Ejercicios Tema 5: Backtracking

Jorge Guillamón y Miguel Ángel Guerrero

16 de junio de 2020

Resumen

Ejercicios 3 y 6 del tema 5.

Ejercicio 3

Se tiene un número de Tam cifras almacenado en una cadena de texto; por ejemplo, la cadena dato = 1151451.
Diseñar un algoritmo que mediante técnicas de Backtracking encuentre, de la manera más eficiente posible, todos los números distintos de N cifras que puedan formarse con los números de la cadena sin alterar su orden relativo dentro de la misma.
Por ejemplo, si N = 4, son números válidos 1151, 1511 y 1541, pero no 4551 o 5411 que aunque pueden formarse con los dígitos de la cadena dato implican una reordenación.

```
18  original = input("Cadena: ")
19  n = int(input("Número de caracteres de los resultados: "))
20
21  #Se usara una pila en lugar de recursividad
22  pila = [[]]
23  resultado = set() #Para eliminar resultados duplicados
24
25
26  while len(pila) != 0:
27      actual = pila.pop()
28      ultimo = -1 if len(actual) == 0 else actual[-1] #Correccion de caso ini
29      for i in range(ultimo +1, len(original) - (n - len(actual)) + 1): #Desd
30          nuevo = actual.copy()
31          nuevo.append(i)
32          if(len(nuevo) == n): #Tenemos un nuevo resultado
33              res = ""
34              for i in nuevo:
35                  res+=original[i]
36              resultado.add(res)
37          else: #Hay que seguir elaborando
38              pila.append(nuevo)
39
40  print(str(len(resultado)) + " resultados distintos: " + str(resultado))
```

En primer lugar, hemos manejado el recorrido del árbol mediante una pila en memoria donde los elementos de la pila son subcadenas de la cadena original. Estas subcadenas se representan usando índices de la cadena original.
Al seleccionar un elemento de la pila se desarrollan sus nodos hijos y se apilan o se guardan como resultado. Mediante los límites superiores e inferiores del `for` (linea 29) nos hemos asegurado que solo se desarrollan los nodos que darán al menos una combinación válida.

Ejercicio 6

Se tiene la tabla de sustitución que aparece a continuación

	a	b	c	d
a	b	b	a	d
b	c	a	d	a
c	b	a	c	c
d	d	c	d	b

que se usa de la manera siguiente: en una cadena cualquiera, dos caracteres consecutivos se pueden sustituir por el valor que aparece en la tabla, utilizando el primer carácter como fila y el segundo carácter como columna. Por ejemplo, se puede cambiar la secuencia ca por una b, ya que M[c,a]=b. Implementar un algoritmo Backtracking que, a partir de una cadena no vacía texto y utilizando la información almacenada en una tabla de sustitución M, sea capaz de encontrar la forma de realizar las sustituciones que permite reducir la cadena texto a un carácter final, si es posible.

Ejemplo: Con la cadena texto=acabada y el carácter final=d, una posible forma de sustitución es la siguiente (las secuencias que se sustituyen se marcan para mayor claridad): aca*ba*da → a*ca*cda → abc*da* → *ab*cd → b*cd* → *bc* → d.

```
def sustituye(str0):
    matriz = [['b','b','a','d'],['c','a','d','a'],['b','a','c','c'],['d','c','d','b']]
    a = ord(str0[0]) - 97
    b = ord(str0[1]) - 97
    return matriz[a][b]
```

Figura 1: Función matriz

```
1  #¿Es la cadena reducible?
2  for i in s:
3      if(ord(i) - 97 >  4): # Si mayores que 'd' en ASCII
4          print("La cadena es irreducible con esta matriz")
5          exit(-1)
6
7  pila = [s] #Usaremos una pila en memoria para backtracking
8  res = []
9
10 while len(pila) != 0:
11     actual = pila.pop()
12     for i in range(len(actual) - 1):
13         aux = actual[:i] + sustituye(actual[i:i+2]) + actual[i+2:]
14         if(len(aux) == 1): #Uno más al resultado
15             res.append(aux)
16         else: #Toca seguir reduciendo
17             pila.append(aux)
18
19 print(str(len(res)) + " resultados: " + str(res))
```

Figura 2: Funcion principal

La estrategia empleada para este ejercicio ha sido muy similar a la anterior aunque en esta ocasión hemos usado una lista en lugar de un set para guardar los resultados.
En la línea 2 se puede apreciar un pequeño control para detectar cadenas no reducibles.