|  |  |
| --- | --- |
|  | **2015** |
|  | Universidad de Chile, Facultad de Ciencias Físicas y Matemáticas, Departamento de Computación  Cristian Andrade Muñoz Javier Liberman Salazar |

|  |
| --- |
| **[Diseño y Análisis de algoritmos:**  **Diccionarios en**  **Memoria Secundaria]** |
| Análisis cuantitativo del desempeño de algoritmos y estructuras de almacenamiento en disco (BTree), de acuerdo a su tiempo de ejecución y comparaciones realizadas sobre textos dados. |

Resumen Ejecutivo:

Se asignó la construcción del código fuente para la implementación de algoritmos y estructuras de almacenamiento de texto en memoria secundaria, específicamente estructuras BTree, Hash Extendible y Hash Lineal (en 2 versiones, dependiendo de su método de expansión). Para esto, se utilizó el lenguaje Java, con implementación en Clases para los distintos algoritmos y estructuras, y empaquetamiento en JAR para ejecución headless en un computador de 64 bits, con procesador de 4 núcleos de 2.7GHz, con 32 Gb de RAM a 1600Mhz. Para almacenamiento externo se utilizó un disco de estado sólido (SSD) conectado mediante SATA 3, con bloques de memoria de 512 bytes.

Para los archivos de búsqueda, se utilizaron archivos de cadenas de ADN) generados a partir de código provisto en ftp://ftp.ncbi.nih.gov/genomes/, específicamente genoma humano, Conejo Europeo, Macaco Cangrejero y Perro, y ADN sintético (en su formato de caracteres c-g-a-t).

Los resultados del algoritmo utilizado indican que el algoritmo/estructura con mejor desempeño *over all* es el algoritmo XXXXX, mostrando tiempos de ejecución y comparaciones de entre un XX% y hasta un XX% más bajos que sus competidores en las pruebas más complejas.

Introducción:

La búsqueda en texto es una tarea del día a día en los tiempos modernos, donde es necesario aplicarla en situaciones desde un procesador de texto básico, hasta en procesos de análisis de datos de gran envergadura, en laboratorios, observaciones astronómicas, servicios de búsqueda de datos online, entre otras. Es necesario medir, en estos casos, el desempeño de los algoritmos utilizados, buscando con ello el más adecuado en cuanto al tiempo necesario para ejecutarlo, la cantidad de datos revisados, y la dificultad de implementarlos.

Se indicó para esta tarea que se debía realizar la implementación de 3 algoritmos de búsqueda en texto, donde se debía especificar métodos para analizar y cuantificar su desempeño en distintas situaciones. En particular, estos algoritmos fueron:

Algoritmo de Fuerza Bruta:

Este algoritmo es el más básico de todos, consistente en comparar caracter por caracter el patrón con el texto de búsqueda, desplazando en 1 unidad cada vez que el patrón no calce.

Algoritmo Knuth-Morris-Pratt

Muy similar a Fuerza Bruta, realiza las comparaciones de izquierda a derecha, caracter por caracter con la diferencia que, para el desplazamiento del patrón sobre el texto, se utiliza una función de *fracaso* precalculada, la cual busca cuantificar los prefijos y sufijos concordantes que pudiera presentar el patrón, y sobre ellos maximizar el desplazamiento.

Algoritmo Boyer-Moore-Horspool

A diferencia de los anteriores, este algoritmo recorre el patrón de forma inversa, pero también busca maximizar el desplazamiento mediante el calce de la última letra comparada con su equivalente en el patrón, si existe, o desplazar el patrón completamente, en caso contrario.

Hipótesis:

