Homework 2015

Juana Paulina Águila Hernández

Características del equipo en donde se hicieron las pruebas:

* **Sistema operativo:** OS X Yosemite (10.10.4)
* **Procesador:** 2.5 GHz Intel Core i5
* **Memoria:** 4 GB 1600 MHz DDR3
* **Versión de Java:** 1.6.0\_65-b14-466.1-11M4716
* **Memoria disponible:** 81 MB (Información proporcionada por la clase *Runtime*).
* **Memoria máxima disponible:** 123 MB (Información proporcionada por la clase *Runtime*).

Archivo donde se ejecutaron las pruebas: package/Principal/TestZone.java

Repositorio en github: <https://github.com/vegenisennawa/Homework2015.git>

El tiempo para las pruebas está en milisegundos.

Ejercicios:

1. Implementar la siguiente recursión:
   1. Dado un elemento de la clase *String* de Java, se necesita codificar una función recursiva que cuente todos los caracteres en un objeto *String*.

Este ejercicio se puede ver dentro de la clase RecSring [package/MyString/RecString.java]

Se tomó en cuenta un índice que se ingresa a la función *StrLen* y la cadena a medir.

Si el iterador ingresado es menor a la longitud de la cadena, entonces va llamando a la función recursivamente hasta que no se cumpla esa condición, regresando el nuevo iterador.

Si el iterador es mayor a la longitud de la cadena, la función regresa -1 para indicar un error.

* 1. Dé la recursión calculando T(n).

*Cálculo para cuando la cadena tiene longitud 0.*

**Código** **Pasos**

StrLen(index, cadena)

if(index < cadena.length()) 1

return *StrLen*(++index, cadena); 0

else if(index > cadena.length()) 0

index = -1; 0

return index; 1

Sumando: 1 + 1 = 2.

*Cálculo para cuando la cadena tiene longitud 1.*

**Código** **Pasos**

StrLen(index, cadena)

if(index < cadena.length()) 1

return *StrLen*(++index, cadena); 1

else if(index > cadena.length()) 0

index = -1; 0

return index; 1

Sumando: 1 + 1 + 1 = 3.

*Cálculo para cuando la cadena tiene longitud mayor a la longitud de la cadena*

**Código** **Pasos**

StrLen(index, cadena)

if(index < cadena.length()) 1

return *StrLen*(++index, cadena); 0

else if(index > cadena.length()) 1

index = -1; 1

return index; 1

Sumando: 1 + 1 + 1 +1 = 4

*Cálculo para cuando la cadena tiene longitud = 0 y longitud = n.*

**Código** **Pasos**

StrLen(index, cadena)

if(index < cadena.length()) 1

return *StrLen*(++index, cadena); n-1

else if(index > cadena.length()) 0

index = -1; 0

return index; 1

Sumando: 1 + n - 1 +1 = 1 + n

* 1. Pruebe con una muestra suficientemente grande.

Se tomó como referencia a la cadena a, aumentando su longitud dependiendo de lo que se requería.

Para el tiempo, se tomó el promedio de 10 ejecuciones con cada longitud.

|  |  |
| --- | --- |
| **Longitud de la prueba** | **Tiempo** |
| 1 x 100 | 0.1 |
| 1 x 101 | 0.0 |
| 1 x 102 | 0.1 |
| 1 x 103 | 0.2 |
| 1 x 104 | java.lang.StackOverflowError |

1. Implemente la representación de la cadena y usando ideas de la clase, necesitas probar el tiempo promedio para cada método, add y remove, sobre la representación de la cadena usando un tiempo adecuado y un número random adecuado de integers.

El tiempo escrito en la tabla representa el promedio de una ejecución de la definición de la lista con cada tamaño n de la muestra.

| **n** | **Tiempo** |
| --- | --- |
| 1 x 100 | 1.0 |
| 1 x 101 | 13.0 |
| 1 x 102 | 1.0 |
| 1 x 103 | 1.0 |
| 1 x 104 | 1.0 |
| 1 x 105 | 0.0 |
| 1 x 106 | 0.0 |

El tiempo escrito en la tabla representa el promedio de n inserciones en la lista.

| **n** | **Tiempo** |
| --- | --- |
| 1 x 100 | 0.0 |
| 1 x 101 | 1.7 |
| 1 x 102 | 0.01 |
| 1 x 103 | 0.029 |
| 1 x 104 | 0.0136 |
| 1 x 105 | 0.11056 |
| 1 x 106 | Se detuvo la prueba |

El tiempo escrito en la tabla representa el promedio de un rango de intentos de eliminación al hacer eliminaciones en la lista.

Cuando n va de 1 a 10, se toma ese número de intentos de eliminación.

Si n es mayor a 10, entonces se toma el numero resultante de dividir n entre 10.

| **n elementos en la lista** | **Tiempo** |
| --- | --- |
| 1 x 100 | 0.0 |
| 1 x 101 | 0.0 |
| 1 x 102 | 0.0 |
| 1 x 103 | 2.0 |
| 1 x 104 | 27.0 |
| 1 x 105 | 2447.0 |
| 1 x 106 | Se detuvo la prueba |

1. Imagina que tienes una representación de array de arrays de una matriz triangular inferior, por favor desarrolle una clase para este tipo de matrices con los siguientes elementos y métodos [ver los elementos y métodos en la clase].

El tiempo escrito en la tabla representa el promedio de una ejecución de la definición de la matriz con cada tamaño n de la muestra.

| **n** | **Tiempo** |
| --- | --- |
| 1 x 100 |  |
| 1 x 101 |  |
| 1 x 102 |  |
| 1 x 103 |  |
| 1 x 104 |  |

El tiempo escrito en la tabla representa el promedio de 10 ejecuciones al colocar datos de la matriz con cada tamaño n de la muestra.

| **n** | **Tiempo** |
| --- | --- |
| 1 x 100 |  |
| 1 x 101 |  |
| 1 x 102 |  |
| 1 x 103 |  |
| 1 x 104 |  |

El tiempo escrito en la tabla representa el promedio de 10 ejecuciones al obtener datos de la matriz con cada tamaño n de la muestra.

| **n** | **Tiempo** |
| --- | --- |
| 1 x 100 |  |
| 1 x 101 |  |
| 1 x 102 |  |
| 1 x 103 |  |
| 1 x 104 |  |

El tiempo escrito en la tabla representa el promedio de 10 ejecuciones al eliminar datos de la matriz con cada tamaño n de la muestra.

| **n** | **Tiempo** |
| --- | --- |
| 1 x 100 |  |
| 1 x 101 |  |
| 1 x 102 |  |
| 1 x 103 |  |
| 1 x 104 |  |

El tiempo escrito en la tabla representa el promedio de 10 ejecuciones al mostrar datos de la matriz con cada tamaño n de la muestra.

| **n** | **Tiempo** |
| --- | --- |
| 1 x 100 |  |
| 1 x 101 |  |
| 1 x 102 |  |
| 1 x 103 |  |
| 1 x 104 |  |

1. Cuatro
2. Cinco