Neal Patel, Duy Nguyen

CSCE 313 – 503

September 8, 2016

MP 1 Report

Team Members: Neal Patel, Duy Nguyen

Section #: 503

Findings: The problem we were trying to solve was to improve the performance of the linked list by reducing the number of interactions it dealt through the operating system's memory manager to one time; that one time being an initial call to allocate memory from the system. After this initial call, we used our program to deal with any operations afterwards. By taking precautions prior to the linked list operations, no errors are witnessed within the program. By only making one call to the memory manager, we increase the uptime of the execution of function calls, which are dealt through the processor; this reason is responsible for producing a more efficient performance of the linked list.

Question: Do you notice any wastage of memory when items are deleted?

Answer: Originally, our linked list would have empty spaces that would never be filled due to deleting at the head, but after adding a beginning pointer, which keeps track of the beginning of the memory block, we do not find any wastage of space.

Question: If so, can your program avoid such wastage? How would you do so?

Answer: As stated in the previous question, we were able to fill the "always deallocated space" by keeping a beginning pointer. Because the head pointer points at the first filled node, that

leaves blocks of space available for insertion before the head if items are deleted at the head. By creating a function called “gap_check(),” we are able to check for any gaps. The head pointer does not necessarily mean the beginning of the memory block so that’s the reason for keeping a beginning pointer at the start of the memory block whereas the head pointer may move depending on insertions and deletions.

Question: Can you think of a scenario where there is space in the memory but no insertion is possible?

Answer: Before the beginning pointer, along with the adjustment of the insertion function, insertion is not possible in the spaces before the head pointer if the delete function was called on the first node.

Question: What is the maximum size of the value when the pointers are 8 bytes?

Answer: Considering that the basic block size is 128 bytes (since it is not specified, go with the default size), the header pointer is 8 bytes, the payload, which consists of the key (4 bytes), the value len (4 bytes), and the value rest, which is what we’re trying to find, the maximum size of the value when the pointers are 8 bytes would be 128 bytes – (8 bytes + 4 bytes + 4 bytes) = 112 bytes. Considering if the basic block size b was specified, the calculation for the maximum size of the value would be: b – (8 bytes + 4 bytes + 4 bytes).