Tarea 1 -Memoria cache y RAM

Profesor: Gonzalo Navarro
Auxiliar: Ricardo Córdova
Ayudantes: Nicolás Canales, Gabriel Iturra, Yuval Linker, Natalia Quinteros
Estudiantes: Nombre de los integrantes del equipo aquí

Septiembre 2022

1. Introducción [1.0pts]

El objetivo principal de esta tarea es que podamos evidenciar empíricamente las diferencias entre los distintos niveles de memoria caché y memoria principal. Para ello, usaremos el problema de la distancia de edición: Dados dos strings $X = x_1x_2...x_n$ e $Y = y_1y_2...y_m$ sobre un alfabeto finito Σ, se define la *distancia de edición* entre X e Y como el mínimo número de operaciones requeridas para transformar una cadena en la otra. Estas operaciones pueden ser: $delete(x_i)$, que elimina x_i de X; $insert(y_j$, que inserta y_j en X; y $substitute(x_i, y_j)$, que reemplaza x_i por y_j en X.

Para esta tarea, su trabajo será el analizar, implementar y evaluar experimentalmente los algoritmos que sean descritos para computar la distancia de edición entre dos cadenas de igual longitud n en memoria principal. Cada algoritmo debe ser analizado detalladamente, haciendo énfasis en: complejidad computacional; espacio utilizado, tanto en memoria principal como en memoria secundaria. Recuerde que deben confeccionar y entregar un documento donde se observe claramente la estructura correspondiente de un informe.

En esta tarea pueden trabajar en grupos de hasta tres personas. Deben dejar claro en su entrega qué persona se enfocó en qué problema y cada integrante del grupo debe ser capaz de explicar todas las soluciones que hay en la entrega, no sólo la que desarrolló individualmente.

Hipótesis Deben presentar una hipótesis respecto a la complejidad computacional y el espacio utilizado por cada uno de los algoritmos que se presentaran en la siguiente sección.

(a) Aristas Sur, Este y Sureste. (b) Caminos que parten desde (0, 0).

Figura 1: Grafo de cuadrícula implícito para las cadenas *banana* y *ananas*.

2. Desarrollo [1.5pts]

2.1. Grafo de Cuadrícula Implícito

Sean X e Y dos strings de longitud n, el grafo de cuadrícula implícito para X e Y se define como:

- Una cuadrícula o grilla 2D de $(n+1) \times (n+1)$ nodos.
- Todo nodo tiene una arista a sus vecinos en el Sur(S), en el Este(E) y en el Sureste(SE).
- Las E-aristas y S-aristas tienen peso 1.
- La SE-arista (La arista que conecta los nodos $(i-1, j-1) \rightarrow (i, j)$tiene peso 0 si $X[i] = Y[j]$, y tiene peso 1 en otro caso.

Teorema 1. *La longitud del camino más corto desde la celda* $(0, 0)$ *a la* (n, n) *en el grafo de cuadrícula implícito para* X *e* Y *es la distancia de edición entre* X *e* Y*.*

Ejemplo 1. Sean $S =$ *banana* y $T =$ *ananas*, la figura 1a muestra el grafo de cuadrícula implícito correspondiente. Las aristas negras tienen peso 1 y las rojas tienen peso 0. En la figura 1b se muestran los pesos de todos los caminos que parten desde la celda $(0, 0)$. La distancia de edición entre las cadenas *banana* y *ananas* es 2 y aparece en la celda $(6, 6)$.

2.2. Algoritmo 1: Programación Dinámica.

1. Construir una matriz de $(n+1) \times (n+1)$ nodos, donde la casilla (i, j) tendrá la longitud del camino más corto desde $(0, 0)$ hasta (i, j).
2. Llenar la primera fila y la primera columna con los valores de 0, 1, ..., n consecutivamente.
3. Calcular el valor para cada casilla, secuencialmente, de izquierda a derecha, y de arriba hacia abajo. Este valor se obtiene usando las casillas anteriores $(i-1, j)$, $(i, j-1)$ y $(i-1, j-1)$.
4. Retorna el valor de la casilla (n, n).

