Algoritmos de empate de cadenas

Observemos la clase String y en específico el método equals:

```
public boolean equals(Object anObject) {
    if (this == anObject) {
        return true;
    }
    if (anObject instanceof String) {
        String anotherString = (String)anObject;
        int n = value.length;
        if (n == anotherString.value.length) {
            char v1[] = value;
            char v2[] = anotherString.value;
            int i = 0;
            while (n-- != 0) {
                if (v1[i] != v2[i])
                    return false;
                i++;
            }
            return true;
        }
    }
    return false;
}
```

Como podemos ver es fácil darnos cuenta de que tipo de algoritmo ocupa y es que se compara cada valor perteneciente a cada cadena, esta comparación se encuentra en el ciclo while que podemos ver y en el caso de que se encuentre algún carácter que no coincida se regresa falso automáticamente y ya no continua la comparación, pero antes se realizan otras validaciones por ejemplo que tenga la misma longitud o si se trata del mismo objeto.

Dado lo anterior podíamos afirmar que ocupan el algoritmo ingenuo, algo poco conveniente para String de tamaño muy grande, pero eficiente si queremos comparar String de tamaño chico.

Ahora a completando la investigación anterior es que se puede sobre escribir el método equals y hashCode para así poder comparar el valor de hash de cada uno y si ambos valores de hash son iguales entonces las cadenas son iguales, en caso contrario no lo son.

Ahora veamos el caso de C++:

Para ser sinceros no entendí en su totalidad como C++ maneja la comparación de String viendo su código, investigando y leyendo varias paginas web, buscando en foros como StackOverflow pareciera que sigue el mismo algoritmo que ocupa Java

si en un recorrido de un String encuentra una diferencia con el String a comparar entonces devolverá un número menor a 0 en caso contrario será mayor a 0, cuando hay match da el resultado es 0.

La verdad es que este ultimo si me costro trabajo entenderle, porque en otra referencia que encontré pareciera que usara Rabin-Karp ya que toma el tamaño del patrón y lo compara tomando el mismo tamaño del patrón con el texto, pero no mencionaban nada de usar hash.

Fue interesante buscar los algoritmos que ocupa cada lenguaje y como conclusión considero que aun a pesar se ocupar un algoritmo parecido los dos lenguajes considero que es más rápida la implementación que realiza C++.

Nota: El algoritmo ingenuo imprime las ocasiones en donde se encuentra un match entre un carácter del patrón y otro del texto, pero la respuesta se encuentra en donde se repite el mimo numero el numero de veces de la longitud del patrón, es decir si el patrón es de longitud 5, y se encuentra una ocurrencia en el índex 3, entonces el número 3 se repetirá 5 veces. La clase main para probar el programa es EmparejamientoCadenas.

Texto que se ocupo para probar los algoritmos: “AABAACAADAABAAABAA”

Patrón que se ocupo para probar los algoritmos: “AABA”