Brandon Mouser

Programming Assignment 07

CS-2420

Professor Meyer

Analysis Document

Who is your programming partner? Which of you submitted the source code of your program?

My programming partner was David Ord. I submitted the source code of our program on Thursday, March 3rd.

Have you worked with more than one partner yet? Remember, you are required to switch at least once this semester.

Yes, David is my second partner this semester and I am planning on switching off again after this assignment.

In the LinkedListStack class, the stack data structure is implemented using a doubly-linked list. Would it be better to use a singly-linked list instead? Defend your answer.

I have not tested this, but knowing what I do about both Doubly Linked Lists and Singly Linked List, I would say it would be better to use a Singly Linked List instead of Doubly Linked List because for the Linked List Stack that we create, items are being added and removed only in the very first spot of the Stack. Which is the very beginning position of the Doubly Linked List. Doubly Linked Lists are good because they have a reference to

the Node before and after the Node that you are at. Singly Linked Lists just have a reference to the next Node in the List. While this is fine, the Stack does not need those references to the previous Node like the Doubly Linked List has. Also, a Doubly Linked list can place items anywhere in the list, but once again, we don't need to do that for the Linked List Stack, we just have to be able to add and remove items from the first spot, which Singly Linked List can do a lot easier than Doubly Linked List.

Would it be possible to replace the instance of DoublyLinkedList in the LinkedListStack class with an instance of Java's LinkedList? Why or why not?

I am going to say yes, it is possible. I changed my Linked List Stack to use a LinkedList instead of a DoublyLinkedList and everything still ran and passed every test. Unless I tested it wrong, the LinkedList class should work for everything because all we are doing is adding and removing elements from the first position, which a LinkedList can do easily. So yes, it will work.

Comment on the efficiency of your time spent developing the LinkedListStack class.

Honestly, we made the LinkedListStack class in less than 20 minutes. It was really easy because everything was already done for us in our DoublyLinkedList class. We just had to call on those functions and the amount of coding that it took to do this was minimal in comparison to the BalancedSymbolChecker class.

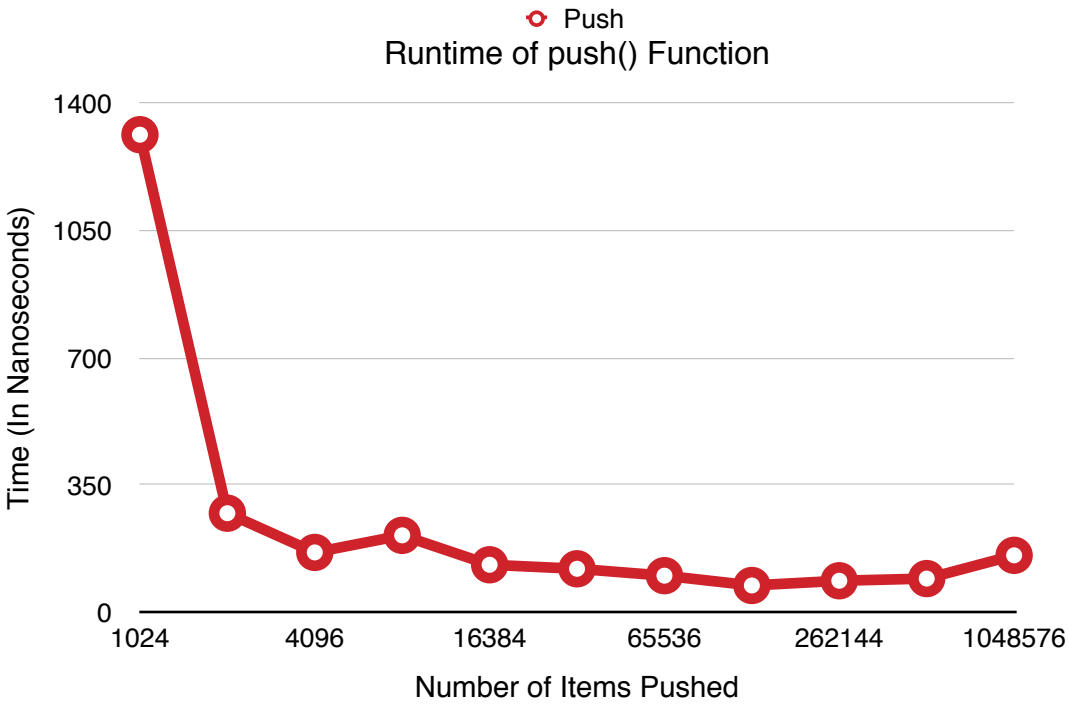
Explain how you would also keep track of the line and column number of the unmatched opening symbol. For example, in Class1.java, the unmatched symbol is detected at line 6 and column 1, but the original '(' is located at line 2 and column 24.

