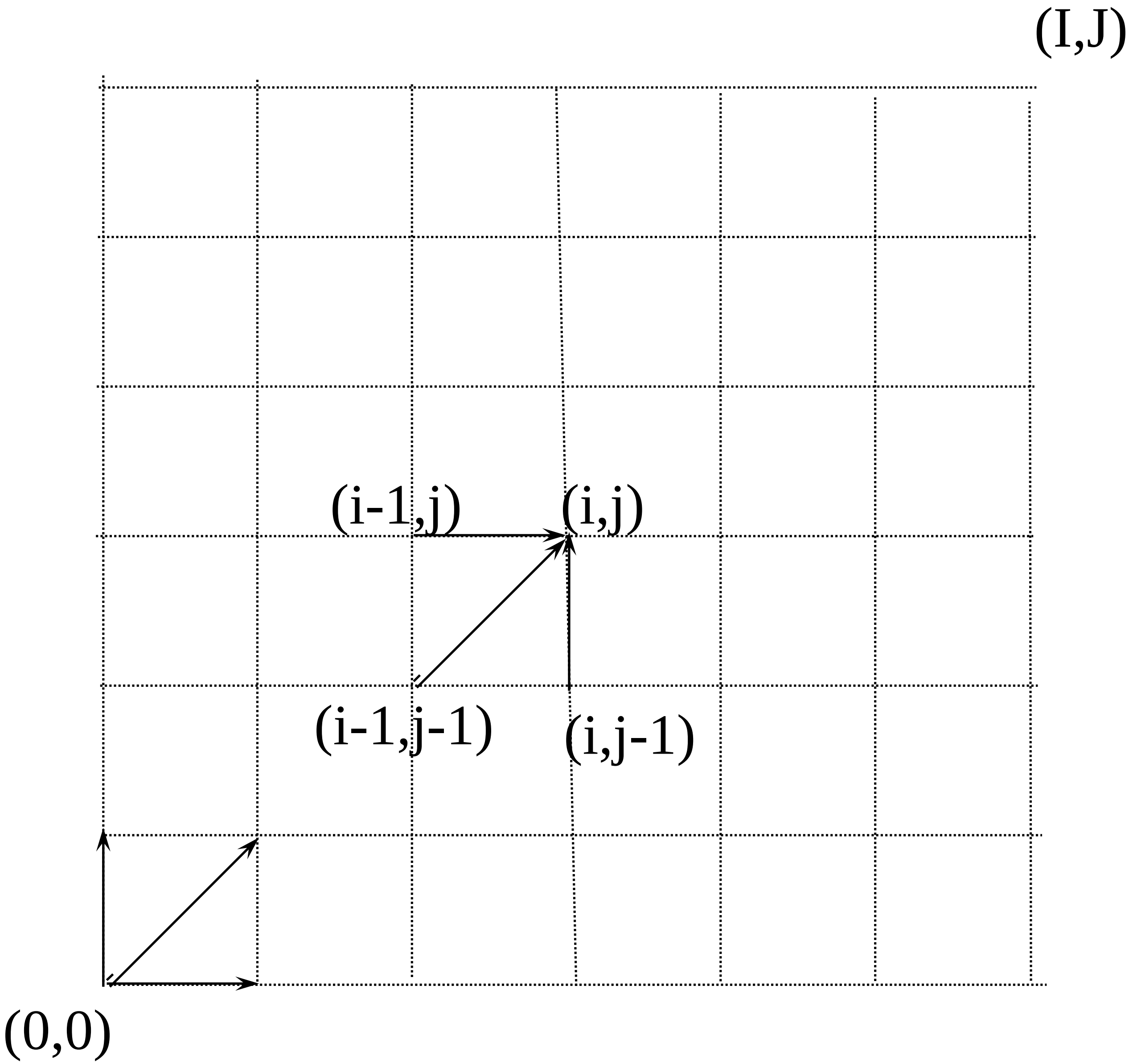
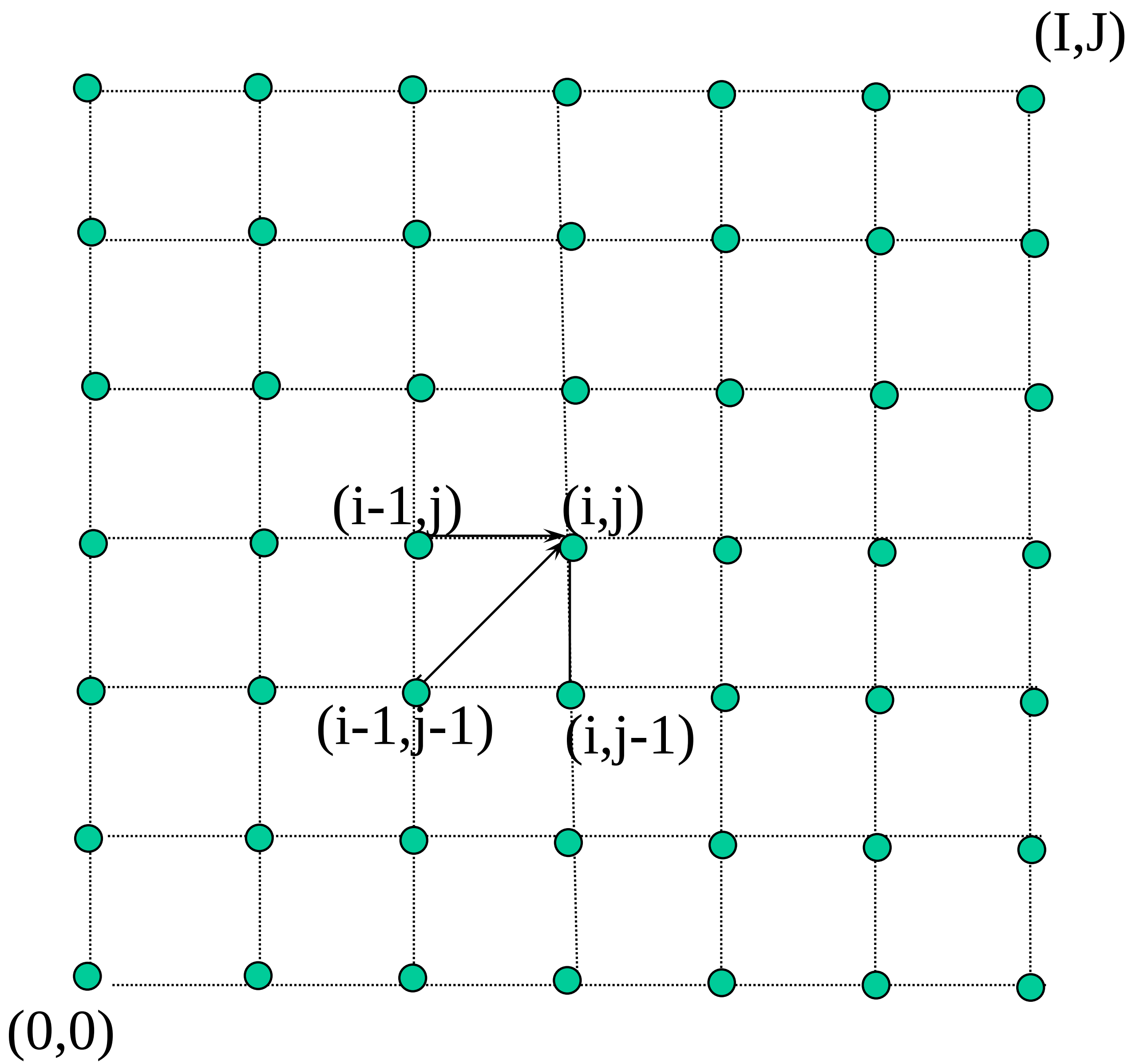
Problema: Encontrar el camino más corto entre (0,0) y (I,J)

