

**FACULTAD DE INGENIERÍA**

CARRERA: Ingeniería de Sistemas Computacionales

**Silabeo de palabras**

Informe académico

**Alumnos (es):**

Carlos Chávez Laguna

Gustavo Ríos Morales

**Curso:**

Análisis y diseño de algoritmos

**Docente:**

Ernesto Cárdenas Cangalagua

Lima – PERÚ

2015-1

**INTRODUCCIÓN**:

Como sabemos todo algoritmo busca solucionar un problema y como tal este debe ser óptimo por ello en el presente informe vamos a exponer los pasos que hemos estado realizando en cuanto al proceso de construcción del algoritmo de silabeo de palabras.

**INFORME**

Inicialmente habíamos comenzado el diseño con fuerza bruta posteriormente hemos estado analizando y aplicando algunos métodos de optimización como el voraz o backtraking, a continuación vemos la secuencia de nuestro código:

1. Hemos estado aplicando un método voraz pues al cambiar el orden de las restricciones la cantidad de silabas varia y evalúa casos innecesarios

2. Por el momento el voraz nos está funcionando pero con una complejidad del O(n2)

3. Hemos generado una versión 1.1

-en el archivo gramatica2 colocamos los guiones al comienzo del conjunto de letras a silabear.

-se agrega un carácter, comodín, al principio de la palabra capturada. Siendo así posible de cumplir la restricción de "restConsonantes" y poder separar la primera letra de las demás cuando empiece con vocal.

-se imprimen las restricciones cumplidas por la palabra.

4. Hemos generado una versión 1.2 como importa el orden hemos visto que primero va la restricción del hiato luego la restricción de la monosílaba, Pues hay palabras que parecen monosílabas pero que son hiatos

5. Los últimos cambios realizados

Version 1.2, cambios:

-en la restricción, "restDobleFusion", mejora esta para que solo valide a conjunto de letras válidas por la gramática española: ch,ll,rr.

Version 1.3, cambios:

-Acepte caracteres propios del idioma español.

Version 1.4, cambios:

-nueva restricción de vocales (í,ú) como vocales fuertes para el hiato.

-se agregó al caracter ü como vocal débil.

ANEXO para ver las versiones

https://github.com/CarlosDevlp/SeparadorSilabasEs

**LA COMPLEJIDAD**

Nuestra complejidad depende de la longitud de cada palabra

Suponiendo que su longitud fuera n entonces su complejidad seria de 2n+2

asumiendo que la cantidad de datos es n

\*void separar:

for 2n+2

for (2n+2)n

for (2n+2)n

1

for (2n+2)n

total 2n+2+3(2n+2)n = 2n+2+6n^2+6n = 6n^2+8n+2 = n^2

seria O(n^2)

\*void dividir

for 2n+2 = O(n)

\*void restMonosilabo

1

2n+2

n

total 1+2n+2 +n = 2n+3 = O(n)

\*void restConsonante

2n+2 = O(n)

\*void rest2RLConsonante

2n+2 = O(n)

\*void rest3Consonante

1

2n+2

total 1+2n+2 = 2n+3 = O(n)

\*void rest4Consonante

1

2n+2

total 1+2n+2 = 2n+3 = O(n)

\*void restDobleFusion, void restDiptongo(),void restTriptongo, void respHiato y \*void imprimir

2n+2 = O(n)

Concluimos que nuestra complejidad es lineal