ALGORITMOS Y ESTRUCTURAS DE DATOS II GRADO INGENIERÍA INFORMÁTICA,CURSO 16/17 PRÁCTICA DE DIVIDE Y VENCERÁS Y ANÁLISIS DE ALGORITMOS.

Memoria de la actividad:

Nombre: Ricardo Ramírez García
Grupo y Subgrupo: 2.2
Email: ricardo.ramirezg@um.es

Nombre: Pablo Salinas Rodríguez
Grupo y Subgrupo: 2.2
Email: pablo.salinasr@um.es

Pseudocódigo y análisis del algoritmo:

esConsonante(char c) : bool

```
if (c ==’a’ OR c == ‘e’ OR c == ‘i’ OR c == ‘o’ OR c == ‘u’) return false
sino return true
```

esVC(String p) : bool

```
if(!esConsonante(a[0])&&esConsonante(a[1])) return true;
sino return false;
```

esCV(String p) : bool

```
if((esConsonante(a[0]))&&(!esConsonante(a[1]))) return true;
sino return false;
```

max(String cad1, String cad2) : String

```
if(cad1.size() >= cad2.size()) return cad1;
sino return cad2;
```

max(String cad1, String cad2, String cad3) : String

```
if((cad1.size() >= cad2.size()) && (cad1.size() >= cad3.size())) return cad1;
sino if((cad2.size() > cad1.size())&&(cad2.size() > cad3.size())) return cad2;
sino return cad3;
```

maxConsVocal(string cadena,int tamano) : String

```
if tamano <= CasoBase then
      return iterativo(cadena)
end
      cad1 = cadena.substr(0,tamano/2);
      cad2 = cadena.substr(tamano/2,tamano-(tamano/2));
      return  combinar(cadena,maxConsVocal(cad1,(tamano/2)),maxConsVocal(cad2,(tamano-(tamano/2))))
```

iterativo(string cadena) : String

```
cadenaActual = ""
cadenaMaxima = ""
buscando = 1

i = 0
while NOT esConsonante(cadena[i]) AND NOT finCadena(cadena)  do  i = i + 1
if NOT finCadena(cadena) then
      buscando = 1
      cadenaActual = cadena[i]

      for j = i + 1 to cadena.length() do
            if buscandoConsonantes(buscando) then
                  if esConsonante(cadena[j]) then
                        cadenaActual = cadenaActual + cadena[j]
                        buscando = 1
                  else cadenaActual = ""
            else if buscandoVocales(buscando) then
                  if !esConsonante(cadena[j])  then
                        cadenaActual = cadenaActual + cadena[j];
                        actualizarCadenaMaxima(cadenaActual);
                        buscando = 0
                  sino
                        cadenaActual = cadena[j]     buscando = 1
                  end
            end
      end

            return cadenaMaxima

else   return ""
```

```
combinar(String cadena, String izq, String der) : String
//n = cadena.size()
calcularSilabaCentral(cadena)
if cadena.size() <= 7 then return cadenaMaximaValida(cadena)
else if esVC(silabaCentral) then
            i:= 1
            while i <= n-(n/2) AND esCV(siguienteSilabaDer) do
                  solucionDer = solucionDer + siguienteSilabaDer
                  i = i + 2
            end
            j:= 1
            while (n/2)-j-2 >=0 AND esCV(siguienteSilabaIzq) do
                  solucionIzq = solucionIzq + siguienteSilabaIzq
                  j = j + 2
            end
            return max(solucionDer+silabaCentral+solucionIzq,izq,der)
      else
            calcularSilabaCentral(cadena)
            if not esValida(SilabaCentral) then return max(der,izq)
            y := 2
            mientras y <= n-(n/2) AND esCV(siguienteSilabaDer) do
                  solucionDer = solucionDer + siguienteSilabaDer
                  y = y + 2
            end

            z := 2
            mientras (n/2)-2-z >=0 AND esCV(siguienteSilabaIzq) do
                  solucionIzq = siguienteSilabaIzq + solucionIzq
                  z = z + 2
            end
            return max(solucionIzq + cadenaCentral + solucionDer, der, izq)
      end
      return max(der,izq)
end
```

Explicación del algoritmo

esConsonante :
Esta función devuelve verdadero o falso en función de si el carácter que se le pasa como parámetro es consonante o vocal respectivamente.

esVC :
Esta función comprueba si silaba que se le pasa como parámetro está formada por una vocal – consonante, si es así devuelve true, en caso contrario false.

esCV :
Esta función comprueba si la silaba que se le pasa como parametro está formada por una consonante – vocal, si es así devuelve true, en caso contrario false.

max :
Esta función está sobrecargada ya que se puede llamar con dos o tres palabras, una vez que se le pasan las cadenas como parámetros devuelve la cadena de mayor longitud.

maxConsVocal :
Consta de dos partes, el caso base que se resuelve aplicando la función iterativa y el caso recursivo que consta de dos llamadas recursivas, una con la mitad derecha de la cadena y otra con la mitad izquierda, el resultado de las dos llamadas iterativas se le pasa a la función combinar junto a la cadena entera que se le ha pasado como parámetro a la función actual, esta función se encarga de combinar los resultados parciales para conseguir el resultado, finalmente se devuelve el resultado del combinar.

iterativo :
Es un función que devuelve la subcadena más larga que contenga un numero par de consonantes seguidas de vocal consecutivas. Primero buscamos la primera consonante, si no hay consonantes devolver directamente la cadena vacía, una vez encontrada esa consonante tenemos una condición boolean que nos ayuda a analizar en que estado nos encontramos , es decir ; si estamos buscando una vocal o una consonante y dentro de estos casos nos encontramos ante diferentes situaciones que analizamos, al mismo tiempo que vamos alternando entre consonantes y vocales , actualizamos la cadenaActual(Válida) y sólo en el caso en el que el carácter actual sea vocal y la cadenaActual sea de mayor longitud que la cadenaMáxima(Válida) se actualiza la cadenaMáxima.

Combinar:
Esta función se encarga de combinar las tres cadenas pasadas como parámetro (cadena, MaxDer, MaxIzq), buscando la cadena mas larga valida que se pueda formar en el centro de la cadena completa, por donde se ha partido la cadena, y comprobar cual es el máximo de los tres, la cadena central, la máxima que se ha obtenido de la mitad derecha o la cadena máxima que se ha obtenido de la mitad izquierda. Para calcular la cadena central se tratan todos los casos posibles comprobando si la cadena central esVC o esCV y recorriendo silaba a silaba por ambos lados de la cadena añadiendo silabas validas a la solución hasta que se añada toda la cadena o hasta que no haya más silabas validas que se puedan añadir.