# ALGORITMOS Y ESTRUCTURA DE DATOS 2

## Documentación - Obligatorio



Gonzalo Otheguy

206324

Melissa Pais

181257

Grupo N4B

Docente: Francisco Bouza

09/05/2018

Índice

[ALGORITMOS Y ESTRUCTURA DE DATOS 2 1](#_Toc512431232)

[Documentación - Obligatorio 1](#_Toc512431233)

[Interfaz Sistema: Pre y post condiciones 3](#_Toc512431234)

[Solución escogida 4](#_Toc512431235)

[Diagrama de la estructura de datos 4](#_Toc512431236)

[Justificación 4](#_Toc512431237)

[Operación Ranking de Palabras 5](#_Toc512431238)

[Testing 5](#_Toc512431239)

# Interfaz Sistema: Pre y post condiciones

## Inicializar Sistema

//PRE CONDICIONES: Se recibe la cantidad máxima de palabras a almacenar (maxPalabras). "MaxPalabras" debe ser un entero.

//POS CONDICIONES: Retorna ERROR\_1 si maxPalabras <= 0, u OK si maxPalabras > 0 y se crearon las estructuras correspondientes.

**public Retorno inicializarSistema (int maxPalabras)**

## Destruir Sistema

//PRE CONDICIONES: Se consideran creadas todas las estructuras

//POS CONDICIONES: Se devuelve OK, si se logró eliminar correctamente

**public Retorno destruirSistema()**

## Analizar texto

//PRE CONDICIONES: Se recibe una cadena de caracteres.

//POS CONDICIONES: Retorna ERROR\_1 si el texto es vacío, u OK si se pudo analizar el texto y grabarlo en las estructuras correspondientes.

**public Retorno analizarTexto(String texto)**

## Ranking palabras

## //PRE CONDICIONES: Se recibe un valor entero.

## //POS CONDICIONES: Retorna ERROR\_1 si "n" es menor o igual a cero, ERROR\_2 si hay menos de "n" palabras diferentes en el texto, y un listado de las "n" palabras más usadas en otro caso.

## public Retorno rankingPalabras(int n)

## Apariciones palabra

//PRE CONDICIONES: Se recibe una cadena de caracteres.

//POS CONDICIONES: Retorna ERROR\_1 si el texto es vacío, u OK si se pudo analizar el texto y grabarlo en las estructuras correspondientes.

**public Retorno aparicionesPalabra(String palabra)**

## Todas palabras

//POS CONDICIONES: Retorna un listado de todas las palabras almacenadas, o vacío si no hay palabras.

**public Retorno todasPalabras()**

# Solución escogida

## Diagrama de la estructura de datos

## 

## Justificación

|  |  |
| --- | --- |
| **PALABRAS** | |
| Estructura | Árbol binario de búsqueda. |
| Justificación | Se optó por un árbol por ser una estructura dinámica, ya que  no vamos a tener una cantidad fija de palabras a almacenar. Además, permite mantener un orden (que es necesario para luego armar el listado de palabras ordenado alfabéticamente), y muy importante es que se solicitaba en la letra que tal listado se hiciera en orden (n) promedio, al igual que para otro método, la búsqueda debía ser en orden (log n) promedio, que fue lo que nos dio la pauta de utilizar esta estructura. |

Para cumplir con todos los requisitos de eficiencia, decidimos tener dos ABB, uno inserta el nodo que contiene la palabra y la cantidad de repeticiones en un árbol que ordena por palabra alfabéticamente, y otro ordena por cantidad de repeticiones.

Para el primer árbol, se le va actualizando la cantidad de repeticiones cada vez que se encuentra una palabra repetida en el texto analizado, y en el otro, se borra el nodo con la palabra, y se vuelve a insertar el nodo con la cantidad de repeticiones actualizada.

## Operación Ranking de Palabras

Explique detalladamente cómo realizó esta operación.

## Testing

## Tests efectuados por nosotros (Clase SistemaTest)