Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Depto. de Ciencias de la Computación
CC4102 - Diseño y Análisis de Algoritmos

# Tarea 2

Integrantes : Rodrigo Delgado
Belisario Panay
Gabriel Sanhueza
Profesor : Gonzalo Navarro
Ayudantes : Sebastián Ferrada
Willy Maikowski
Auxiliar : Jorge Bahamondes

# Índice

# 1. Introducción

// TODO: Definir qué es un suffix tree
// TODO: Decir que queremos probar Ukkonen, y que es Ukkonen
Se pide que plantee una hipótesis con respecto al tiempo amortizado de construcción de una estructura de este tipo y al tiempo de búsqueda, y la ponga a prueba de forma experimental. Se espera que se implemente la estructura y los algoritmos correspondientes, y se entregue un informe. En el presente informe se muestra el diseño, implementación y experimentación para resolver la Tarea 2 del curso CC4102 Diseño y Análisis de Algoritmos de la carrera de ingeniería civil en Computación de la Universidad de Chile, la cual consiste en programar un algoritmo de orden lineal para la creación de Suffix Trees, una estructura de datos de datos del tipo Arbol, la que esta formada por Nodos que poseen información interna y referencias a sus hijo. En particular el SuffixTree almacena los sufijos de un string en forma de arbol, de modo que cada arco contiene los caracteres para formar una palabra, si se llega a una hoja ( nodo terminal ) es porque recorrimos los arcos necesarios para formar un sufijo. La idea de la tarea es que el algoritmo a desarrollar es on-line, de orden lineal y con una implementación mas sencilla con respecto a algoritmos similares ( algoritmo de Weiner y algoritmo de McCreight ).

## 1.1. Problema a resolver

// TODO: Explicar que queremos desarrollar un suffix tree para acelerar busqueda de texto
// TODO: Decir que queremos que sea rápido
El problema consiste en realizar los pasos necesarios para la buena implementación del algoritmo, ya que pequeños errores en código pueden producir un algoritmo de mayor orden de magnitud y con ello se falla en el objetivo. Los pasos para realizarlos estan detallados en el enunciado de la tarea y son los pasos que se desarrollan en el paper del algoritmo de ukkonen, el cual se encuentra disponible en el enunciado de la tarea. Como resumen, se puede detallar que los pasos consisten en pasar un algoritmo de Orden n a orden lineal.

## 1.2. Hipótesis

// TODO: Decir que creemos que Ukkonen será más rápido que un suffix tree normal, probablemente será lineal Si bien es un algoritmo lineal, el algoritmo posee una constante que hace que el tiempo pueda crecer de manera rapida, tambien como el algoritmo será escrito en java como lenguaje, posee caracteristicas como el garbage collector que reducen la eficiencia del algoritmo al tomar la Cpu para hacer realizar sus funciones. Se espera demostrar que la constante del algoritmo varía con respecto al largo por motivos externos al algoritmo, esto se puede mostrar si para números pequeños el agoritmo si demuestra un comportamiento lineal.

# 2. Diseño Teórico

// TODO: Mostrar las distintas clases creadas y para qué sirven (cada clase puede tener su propia subsection)
Para explicar el diseño teórico del algoritmo primero se deben definir algunos conceptos:

- Suffix Tree Implicit es un Suffix Tree al cual se le remueven sus simbolos terminales ( en algunos casos se utiliza el signo $ como terminal)

## 2.1. Metodología

## 2.2. Structs

### 2.2.1. Rectangle?

### 2.2.2. Node

## 2.3. Constantes

## 2.4. Funciones

# 3. Presentación de los Resultados

## 3.1. Tiempo de Creación del Suffix Tree

// TODO: Mostrar cuanto se tarda en crear un suffix tree, para N = 2 ** 15..21 palabras (aprox).

### 3.2. Desempeño de operación *Buscar*

// TODO: Mostrar cuanto se demora en buscar N = (2 ** 15..21) / 10 palabras (aprox).

# 4. Análisis y Conclusiones

## 4.1. Construcción del Suffix Tree

// TODO: Verificar si el Suffix tree fue lineal o no

## 4.2. Búsqueda en el Suffix Tree

// TODO: Verificar si los tiempos están acotados o se fueron a la chu*** cada vez que aumentamos el número

## 4.3. Conclusiones

// TODO: Resumir lo “aprendido” (?)