Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 — Estructuras de Datos y Algoritmos
2019 - 2

Tarea 1

Fecha de entrega código: 17 de Agosto
Fecha de entrega informe: No hay informe

Objetivos

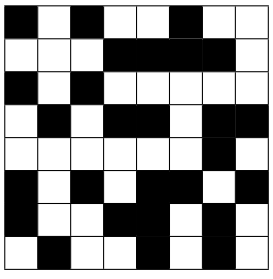
- Optimizar un algoritmo de consultas usando un árbol
- Investigar una estructura de datos no vista en clases e implementarla

Introducción

Los científicos de la NASA acaban de hacer un descubrimiento asombroso sobre el ruido que ayudaste a limpiar de las imágenes de la sonda Yadrinni: el ruido no es producto de un asteroide que daño la sonda, sino que es un mensaje enviado por extraterrestres a través de la sonda para comunicarse con los científicos de la tierra.

Cómo interpretar este mensaje, aún no lo sabemos, pero ya que ayudaste a la NASA a limpiar con éxito las imágenes, te pidieron a ti que descubras las palabras insertadas en ellas. La manera de extraer el mensaje de las imágenes es tomar cada píxel y remplazar los puntos blancos por un 1 y el resto como un 0. Esto genera una matriz de 0s y 1s del tamaño de la imagen a la que llamaremos **SOPA**[1]. Para efectos de este enunciado, visualizaremos la sopa como una matriz de celdas blancas y negras.

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} =$$



Para averiguar cual es el mensaje, la NASA te da una lista de palabras en binario que los extraterrestres podrían haber enviado, por lo que tu tarea es buscar si estas palabras se encuentran o no en la sopa.

[1] Por las siglas del proyecto: Sistema Oculto de Palabras Alienígenas

Definición formal del problema

*Dada una lista de palabras en binario y una **SOPA**, determina cuantas veces se repite cada palabra en una vertical, horizontal o diagonal de la **SOPA**.*

Las palabras pueden estar escritas de arriba a abajo, de izquierda a derecha o en diagonal hacia la derecha y abajo pero no pueden estar en el resto de las direcciones, como se muestra en la siguiente figura:

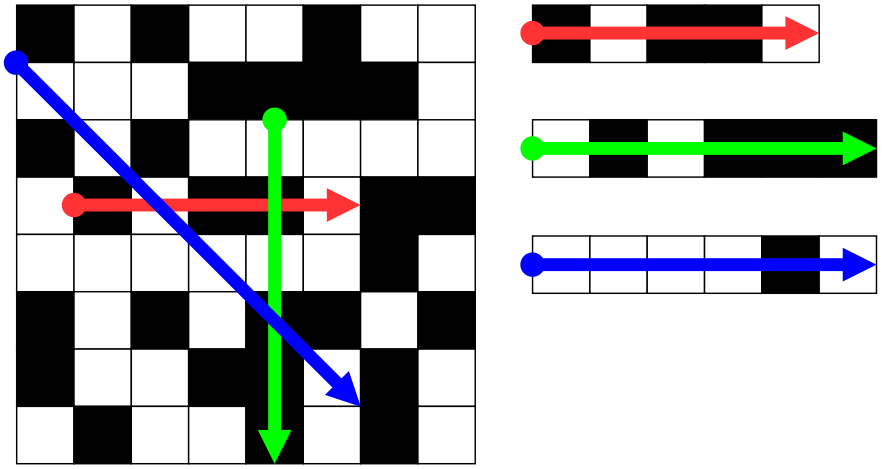


Figura 1: Ejemplos de palabras en la sopa en las 3 direcciones válidas

Para resolver el problema de manera eficiente es necesario preprocesar la sopa usando una estructura de datos llamada **Suffix Trie**. Para entender esta estructura de datos primero se definen los conceptos de prefijo y sufijo en la siguiente sección. En la figura 2 se muestra un ejemplo en el que una misma palabra aparece 5 veces.

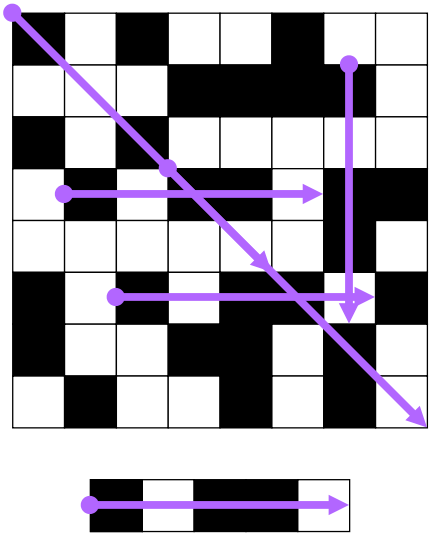


Figura 2: Las posiciones donde se encuentra la secuencia 01101

Prefijos y Sufijos

Para un string S tal que

$$S = \{a_1, a_2, \cdots, a_n\}$$

Definimos:

- El i-ésimo **sufijo** de S como el substring s_i que comienza en a_i y termina en a_n
- El i-ésimo **prefijo** de S como el substring p_i que comienza en a_1 y termina en a_i

Todo substring $s \in S$ es **prefijo** de algún **sufijo** de S. Por lo tanto podemos cambiar el problema de buscar un substring en S por revisar si es prefijo de alguno de sus sufijos. Si el problema requiere buscar muchos substrings es posible generar los sufijos de antemano y luego revisar sólo los que correspondan.

Para eso necesitamos tenerlos organizados en una estructura de datos; en este caso queremos organizarlos en un Trie de sufijos, un **Suffix Trie**.

