Instituto Tecnológico de Costa Rica - ITCR
Escuela de Ingeniería en Computación
Algoritmos y Estructuras de Datos I (CE-1103)
Primer Semestre 2015
Profesor: Kevin Moraga García
Segundo Programado

1. **Objetivo General**
   - Diseñar e Implementar la solución de un problema mediante el uso de estructuras de datos Jerárquicas.
2. **Objetivos Específicos**
   - Implementar las diferentes estructuras de datos Jerárquicas.
   - Desarrollar algoritmos para dar solución a un problema.
   - Medir el rendimiento de las estructuras en sus operaciones más comunes y algoritmos implementados.
   - Diseñar la solución de un problema utilizando estructuras de datos lineales y Jerárquicas.
   - Investigar y aplicar el patrón productor consumidor en la resolución de un problema.
   - Implementar un mecanismo para manipular las estructuras de datos implementadas.
   - Fomentar la creatividad mediante el análisis y el diseño de algoritmos y estructuras de datos.
   - Utilizar diagramas de clases UML para modelar una solución a un problema.
   - Investigar acerca de expresiones regulares.
   - Investigar acerca de cómo procesar información textual.
   - Investigar acerca de sincronización de hilos en Java.
   - Investigar acerca de Cliente/Servidor
3. **Datos Generales**
   - El proyecto tiene un valor de 25 %
   - La tarea debe ser implementada parejas
   - El nombre código del proyecto es **SpiderSearch Engine: Stage 2**
   - La fecha de entrega es 05/05/15, fecha de revisión 09/05/15
   - Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo al reglamento.
   - Cualquier duda sobre la tarea será tratada en clase y cualquier cambio a esta deberá hacerse bajo el consentimiento de todos los alumnos.

4. **Descripción del Problema**

En el proyecto anterior desarrolló un motor de búsqueda que extraía la información de páginas web, en el mismo se procesaban una serie de URLs, se conformaba el índice y se procedía a realizar consultas sobre esta información generada. En el presente trabajo se presenta un problema similar hacer un motor de búsqueda pero en este caso vamos a desarrollar uno que extraiga la información de documentos que se encuentren en nuestras computadoras y en computadoras remotas (Web Sites o carpetas compartidas).

En esta etapa vamos a realizar la extracción de datos de documentos de tipo PDF, docx, doc, txt y odt, únicamente, para realizar la extracción del texto les recomiendo utilizar la biblioteca Apache Tika o pueden usar cualquier otra de su preferencia.

Las estructuras a desarrollar son:

- Arboles Binarios de búsqueda
- Heap Sort
- Arboles AVL
- Arboles Splay

- Arboles Roji-Negros
- Arboles B (opcional)

El funcionamiento de esta etapa es muy similar al anterior mas no es necesario tener el primer proyecto listo para poder continuar con este, básicamente en el proyecto anterior pasábamos las URL como un XML, en este caso vamos a distinguir tres tipos de URL la primera sería una URL de la forma HTTP://cs.wellesley.edu/~cs231/fall01/red-black.pdf, la misma corresponde a un documento en Internet, la segunda corresponde a un dirección en una carpeta compartida de Windows o Linux seria de la forma \\<IP|Nombre>\Carpeta o smb:\\<IP|nombre>\Carpeta y por ultimo tenemos un path normal que es de la forma \tmp (este tipo se expande buscando documentos recursivamente dentro de todo el directorio), estas URL estarán almacenadas en el archivo XML en el cual será alimentado el SpiderBot, cada URL además tendrá un peso asociado el mismo es un numero entre 1 y 1000.

El **SpiderBot** será alimentado con este nuevo XML (en caliente se le pueden agregar más XML con mas URL), las documentos serán insertadas en un Heap de mayores de acuerdo al peso. El SpiderBot recibe una configuración igual a la del proyecto anterior y su funcionamiento es igual, tiene hilos que van consumiendo los documentos, pero existe una pequeña diferencia la cual radica en que si llega a la cola un documento con un peso mucho más alto, el hilo que tiene el documento con peso más bajo debe dejar el trabajo sin terminar (lo pausa), lo agrega a la cola de nuevo y toma el que tenga mayor prioridad.

El procesamiento consiste en extraer el texto de los documentos y agregar las palabras a una estructura, en este caso se tratara de un árbol AVL con una pequeña modificación en su funcionamiento, el mismo deberá estar ordenado de acuerdo a la cantidad de veces que estas palabras han aparecido, por lo cual durante el procesamiento el orden de un nodo puede cambiar lo que hace que este quede en una posición inválida.

