ESTRUCTURAS DE DATOS

TRABAJO PRÁCTICO N ° 3

Pilas y Colas

Departamento de Ciencias e Ingeniería de la Computación - U.N.S.
Primer cuatrimestre de 2019

Bibliografía:

[GT] Michael Goodrich & Roberto Tamassia. Data Structures and Algorithms in Java. Fourth Edition. John Wiley and Sons. 2006.
[W12] Mark A. Weiss. Data Structures and Algorithm Analysis in Java. Third Edition. Addison-Wesley Pearson Education, Inc. 2012.

En la página web de la materia se encuentran disponibles los archivos TdaPila.jar y TdaCola.jar provistas por la cátedra. Estas implementaciones pueden utilizarse para resolver los ejercicios marcados con ✪.

Ejercicio 1:
Defina el TDA pila, es decir, indique el modelo sobre el cual está basado y un conjunto de operaciones con sus descripciones precisas para manipular datos de este tipo. Para ello, utilice una interfaz llamada *Stack*. También, la pila debe tener un parámetro genérico de tipo E que representará el tipo de los elementos de la pila.

Ejercicio 2:
Implemente una clase *PilaArreglo* donde la pila se representa con un arreglo para almacenar los elementos de la pila y un entero para indicar el índice del tope en dicho arreglo. La clase PilaArreglo debe implementar la interfaz Stack definida en el ejercicio 1. Bajo esta implementación, ¿qué consideraciones especiales se deben tener al apilar un nuevo elemento?

Ejercicio 3:

a) Utilizando la clase PilaArreglo implementada en el Ejercicio 2, programe un método llamado invertir(P) que invierta el contenido de la pila P que se recibe como parámetro. De considerarlo necesario, puede recurrir al uso de otras pilas. Calcule el orden del tiempo de ejecución de su solución.✪
b) Agregue un método a la clase PilaArreglo programada en el Ejercicio 2 que permita invertir el contenido de la pila receptora del mensaje. Compare la signatura de este método con la dada en el inciso (a).
c) Suponiendo que posee la clase *Persona,* implemente un método Invertir(A) que reciba un arreglo de personas y utilice la clase PilaArreglo para invertir el contenido del arreglo A. Calcule el orden del tiempo de ejecución de su solución.
d) Resuelva nuevamente del problema (3.a) pero utilizando la clase provista por Java llamada *java.util.Stack*.

Ejercicio 4:
Programe una clase llamada *PilaConEnlaces* que implemente la interfaz Stack. En este caso, la pila se implementa con una lista enlazada para almacenar los elementos de la pila.

Ejercicio 5:
Defina el TDA cola, es decir, indique el modelo sobre el cual está basado y un conjunto de operaciones con sus descripciones precisas para manipular datos de este tipo. Para ello, utilice una interfaz llamada *Queue*. También, la cola debe tener un parámetro genérico de tipo E que representará el tipo de los elementos de la cola.

Ejercicio 6:
Programe una clase llamada *ColaConArregloCircular* que implemente la interfaz Queue definida en el ejercicio (5). En este caso, la cola se debe implementar internamente con un arreglo circular.

Ejercicio 7:
Programe una clase llamada *ColaConLista* que implemente la interfaz Queue definida en el ejercicio (6). Esta clase, debe modelar internamente la cola con una lista enlazada, manteniendo referencias al primer y último elemento de la lista.

Ejercicio 8: ⚽
Programe una clase llamada *ColaConPila* que implemente la interfaz Queue utilizando una pila.

Ejercicio 9: ⚽
Suponiendo que posee la implementación de los TDA Pila y TDACola (como se definen en [GT]) implemente, solamente en términos de las operaciones de los TDA Pila y Cola, un método que recibe una cola de caracteres Q y un caracter X y determina si Q respeta el siguiente formato: AXA'A, donde A es una cadena de caracteres formada por 1 o más caracteres y A' corresponde al inverso de A. Puede asumir que el carácter X pasado por parámetro no está presente en A.

Ejercicio 10: ⚽
Sean Cin1 y Cin2 **dos colas de pilas** de caracteres cuyos elementos se encuentran ordenados de menor a mayor; se desea crear una nueva cola Cout que sea la unión de los elementos de Cin1 y Cin2. Los elementos de Cout deben quedar ordenados de menor a mayor. Escriba un procedimiento en Java que reciba dos colas (Cin1 y Cin2) y resuelva el problema planteado exclusivamente en términos de las operaciones de los TDA Pila, TDA Cola y clase Character. Se considera que una pila P1 es menor que otra pila P2 cuando P1 tiene menos elementos que P2.

a) Calcule el orden del tiempo de ejecución de su solución.
b) Resuelva nuevamente el inciso (a) pero ahora sin poder utilizar la operación Size() de ninguno de los TDA Pila y Cola. ¿Cómo afecta esto al orden del tiempo de ejecución de su nueva solución?

Ejercicio 11: ⚽
Escriba un método que reciba una pila de pilas de enteros PPE y devuelva una pila de enteros PE donde el contenido de PE es el contenido de PPE en el mismo orden en que se hallan almacenados en PE. Es decir, si

$$PPE = < <e^1_1, \ldots, e^1_{n1}>, <e^2_1, \ldots, e^2_{n2}>, \ldots., <e^m_1, \ldots, e^m_{nm}>>, \text{ entonces:}$$

$$PE = < e^1_1, \ldots, e^1_{n1}, e^2_1, \ldots, e^2_{n2}, \ldots., e^m_1, \ldots, e^m_{nm} >$$

a) Calcule el orden del tiempo de ejecución de su solución.

Ejercicio 12: ⚽
Implemente un método que reciba dos **pilas de colas de enteros** P_1 y P_2, y retorne una nueva pila de colas P_{out} que sea la unión de los elementos de P_1 y P_2. Cabe aclarar que los elementos de P_1 y P_2 están ordenados de menor a mayor en función de su tamaño, y que la pila P_{out} debe quedar ordenada del mismo modo. En el tope de las pilas se encuentra el elemento de mayor tamaño. Resuelva el problema planteado exclusivamente en términos de las operaciones de los TDA Pila, TDACola y clase Integer.

Ejercicio 13: ⚽
Dado el conjunto de símbolos *{a,b}*, llamaremos *{a,b}** al conjunto de todas las cadenas constituidas por 0 o más símbolos de *{a,b}*. Ejemplo las cadenas *aaab, b, a, bababb* y *bbbb* pertenecen a *{a,b}**. Escriba un programa que tome como entrada una cadena de caracteres y determine si es de la forma: $A_1 \, x \, A_1' \, x \, A_2 \, x \, A_2' \, x \, \ldots\ldots\ldots \, x \, A_n \, x \, A_n'$
donde:

- el metasímbolo A_i representa una cadena de la forma C z CC, con C perteneciente a {a,b}*, es decir, si C = abb entonces, CzCC = abbzabbabb.
- el metasímbolo A_i' representa la cadena Ai invertida, es decir si A_1 = bzbb entonces A_1' = bbzb.

Los símbolos x y z son parte del lenguaje y son considerados como separadores.
Ejemplos de cadenas válidas: bzbbxbbzb - azaaxaaza - abzababxbabazba - bzbbxbbzbxazaaxaaza - aaabzaaabaaabxbaaabaaazbaaaxabzababxbabazba