Laboratorio Nro. 3
Listas Enlazadas y Vectores Dinámicos

Alejandro Villada Toro
Universidad Eafit
Medellín, Colombia
avilladat@eafit.edu.co

Cristian Alzate Urrea
Universidad Eafit
Medellín, Colombia
calzateu@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

3.1

Punto 1.1 métodos:
Leer vértices O(n^2), donde n es el número de líneas del archivo .txt.
Leer arcos O(n^2), donde n es el número de líneas del archivo .txt.
La complejidad de getSuccessors() es $O(n^2)$, donde n es la cantidad de grafos que contiene la Estructura de datos.
La complejidad de getWeight() es O(n), donde n es la cantidad de grafos que contiene la Estructura de datos.

Punto 1.2 métodos:
La complejidad de consultarPorCurso(String curso, String año_semestre) es O(n + m*z), donde n es el número de semestres que hay registrados, m es el número de estudiantes en el semestre buscado y z es el número de materias que tiene el estudiante analizado en dicho semestre.
La complejidad de consultarPorEstudiante(String estudiante, String año_semestre) es O(n + m*z), donde n es el número de semestres que hay registrados, m es el número de estudiantes en el semestre buscado y z es el número de materias que tiene el estudiante buscado.
La complejidad de agregarEstudianteAlaBaseDeDatos(String año_semestre, Estudiante estudiante) es O(n*m), donde n es el número de semestres y m son las materias del estudiante en el semestre requerido.
La complejidad de agregarMateriaAUnEstudiante(String año_semestre, String estudiante, Materia curso) es O(n*m*z), donde n es el número de semestres, m es el número de estudiantes en el semestre y z es el número de materias que posee el alumno requerido en el semestre.

Punto 1.3 métodos:
La complejidad de staticBalance() es $O(n^3)$, donde n es la longitud de la lista recibida como parámetro, “la viga”. Esto es así porque hay dos ciclos anidados y el metodo get en una lista enlazada tiene complejidad O(n).

Punto 1.4 métodos:

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

La complejidad de assigned() es O(n*m), donde n es la cantidad de neveras y m es la cantidad de solicitudes. Esto es así porque hay dos ciclos anidados y los métodos de añadir y remover en una pila y en una cola se hacen en tiempo constante.

Punto 1.5 métodos:
La complejidad del metodo de insertar un dato al principio o al final de la lista es O(1).
La complejidad del metodo de insertar un dato en una posición arbitraria es O(n), donde n es la longitud de la lista.
La complejidad del metodo de borrar un dato al principio o al final de la lista es O(1).
La complejidad del metodo de borrar un dato en una posición arbitraria es O(n), donde n es la longitud de la lista.
La complejidad del metodo de verificar si un dato esta en la lista es O(n).

3.2

El problema del teclado roto

El problema del teclado roto consiste en que estás escribiendo un texto largo con un teclado roto. El único problema es que algunas veces la tecla “Inicio” o la tecla “Fin” se presionan solas. Usted no es consciente de este problema, ya que está concentrado en el texto y ni siquiera mira el monitor. Luego de que usted termina de escribir puede ver un texto en la pantalla.
Lo que se busca es crear un metodo que permita ver como quedaría texto, considerando el problema anterior; se da una cadena de caracteres, donde aparezca “[“ significa que se ha presionado la tecla “inicio”, y todos los caracteres a partir de ese punto deben ingresarse en el inicio de la cadena de caracteres, por otro lado, donde aparezca “]” significa que se ha presionado la tecla “fin”, y todos los caracteres a partir de ese punto deben ingresarse en el fin de la cadena de caracteres.
El metodo *brokenKeyBoard* da solución a este problema, recibe como parámetro un String *text*, que es al que se le realizara el procedimiento; su funcionamiento es el siguiente:
Primero inicializa una lista doblemente enlazada *list* donde se guardará el texto tomando el problema expuesto, luego inicializa un entero *i* en cero, que nos servirá como iterador. Después mediante un ciclo que se repetirá la longitud del texto, repasaremos cada carácter de este. Dentro del ciclo, el algoritmo se pregunta, si el carácter en la posición *i* es igual a “[“se inicializa un String *wrongposition* con cero caracteres, luego se inicializa un entero *j* como *i+1*. Posteriormente mediante otro ciclo que se repetirá siempre que *j* sea menor que la cantidad de caracteres del texto y que el carácter en la posición *j* sea distinto a “[“o “]”, dentro de este ciclo se le ira sumando a *wrongposition* el carácter en la posición *j* y se le suma 1 a *j*. Terminado el ciclo se le añade al principio de *list* el String *wrongposition* y el iterador *i* se vuelve igual a *j*. Por otro lado, si el carácter en la posición *i* es igual a “]” se inicializa un entero *j* como *i+1*. A continuación mediante otro ciclo que se repetirá siempre que *j* sea menor que la cantidad de caracteres del texto y que el carácter en la posición *j* sea distinto a “[“o “]”, dentro de este ciclo se le añade al final de *list* el carácter en la posición *j* de *text* y se le suma 1 a *j*. Terminado el ciclo el iterador *i* se vuelve igual a *j*. Finalmente, si no se cumplen ninguna de las condiciones anteriores, simplemente se añade al final de *list* el carácter de en la posicon *i* de *text* y se le suma 1 a *i.* Al realizar este procedimiento con cada uno de los caracteres de *text,* el metodo retorna *list.*

