Universidad Nacional Autónoma de México

Facultad de Ciencias

Modelado y Programación

Tercer Problema Laboratorio | Suma minima

López Molina Andrés Daniel | 319117026

10 de septiembre de 2024

1. Resolución del problema

Para la resolución de problema, primero hacemos el caso en donde la matriz esta vacía para evitar conflictos en la resolución.

Una vez establecido lo anterior, dada la matriz de $n \times n$ pensé en ocupar el algoritmo de Dijkstra pero este no se adapto a las necesidades del problema. Por otra parte, investigue y quise adaptar el algoritmo de Kadane, sin embargo, tampoco se adapto a las necesidades del problema.

Al leer los algoritmos anteriores y escuchar los hints dados en clase, me dio una idea para realizar el ejercicio. Para ello, esa idea fue realizar una matriz temporal (dp) inicializada con valores 0 en donde guardara los costos de cada suma de índice de la matriz de manera que se fueran calculando con la función a realizar. Una vez dado eso, llenamos la matriz temporal con el valor del índice (0,0) de la matriz dada y posteriormente llenamos la primera fila y la primera columna.

Ahora, el problema radicaba en como llenar el resto de la matriz que no estaba en la primera fila o columna. No obstante, un detalle muy importante es que para obtener los otros valores de costos, podemos llegar desde la celda de arriba o desde la celda a la izquierda de manera que para cada celda, tomamos el valor mínimo entre la celda de arriba y la de la izquierda y posteriormente le sumamos el valor del índice de la matriz dada. Así pues, haciéndolo iterativamente íbamos a llegar al ultimo valor (n,n).

He aquí un ejemplo de la explicación anterior para un mejor detalle:

- Matriz dada:

$$\text{m} = \begin{array}{|c|c|c|}\hline 1 & 3 & 1 \\ \hline 1 & 5 & 1 \\ \hline 4 & 2 & 1 \\ \hline \end{array}$$

- Matriz temporal:

$$\text{dp} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

- Inicialización de la celda de inicio:

$$\text{dp} = \begin{array}{|c|c|c|}\hline 1 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

- Llenando de la primera fila con $dp[0][i] = dp[0][i-1] + matriz[0][i]$:

1. $dp[0][1] = dp[0][0] + matriz[0][1] = 1 + 3 = 4$
2. $dp[0][2] = dp[0][1] + grid[0][2] = 4 + 1 = 5$

$$\text{dp} = \begin{array}{|c|c|c|}\hline 1 & 4 & 5 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

- Llenando de la primera columna con $dp[i][0] = dp[i-1][0] + matriz[i][0]$:

1. $dp[1][0] = dp[0][0] + matriz[1][0] = 1 + 1 = 2$
2. $dp[0][2] = dp[1][0] + matriz[2][0] = 2 + 4 = 6$

$$\text{dp} = \begin{array}{|c|c|c|}\hline 1 & 4 & 5 \\ \hline 2 & 0 & 0 \\ \hline 6 & 0 & 0 \\ \hline \end{array}$$

- Finalmente llenamos el resto de la matriz que no están en la primera fila o columna. Para ello, para cada celda (i, j), tomamos el valor mínimo entre $dp[i-1][j]$ (celda de arriba) y $dp[i][j-1]$ (celda de la izquierda), y le sumamos el valor de $matriz[i][j]$:

1. Para la celda (1,1): $dp[1][1] = min(dp[0][1], dp[1][0]) + matriz[1][1] = min(4, 2) + 5 = 2 + 5 = 7$
2. Para la celda (1,2): $dp[1][2] = min(dp[0][2], dp[1][1]) + matriz[1][2] = min(5, 7) + 1 = 5 + 1 = 6$
3. Para la celda (2,1): $dp[2][1] = min(dp[1][1], dp[2][0]) + matriz[2][1] = min(7, 6) + 2 = 6 + 2 = 8$
4. Para la celda (2,2): $dp[2][2] = min(dp[1][2], dp[2][1]) + matriz[2][2] = min(6, 8) + 1 = 6 + 1 = 7$

$$\text{dp} = \begin{array}{|c|c|c|}\hline 1 & 4 & 5 \\ \hline 2 & 7 & 6 \\ \hline 6 & 8 & 7 \\ \hline \end{array}$$

- Y así, obtenemos el valor del costo mínimo para llegar a la esquina inferior derecha.

2. Ejemplos que se probaron en el programa

```
1
2  import pytest
3  from sumaMinima import caminoSumaMin
4
5  def test_sumaMinima4x4():
6      cuad = [[1,1,2,2],[1,1,3,6],[5,10,1,1],[7,7,4,1]]
7      result = caminoSumaMin(cuad)
8      assert result == 9
9
10 def test_sumaMinima3x3():
11     cuad = [[1, 4, 5],[2, 7, 6],[6, 8, 7]]
12     result = caminoSumaMin(cuad)
13     assert result == 23
14
15 def test_sumaMinima1x1():
16     cuad = [[1]]
17     result = caminoSumaMin(cuad)
18     assert result == 1
19
```

```
20 def test_sumaMinimaError1():
21     cuad = [[1,1,2,2],[1,1,3,6],[5,10,1,1],[7,7,4,1]]
22     result = caminoSumaMin(cuad)
23     assert result != 8
24 
25 def test_sumaMinimaError2():
26     cuad = [[1,1],[1,1]]
27     result = caminoSumaMin(cuad)
28     assert result != 4
29 
30 def test_sumaMinimaError2():
31     cuad = []
32     result = caminoSumaMin(cuad)
33     assert result == 0
```

Listing 1. Pruebas que se hicieron para la comprobacion