El problema de la función exponencial

Julio Javier Castillo, Diego Javier Serrano
Laboratorio de Investigación de Software MsLabs, Dpto. Ing. en Sistemas de Información, Facultad Regional Córdoba - Universidad Tecnológica Nacional (Argentina)

<jotacastillo@gmail.com>,
<diegojserrano@gmail.com>

Este es el enunciado del problema A de los planteados en la Segunda Competencia de Programación de la Facultad Regional de Córdoba (Universidad Tecnológica Nacional, Argentina) UTN-FRC celebrada el 24 de noviembre de 2010.

Nivel del problema: Sencillo

Se conoce que la función exponencial puede ser definida de múltiples maneras, entre ellas, una definición posible es:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Esta función es de gran importancia en el estudio del plano complejo, y en el campo del análisis, ya que es la única función que es siempre igual a su derivada.

Para su referencia se sabe que:

$e^1 \backsimeq 2,708333333$, aproximado con 5 sumandos

$e^2 \backsimeq 7,380952381$, aproximado con 8 sumandos

Usted debe determinar la suma de los valores de los decimales a, b comprendidos del número e^x (calculado con la aproximación superior). Con a, b, x enteros, y $a \leq b$, calculado e^x como una aproximación de los primeros k - términos según la definición anterior.

Por ejemplo para x=1, k=5, a=1 y b=3, el resultado será 15 pues es la suma de los tres primeros decimales para el cálculo de la función exponencial con los 5 primeros sumandos.

Entrada:
Por cada caso de prueba se reciben cuatro números enteros x, k, a y b, siempre en ese orden. Donde x indica que se desea e^x, k indica que se lo va a calcular con la aproximación de k-sumandos, a indica la posición del primer decimal que se tendrá en cuenta para la suma, y b es la posición del segundo decimal. Se sabe que siempre a<=b.

Salida:
Por cada caso de prueba se debe imprimir un número entero con el valor de la suma de los decimales ubicados entre los lugares a-ésimo y b-ésimo de la función e^x aproximada con los k-primeros sumandos.

Ejemplo:

Entrada:

```
1 5 1 3
2 8 2 4
```

Salida:

```
15
17
```

El problema del decodificador

Julio Javier Castillo, Diego Javier Serrano
Laboratorio de Investigación de Software MsLabs, Dpto. Ing. en Sistemas de Información, Facultad Regional Córdoba - Universidad Tecnológica Nacional (Argentina)

<jotacastillo@gmail.com>, <diegojserrano@gmail.com>

El enunciado de este problema apareció en el número 211 de ***Novática*** (mayo-junio 2011, p. 75).

Recordemos que el problema requería decodificar cierta información provista como entrada mostrando el mensaje decodificado en la salida. Sabiendo que el mensaje original es una frase en español no conteniendo en ningún caso los símbolos '(', ')', '{', '}', '[', ']', ',', y '_'. El mensaje sigue ciertas reglas de codificación definidas en el enunciado.

Solución

El problema se resuelve básicamente realizando el procedimiento inverso al de codificación. La resolución comienza tomando cada una de las subcadenas, que se conoce donde comienzan y terminan puesto que estarán separadas por un espacio.

Luego, cada carácter '_' es reemplazado por una 'p', y de esta forma invertimos el paso 5 del proceso de codificación.

Seguidamente, determinamos la cantidad de palabras que han sido generadas de acuerdo al paso 3, ya que el resultado del mismo determinará si el paso 4 se aplicó o no en el proceso de codificación. Entonces una función en línea nos determinará si los dos primeros y los dos últimos caracteres son iguales entre sí en una cadena determinada. Dicha cadena será siempre de longitud 4, de acuerdo a las restricciones del enunciado.

Si este proceso se aplicó una cantidad par (y positiva a los efectos de evitar las 0-veces), entonces se procede a reemplazar los caracteres 'u' por 'o', 'o' por 'e', 'e' por 'a', y 'a' por 'u'. Nótese que este reemplazo tiene que ser simultáneo, de aquí que se realice dicha conversión a nivel de caracteres y no de reemplazo de subcadenas.

Luego, se recortan las palabras que hayan sido generadas de acuerdo al paso 2, por lo cual tendremos que palabras de 4 caracteres de longitud se convierten en palabras con 2 caracteres de longitud.

A continuación se aplica la inversa de la función F enunciada en el paso 3 del proceso de conversión. De aquí que una palabra $p = a_k a_{k-1} \dots a_1 a_n a_{n-1} \dots a_{k+1}$. se deba convertir a una palabra $p' = a_1 a_2 \dots a_{k-1} a_k a_{k+1} \dots a_n$.