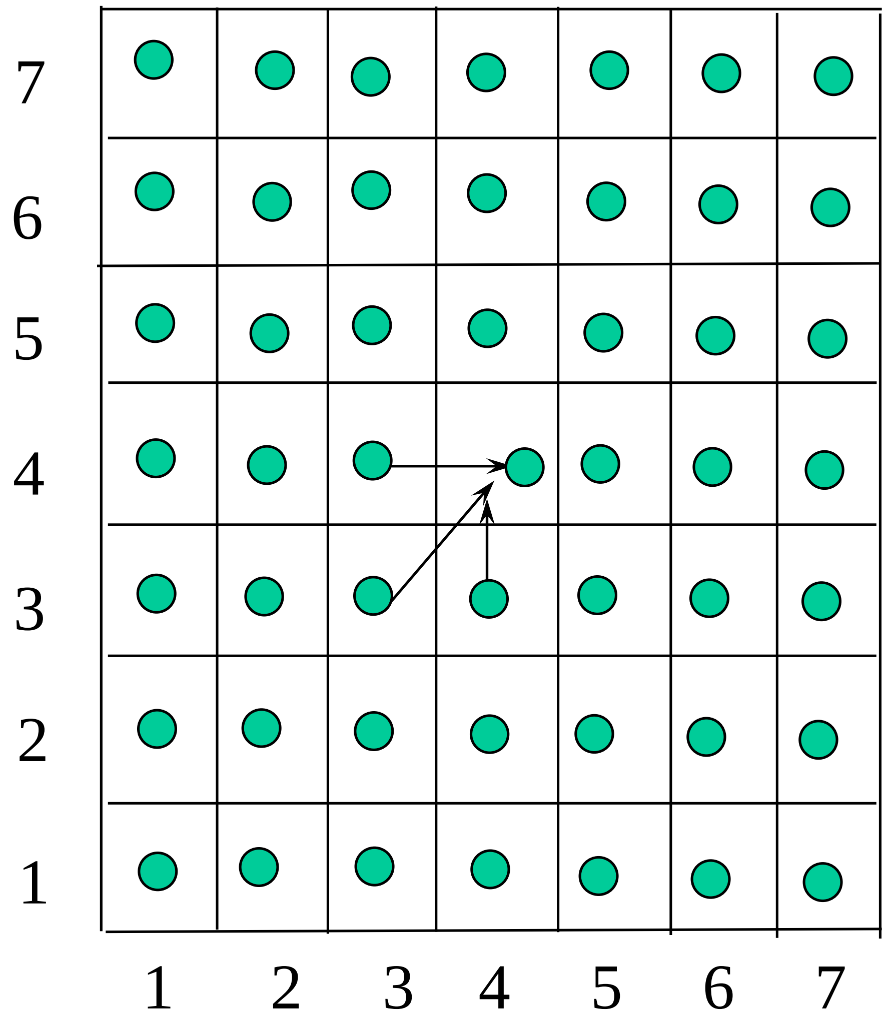


$$
\mathrm{cos}te(i,j)=\begin{cases}
0 & (i=1)\wedge(j=1)\\
\mathrm{cos}te(i,j-1)+d[(i,j),(i,j-1)] & (i=1)\wedge(j>1)\\
\mathrm{cos}te(i-1,j)+d[(i,j),(i-1,j)] & (i>1)\wedge(j=1)\\
\min\left\{\begin{array}{l}\mathrm{cos}te(i,j-1)+d[(i,j),(i,j-1)],\\ \mathrm{cos}te(i-1,j)+d[(i,j),(i-1,j)],\\ \mathrm{cos}te(i-1,j-1)+d[(i,j),(i-1,j-1)\end{array}\right\} & (i>1)\wedge(j>1)
\end{cases}
$$

Solución: coste(I,J)


(I,J)
(i-1,j)
(i,j)
(i-1,j-1)
(i,j-1)
(0,0)