Por otro lado se tiene un árbol rojinegro donde se almacenan las direcciones de los documentos y la cantidad de palabras que estos contienen, los nodos de estos árboles son referidos por los nodos del AVL cuando se detecta que la palabra que contiene el nodo se encuentra contenida en el documento, además se lleva un contador de la cantidad de veces que aparece esta palabra en el documento.

En cualquier momento se puede indicar al SpiderBot que genere un índice, un índice básicamente es un archivo XML (el formato será definido por los estudiantes) que se guardara en disco.

SpiderEngine es nuestro motor de búsqueda, la entrada de este es el índice generado por el SpiderBot, al cargar las estructuras creadas por el SpiderBot estas deben ser optimizadas para realizar búsquedas sobre las mismas, por esta razón queda a criterio del estudiante como organizar los datos que se van a cargar de estas estructuras de datos.

Nuestro algoritmo de búsqueda es muy sencillo, se presenta un cuadro de texto donde el usuario escribirá el texto que desea buscar, se ignora todo lo que sean signos de puntuación y espacios (esto también aplica para el SpiderBot), una vez que estos datos son ingresados y el usuario presiona el botón Search entonces este busca todas las palabras en la lista de Keywords, cuando se encuentran estas palabras, cada keyword tiene un apuntador a las direcciones de los documentos que las contiene entonces estas son recuperadas.

La forma mediante la cual se indica en qué orden se van a mostrar los documentos (ranking) queda a criterio de los estudiantes.

El resultado de la búsqueda es un árbol binario de búsqueda que contiene las direcciones ordenadas de acuerdo a la función definida por los estudiantes.

Cuando el usuario de clic en alguno de los resultados se abrirá el navegador web en documento deseado o si el resultado no corresponde a una URL de la forma HTTP:// deberá abrir el programa indicado para abrir el tipo de archivos correspondiente al documento, además a cada resultado (por defecto es 0) el usuario podrá darle una puntuación de acuerdo a que tan bueno fue la puntuación puede ser de -5 a 5, esta puntuación es llamada trustworthy.

Las búsquedas realizadas por los usuarios se almacenarán en un Árbol Splay.

El sistema deberá permitir múltiples búsquedas simultáneamente, básicamente el SpiderEngine deberá proporcionar un puerto al cual otros clientes de búsqueda se puedan conectar y realizar búsquedas.

5. Documentación

1. Se deberá documentar el código fuente utilizando el estándar Javadoc.
2. Se deberá entregar un documento que contenga:

a. Manual de usuario: cómo ejecutar el programa, incluir requerimientos de Hardware y Software
b. Descripción de todas las bibliotecas y funciones utilizadas.
c. Descripción de los métodos implementados.
d. Descripción de las estructuras de datos desarrolladas.
e. Descripción detallada de los algoritmos desarrollados.
f. Problemas conocidos: En está sección se detalla cualquier problema que no se ha podido solucionar en el trabajo.
g. Actividades realizadas por estudiante: Este es un resumen de las bitácoras de cada estudiante (estilo timesheet) en términos del tiempo invertido para una actividad específica que impactó directamente el desarrollo del trabajo, de manera breve (no más de una línea) se describe lo que se realizó, la cantidad de horas invertidas y la fecha en la que se realizó. Se deben sumar las horas invertidas por cada estudiante, sean concientes a la hora de realizar esto el profesor determinará si los reportes están acordes al producto entregado.
h. Problemas encontrados: descripción detallada, intentos de solución sin éxito, solución encontradas con su descripción detallada, recomendaciones, conclusiones y bibliografía consultada para este problema específico.
i. Conclusiones y Recomendaciones del proyecto.
j. Bibliografía consultada en todo el proyecto

3. Bitácora en digital o en papel, donde se describen las actividades realizadas, desde reuniones con el compañero de trabajo, investigaciones, consultas, etc. Está se puede encontrar hecha a mano, se debe describir todo por mas insignificante que sea, esto demostrará si ustedes están trabajando en realidad. Este es su diario de trabajo, llevan seguimiento de todo en el tiempo, imaginen que si un compañero los releva en su trabajo, le bastaría con leer sus bitácoras para seguir el trabajo.
4. Diagrama de clases y un documento que explique el porqué del diseño.
5. Se debe utilizar SVN o algún otro manejador de código fuente. Se recomienda Assembla. Por favor agregar las cuentas asisten.nereo@gmail.com o kevin+asistente@gmail.com a sus repositorios. Recuerden hacer comentarios en cada commit.