Para ello se debe tomar: el primer carácter, y formar dos subcadenas con los caracteres restantes de la palabra p. Finalmente, estas tres subcadenas se deben ordenar convenientemente para poder generar p'.

De manera análoga se resuelve el caso en el que la cadena presenta una longitud impar.

El último paso consiste en rotar cada uno de las subcadenas para revertir el paso 1 del proceso de codificación.

Finalmente debe notarse que cada una de las subcadenas se manipulan en un vector de *strings*. Esto es conveniente para respetar los espacios que pueden estar presentes en la cadena original, y también nos permite manipular individualmente cada una de estas cadenas facilitando así el proceso de rotación, acortamiento, y reemplazo de caracteres en cadenas.

El código fuente de la solución se provee a continuación :

```
import  java.util.Scanner;

public  class  Main  {

      public  static  void  main(String[]  args)  {
                   Scanner  sc  =  new  Scanner(System.in);
                  String  line="";
                  String  []cads;
                     while(  (line=sc.nextLine()).compareTo("")!=0)
        {
                          cads=line.substring(0,  line.length()-1).split("  ");

                        //Primero se reemplazan _ por p.
                        int  cantidadGeneradas=0;
                        for(int  i=0;  i<cads.length;i++){
                             cads[i]=cads[i].replace('_',  'p');
                              //se cuentan la cantidad de palabras generadas en el pto.3.
                              //Para evitar procesar los "" nos aseguramos que tenga longitud positiva.
                              if(cads[i].length()  >  1  &&  palabraGenerada(cads[i]))
                                 cantidadGeneradas++;
                      }
                         for(int  i=0;  i<cads.length;i++){
                               if(cantidadGeneradas!=0  &&  cantidadGeneradas%2==0  &&
palabraGenerada(cads[i])){
                                     cads[i]=reemplazoCaracteres(cads[i]);
                           }
                              if(palabraGenerada(cads[i]))
                                   cads[i]=palabraRecortada(cads[i]);
                              cads[i]=transformacionF(cads[i]);
                             cads[i]=rotar(cads[i]);
                      }
                        //imprimimos respetando los espacios
                       int  t=0;
                        for(int  i=0;  i<cads.length;i++){
```

```
                                if(cads[i].compareTo("")==0){
                                   System.out.print(' ');
                                  continue;
                                }
                                else{
                                     System.out.print(cads[i]);
                                     if(i+1!=cads.length)
                                         System.out.print(' ');
                                }
                           }
                             System.out.println(".");
                }
        }

         public static String rotar(String c)
        {
               return new StringBuffer(c).reverse().toString();
        }

         public static String palabraRecortada(String c)
        {
              return ""+c.charAt(0)+ c.charAt(3);
        }

         public static boolean palabraGenerada(String c)
        {
             boolean generada=false;
             if(c.length()==4){
                   if(c.charAt(0) == c.charAt(1) && c.charAt(2) == c.charAt(3))
                     generada=true;
                }
              return generada;
        }

         public static String reemplazoCaracteres(String c)
        {
               StringBuilder res=new StringBuilder(c);
               for(int i=0; i<res.length();i++){
                   //reemplazo simultaneo
                   switch(res.charAt(i)){
                      case 'a':
                            res.setCharAt(i, 'u');
                           break;
                      case 'u':
                            res.setCharAt(i, 'o');
                           break;
                      case 'o':
                            res.setCharAt(i, 'i');
                           break;
                      case 'i':
                            res.setCharAt(i, 'e');
                           break;
                      case 'e':
                            res.setCharAt(i, 'a');
                           break;
                    }
                }
                 return res.toString();
        }

         public static String transformacionF(String c)
        {
              String res="";
             int k=0;
               if(c.compareTo("")==0) return "";

               if(c.length()%2==0)
             {
                   k=c.length()/2;
                    k=k-1; //posicion del caracter
                     //recordar que aqui debemos aplicar la F-1, i.e: la funcion de decodificacion:
                       res=rotar(c.substring(1, k+1)) + c.charAt(0) +  rotar(c.substring(k+1, c.length()));

             }
             else{
                   k=((c.length()-1)/2) + 1;
                   k=k-1; //posicion del caracter
                      res= rotar(c.substring(0, k)) + c.charAt(k)+ rotar(c.substring(k+1, c.length()));
             }
              return res;
        }
}
```