T: matriz[1..I,1..J] de **R**;



Función coste (I,J:**N**):**R**;
var T: matriz[1..I,1..J] de **R**;

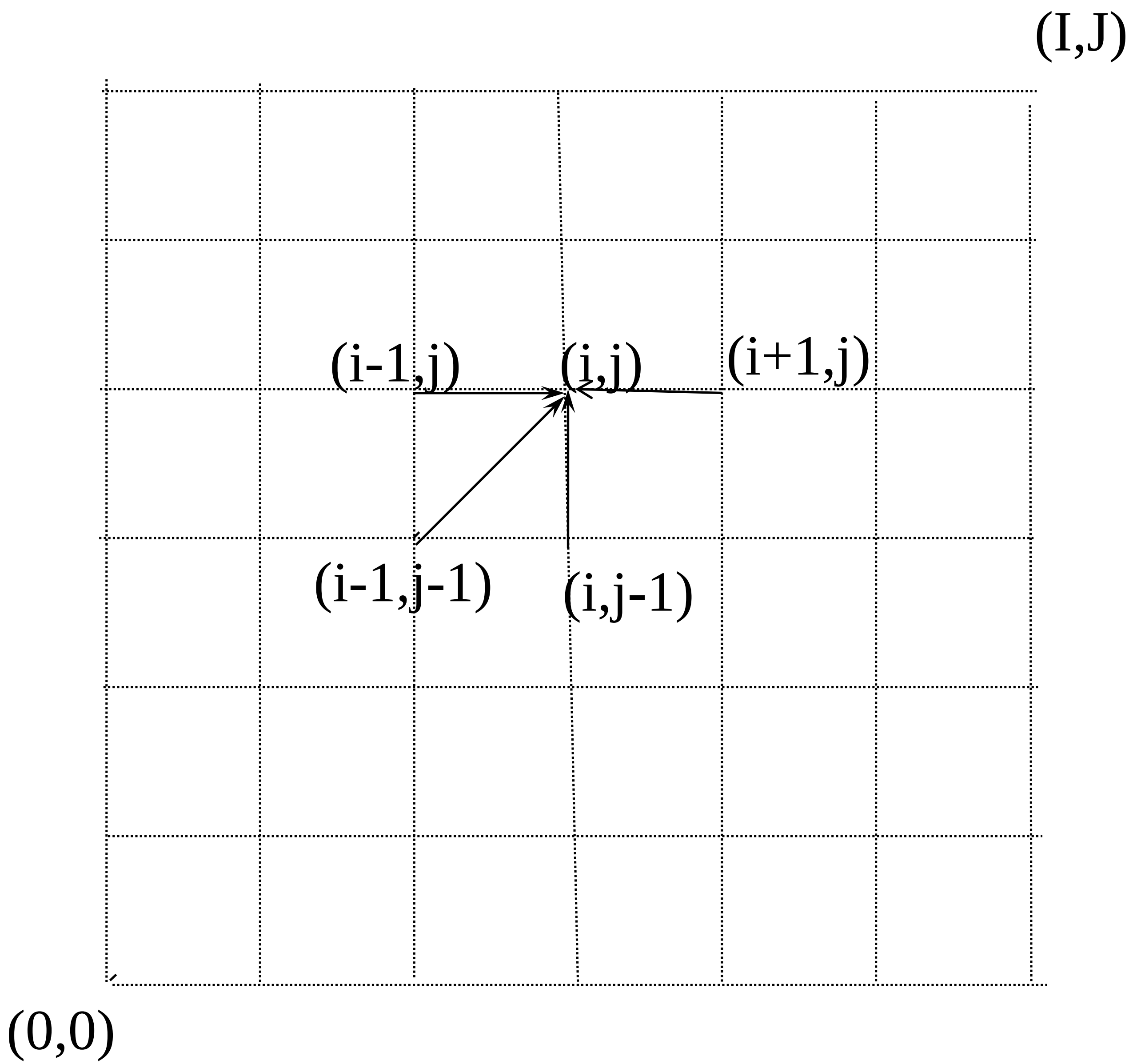
T[1,1]=0
para j=2 **hasta** J **hacer** T[1,j]=T[1,j-1]+d[(1,j),(1,j-1)] **fpara**
para i=2 **hasta** I **hacer**
    T[i,1]=T[i-1,1]+d[(i,1),(i-1,1)]
    para j=2 **hasta** J **hacer**

$$T[i,j]=\min\left\{\begin{matrix} T[i-1,j]+d[(i,j),(i-1,j)], \\ T[i,j-1]+d[(i,j),(i,j-1)], \\ T[i-1,j-1]+d[(i,j),(i-1,j-1)] \end{matrix}\right\}$$

    fpara
fpara
coste=T[I,J];
fin

Coste O(IJ)

¿Qué ocurriría si añadimos un nuevo movimiento?



El problema del camino más corto en grafos sin ciclos negativos: el algoritmo de Bellman-Ford

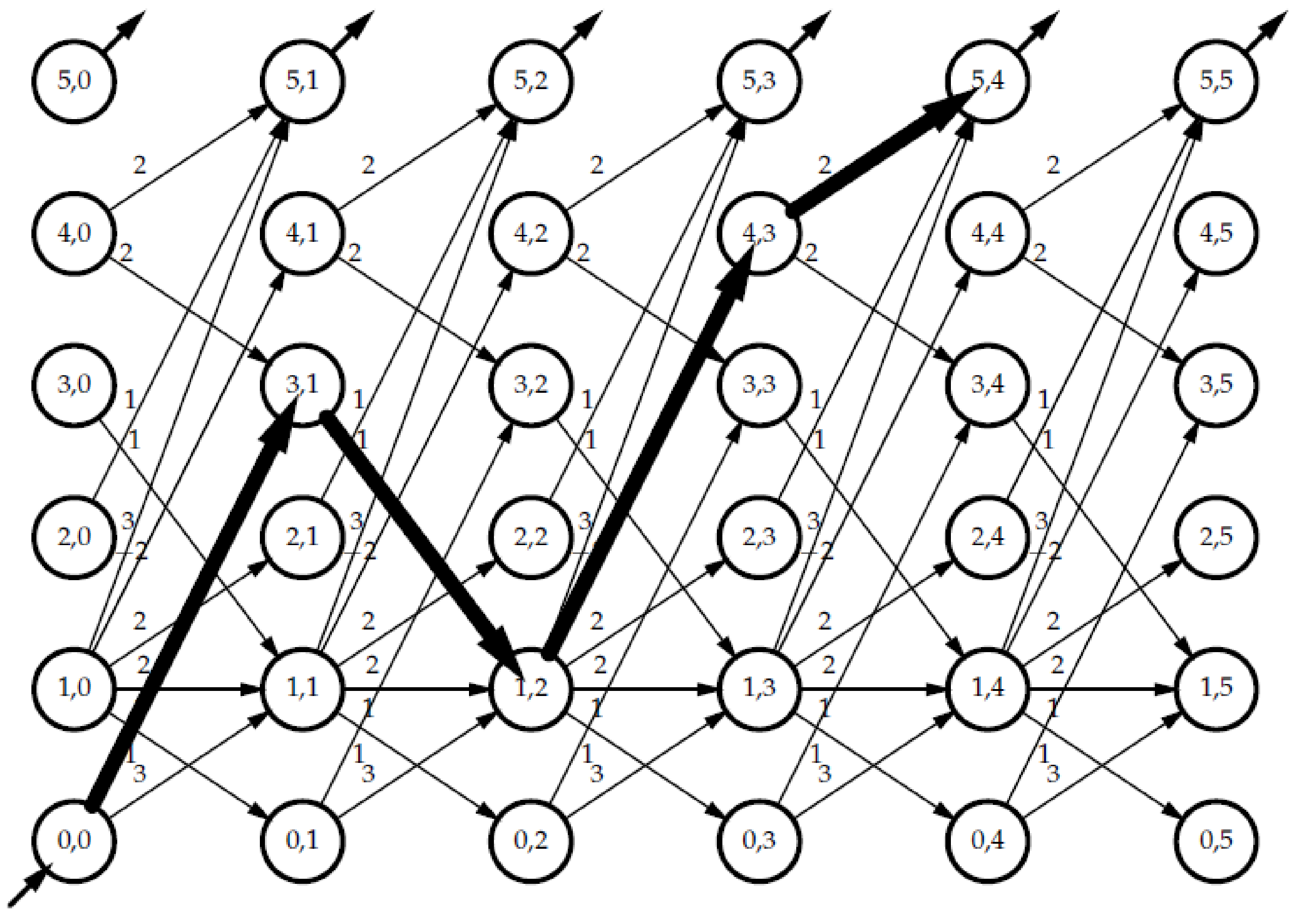
Sólo impondremos una condición: que el peso de un ciclo no sea negativo. Si lo fuera, el problema estaría mal definido, pues el camino óptimo tendría un número infinito de aristas

Solución recursiva:

$$D(v,k) = \begin{cases} 0, & \text{si } v = s \text{ y } k = 0; \\ +\infty, & \text{si } v = s \text{ y } k > 0; \\ +\infty, & \text{si } v \neq s \text{ y } \nexists (u,v) \in E; \\ \min_{(u,v)\in E} \left(D(u,k-1) + d(u,v) \right), & \text{en otro caso.} \end{cases}$$

Hemos de calcular el valor de la expresión $\min_{0\leq k<|V|} D(t,k)$ para conocer el peso del camino más corto con cualquier número de aristas.

Grafo multietapa: Cada columna representa todos los nodos del grafo (puntos de la retícula), y en cada etapa se almacena el mejor coste de alcanzar los nodas.

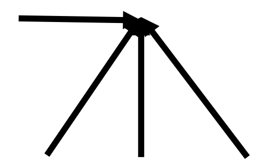


bellman_ford.py

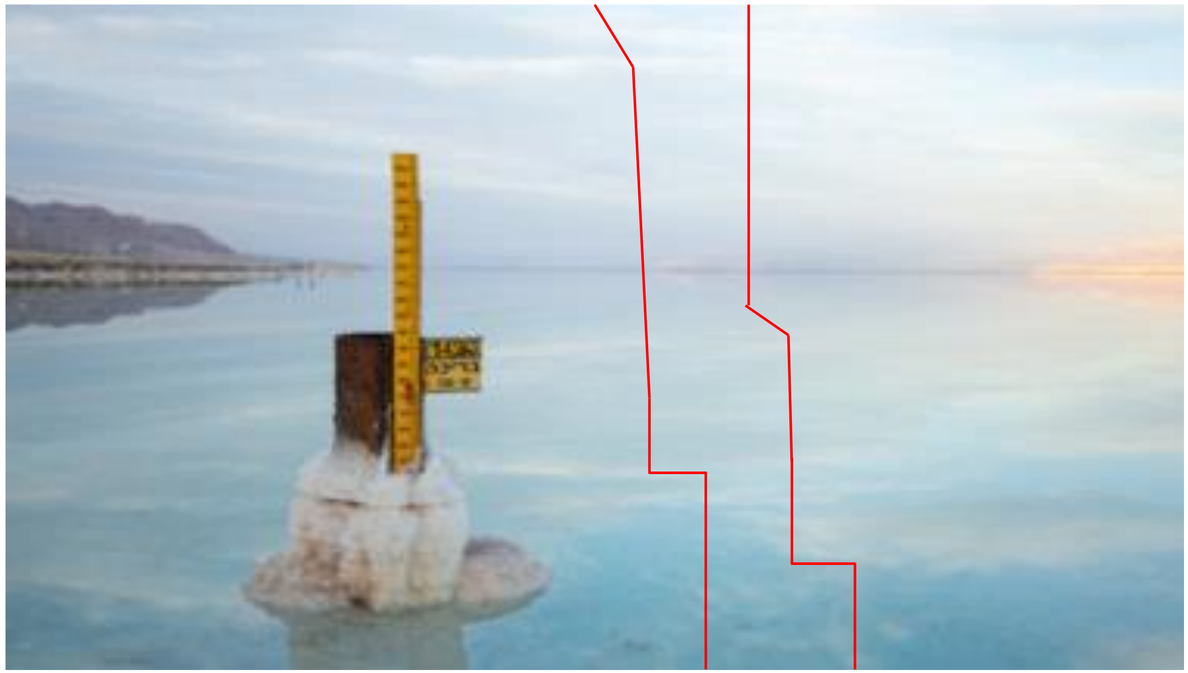
```
def shortest_distance(G, d, s, t, infinity=3.4e+38):
    D = dict( ((v,0), infinity) for v in G.V )
    D[s, 0] = 0
    for i in xrange(1, len(G.V)):
        for v in G.V:
            if G.in_degree(v) == 0:
                D[v, i] = infinity
            else:
                D[v, i] = min( [ D[u, i-1] + d(u,v) for u in G.preds(v) ] )
    return min(D[t, i] for i in xrange(len(G.V)))
```

Problema: Encontrar el camino más corto entre cualquier punto (i,0) y cualquier punto (i,J)

(I,J)



(0,0)



$$
\text{coste}(i, j) = \begin{cases} 0 & (j = 0) \\ \infty & (i < 0) \vee (i > I) \\ \min \left\{ \begin{array}{l} \text{coste}(i, j-1) + d[(i, j), (i, j-1)], \\ \text{coste}(i-1, j) + d[(i, j), (i-1, j)], \\ \text{coste}(i-1, j-1) + d[(i, j), (i-1, j-1)) \\ \text{coste}(i+1, j-1) + d[(i, j), (i+1, j-1)] \end{array} \right\} & (j > 0) \end{cases}
$$

Solución: $\max\limits_{\forall i} \{\text{coste(i,J)}\}$

Función coste (I,J:**N**):**R**;
var T: matriz[-1..I+1,0..J] de **R**;

para i=0 **hasta** I **hacer** T[i,0]=0 **fpara**

para j=0 **hasta** J **hacer** T[-1,j]=∞; T[I+1,j]=∞ **fpara**

para j=1 **hasta** J **hacer**

    para i=0 **hasta** I **hacer**

$$T[i,j] = \min\begin{Bmatrix} T[i-1,j]+d[(i,j),(i-1,j)], \\ T[i,j-1]+d[(i,j),(i,j-1)], \\ T[i+1,j-1]+d[(i,j),(i+1,j-1)] \\ T[i-1,j-1]+d[(i,j),(i-1,j-1)] \end{Bmatrix}$$

    fpara

fpara

coste= $\max\limits_{\forall i}$ T[i,J];

fin

Ejemplos de aplicación

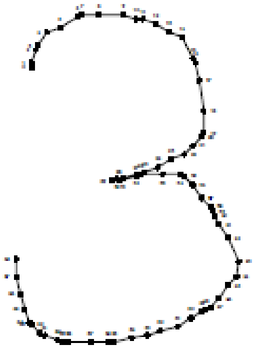


Figura 8.51: Cada trazo es una secuencia de puntos en el plano.

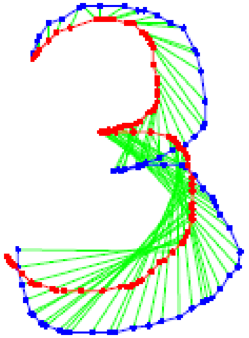


Figura 8.52: Alineamiento punto a punto entre dos trazos del dígito 3. Quedan puntos sin alinear y el alineamiento empareja puntos de zonas «no equivalentes».

Problema: Alineamiento temporal