Trie

El **Trie** es una estructura de datos que se implementa como un árbol y que almacena palabras de largo arbitrario. Permite determinar si existe o no una palabra en un grupo de palabras. En la figura 1 se puede ver un Trie que almacena 5 palabras. Los nodos con doble círculo denotan el fin de palabra.
Para revisar si existe una palabra en el Trie solo hay que recorrer desde la raíz siguiendo el camino de la letra correspondiente hasta un nodo marcado. Si no existe el camino dado por la letra que sigue simplemente la palabra no está en el Trie.

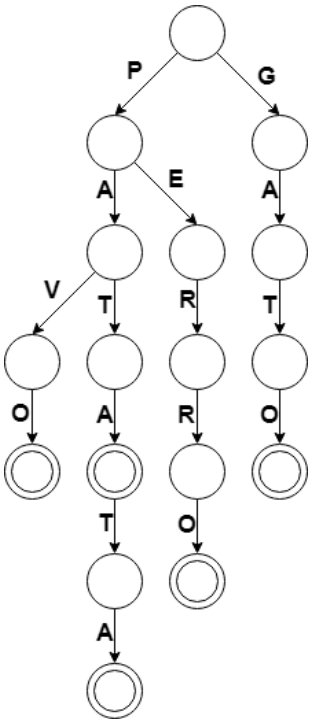


Figura 3: Trie con palabras patata, pata, pavo, gato y perro.

Un **Suffix Trie** es un a versión particular de un **Trie** que almacena palabras y sus sufijos. Esto permite encontrar substrings en las palabras, como se dijo en la sección anterior. Por ejemplo la palabra "tetera" tiene como substring la palabra "eter". Si creamos un **Suffix Trie** con la palabra "tetera" podemos saber si "eter" es substring de "tetera" siguiendo los nodos con las letras correspondientes. Esto se muestre en la siguiente figura:

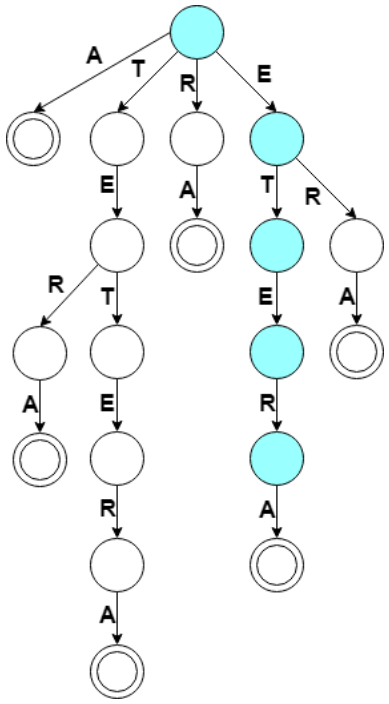


Figura 4: Suffix Trie con palabra tetera

Lo que se espera de tu programa es que implementes un Suffix Trie que almacene palabras cuyas letras son 0s o 1s para almacenar las palabras de las filas, columnas y diagonales de la matriz de datos y sus respectivos sufijos. Con esto puedes saber con rapidez si alguna secuencia de 0s y 1s está en la matriz, y, en el caso de esta tarea, cuantas veces está.

Para crear el Suffix Trie desde la sopa debes iterar por cada fila insertando todos los sufijos de cada fila en el Suffix Trie. Luego haces esto para cada columna y para cada diagonal. Una vez terminado tu Suffix Trie está listo para saber si una secuencia de 1s y 0s está o no en la sopa. Si además guardas un contador en cada nodo puedes responder cuantas veces está la secuencia en la sopa.

Input/Output

El programa debe recibir como input una imagen en formato png y un archivo con las palabras a consultar. Además recibe el nombre del archivo de output.
El output del programa se escribe en el archivo de output indicando las repeticiones de cada palabra de la lista.
El archivo con las consultas sigue el siguiente formato:

- Primera linea: número entero que indica la cantidad de palabras en el archivo.
- Las siguientes líneas parten con un número entero que indica la cantidad de letras de la palabra. Puedes asumir que este número está entre 16 y 512.
- El resto de la línea es la palabra binaria separada por espacios.

El archivo de output tiene un número por cada palabra del archivo de consultas en el mismo orden que las palabras y separadas por un salto de línea.
OJO: El código base de la tarea ya maneja la lectura del input y la escritura del output por lo que no debes preocuparte de como leer el input y escribir el output.

Cálculo de nota

En esta tarea no habrá informe para que disfrutes de las fiestas patrias por lo que el 100% de la nota corresponde al código. Se evaluará usando distintos tests de tamaño variable y se ejecutarán con un tiempo máximo de 10 segundos por test. Además se ejecutará tu programa usando valgrind y este valdrá el 10% de tu nota.

Secreto

Cada test efectivamente tiene un mensaje secreto que puede ser interpretado. Suerte encontrándolos.

Entrega

- **Código:** GIT - Repositorio asignado (asegúrate de seguir la estructura inicial de éste). Si se crean varios repositorios cuando accedes al link para crear repositorios, trabaja en el que no tenga un número al final y borra el resto de los repositorios.
  - En la carpeta *Programa* debe encontrarse el código.
  - Se recogerá el estado en la rama *master*.
  - Se espera que el código compile con `make` dentro de la carpeta *Programa* y genere un ejecutable de nombre `decode` en esa misma carpeta.
- **Hora Límite:** 1 minuto pasadas las 23:59 del día de la entrega.
- No se permitirán entregas atrasadas, **seremos estrictos con esto** por lo que no dejen para último instante el subir su tarea.