Algoritmia básica - Practica 2 (2016), pp. 1–2
Manuel Lagunas Arto

¿Cuánto cuesta transformar una cadena en palíndromo?

Manuel Lagunas[1]

[1]Universidad de Zaragoza, Algoritmia básica

Abstract

El siguiente documento contiene la información referente a la práctica 2 de la asignatura de Algoritmia básica. El objetivo de esta es calcular el coste entendido como inserciones, borrados o cambios de letras en una cadena para crear un palíndromo. Las siguientes lineas exponen la estrategia seguida para su desarrollo asi como las decisiones tomadas. El código ha sido desarrollado por el propio autor de este documento.

En el documento se usa $len(x)$ para hacer referencia a la longitud de x.

El código se encuentra disponible de manera abierta en : https://github.com/mlagunas/Palindrome_creation

1. Menor distancia de edición

Para obtener un palíndromo dada una palabra, esta tiene que ser transformada de manera que su primera mitad sea igual a la segunda pudiendo existir un carácter intermedio. Para transformar una palabra en otra se ha hecho uso de la mínima distancia de edición considerando la posibilidad de añadir, borrar o intercambiar caracteres e introduciendo como entrada dos subcadenas de la palabra en cuestión. Para ello se ha empleado el algoritmo de Levenshtein*. La matriz empleada para guardar las sub soluciones del problema tiene el tamaño de las longitudes de las cadenas de entrada más uno $DP = [len(str1)+1][len(str2)+1]$ considerando la primera fila y columna como la cadena vacía cuyo valor será $len(str2_j)$ si $i = 0$ y $len(str1_i)$ si $j = 0$. El resto de posiciones (i, j) almacenan la mínima distancia de edición para las subcadenas $str1_i$ y $str2_j$. El resultado final se encontrará en la posición $[len(str1)][len(str2)]$ (se considera la primera posición con indice 0). Teniendo la fila y columnas de índice 0 con su resultado ya obtenido habrá que continuar calculando el resto. Para ello se tienen en cuenta las tres operaciones, sabiendo que cada índice de la matriz está representando una subcadena de las cadenas de entrada se puede deducir que los cambios en el eje x representan inserciones, mientras que los cambios en el eje y serán borrados. Si se considera la diagonal serán sustituciones si su valor es mayor al actual o se mantendrán las mismas cadenas de caracteres si su valor es el mismo. Para poder rellenar la matriz, será necesario comparar entonces los resultados obtenidos anteriormente para obtener el mínimo coste posible teniendo la siguiente fórmula:

Tras acabar el proceso se obtiene una matriz con las mismas características que la que se muestra a continuación. Dado el proceso

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

Figure 1: *Fórmula de la distancia de Levenshtein. Imágen obtenida de wikipedia.*

y su necesidad de almacenar la información relativa a todas las subcadenas posibles para su reconstrucción, el coste asintótico tanto en espacio como en tiempo es $O(n^2)$, aunque es posible realizar con coste $O(n)$ en espacio siendo n la longitud de la cadena más larga. Para realizarlo de esta forma se irán actualizando los valores sobre el mismo vector perdiendo los previamente calculados, por ello, no es viable con esta solución.

	_	p	a	l	i	n
_	0	1	2	3	4	5
o	1	1	2	3	4	5
m	2	2	2	3	4	5
o	3	3	3	3	4	5
r	4	4	4	4	4	5
d	5	5	5	5	5	5

Table 1: *Ejemplo de matriz con las mínimas distancias de edición.*

1.1. Reconstruyendo las operaciones

Una vez se ha obtenido la tabla con los cálculos de cada subproblema, obteniendo también la mínima distancia de edición será

* Teoría usada: https://web.stanford.edu/class/cs124/lec/med.pdf

necesario reconocer los pasos realizados con el fin de poder modificar la palabra de entrada hasta que sea un palíndromo. Para ello, de nuevo hay que recordar que los cambios en el eje *y* son borrados y en el eje *x* inserciones mientras que las diagonales, dependiendo de su valor, son reemplazamientos o simplemente se mantiene como estaba. Se comienza en la posición de la esquina inferior derecha (donde se obtiene la mínima distancia de edición) y se comparan el valor de las inserciones, borrados y reemplazamientos o mantenerlo obteniendo el mínimo, que será la operación realizada. Se debe iterar hasta alcanzar la posición de inicio (0,0), punto en el que se obtiene las instrucciones a realizar, pudiendo existir más de un camino válido. El algoritmo usado almacena una letra para cada operación correspondiendo la S a sustituciones, A a añadir y D a borrados, si se encuentra un espacio en blanco se indicará que el carácter debe permanecer como estaba, teniendo al final una cadena con longitud igual al número de pasos a realizar.

	_	p	a	l	i	n
_	***0***	1	2	3	4	5
o	1	***1***	2	3	4	5
m	2	2	***2***	3	4	5
o	3	3	3	***3***	4	5
r	4	4	4	4	***4***	5
d	5	5	5	5	5	***5***

Table 2: *Camino seguido para la obtención de las operaciones realizadas, que en este caso son 5 reemplazos de caracteres. No tiene porque ser la diagonal siempre*

2. Obtención del palíndromo

Se tienen las operaciones realizadas como una cadena de caracteres, ahora el siguiente paso será aplicarlas a la primera subcadena de manera que esta se transforme y se obtenga un palíndromo. Para ello simplemente se leen las operaciones en orden inverso (están ordenadas de ultimo a primero) aplicándolas a la primera subcadena, teniendo en cuenta el valor de los índices en operaciones como añadir o borrar para que apunten al carácter en cuestión de ambas. Al acabar la transformación la cadena de entrada se habrá convertido en un palindromo

3. Resultados

Con el fin de obtener la mínima distancia de edición para crear el palíndromo se han probado todas las posibles subcadenas de la palabra inicial quedándonos tan solo con la configuración que devuelve el menor numero de pasos en la distancia de edición. Obteniendo los siguientes resultados:

```
Numero minimo: 8
Cadena original: algoritmia basica
Cambiar pos 2 por c: acgoritmia basica
Cambiar pos 3 por i: acioritmia basica
Cambiar pos 4 por s: acisritmia basica
Cambiar pos 5 por a: acisaitmia basica
Añadir b en pos 5: acisabitmia basica
Añadir   en pos 6: acisab itmia basica
Añadir a en pos 7: acisab aitmia basica
Cambiar pos 10 por m: acisab aimmia basica
acisab aimmia basica
es Palindromo? True

Numero minimo: 11
Cadena original: Esto es una cadena larga
Cambiar pos 1 por a: asto es una cadena larga
Cambiar pos 2 por g: agto es una cadena larga
Cambiar pos 3 por r: agro es una cadena larga
Cambiar pos 4 por a: agra es una cadena larga
Añadir l en pos 4: agral es una cadena larga
Cambiar pos 7 por a: agral as una cadena larga
Cambiar pos 8 por n: agral an una cadena larga
Cambiar pos 9 por e: agral aneuna cadena larga
Cambiar pos 10 por d: agral anedna cadena larga
Borrar posición 11: agral aneda cadena larga
Cambiar pos 13 por c: agral anedaccadena larga
agral anedaccadena larga
es Palindromo? True

Numero minimo: 4
Cadena original: 12342331234
Cambiar pos 1 por 4: 42342331234
Borrar posición 2: 4342331234
Borrar posición 4: 432331234
Añadir 1 en pos 5: 4321331234
4321331234
es Palindromo? True
```