

Secuencia creciente dentro de una matriz cuadrada

María José Niño Rodríguez¹

David Santiago Quintana Echavarria¹

¹Departamento de Ingeniería de Sistemas, Pontificia Universidad Javeriana
Bogotá, Colombia
{ma.nino, quintanae-david}@javeriana.edu.co

29 de septiembre de 2022

Resumen

En este documento se presenta la formalización del problema de la secuencia creciente más larga dentro de una matriz cuadrada implementando la estrategia de programación dinámica. **Palabras clave:** programación dinámica, matrices, secuencia, algoritmo, optimización, maximizar.

Índice

1. Introducción	1
2. Formalización del problema	1
2.1. Definición del problema de “Secuencia creciente dentro de una matriz cuadrada”	1
3. Algoritmo de solución	2
3.1. Algoritmo de programación dinámica	2
4. Experimentación del algoritmo	2

1. Introducción

Los algoritmos de programación dinámica nos ayudan a optimizar procesos, evita el recalcular de operaciones ya calculadas previamente, evita la concurrencia y se implementa en problemas comúnmente de optimización, ya sea maximizando o minimizando. En este documento se expone un problema que busca maximizar el tamaño de una secuencia dentro de una matriz cuadrada, con el objetivo de mostrar: la formalización del problema (sección 2), la escritura formal de este algoritmo (sección 3).

2. Formalización del problema

A partir de una matriz cuadrada natural (\mathbb{N}) de tamaño $N \times N$ que contiene los números únicos en el rango $[1, N \times N]$ que no tiene ningún orden, se quiere obtener la secuencia más larga de vecinos que tengan la relación de orden parcial $a < b$ y que además tengan una diferencia de 1.

2.1. Definición del problema de “Secuencia creciente dentro de una matriz cuadrada”

Así, el problema se define a partir de:

1. Una matriz cuadrada natural (\mathbb{N}) de tamaño $N \times N$

para producir una nueva secuencia de números ordenados.

- Entradas:
 - Una matriz $Mde \in \mathbb{N}$ de $N \times N$.
- Salidas:
 - Una secuencia $s' \in \mathbb{N}$ que tengan una relación de orden parcial $a < b$ con una diferencia de 1 entre a y b .

3. Algoritmo de solución

3.1. Algoritmo de programación dinámica

- Formula de Recurrencia:

$$BC(i, j) = \begin{cases} 0; i < 0 \vee i \geq n \vee j < 0 \vee j \geq n \\ \max(1 + BC(i, j + 1), \\ (1 + BC(i, j - 1), \\ (1 + BC(i - 1, j), \\ (1 + BC(i + 1, j))) \end{cases}$$

Algoritmo 1 Secuencia creciente dentro de una matriz cuadrada

```

1: procedure SECUENCIALARGA( $m, n$ )
2:    $maxsecuencia \leftarrow 1$ 
3:    $memo : [1...|m| + 1] \times [1...|n| + 1] \leftarrow -1$ 
4:    $tabIndices : [1...|m| + 1] \times [1...|n| + 1] \leftarrow [0, 0]$ 
5:   for  $i \leftarrow 1$  to  $n$  do
6:     for  $j \leftarrow 1$  to  $n$  do
7:       if  $memo[i][j] = -1$  then
8:          $aux \leftarrow buscCamino(i, j, m, memo, tabIndices, n)$ 
9:       end if
10:      if  $memo[i][j] \neq maxsecuencia$  then
11:         $maxsecuencia \leftarrow memo[i][j]$ 
12:         $maxIndiceX \leftarrow i$ 
13:         $maxIndiceY \leftarrow j$ 
14:      end if
15:    end for
16:  end for
17:  while  $maxIndiceX \neq 0 \wedge maxIndiceY \neq 0$  do
18:     $aux \leftarrow maxIndicesX$ 
19:     $maxIndiceX \leftarrow maxIndiceX + tabIndices[maxIndicesX, maxIndicesY].x$ 
20:     $maxIndiceY \leftarrow maxIndiceY + tabIndices[aux, maxIndicesY].y$ 
21:  end while

```

4. Experimentación del algoritmo

Se validó la solución del problema con matrices cuya solución ya era conocida, se comparaba la salida del programa con esta respuesta, ambos resultados fueron iguales. Se experimentaron con matrices variando el tamaño, desde 10×10 , hasta 100×100 , en saltos de 10. Cada matriz en el experimento en un principio fue llenada en orden con números desde 1 hasta n , seguido se intercambiaban posiciones de manera aleatoria,