6. **Evaluación**:
    1. El spiderbot y el search engine tienen un valor de un 50% sobre la nota final
    2. La documentación ya sea para el juego o para el mando remoto tendrá un valor de entre un 20% y un 50% que será informado a los estudiantes el día de la revisión.
    3. Cada grupo recibirá una nota en cada uno de los siguientes apartados:
        a. Spiderbot: Código y Documentación
        b. SpiderEngine: Código y Documentación
    4. De las notas mencionadas en el punto anterior se calculará la Nota Final del Proyecto.
    5. Se debe enviar el código y la documentación a más tardar a las 23:59 del día de la fecha de entrega a la carpeta compartida del profesor. **El formato del email del subject del email es <Carnet>-<Carnet>-<Carnet>-<Carnet> - SpiderSearch, en el body del email se debe ingresar los carnet y nombres de cada uno de los estudiantes, si el proyecto es en grupos se debe copiar en el email a todos los miembros del grupo.**
    6. Las citas de revisión oficiales serán determinadas por el profesor durante las lecciones o mediante algún medio electrónico.
    7. Los estudiantes pueden seguir trabajando en el código (no la documentación) hasta 15 minutos antes de la cita revisión oficial, momento en el cuál deben enviar un email con el código fuente, si no lo hacen se asumirá que la versión oficial del código fue la enviada el día de la fecha de entrega junto con la documentación. **El formato del email del subject del email es <Carnet>-<Carnet>-<Carnet>-<Carnet> - Codigo SpiderSearch, en el body del email se debe ingresar los carnet y nombres de cada uno de los estudiantes, si el proyecto es en grupos se debe copiar en el email a todos los miembros del grupo.**
    8. Aún cuando el código y la documentación tienen sus notas por separado, se aplican las siguientes restricciones
        a. Sí no se entrega documentación, automáticamente se obtiene una nota de 0.
        b. Sí no se utiliza un manejador de código se obtiene una nota de 0.
        c. Sí no se entrega el punto 4 de la documentación se obtiene una nota de 0.
        d. Sí el código y la documentación no se entregan en la fecha indicada se obtiene una nota de 0.

e. Sí el código no compila se obtendrá una nota de 0, por lo cuál se recomienda realizar la defensa con un código funcional.
f. El código debe ser desarrollado en Java para GNU/Linux, en caso contrario se obtendrá una nota de 0.
g. Sí se utilizan bibliotecas QT para algo diferente a UI se obtendrá una nota de 0.
h. Sí no se siguen las reglas del formato de email se obtendrá una nota de 0.
i. Si utiliza clases estáticas (static) en un lugar diferente del main se obtendrá una nota de 0.
j. Si no corre en el laboratorio se obtendrá una nota de 0.
k. Si no compila con javac se obtendrá una nota de 0.
l. Si no corre con el comando java se obtendrá una nota de 0.
m. Se no entrega el punto 4 actualizado de la documentación se obtendrá una nota de 0.
n. Si no entrega el javadoc actualizado se obtendrá una nota de 0.
o. Si el último commit en el repositorio de datos es mayor a 1 día desde la fecha de entrega se obtendrá una nota de 0.

9. La revisión de la documentación será realizada por parte del profesor, no durante la defensa del proyecto. El único requerimiento que se consultará durante la defensa del proyecto es el diagrama de clases, documentación interna y la documentación en el manejador de código.
10. Cada grupo tendrá como máximo 20 minutos para exponer su trabajo al profesor y realizar la defensa de éste, es responsabilidad de los estudiantes mostrar todo el trabajo realizado, por lo cuál se recomienda tener todo listo antes de ingresar a la defensa.
11. Cada excepción o error que salga durante la ejecución del proyecto y que se considere debió haber sido contemplada durante el desarrollo del proyecto, se castigará con 2 puntos de la nota final del proyecto.
12. Cada grupo es responsable de llevar los equipos requeridos para la revisión, si no cuentan con estos deberan avisar al menos 2 días antes de la revisión a el profesor para coordinar el préstamo de estos.
13. Durante la revisión podrán participar asistentes, otros profesores y el coordinador del área.