ESTRUCTURAS DE DATOS

TRABAJO PRÁCTICO N ° 4

Listas e Iteradores

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.
[W] Mark A. Weiss. *Estructuras de datos*. Addison-Wesley Iberoamericana España, S.A. 2000.
[W12] Mark A. Weiss. *Data Structures and Algorithm Anlysis in Java*. Third Edition. Addison-Wesley Pearson Education, Inc. 2012.

En la página web de la materia se encuentra disponible el archivo TdaLista.jar provisto por la práctica.
Esta implementación puede utilizarse para resolver los ejercicios marcados con ✪.

Ejercicio 1:

a) Defina el TDA Lista, es decir, indique el modelo matemático sobre el cual está basado, y un conjunto de operaciones, con sus respectivas descripciones, para manejar datos de este tipo. Hacerlo siguiendo la propuesta de [GT].
b) Defina el concepto de posición para el caso de una lista enlazada.

Ejercicio 2: LISTA SIMPLEMENTE ENLAZADA

a) ¿Qué estructuras de datos podrían usarse para implementar el TDA Lista? Analice las ventajas y desventajas de cada una de ellas.
b) Defina en Java la clase Nodo que permite enlazar nodos en forma simple hacia adelante y cuyos elementos son de tipo genérico E.
c) Implemente en Java el TDA Lista utilizando una estructura simplemente enlazada (se recomienda usar la clase Nodo definida en el inciso (b)). Analice los tiempos de ejecución de las operaciones.
d) Escriba un programa de prueba en Java, para verificar el correcto funcionamiento de los TDA implementados.

Ejercicio 3:

a) Utilizando la implementación del TDA Lista del ejercicio (2), resuelva el siguiente problema implementándolo en Java. Dadas dos listas de enteros ordenadas L1 y L2, se desea obtener una tercera lista ordenada U (la intercalación de L1 y L2) que contenga los elementos de L1 y L2 (quedando sin elementos repetidos). ✪
b) Escriba un programa que tenga como entrada L1 y L2 y muestre la lista resultante de la intercalación de ambas. Analice además el tiempo de ejecución del programa, en términos de las longitudes de las listas. ✪
c) Implemente un iterador para la lista implementada en el ejercicio (2).
d) Resuelva nuevamente el ejercicio (3.b) pero ahora utilizando el iterador definido en el inciso (c).
e) Resuelva nuevamente el mismo ejercicio, pero ahora implementando un iterador de posiciones.

Ejercicio 4:

El siguiente método se hizo con el propósito de eliminar todas las apariciones de un elemento x de una lista L. Explique por qué no siempre funciona y sugiera una manera de arreglarlo para que realice la tarea propuesta:
Versión 1:

```
public static void Elimina_apariciones( Integer x, Lista<Integer> l )
{
        Position <Integer> p;
        Integer i;
        p = l.first();
        while( p != l.last() ) {
          if(  p.element().equals(x) )
                i = l.remove(p);
          p = l.next(p);
        }
}
```

Versión 2:

```
public static void Elimina_apariciones( Integer x, List<Integer> l )
{
    Position p, a;
    p = l.first();
    while( p != l.last() )
    {
        if( p.element().equals( x ) )
        {
           a = p;
           p = l.next( p );
           l.remove( a );
        }
        a = p;
        p = l.next( p );
    }
}
```

Ejercicio 5:

a) Escriba en Java un método no recursivo INVERTIR(L) que, dada una lista L, devuelve a L con sus elementos en el orden inverso al que estaban. Para realizar esta operación deberá utilizar ÚNICAMENTE las operaciones del TDA Lista visto en clase. Puede utilizar si fuera necesario una o varias listas auxiliares. Determine el orden del tiempo de ejecución de la solución dada.
b) Escriba en Java un método recursivo INVERTIR-REC(L) para realizar la misma tarea que la del inciso anterior, utilizando ÚNICAMENTE las operaciones del TDA Lista. La diferencia es que ahora NO puede utilizar ninguna estructura auxiliar. Determine el orden del tiempo de ejecución de su solución.
c) Agregue una operación al TDA Lista visto en clase para invertir la lista. Para implementarlo, asuma que la lista se representa con una lista simplemente enlazada y sin celda de encabezamiento. Calcule el orden del tiempo de ejecución de la solución que propone.

Ejercicio 6: LISTA DOBLEMENTE ENLAZADA

a) Implemente el TDA Lista utilizando una estructura de nodos doblemente enlazada. Ayuda: complete la implementación presentada en [GT]; para ello puede utilizar las porciones de código presentadas en la sección 6.2.4.
b) Analice el orden de los tiempos de ejecución de las operaciones de este TDA. Compárelos con los obtenidos para el TDA Lista con una implementación de enlaces simples hacia adelante.
c) Indique las ventajas y desventajas del doble enlace.

Ejercicio 7:

Realice un programa que mantenga una lista de nombres de alumnos, y para cada uno de ellos, una lista con los números de código de las materias que ha cursado y otra lista con los códigos de las materias que ha aprobado. El programa deberá permitir la carga de nuevas materias cursadas y aprobadas, verificando que no estén ya cargadas en la lista. En el caso de las materias aprobadas deberá verificar además que estas hayan sido cursadas.

a) Realice su solución utilizando el TDA visto en clase.
b) Realice una solución alternativa utilizando el tipo ArrayList provisto por Java.

