

Complejidad de los algoritmos kmp y Boyer moore

Complejidad Algoritmo Kmp:

Como podemos ver en la primera parte del código mostrada abajo primero instanciamos un arraylist por ende hay que tener en cuenta esto.

Después asignamos un valor al arraylists si la condición es verdadera y asignamos un mas 1 al contador. por ende en el peor de los casos esta operación tendría un peso de 3 con la comparativa de la condición.

Ahora también creamos un array de caracteres los cual también nos da un peso de 1 y seguidamente otro array de caracteres al mismo tiempo esto nos da peso de 1.

Ahora en la parte más compleja de nuestro código tenemos una iteración la cual tiene tres operaciones que siempre van a dar un peso instanciación incremento y comparación. Nuestro caso la instancia tiene un peso de 1 el comparativo un peso igual a el tamaño del array patrón y el incremento tendrá este valor más 1.

Ahora en en cada iteración tenemos que hacer un segmento de operaciones los cuales con instanciación de un valor lo cual lo tomamos como 1 y un cilco locual lo tomamos con un log de 2 y un otra instanciación lo cual lo tomamos como dos pues esta dentro una un comparativo

Ahora repasando lo que tenemos tendríamos lo siguiente :

$1+1+1+1+1+1+(patrón.length)+patron.length(1+\log_2(Patron.length)+3)$ lo cual nos daris como resultado lo siguiente: $5+(patrón.length)+patron.length(\log_2(Patron.length)+4)$

```
ArrayList<Integer> p =new ArrayList<Integer>();
int cont = 0;
if (patron == null || patron.length() == 0) {

    p.add(0);
    cont++;

}

char[] chars = patron.toCharArray();

int[] next = new int[patron.length() + 1];
for (int i = 1; i < patron.length(); i++) {
    int j = next[i + 1];

    while (j > 0 && chars[j] != chars[i]) {
        j = next[j];
    }

    if (j > 0 || chars[j] == chars[i]) {
        next[i + 1] = j + 1;
    }
}
```

Ahora vamos a calcular el algoritmo de Boyer moore :

Como podemos ver en código de abajo tenemos tres instanciaciones al inicio lo cual nos da un valor de 3

Un while que lo calculamos comparando pos con el la longitud del texto lo cual nos da $\log_2(\text{longitud del texto})$ dentro del ciclo while tenemos lo siguiente 4 instanciaciones al mismo tiempo otro ciclo while lo cual es $\log_2(\text{variable l y comparación})$ lo cual resumiendo nos da 2.

una llamada a un función lo cual nos da un valor de 1 luego una asignación con una comparación y el razon logica lo cual nos da un valor de 3

y una asignación . después una suma lo cual lo podemos tomar con un peso de 1.

Tendríamos lo siguiente.

$3 + (\log_2(\text{txt.length}) * (4 + \text{txt.length}(3) + 3 + 1))$ lo cual nos quedaría con

$3 + (\log_2(\text{txt.length}) * (4 + \text{txt.length}(3) + 4))$

```
HashMap<Character, Integer> mapa=patron(patron);

ArrayList<Integer> lista=new ArrayList<Integer>();

while(pos <= (text1.length-1)) {
    int sum=0;
    int inercount=0;
    int l=patron.length()-1;
    int j=pos;
    while(l>=0 && (text1[j] == chars[l])) {
        inercount++;
        l--;
        j--;
    }
    if(inercount != patron.length()) {
        sum =(mapa.get(text1[j]) == null)? patron.length(): mapa.get(text1[j]);
    }else {
        sum = patron.length();
        lista.add(j+1);
    }
    pos=pos+sum;
}
return lista;
```