* Si las condiciones de ejecución para los tres algoritmos son iguales y no dependen de los textos ni el patrón a buscar, entonces los resultados tendrán forma similar en todas las situaciones, y las variaciones que se produzcan van a ser el resultado del desempeño específico de los algoritmos frente a las características del lenguaje/patrón utilizado.
* El tamaño del alfabeto es uniforme para todas las ejecuciones, por lo que no representa una variable relevante al momento de comparar los algoritmos.
* Por otra parte, debería existir una variación entre las ejecuciones sobre ADN real y ADN sintético, debido a las limitaciones que presentan las bases nitrogenadas en su disposición en las cadenas, las cuales no se reflejan en la generación de archivos de genoma sintético.
* Consideramos B = 512 y N = n° de elementos en la simulación (entre 1.048.576 y 33.554.432 elementos).
* Cada string contiene 15 caracteres, los cuales, por implementación, se guardan directamente en el caso de BTrees, y de forma binaria en el caso de los hash (convirtiendo cada letra a una codificación de 2 bits). Esto entrega, para el segundo caso, un aproximado virtual de 128 strings por página en disco.
* A medida que el texto crece, tanto la inserción como la búsqueda sobre la estructura BTree debería tender a O(logB n), mientas que la búsqueda en Hash Extensible y Lineal Versión 1 debería ser constante, dependiendo su tiempo de ejecución sólo en la expansión de cada uno.
* Por otra parte, suponemos que las tasas de ocupación se mantendrán sobre el 50% en el caso de todas las estructuras. Sin embargo, experimentalmente se ha demostrado que la ocupación de Hash Extensible se mantiene (en promedio) en torno al 69%, lo cual será revisado con este experimento.

Diseño e Implementación:

Se utilizó el lenguaje Java, bajo el ambiente de desarrollo Eclipse. Se utilizaron metodologías de desarrollo tipo Template para la generación de los algoritmos de búsqueda, y se empaquetó el software en formato JAR para ser ejecutado en una máquina Intel de 64 bits, con 4 núcleos de 2.93Ghz y 6Gb de RAM.

Se realizaron pruebas para tamaños de patrón de texto entre 4 y 128 caracteres, con textos de aproximadamente 1Mb cada uno. Se iteró 50.000 veces cada experimento, con patrones elegidos aleatoriamente de acuerdo a las indicaciones (patrones binarios generados por código, patrones de texto extraídos de la muestra), de forma de reducir el error asociado a las muestras (de tiempo y de comparaciones). Se utilizó la librería Apache Commons Math (archive.apache.org/dist/commons/math/) para el cálculo automático de promedio, varianza y desviación estándar.

Los resultados se volcaron en un archivo de texto, el cual fue volcado manualmente en los gráficos mostrados a continuación. Se incluye error estándar en todos los gráficos.

Análisis

De acuerdo a los datos entregados por el código ejecutado, se tienen los siguientes resultados:

Texto Binario:

De acuerdo a estos resultados (tabla en Anexo), el desempeño entre Fuerza Bruta y Knuth-Morris-Pratt es muy similar en cuanto a comparaciones, y levemente mejor para el segundo en cuanto a tiempo de ejecución. Sin embargo, para ambas medidas, el algoritmo Boyer-Moore-Horspool tiene peor desempeño para patrones cortos, pero mejora levemente en comparaciones, y drásticamente en tiempo de ejecución, en cuanto aumenta el tamaño del patrón.

ADN real:

Con respecto al ADN real, se da cuenta que el algoritmo de Fuerza Bruta es relativamente constante en su accionar (tanto en comparaciones como en tiempo de ejecución), dando cuenta de su O(n\*m). Con respecto a Knuth-Morris-Pratt y Boyer-Moore-Horspool se da cuenta de rendimientos parecidos, gracias al crecimiento del tamaño del alfabeto (de 2 a 4 caracteres). Sin embargo, el segundo ocupa cerca de 3/5 del tiempo del primero (y menos de la mitad de Fuerza Bruta), aunque toma cerca de la misma cantidad de comparaciones. Esto supondría que calcular bajo necesidad la función de salto en BMH es más eficiente que tenerla pre-calculada como KMP.

ADN sintético:

Con respecto al ADN sintético, el comportamiento es muy similar al de ADN real. Se nota un aumento en el tiempo empleado y las comparaciones realizadas por los algoritmos, dado que este archivo tiene menos limitaciones en su configuración (presencia azarosa de los caracteres, sin las limitaciones químicas del ADN real). Aún así, se aprecia un acercamiento en los rendimientos de Knuth-Morris-Pratt y Boyer-Moore-Horspool, bajo las mismas razones expuestas anteriormente.

Texto Plano en alfabeto inglés:

Con respecto al desempeño en texto plano, las iteraciones en Fuerza Bruta sufren una caída importante en tiempo de ejecución y comparaciones, lo cual puede deberse a las características del texto (eventualmente más corto). Con respecto a Knuth-Morris-Pratt se nota un desempeño muy acorde a lo visto en las pruebas anteriores, con tiempo y comparaciones ascendentes de acuerdo al tamaño del patrón. Sin embargo, Boyer-Moore-Horspool muestra un desempeño convexo

Texto Plano sintético en alfabeto inglés: