Programación Dinámica: Ejercicio 1

Se dispone de una matriz M de tamaño FxC (F es la cantidad de filas y C la cantidad de columnas), cuyas celdas tienen un valor numérico entero positivo. Por ejemplo, la matriz con 4 filas y 5 columnas siguiente:

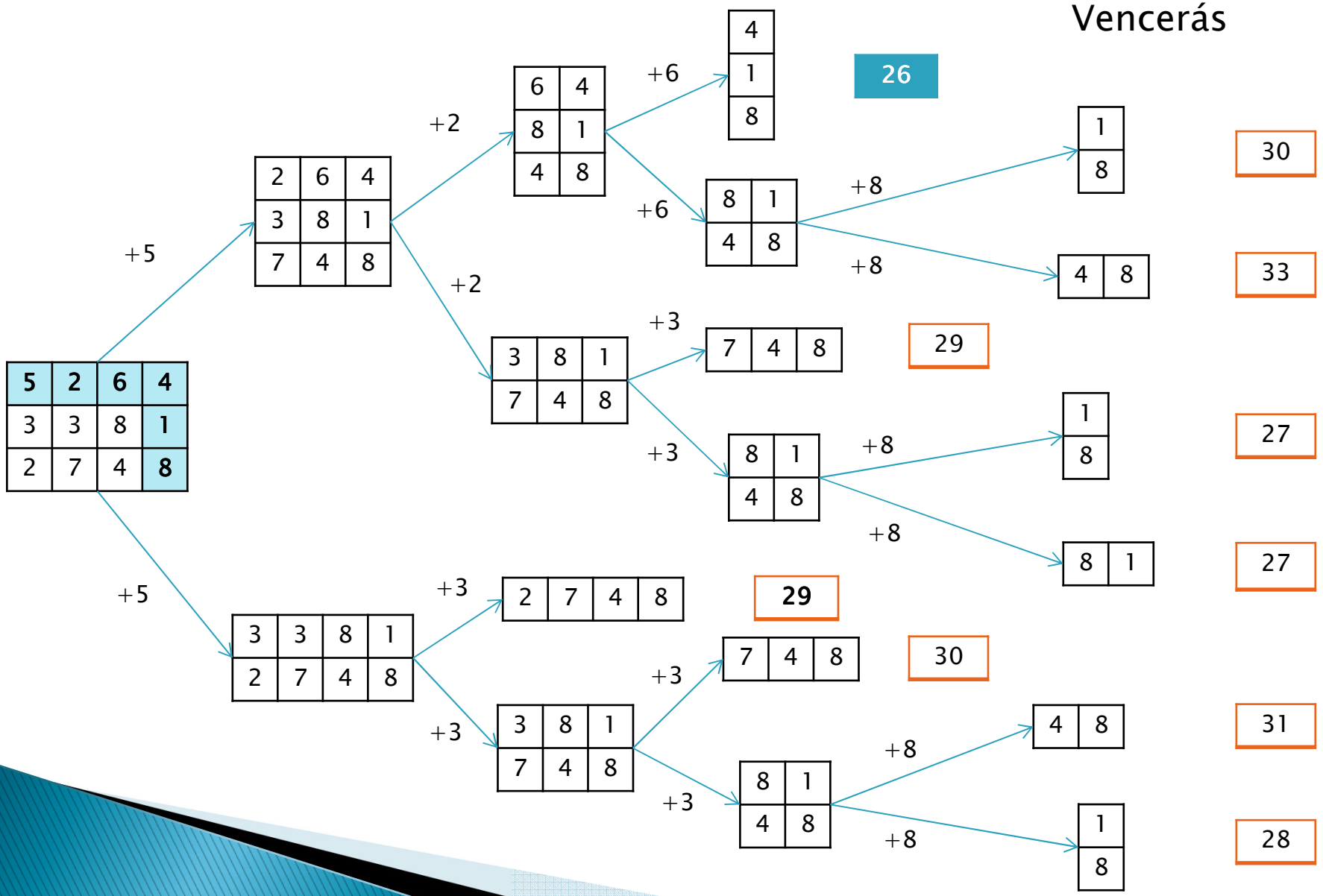
3	2	3	2	5
1	2	4	5	6
4	3	6	2	4
2	5	3	4	3

Un movimiento entre las casillas Mij y Mpq es válido solamente si (p=i && q=j+1) o bien si (p=i+1 && q=j). De denomina camino de la matriz como la sucesión de movimientos que llevan de la casilla M_{11} a la casilla MFC y el coste de un camino es igual a la suma de los valores de las casillas que recorre.