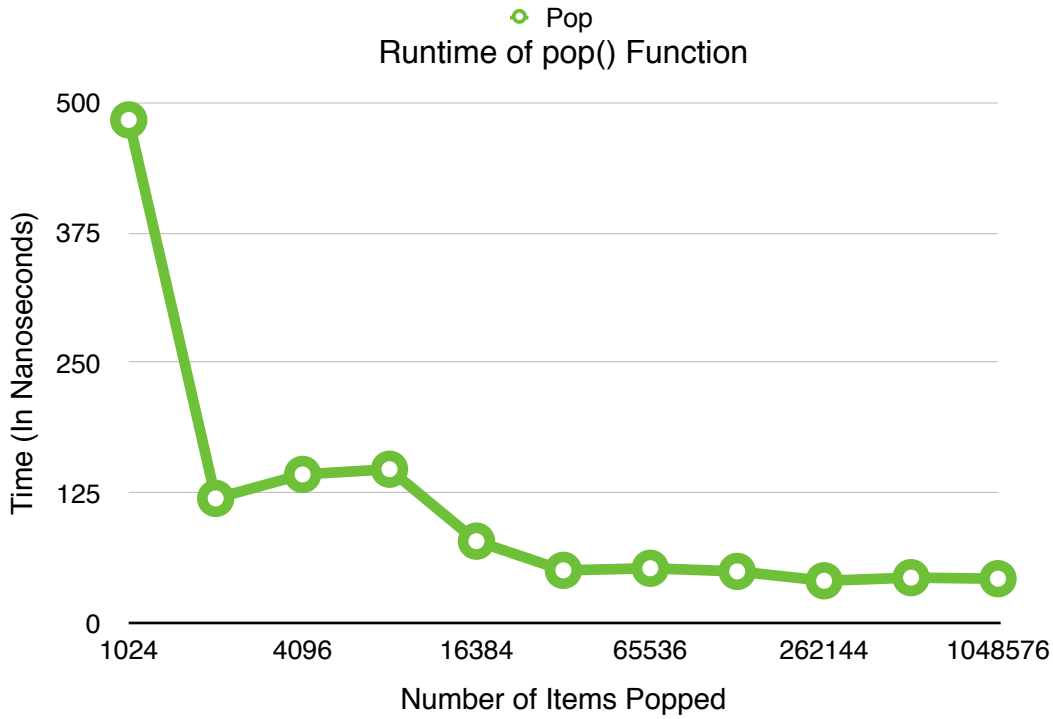
There are many different ways I can think of to do this, but I think the best way would be to have a separate Stack that holds the row and column number. Or two different Stacks, one that has line numbers and one that has column numbers, but you would only need one because you could push() both times to the stack then call pop() twice to get both numbers. So when the opening symbol is hit, use the current line and column that you are at, and add it to the new Stack. When the closing mark comes, when you pop() the symbol in the first stack you could also pop() the numbers from the second Stack. So if there is an unmatched symbol at the end, pop() the numbers in the other Stack to get where the original error went wrong.

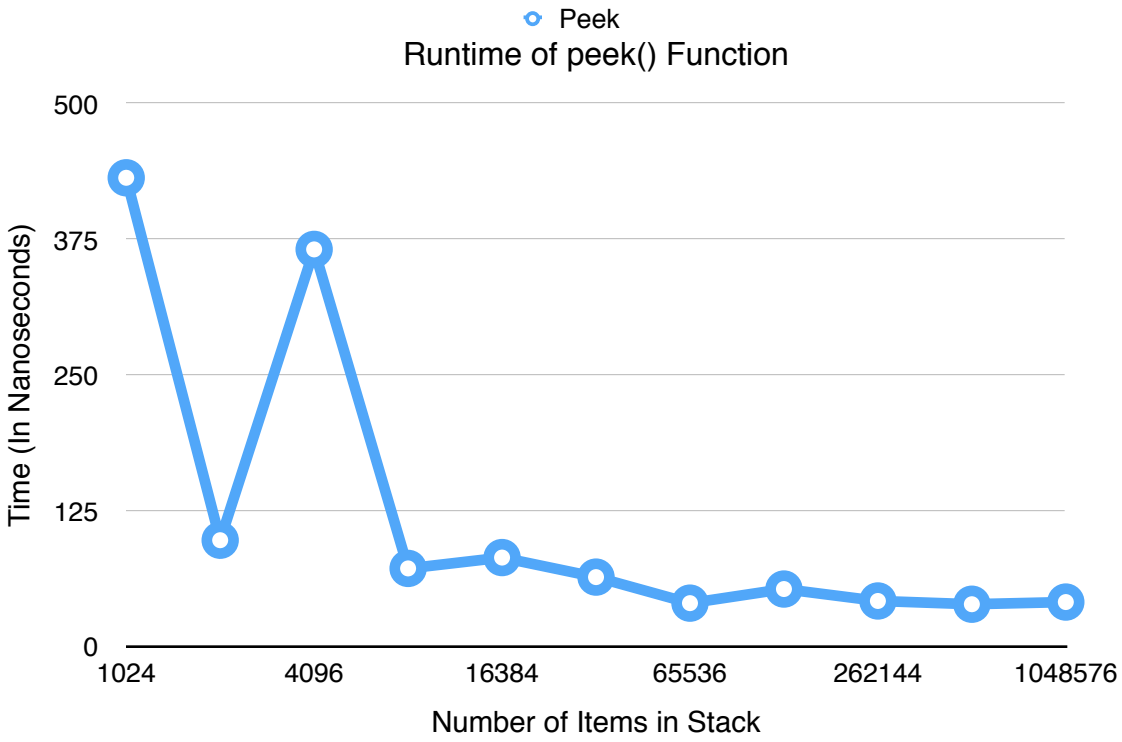
Collect and plot running times in order to determine if the running times of the LinkedListStack methods push, pop, and peek are O(1) as expected.

Below are the running times of push, pop and peek. From what can be seen by these graphs, all of these methods are O(1) as expected. The time is constant no matter how big of a Stack we have and how many elements are being added in. Except for the few abnormalities in timing, which every timing program I've ran has had, the time to push, pop, or peek a single element from a Stack of different sizes is always about 0 seconds, (or around 0-100 nanoseconds), showing that these functions are indeed constant with

a Big-O complexity of O(1). The numbers below reflect the time it takes to push,peek, or pop a single item in the Stack.







How many hours did you spend on this assignment?

Overall we probably spent around 10-12 hours on this assignment.