Universidad de Chile
Departamento de Ciencias de la Computación
CC4102 - Diseño y Análisis de Algoritmos

Tarea 2 - "Consultas sobre Strings"

Profesores: Pablo Barceló
Gonzalo Navarro
Auxiliar: Dustin Cobas
Ayudantes: Francisco Felipe Sanhueza Matamala
Ignacia Parra
Nicolás Higuera

1 Problema

Un *texto* $T[1, n]$ es una secuencia de n símbolos sobre un alfabeto Σ, con $|\Sigma| = \sigma$. Sobre T podemos definir un conjunto bastante amplio de operaciones. Para esta tarea estamos interesados en operaciones relacionadas con encontrar las *ocurrencias* de un *patrón* $P[1, m]$ [1] en T. Diremos que P *ocurre* en T en la posición i si $T[i, i + m] = P$. Específicamente queremos resolver las siguientes consultas:

* `count(`T`, `P`)`: Retorna la cantidad de posiciones donde el string $P[1, m]$ ocurre en $T[1, n]$.
* `locate(`T`, `P`)`: Retorna la lista con las posiciones donde el string $P[1, m]$ ocurre en $T[1, n]$.
* `top-k-q(`T`, `k`, `q`)`: Retorna la lista con los k strings de largo q que ocurren más veces en $T[1, n]$.

Note que, para un mismo texto T, queremos ejecutar repetidamente las consultas anteriores con diferentes patrones P y diferentes combinaciones de los parámetros k y q. En este caso, es conveniente construir un *índice* sobre T que nos permita realizar dichas consultas eficientemente. El *árbol de sufijos* está entre los índices más ampliamente usados para resolver estas operaciones.

El objetivo de esta tarea es implementar y evaluar experimentalmente un índice formado por un árbol de sufijos sobre el texto $T[1, n]$ que soporte las consultas `count`, `locate` y `top-k-q`. Deberá evaluar los tiempos construcción y el espacio utilizado, así como los tiempos de ejecución para cada una de las consultas. Además, deberá confeccionar un informe donde indique claramente los siguientes puntos:

1. Las *hipótesis* escogidas antes de realizar los experimentos.
2. El *diseño experimental*, incluyendo los detalles de la implementación de los algoritmos, la generación de las instancias y las medidas de rendimiento utilizadas.
3. La *presentación de los resultados* en forma de una descripción textual, tablas y/o gráficos.
4. El *análisis e interpretación* de los resultados.

[1] Al igual que T, el patrón P es un string pero de largo m.

2 Estructuras de Datos

En esta sección describiremos brevemente las estructuras datos y conceptos básicos para construir nuestro índice. Una información más detallada puede ser encontrada en los Apuntes del Curso.

2.1 Tries

Un *trie* (*árbol digital*) es una estructura de datos para almacenar un conjunto de strings $\mathcal{S}$ sobre un alfabeto Σ con $|\Sigma| = \sigma$. Este es un árbol *etiquetado* donde cada string de $\mathcal{S}$ puede ser leído a lo largo del camino desde la raíz hasta una hoja diferente.

Los strings de $\mathcal{S}$ se suponen terminados en el carácter especial "$" (lexicográficamente menor que todos los otros símbolos de Σ), lo que asegura que ningún string es prefijo de otro, es decir, ningún string termina en un nodo interno del trie. Por esto, un trie que almacena n strings, tiene exactamente n hojas. Cada nodo puede tener hasta $\sigma + 1$ hijos. La arista hacia cada hijo está etiquetada por un carácter del alfabeto, y no puede haber dos aristas saliendo de un mismo nodo y etiquetadas por el mismo carácter.

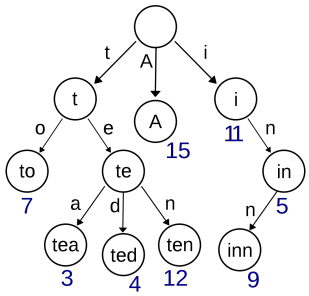

Figure 1: Ejemplo de Trie

Para buscar un determinado prefijo $P[1, m]$ en un trie, partimos de la raíz v_0 y bajamos por la arista rotulada $S[1]$ para llegar al nodo v_1. De v_1 bajamos por la arista rotulada $S[2]$ para llegar al nodo v_2, y así sucesivamente. La búsqueda de P requiere un tiempo $\mathcal{O}(m)$.

2.2 Árbol Patricia

Un *árbol Patricia* (también conocido como *blind trie*, *compressed prefix tree*, entre otros) es una variante de trie en el que se han comprimido las ramas unarias, reemplazándolas por arcos. En otras palabras, se reemplazan caminos unarios de la forma $v_i \xrightarrow{c_i} \ldots \xrightarrow{c_k} v_{k+1}$ por $v_i \xrightarrow{c_i \ldots c_k} v_{k+1}$. [2]

[2]Existen diferentes formas de ver los árboles Patricia. En algunas, la palabra completa se encuentra almacenada en las hojas; en otras, los nodos intermedios almacenan subcadenas. Otras variantes utilizan etiquetas numéricas en vez de subcadenas. Estos cambios alteran ligeramente los algoritmos de inserción y búsqueda. Si decidiera utilizar una variante diferente para sus implementaciones, explicítela en el informe, detallando las diferencias con las versiones aquí expuestas y justificando su decisión.

