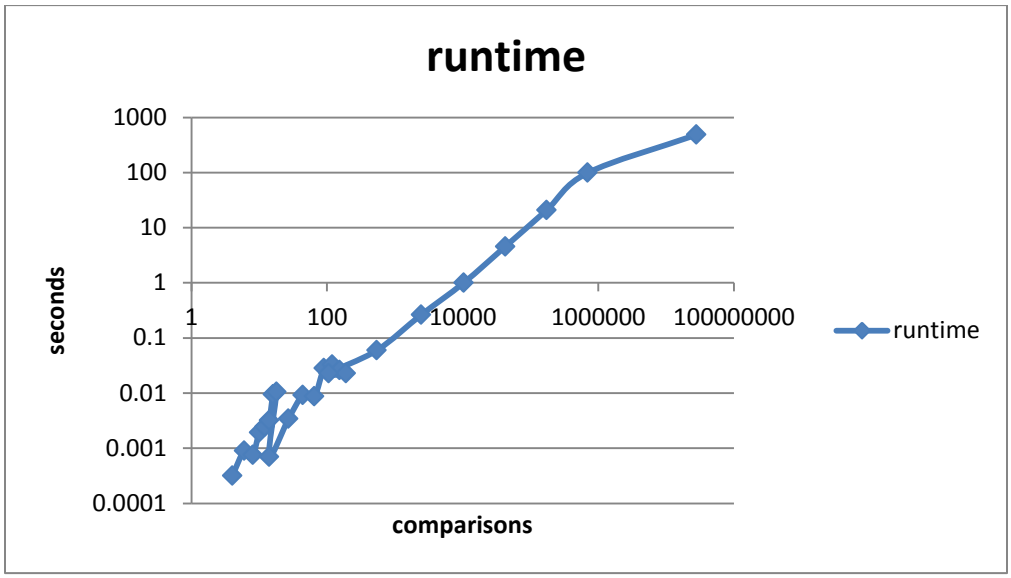
9/18/2013

Machine Problem 1

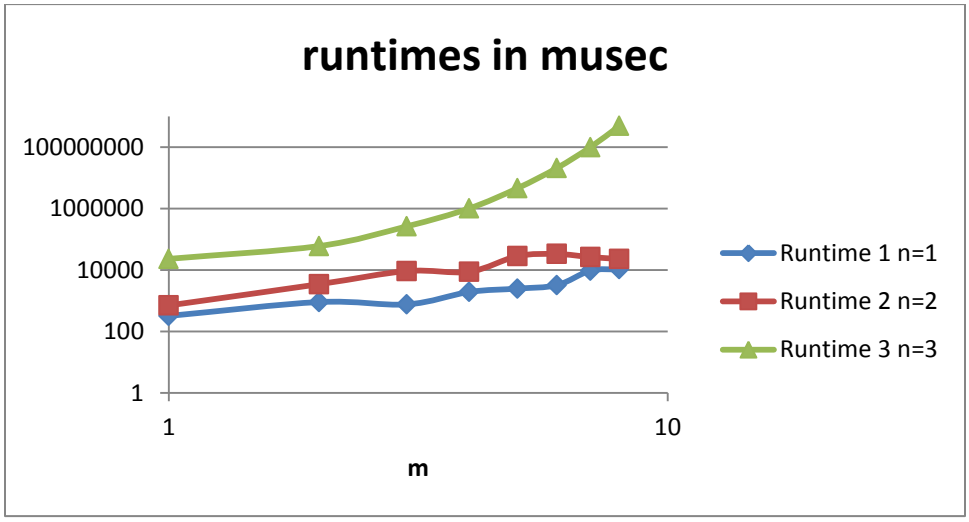
CSCE 313-501

Prof Amin Hassanzadeh

For my memtest using my allocator the relationship of run time with respect to number of comparisons is fairly linear, when you take away the outliers for small comparisons due to less calculations allowing for greater chance of average value to be extremely different. For results see the Graph below in a log base 10 scale along both axis.



My Memtest uses an implementation of the ackerman function for testing. The results of the tests were as such. It is much easier to see the exponential growth in n=3. Graph below is in a log base scale along both axis.



The bottle necks in the system are definitely when large amounts of memory are initialized with only a few small block sizes and small amounts of memory are requested from the system, because the free blocks are large a lot of time must be spent breaking the blocks down into smaller block sizes. And there is a lot of increased internal fragmentation. The efficiency of the memory management is also much greater for bigger block sizes due to that every block has a small constant size added for Headers that does not change with the size of the block. The biggest implementation bottlenecks are in the joining of blocks because even when it cannot be joined it still checks for the possibility, and it also does this everytime a block is free'd as compared to when a block is allocated only when it needs to does it have to break a block.

As stated above the slowest part about the implementation is the freeing and joining of blocks. One possible solution to this is to rewrite the code so that it only joins blocks when there is no memory available of that block size to allocate. So all the joining and breaking are both done in the the allocate function. The benefit of this is that it can skip joining blocks for every free, but it also has a downside as well. Without specific blocks being freed, the operation of finding buddys to join would be much harder to implement and a more expensive operation. For this change in the algorithm to be significantly more efficient it would have to have rare needs of joining blocks. For example a program that consistently allocates and free blocks from your smallest block size would theoretically only need to break blocks not join them.