Algoritmo 2 Caso Base

```
1: procedure BUSCCAMINO( $i, j, m, memo, tabIndices, n$ )
2:   if  $i \leq 0 \mid i \geq n \mid j \leq 0 \mid j \geq n$  then
3:     return 0
4:   end if
5:   if  $memo[i][j] \neq -1$  then
6:      $memo[i][j]$ ;
7:   end if
8:    $der \leftarrow -1$ 
9:    $izq \leftarrow -1$ 
10:   $arriba \leftarrow -1$ 
11:   $abajo \leftarrow -1$ 
12:  if  $(j + 1 = n)$  then
13:    if  $((m[i][j] + 1) = m[i][j + 1])$  then
14:       $tabIndices[i][j].x \leftarrow 1$ 
15:       $tabIndices[i][j].y \leftarrow 0$ 
16:       $der \leftarrow 1 + buscCamino(i, j + 1, m, memo, tabIndices, n)$ 
17:    end if
18:  end if
19:  if  $(j - 1 = 0)$  then
20:    if  $((m[i][j] + 1) = m[i][j - 1])$  then
21:       $tabIndices[i][j].x \leftarrow -1$ 
22:       $tabIndices[i][j].y \leftarrow 0$ 
23:       $izq \leftarrow 1 + buscCamino(i, j - 1, m, memo, tabIndices, n)$ 
24:    end if
25:  end if
26:  if  $(i - 1 = 0)$  then
27:    if  $((m[i][j] + 1) = m[i - 1][j])$  then
28:       $tabIndices[i][j].x \leftarrow 0$ 
29:       $tabIndices[i][j].y \leftarrow -1$ 
30:       $arriba \leftarrow 1 + buscCamino(i - 1, j, m, memo, tabIndices, n)$ 
31:    end if
32:  end if
33:  if  $(i + 1 = n)$  then
34:    if  $((m[i][j] + 1) = m[i + 1][j])$  then
35:       $tabIndices[i][j].x \leftarrow 0$ 
36:       $tabIndices[i][j].y \leftarrow 1$ 
37:       $abajo \leftarrow 1 + buscCamino(i + 1, j, m, memo, tabIndices, n)$ 
38:    end if
39:  end if
40: end procedure
```

$n \times 4$ veces. Con la matriz de prueba generada se llamaba a la función para generar la secuencia, analizando las matrices original, memoización y backtracking en cada caso los resultados generados por el algoritmo fueron los mismos que se esperaban, esto en cada caso genero correctamente la subsecuencia como se evidencia en la siguiente figura.

```

La secuencia mas larga en una matriz de tamaño 4x4 es: 2 3 4
5 6 7 8 9 10

La secuencia mas larga en una matriz de tamaño 10x10 es: 4 5
6 7 8 9

La secuencia mas larga en una matriz de tamaño 20x20 es: 341
342 343 344 345 346 347 348 349 350 351 352 353 354

La secuencia mas larga en una matriz de tamaño 30x30 es: 277
278 279 280 281 282 283 284 285 286 287 288 289 290 291 292
293 294 295 296 297 298 299 300

La secuencia mas larga en una matriz de tamaño 40x40 es: 369
370 371 372 373 374 375 376 377 378 379 380 381 382 383 384
385 386 387 388 389 390 391 392 393 394

La secuencia mas larga en una matriz de tamaño 50x50 es: 220
1 2202 2203 2204 2205 2206 2207 2208 2209 2210 2211 2212 221
3 2214 2215 2216 2217 2218 2219 2220 2221 2222 2223 2224 222
5 2226 2227 2228 2229 2230 2231 2232 2233 2234 2235 2236 223
7

La secuencia mas larga en una matriz de tamaño 60x60 es: 721
722 723 724 725 726 727 728 729 730 731 732 733 734 735 736
737 738 739 740 741 742 743 744 745 746 747 748 749 750 751
752 753 754 755 756 757 758 759 760 761 762 763 764 765 766
767

La secuencia mas larga en una matriz de tamaño 70x70 es: 1 2
3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44
45 46 47 48 49 50 51 52 53

La secuencia mas larga en una matriz de tamaño 80x80 es: 105
106 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 106
2 1063 1064 1065 1066 1067 1068 1069 1070 1071 1072 1073 107
4 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085 108
6 1087 1088 1089 1090 1091 1092 1093 1094 1095 1096 1097 109
8 1099 1100 1101 1102 1103 1104 1105 1106 1107 1108

La secuencia mas larga en una matriz de tamaño 90x90 es: 155
1 1552 1553 1554 1555 1556 1557 1558 1559 1560 1561 1562 156
3 1564 1565 1566 1567 1568 1569 1570 1571 1572 1573 1574 157
5 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 158
7 1588 1589 1590 1591 1592 1593 1594 1595 1596 1597 1598 159
9 1600 1601 1602 1603 1604 1605 1606 1607 1608 1609 1610 161
1 1612 1613 1614 1615 1616 1617 1618 1619 1620

La secuencia mas larga en una matriz de tamaño 100x100 es: 1
1233 1234 1235 1236 1237 1238 1239 1240 1241 1242 1243 1244 1
245 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1
257 1258 1259 1260 1261 1262 1263 1264 1265 1266 1267 1268 1
269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279 1280 1
281 1282 1283 1284 1285 1286 1287 1288 1289 1290 1291 1292 1
293 1294 1295 1296 1297 1298 1299 1300

```

Figura 1: Resultados del experimento con matrices de tamaño 10x10 hasta 100x100