**Realizado por:**

Wilder Valencia Ocampo 000375627

Emilio Martínez Rivera 000255600

Luis Esteban Santamaria 000291800

**3.**Estimar la complejidad algoritmica de la operación get, asumiendo una colección de N términos y uno de los algoritmos de comparación analizados en el punto 1.

**a. Operación get**

**public** Bag<String> get(String termino){

//Algoritmo Extraido y adecuado al problema de Brute.java from 5.3 Substring Search de Algorithms, 4th Edition.

Bag<String> similares=**new** Bag<String>();

String S;

termino=termino.toLowerCase();

**int** m = termino.length();

**for**(String N: coleccion) {

S=N.toLowerCase();

**int** n = S.length();

**for**(**int** i=0;i<=n-m;i++) {

**int** j;

**for**(j=0;j<m;j++) {

**if**(termino.charAt(j)!=S.charAt(i+j))**break**;

}

**if**(j==m) similares.add(N);

}

}

**return** similares;

}

Análisis con modelo de costo la comparación de caracteres

Algoritmo Fuerza bruta

Teóricamente la investigación ya ha arrojado que el costo es de

Recorrido de colección de N términos

El recorrido implica recorrer N términos

Por lo tanto el costo Total es:

= Algoritmo Fuerza bruta \* Recorrido de colección de N términos

= Orden de Crecimiento Cubico

**b. Algoritmo Boyer–Moore para N términos**

**public** Bag<String> getBC(String termino){

Bag<String> similares=**new** Bag<String>();

BoyerMoore patron = **new** BoyerMoore(termino);

**for**(String N: coleccion) {

**if**(patron.search(N)!=N.length()) similares.add(N);

}

**return** similares;

}

Análisis con modelo de costo la comparación de caracteres

Algoritmo Boyer–Moore

Teóricamente la investigación ya ha arrojado que el costo es de

donde c es el tamaño del alfabeto (c<<n).

Para un alfabeto razonablemente grande, el algoritmo es

~n/m

Recorrido de colección de N términos

El recorrido implica recorrer N términos

Por lo tanto el costo Total es:

= Algoritmo Boyer–Moore \* Recorrido de colección de N términos

=

**c. Algoritmo Knuth-Morris-Pratt para N términos**

**public** Bag<String> getKMP(String termino){

Bag<String> similares=**new** Bag<String>();

KMP patron = **new** KMP(termino);

**for**(String N: coleccion) {

**if**(patron.search(N)!=N.length()) similares.add(N);

}

**return** similares;

}

Análisis con modelo de costo la comparación de caracteres

Algoritmo Knuth-Morris-Pratt

Teóricamente la investigación ya ha arrojado que el costo es de

incluyendo el preprocesamiento del patrón

Recorrido de colección de N términos

El recorrido implica recorrer N términos

Por lo tanto el costo Total es:

= Algoritmo Knuth-Morris-Pratt \* Recorrido de colección de N términos

= incluyendo el preprocesamiento del patrón