CSC 225 Assignment 3

Linked Lists, Stacks, Queues, and Matching

University of Victoria

February 7, 2022

Due date

The submission deadline is 11:59pm on Monday February 28, 2022.

How to hand it in: Submit your ASSIGNMENT3.PDF and ARRAYMATCH.JAVA files through the Assignment 3 link on the CSC225 BrightSpace page.
IMPORTANT: The files submitted must have a .PDF or a .JAVA extension.

Exercises

Question 1

In pseudocode, describe a recursive algorithm that reverses the elements in a singly linked list.

Assumption: that the recursive algorithm is originally called with the head node in a linked list.

Algorithm reverse(n):
Input: The first node in a sequence of elements forming a singly linked list
Output: A reverse linked list (n) ends up as the last node in the sequence).

Question 2

Consider how the stack ADT could be implemented using two queues, **Q1** and **Q2**.

When a user **pushes** a number of elements to the stack and then **pops** an element, the last element (most recent) inserted should be returned and removed (LIFO).

(a) Describe how the push and pop operations are implemented.

(b) What are the running times of the push() and pop() methods for this implementation?

Question 3

Complete the implementation of the match method in **ARRAYMATCH.JAVA**. This method determines if a match (something we are defining for this particular problem) can be found when examining two arrays, A and B. A and B and are arrays of size n, containing the same number of integer elements.

Two arrays, A and B, are defined to be matches of one another if at least one of the following two conditions is satisfied:

I. $A = B$ (the arrays have the same elements at each index)

II. If n is divisible by 2, A and B and are divided into two sub-arrays of equal size (A is divided into A_1 and A_2, B into B_1 and B_2). Then, at least one of the following conditions is satisfied:

(a) (A_1 matches B_1 AND A_2 matches B_2)

(b) (A_1 matches B_1 AND A_1 matches B_2)

(c) (A_2 matches B_1 AND A_2 matches B_2)

Note: if n is not divisible by 2, condition II is not satisfied.

Additional Information:

You **cannot** change the method signature for MATCH at all (two integer arrays as parameters, and returns a boolean) or you will receive a score of **0**. If your submission fails to compile, you will receive a score of **0**. You are welcome to create additional methods to aid in your implementation, but again, the MATCH method must return a boolean when given two integer arrays.

The methods provided for you will handle file I/O. When executed, the program reads from input files, and outputs whether a match is found based on the array data found in the file.

The program is executed in the following way: **JAVA ARRAYMATCH FILENAME.TXT**

Input files must be three lines, formatted in the following way:
<a single integer representing the size of the arrays>
<integer elements for Array A, where the elements are separated by white space>
<integer elements for Array B, where the elements are separated by white space>

You have been provided with some sample files. It is strongly recommended you add further tests.

File name	Expected output	Reasoning
test01.txt	match found	$A = B$
test02.txt	no matches	$A \neq B$, and no conditions from II are satisfied; when the arrays are split, $A_1 \neq B_1$, $A_1 \neq B_2$, $A_2 \neq B_1$, and $A_2 \neq B_2$, and the size (n) is not divisible by 2 for these arrays, so no further splits are made.
test03.txt	match found	II (b) is satisfied (A_1 *matches* B_1) AND (A_1 *matches* B_2)
test04.txt	match found	II (a) is also satisfied: (A_2 *matches* B_2 trivially. Eventually we will also see A_1 *matches* B_1 after a number of subarrays are created and determined to be *matches* of one another. *I recommend drawing a picture!*

For problem 3 the mark breakdown will include the following:

(a) Correctness: how many *automated* tests your implementation classifies correctly.

(b) Runtime efficiency: $O(n \log n)$ will receive full marks, followed by $O(n^2)$, $O(n^2 \log n)$, $O(n^3)$, etc. Runtime efficiency marks are given separately from marks distributed for part 3(a). Runtime efficiency will be graded by manual inspection.