Diseñar un algoritmo de Programación Dinámica que obtenga el menor de los costes de todos los caminos de una matriz dada como parámetro.

Programación Dinámica: Ejercicio 1

Ejemplo 1:
Solución de Divide y Vencerás



Programación Dinámica: Ejercicio 1

- Contenido de la matriz: M[NF][NC]
- Solo válidos movimientos de avance en fila o en columna
- El coste de cada movimiento se corresponde a la suma de la opción de menor coste: avanzar en fila o avanzar en columna
- Guardo en una matriz auxiliar el coste asociado a cada posición de la matriz original: C[NF][NC]
- Para conocer los movimientos, recorro la matriz auxiliar desde el final, una vez se ha calculado completamente: Mov[1…2][1…NF+NC-1]

Programación Dinámica: Ejercicio 1

Ejemplo 1:
Solución de
Programación Dinámica

- NF = 3, NC =4

M

5	**2**	**6**	**4**
3	3	8	**1**
2	7	4	**8**

C

5	**7**	**13**	**17**
8	10	18	**18**
10	17	21	**26**

- Coste mínimo: 26
- Camino mínimo:
  - Movimientos necesarios: NMOV = NF + NC – 2 = 5
  - Celdas recorridas: NCeldas = NF + NC – 1 = 6
  - Camino mínimo: (1,1)→(1,2)→(1,3)→(1,4)→(4,2)→(4,3)

Programación Dinámica: Ejercicio 1

Ejemplo 2:
Solución de
Programación Dinámica

- NF = 4, NC =5

M

3	**2**	**3**	**2**	5
1	2	4	**5**	6
4	3	6	**2**	**4**
2	5	3	4	**3**

C

3	**5**	**8**	**10**	15
4	6	10	**15**	21
8	9	15	**17**	**21**
10	14	17	21	**24**

- Coste mínimo: 24
- Camino mínimo:
  - Movimientos necesarios: NMOV = NF + NC – 2 = 7
  - Celdas recorridas: NCeldas = NF + NC – 1 = 8
  - Camino mínimo: (1,1)→(1,2)→(1,3)→(1,4)→(2,4)→(3,4)→(3,5)→(4,5)

Programación Dinámica: Ejercicio 1

- Cómo rellenar la matriz de costes auxiliar: C[i][j]
  - Inicialización de la primera casilla:
    - $C[1][1] = M[1][1]$
  - Inicialización de la primera columna:
    - $C[i][1] = M[i][1] + C[i-1][1]$ $\quad \forall\ i$
  - Inicialización de la primera fila:
    - $C[1][j] = M[1][j] + C[1][j-1]$ $\quad \forall\ j$
  - Resto de la matriz:
    - $C[i][j] = M[i][j] + \text{mínimo}\{ C[i-1][j], C[i][j-1] \}$ $\quad \forall\ i > 1, j > 1$

Programación Dinámica: Ejercicio 1

```
const NF, NC = ...
tipos matriz= array[1... NF] [1...NC] de entero
tipos movimientos= array[1... 2] [NF+NC-1] de entero //Para guardar los movimientos
fun CostesMatriz (E M: matriz; E/S C: matriz, E/S Mov: movimientos) dev Coste: entero
    var i, j, nmov: entero
    var M: matriz
    C [1] [1] = M [1] [1]
    desde i=2 hasta NF hacer   C [ i ] [1] = M [ i ] [1] + C [ i - 1] [1]  fdesde //Primera columna
    desde j=2 hasta NC hacer   C [1] [ j ] = M [1] [ j ] + C [1] [ j - 1]  fdesde //Primera fila
    desde i=2 hasta NF hacer //Resto de la matriz
        desde j=2 hasta NC hacer
            C [ i ] [ j ] = M [ i ] [ j ] + Mínimo { C [ i - 1] [ j ] , C [ i ] [ j - 1] }
        fdesde
    fdesde
    Mov [1] [NF+NC-1] = NF ; Mov [2] [NF+NC-1] = NC  ; i = NF; j = NC; //La última casilla es (NF,NC)
    desde nmov = NF+NC-2 hasta 1, con nmov = nmov – 1 //Movimientos desde (NF,NC) hacia atrás
        si  ( ( C [ i – 1] [ j ] < = C [ i ] [ j – 1] ) y ( i > 1) ) entonces
            i = i – 1
        sino si ( ( i ==1 ) o ( j > 1 ) ) entonces
            j = j – 1
        fsi
        Mov [1] [ nmov ] = i ; Mov [2][ nmov ] = j //Guardo las casillas desde las que me muevo
    fdesde
    devolver  C [NF] [NC]
Ffun
```