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

|  |
| --- |
| **[Diseño y Análisis de algoritmos:**  **búsqueda en texto]** |
| Análisis cuantitativo del desempeño de algoritmos e Fuerza Bruta, Knuth-Morris-Pratt y Boyer-Moore-Horspool, 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 de búsqueda en texto, específicamente búsqueda por Fuerza Bruta, algoritmo de Knuth-Morris-Pratt, y algoritmo Boyer-Moore (Boyer-Moore-Horspool). Para esto, se utilizó el lenguaje Java, con implementación en Clases para los distintos algoritmos, y empaquetamiento en JAR para ejecución headless en un servidor de 64 bits, con procesador de 4 núcleos de 2.93GHz, con 6 Gb de RAM a 1333Mhz.

Para los archivos de búsqueda, se utilizaron archivos generados a partir de código provisto en <http://pizzachili.dcc.uchile.cl/texts.html>, específicamente el generador de datos binarios, archivos de código ADN real y sintético (en su formato de caracteres c-g-a-t), y archivos de texto real y sintético en alfabeto inglés alfanumérico.

Los resultados del algoritmo utilizado indican que el algoritmo con mejor desempeño *over all* es el algoritmo [[[Knuth-Morris-Pratt]]], 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:

Asumimos *n* el tamaño del texto, *m* el tamaño del patrón, y *c* el tamaño de cada alfabeto (binario, ADN, y texto alfanumérico en inglés, todos fijos). Asumimos que un alfabeto mayor implica mayores chances de desplazar el patrón en el texto por no calce, lo que lo lleva a hacer menos comparaciones. Asumimos también que los algoritmos recorrerán necesariamente la totalidad del texto, debido a que se solicita que se encuentren todas las ocurrencias del patrón en el texto.

En Fuerza Bruta, asumimos que la cantidad de comparaciones será siempre del orden de *n x m*, con *n* fijo, y *m* variable según el largo del patrón utilizado.

En Knuth-Morris-Pratt dependemos del tamaño *m* de patrón extraído y el tamaño del lenguaje *c*, pues una mayor cantidad de subcadenas similares dentro del texto disminuirá la efectividad de la función de salto, acercando su rendimiento a Fuerza Bruta

En Boyer-Moore-Horspool dependemos también del tamaño del lenguaje *c*, pues un lenguaje reducido aumentará las posibilidades de que un carácter que no calce en el patrón, aún esté presente más adelante. Con un lenguaje más amplio, la probabilidad de que esto ocurra baja significativamente.

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: