Daniel Yoh

CSCI 313-22

#3: Test a stack's speed (with millions of operations) as to which is faster: an array based stack or a linked list based stack

Header files are included for making array based and linked list based stacks, and for timing functions (“myStack.h”, “linkedStack.h”, <chrono>). Then the functions that take in a stack and a size for timing array stack and linked list stack operations are declared, then we reach the main function. I wanted to test the times of the functions multiple times with increasing numbers of elements in the stacks, so I used a sort of iterator integer that would act as the size to pass into the functions, and would change sizes for each iteration. The sizes are a million, five million, then ten million.

A for loop is used to call the functions three times, once for each size. The iterator int changes size depending on the for loop’s iteration (0, 1, 2). Then the stacks are created and initialized. They are passed into the functions with their size, then the functions run millions of push and pop operations on the stacks. The execution times are measured, which are passed back as two integers. The results are shown and compared, with the faster one being stated.

Both the timing functions work pretty much the same. The reference address to the stacks are passed into the function, and the sizes are also given so that the for loop can perform operations that match the size of the array based stack. In the linked list stack’s case, there’s no size given when it is declared in main, so its for loop’s size will match the size of the array stack to ensure fairness. The time starts and stops before and after the for loops, which push and pop elements into the stacks. For simplicity, the elements are i from the for loop. Then the duration of the time is calculated and returned to main.

Running the program multiple times shows that array stacks are faster than linked list stacks. On average, they’re about 3 times faster.