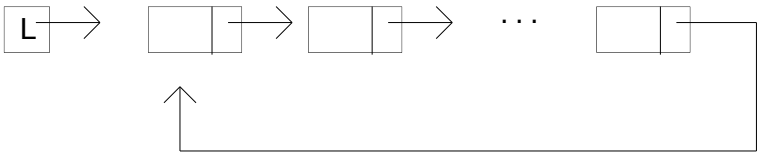
Ejercicio 8:

Se desea escribir un método *Generar_Lista_Resumen(L1,L2,R)* que reciba dos listas L1 y L2 (del mismo tipo elemento), y genere la lista R cuyos elementos serán ternas de la forma:
(elemento E, cantidad de apariciones de E en L1, cantidad de apariciones de E en L2)
Ejemplo:
Sean L1 = <a,b,a,c,d,b> y L2 = <c,a,b,f,c> el comando deberá generar la lista
R=<(a, 2, 1), (b, 2, 1), (c, 1, 2), (d, 1, 0), (f, 0, 1)>

a) Implemente en Java la solución al problema utilizando el TDA Lista definido en clase. Defina una clase auxiliar para representar las ternas que serán los elementos de la lista resultado.
b) Indique el orden del tiempo de ejecución del comando anterior. ¿Sería el mismo si las listas L1 y L2 estuvieran ordenadas? Justifique su respuesta.

Ejercicio 9:
Escriba utilizando únicamente las operaciones provistas por el TDA Lista un método en Java que teniendo como entrada dos listas L1 y L2, devuelva verdadero si L1 está compuesta por los elementos de L2 en el mismo orden, seguidos de los elementos de L2 en orden inverso. Determine el orden del tiempo de ejecución de su solución.
Ejemplo:
Si L1 = <1,2,3,4,4,3,2,1> y L2 = <1, 2, 3, 4> entonces el método deberá devolver verdadero.
Si L1 = <1,2,3,4,4,3,2,1,5> y L2 = <1, 2, 3, 4> entonces el método deberá devolver falso

Ejercicio 10: LISTA CIRCULAR
El campo "de enlace" de la última celda de una lista simplemente enlazada, siempre contiene el valor null. En algunos casos conviene hacer un uso más provechoso de este campo. Por ejemplo, hacer que apunte a la primera celda de la lista. Con esta modificación, la estructura resultante se llama lista circular (o lista circularmente enlazada) y su representación gráfica es la siguiente:



a) Analice las ventajas y desventajas si la anterior lista se tratara de una lista con celda de encabezamiento.
b) Implemente el TDA Lista cuya estructura de datos sea circular con una estructura enlazada. Para ello puede estudiar las porciones de código presentadas en http://java.datastructures.net, enlace Code Fragments, Chapter 6. Considere especialmente cómo modificar las operaciones prev y next en este contexto donde la siguiente de la última posición será la primera y la anterior de la primera será la última.
c) Escriba un método que reciba una lista circular y un número entero n. El método debe solicitar la primera posición de la lista y avanzar n posiciones, luego eliminar este elemento. Avanzar desde allí n posiciones y eliminar el elemento alcanzado. Así seguir sucesivamente hasta que la lista sólo esté conformada por un único elemento.

Ejercicio 11:
Se sabe que el método clone() de la clase Object realiza un clonado superficial (en inglés shallow clone); se desea poseer un método para lograr un clonado en profundidad (en inglés deep clone). El clonado superficial no copia la estructura de nodos asociados a una estructura de datos. En cambio, el clonado en profundidad sí lo hace. Dada la clase que implementa el TDA PositionList usando una estructura de nodos simplemente enlazados, *redefina* el método clone() heredado de la clase Object para que ahora permita generar un clonado en profundidad de la lista receptora del mensaje. Determine el orden del tiempo de ejecución de la implementación propuesta.

Ejercicio 12:
Implemente en Java una operación para la clase lista doblemente enlazada definida en el ejercicio 7 (a) que reciba dos enteros A y B y que modifique la lista L receptora del mensaje de acuerdo a: Eliminar de la lista L todas las apariciones de los elementos A y B ubicados consecutivamente en L. Recuerde que está agregando un método a la clase Lista, por lo tanto, tiene total acceso a su estructura. Esta operación no debe generar una nueva lista, sino que debe modificar la lista A receptora del mensaje.

Ejercicio 13:
Implemente en Java una operación para la clase lista definida en el inciso 6(a) que, reciba una lista B del mismo tipo, y que modifique la lista A receptora del mensaje de acuerdo a: eliminar de la lista A todas las apariciones de aquellos elementos que se encuentren más veces en la lista B que en la lista A. Recuerde que está agregando un método a la

clase Lista, por lo tanto, tiene total acceso a la estructura de la lista receptora del mensaje. Esta operación no debe generar una nueva lista, sino que debe modificar la lista A receptora del mensaje.

Ejercicio 14:

Agregue un método eliminar(PositionList<E> L) a la clase Lista, que modifique la lista que recibe el mensaje de la siguiente manera: primero deberá eliminar de la misma todas las apariciones de los elementos contenidos en L, luego deberá insertar al final de la misma todos los elementos de L pero en orden inverso al que aparecen en L. Puede considerar que L no tiene elementos repetidos.
Considere dos implementaciones posibles: (a) accediendo a la estructura de nodos de la lista receptora, (b) manipulando la lista receptora con operaciones del TDA.

Ejercicio 15:

Implemente un método que, dada una lista de caracteres, determine si la misma cumple con la forma AxA'A, donde:

- A representa una cadena perteneciente a {a,b}*
- A' representa la cadena A invertida, es decir si A = baa entonces A' = aab.
- El caracter x es considerado como separador.

Ejemplo: la cadena "baaxaabbaa" cumple con la condición mientras las cadenas "baaxaabbab" y "gaaxgaaaag" no cumplen la condición.

Puede utilizar las operaciones de los TDALista, TDACola y TDAPila sin implementarlas, pero sí deberá implementar los métodos auxiliares utilizados.