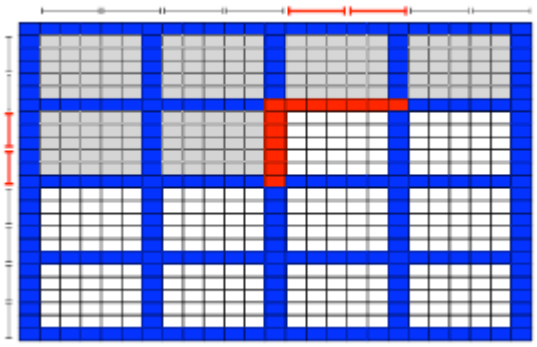

Figura 2: Particionado de la cuadrícula implícita, donde las lineas azules son las fronteras solapadas.

2.3. Algoritmo 2: Algoritmo en Cache.

1. Construir una fila y/o una columna con valores desde 0 a n.
2. Iterativamente, calcular los valores correspondientes a la siguiente celda de una columna y/o una fila, sobrescribiendo los valores que ya no son necesarios. Si hay valores que pueden ser útiles para calcular una celda, guárdelo antes de sobrescribirlos.
3. Retornar el valor de la última celda, sea de la fila o de la columna.

Note que el algoritmo a usar es básicamente el mismo algoritmo descrito en la sección 2.2. La diferencia está en que al computar los valores de la matriz, vamos sobrescribiendo los valores que ya no necesitamos para poder utilizar solamente la memoria cache.

2.4. Algoritmo 3: Particionando cuadrícula.

1. Dividir la cuadrilla o matriz en submatrices de tamaño $x \times x \leq M$ con fronteras solapadas (figura 2).
2. Procesar las subtablas de izquierda a derecha, y de arriba hacia abajo. Por cada subtabla:
   a) Leer las subtablas correspondientes.
   b) Computar los valores usando el algoritmo descrito en la sección 2.2.
   c) Escribir las fronteras de salidas en memoria secundaria.
3. Retornar el valor final de la frontera final.

3. Resultados [2.5pts]

Deben implementar los algoritmos descritos para computar distancia de edición. La descripción dada para los algoritmos es bastante general, por lo que es requerido análisis adicional y explicar partes del diseño de los detalles no explicitados en este documento.

Note que el algoritmo 2.4 requiere conocer el tamaño B de la memoria cache para determinar como particionar la cuadrilla, es decir, cual sería el tamaño de las subtablas. Es recomendable que esto sea un parámetro en su implementación para facilitar la experimentación.

Los experimentos se realizarán utilizando datos sintéticos. Genere aleatoriamente los strings X e Y de tamaños $N \in \{2^3, ..., 2^{15}\}$. Para cada par X, Y ejecute los algoritmos vistos en la sección 2, evalúe sus desempeños; repita este proceso al menos 50 veces para poder obtener resultados estadísticos significativos, graficandolos. Para determinar el tamaño de las submatrices puede realizar distintas pruebas usando solicitudes de espacio en el cache versus solicitudes en la memoria RAM.

Documente las características del hardware utilizado (CPU, RAM, etc.), el sistema operativo, así como el lenguaje utilizado y la versión del compilador. Tenga cuidado con el tamaño de los arreglos; como el trabajo es realizado en parte en memoria RAM y en parte en memoria cache, es posible que no disponga de la suficiente memoria para alguno de los algoritmos. Calcule e indique si esto puede ocurrir (u ocurre en su maquina).

4. Conclusión [1.0pts]

- La tarea puede realizarse en grupos de a lo más 3 personas.
- Para la implementación se recomienda utilizar C, C++ o Java, por tener manejo explícito de memoria. Recuerde desactivar el colector de basura en Java para hacer sus experimentos. Para el informe se recomienda utilizar LaTeX.
- Siga buenas prácticas en sus implementaciones. El código comentado ayuda a que se pueda revisar mejor. Recuerde habilitar optimizaciones al momento de compilar su código.
- Escriba un informe claro y conciso. Las ponderaciones del informe y la implementación en su nota final son las mismas.
- Cada algoritmo implementado será probado al momento de evaluar su código. Asegúrese que retornen lo esperado bajo distintas condiciones de borde.
- La entrega será a través de U-Cursos y deberá incluir el informe junto con el código fuente de la implementación (y todas las indicaciones necesarias para su ejecución.

- Se permiten atrasos con un descuento de 1 punto por día.