3.3

Complejidad de el algoritmo que resuelve el problema del teclado roto.

El metodo charAt() de la clase String es en tiempo contante.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Como son dos ciclos anidados que dependen de la longitud del String *text,* la complejidad es $O(n^2)$ con n igual a longitud del String *t*ext.

3.4

Las variables *n* y *m* son aquellas con las que se representa y entiende el tamaño del problema, es decir, son las variables que aumentan o disminuyen la complejidad del algoritmo ya sea en tiempo o en memoria.

Las n y m, son especificadas y aclaradas en cada uno de los puntos de cálculo de complejidad.

4) Simulacro de Parcial

4.1.
- ***4.1.1. b***
- ***4.1.2. b***

4.2. c

4.3. Pendiente

4.4.
- ***4.4.1. token***
- ***4.4.1. c***

4.5. a

4.6. a

4.7. Faltaaaaaaaaaaa

4.8. d

4.9.
- ***4.9.1. a***
- ***4.9.2. b***
- ***4.9.3. c***

4.10.
- ***4.10.1. d***
- ***4.10.2. a***
- ***4.10.3. b***

4.11.
- ***4.11.1. c***
- ***4.11.2. b***

4.12.
- ***4.12.1. !s1.isEmpty()***
- ***4.12.2. s1.pop()***
- ***4.12.3. s2.pop()***

4.13.
- ***4.13.1. IV***
- ***4.13.2. I***

4.14.
- ***4.14.1. III***
- ***4.14.2. IV***

4.15. III

5) Lectura recomendada (opcional)

Mapa conceptual

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

6) Trabajo en Equipo y Progreso Gradual (Opcional)

Integrante	Fecha	Hecho	Haciendo	Por hacer
Cristian	19/09/2020	Primer vistazo del laboratorio		Realización del laboratorio
Alejandro	19/09/2020	Primer vistazo del laboratorio		Realización del laboratorio
Cristian	22/09/2020	Realización del simulacro de parcial	Resolviendo las preguntas entre los dos	Implementación de una estructura de datos que lea el mapa de una ciudad.
Alejandro	22/09/2020	Realización del simulacro de parcial	Resolviendo las preguntas entre los dos	Implementación del metodo para solución del problema del teclado roto
Alejandro	25/09/2020	Implementación del metodo que soluciona el problema del teclado roto	Escribiendo el código en el lenguaje java	Puntos opcionales del punto 1
Cristian	26/09/2020	Implementación de la estructura de datos que lee el mapa de una ciudad	Escribiendo el código en el lenguaje java	Puntos opcionales del punto 1
Cristian	27/09/2020	Diseño de una estructura de datos para almacenar las notas de cada estudiante en una universidad	Escribiendo el código entre los dos y redactando el informe	Hacer la lectura cada uno
Alejandro	27/09/2020	Implementación de los algoritmos que resuelven puntos opcionales del punto 1	Escribiendo el código entre los dos y redactando el informe	Hacer la lectura cada uno

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473