Esta modificación asegura espacio $\mathcal{O}(n)$ para la estructura, independientemente del largo de los strings de $\mathcal{S}$, pues un árbol de este tipo con n hojas tiene a lo sumo n nodos internos. Además mantiene el tiempo de las operaciones en $\mathcal{O}(m)$.

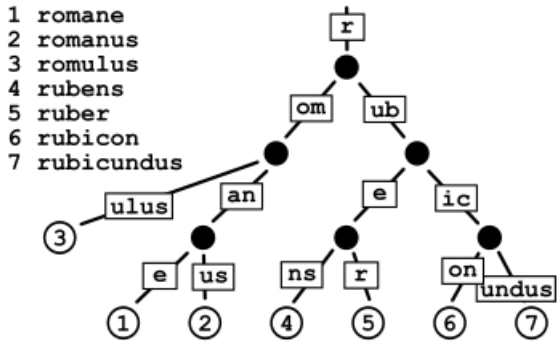

Figure 2: Ejemplo de Árbol Patricia

2.3 Árbol de Sufijos

Un *árbol de sufijos* es un árbol Patricia conteniendo todos los sufijos de un texto $T[1, n]$. Este texto es terminado con $T[n] = \$$, el cual es un caracter especial lexicográficamente menor que todos los otros símbolos. Como este árbol contiene unicamente los sufijos de T, en lugar de almacenar los sufijos completos $T[i, n]$ en las hojas, solo necesitamos mantener la posición inicial i del sufijo correspondiente.

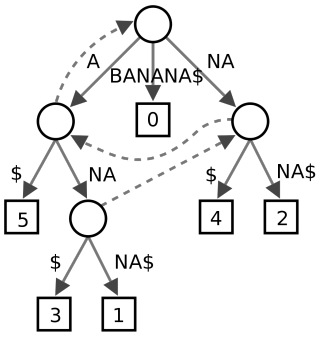

Figure 3: Ejemplo de Árbol de Sufijos

El árbol de sufijos es una herramienta muy útil para encontrar las ocurrencias de un patrón $P[1, m]$ en un tiempo proporcional a $\mathcal{O}(m)$. Para buscar P en T, usamos una búsqueda de prefijo del trie.

3 Implementación

Debe implementar un índice basado en un árbol de sufijos sobre el texto $T[1, n]$ que soporte las consultas `count`, `locate` y `top-k-q` anteriormente definidas. Dichas consultas pueden ser resueltas a través de recorridos sobre el árbol de sufijos.

Hint. Podemos almacenar en cada nodo algunos valores precomputados con el objetivo de evitar el recorrido del subárbol correspondiente al nodo.

4 Experimentación

Para los experimentos debe utilizar textos extraídos de las colecciones `ENGLISH (english texts)` y `DNA (gene DNA sequences)` contenidas en `Pizza&Chili Corpus`, compuestas por textos en idioma inglés y secuencias de ADN respectivamente. Se recomienda en ambos casos, para mayor facilidad, extraer los textos de los dataset de tamaño 50MB. Los textos extraídos deben ser preprocesados: elimine saltos de línea, signos de puntuación y espacios en blanco repetidos; lleve todo a minúsculas; si es necesaria o conveniente otra transformación del texto, efectúela y exponga los detalles en el informe. Asegúrese que sus textos preprocesados tengan largos de $n = 2^i$ símbolos, con $i \in \{10, 11, \ldots, 23\}$.

Para cada texto $T[1, n]$, construya el índice insertando en el árbol de sufijos todos los sufijos de T, midiendo el tiempo de construcción y el espacio completo requerido por su estructura de datos. Recuerde concatenar un símbolo $\$ \notin \Sigma$ (y menor lexicográficamente a cualquier otro símbolo) al final, para evitar que un sufijo sea prefijo de otro.

Para la primera colección (textos en inglés), los patrones P serán $n/10$ palabras del texto T seleccionadas de forma aleatoria. Para la segunda colección (secuencias de ADN), los patrones P serán $n/10$ substrings del texto de tamaños $m = \{8, 16, 32, 64\}$ escogidos aleatoriamente. Las consultas `count` y `locate` serán ejecutadas con estos patrones, registrando los tiempos de búsqueda en función de los largos de los patrones (m). Además, deberá ejecutar estas consultas con strings que no ocurran en los textos para registrar los tiempos de *miss search*, considerando también en este caso los largos de los patrones buscados.

Para la consulta `top-k-q`, deberá usar las combinaciones de los parámetros $k = \{3, 5, 10\}$, con $q = \{4, 5, 6, 7\}$ para la colección de textos en inglés, y con $q = \{4, 8, 16, 32\}$ para la colección de secuencias de ADN.

5 Entrega de la Tarea

- La tarea puede realizarse en grupos de a lo más 3 personas.
- Para la implementación puede utilizar C, C++ o Java. Para el informe se recomienda utilizar LaTeX.

- Siga buenas prácticas (*good coding practices*) en sus implementaciones.
- Escriba un informe claro y conciso. Las ponderaciones del informe y la implementación en su nota final son las mismas.
- Tenga en cuenta las sugerencias realizadas en las primeras clases sobre la forma de realizar y presentar experimentos.
- 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.

6 Links

- En la sección 7.4 de `http://cglab.ca/~morin/teaching/5408/notes/strings.pdf` puede encontrar una versión optimizada en espacio de los `Árboles Patricia`.