Amy Zhang

CSCI-313

HW2- Question 3: Array & Linked List Based Stacks

For this problem, I used the array and linked list based stack classes given to us on blackboard. Additionally, in the stackType class, I created a function called resizing(int capacity) that will increase the array capacity when the stack is full. Within the resizing function, I first created an array pointer object called oldData that will store the original information stored in the array pointer list. Next, I change the maxStackSize to the input capacity, which is what the following code is doing: maxStackSize = capacity. Then, I used a for loop to copy all the information stored in oldData back to the array pointer list. Before exiting this function, I deleted the oldData to prevent memory leakage.

Moving on, I created another function called stackOperations(stackADT<int> & stack, int pushOperations, int popOperations, int rounds). This function will take in a stack declared in the main. Furthermore, it will also take in three int variables, which are pushOperations, popOperations, and rounds. The pushOperations is the number of push operations that will be performed on stack per round. The popOperations is the number of pop operations that will be performed on stack per round. The outer for loop uses the rounds variable, which is the amount of repetition for performing the pushing and popping operations for the input stack. Continuing on, there are two for loops. The first for loop will add values to the stack starting from index 0 all the way up to one less than the pushOperations. Similarly, the second for loop will start at index 0 and remove values from the stack all the way up to one less than the popOperations. Right before this stackOperations function, there is a template TimeFunc that will be used in the main for timing which based stacks are faster.

In the main function, I declared three const int variables, which are pushOperationTotal, popOperationTotal, and rounds. I initialized both pushOperationTotal and popOperationTotal to 50,000 and rounds to 100. These will later be passed into the stackOperations functions as three of the four parameters. Next, I declared an array based stack object called arrStack that will be storing integer values and it will be one of the parameters for the stackOperations functions. The following code: long long stackOperationTimes = TimeFunc([&]() {stackOperations(arrStack, pushOperationTotal, popOperationTotal,rounds); }), will be calling the stackOperations function and perform the pushing and popping operations "rounds" amount of times. At the same time, the TimeFunc is used to time this whole process. When the operation is completed the final time will be stored in the stackOperationTimes and printed. For the linked list part, I declared a linked list based stack object called listStack that will store integer values. The stackOperation and timing part for linked list based stack is the same as the array based stack. In general, to answer the question whether an array or linked list based stack is faster by timing them with millions of operations, I concluded that an array based stack is faster.