1. Podemos representar una ecuación de recurrencia *lineal, de coeficientes constantes, y homogénea* de orden k de la siguiente forma:

$$\forall n \geq k \; f_n + a_k \cdot f_{n-1} + \cdots + a_1 \cdot f_{n-k} = 0$$
$$f_0 = c_1, \ldots, f_{k-1} = c_k$$

donde $a_1, \ldots, a_k$ y $c_1, \ldots, c_k$ son elementos conocidos de un anillo $\mathscr{A} = \langle A, +, \cdot, -, 0, 1 \rangle$. Una ecuación de este tipo queda pues determinada por dos vectores:

- El vector de coeficientes, $\vec{a} = [a_1, \ldots, a_k]$.
- El vector de condiciones iniciales, $\vec{c} = [c_1, \ldots, c_k]$.

Emplee programación dinámica para diseñar un algoritmo que calcule los términos de su solución. El algoritmo recibirá $\vec{a}$, $\vec{c}$, k y n, y devolverá f_n. Analice el algoritmo calculando el número exacto de operaciones de cada tipo sobre $\mathscr{A}$. No emplee más de $\Theta(k)$ espacio.

algoritmo(a, c, k, n) → f_n c[k]

2. Diseñe un algoritmo que reciba la *tabla de subproblemas resueltos de valores* correspondiente a un ejemplar del problema de la mochila discreta y devuelva, en un conjunto, una de las posibles composiciones óptimas de la carga.

algoritmo(p, v,) → S

S ← ∅

mientras

CAPACIDADES

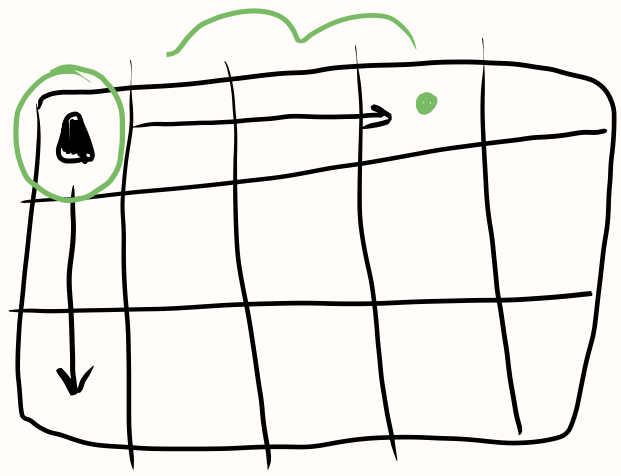
VALORES MÁX.	0	1	2	3	4	5	6	7	8	9	10	P	V
	0	1	1	1	1	1	1	1	1	1	1	1	1
	0	1	6	7	7	7	7	7	7	7	7	2	6
	0	1	6	7	7	18	19	24	25	25	25	5	18
	0	1	6	7	7	18	22	24	28	29	29	6	22
	0	1	6	7	7	18	22	28	29	34	35	7	28

desde i ← n-1 hasta 1
  desde j ← n-1 hasta 1
    si m[i, j] ≠ m[i-1, j]
      S ← S ∪ {i}

3. Diseñe un algoritmo que reciba la *tabla de subproblemas resueltos de valores* correspondiente a un ejemplar del problema del cambio de moneda con suministro ilimitado y devuelva, en un conjunto, una de las posibles composiciones óptimas del cambio.

```
Monedas(TSR, n, C, v) → S
        n: nº distinto monedas
        C: cambio
        v: cuanto vale cada moneda

S ← ∅
(i, j) ← (n, C)
mientras i ≠ 1 ^ j ≠ 0
    si TSR[i,j] ≠ TSR[i-1, j]
        S ← S ∪ {i}
        j ← j - v[i]
    i ← i-1

si v[1] ≤ j
    S ← S ∪ {1}
```

4. Se dispone de un tablero, T, de dimensión $m \times n$ que tiene asociado un valor numérico t_{ij} con cada casilla (i, j). Deseamos mover un robot situado en la casilla $(1, 1)$ al extremo opuesto (m, n) de manera que la suma de los valores de las casillas que atraviesa sea mínima. En cada paso, el robot únicamente puede moverse a la casilla de la derecha o a la de abajo.



a) Resuelva el problema restringido consistente en calcular los valores de los caminos óptimos empleando programación dinámica. Detalle el proceso de obtención del algoritmo y no emplee más espacio que el del propio tablero.

b) Diseñe un algoritmo que a partir del resultado del anterior construya una matriz binaria con 1 en las posiciones que forman parte de *algún* camino óptimo y 0 en las demás.

c) Aplique ambos algoritmos para resolver los siguientes ejemplares:

1	2	4
3	1	1
5	2	4

1	3	5
2	1	2
3	2	4

1	1	1
1	1	1
1	1	1

a) $t(t, n, i, j) \rightarrow s$

```
desde i ← 1 hasta n
    desde j ← 1 hasta n
        si i = 1 ^ j = 1
            S[i][j] ← t[i][j]
        sino
            si i = 1 ^ j > 1
                s[i][j] ← t[i][j] + S[i][j-1]
            sino
                si j = 1 ^ i > 1
                    s[i][j] ← t[i][j] + S[i-1][j]
                sino
                    S[i][j] ← min(t[i][j] + s[i-1][j], t[i][j] + S[i][j-1])
```

$$d(i,j) = \begin{cases} t(i,j) & i=1 \wedge j>1 \\ t(i,j)+d(i-1,1) & i>1 \wedge j=1 \\ t(i,j)+d(i,j-1) & i>1 \wedge j>1 \\ - & i=1 \wedge j=1 \end{cases}$$

b) marcar $(S, n) \rightarrow S2$

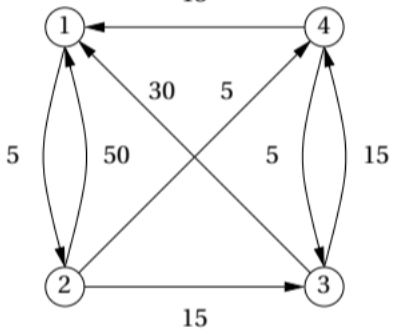
```
S2 ← ∅
i ← n
j ← n

desde z ← 1 hasta n        // rellenar solución a 0
    desde m ← 1 hasta n
        S2[z,m] ← 0

mientras i ≠ 1 ^ j ≠ 1
si i = 1
    S2[i][j] ← S[i, j-1]
    j ← j-1
sino
    si j = 1
        S2[i][j] ← S[i-1][j]
        i ← i-1
    sino
        si S[i][j-1] < D[i-1][j]
            S2[i][j-1] ← 1
            j ← j-1
        sino
            S2[i-1][j] ← 1
            i ← i-1
```

5. Empleando el algoritmo de *Floyd*, diseñe un algoritmo que reciba la *matriz de adyacencia* de un grafo y devuelva su *matriz de accesibilidad*. Utilícelo para calcular la matriz de accesibilidad de:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

6. Demuestre que si el algoritmo de *Floyd* se aplica a un grafo con pesos no negativos en sus aristas, en el paso k del bucle más externo no se modifican ni la fila ni la columna k-ésimas de la matriz de pesos del grafo.

Aplique el algoritmo de *Floyd* al grafo de la figura para calcular los valores de los caminos mínimos entre cada pareja de vértices, mostrando las matrices intermedias que se obtienen en cada paso del bucle más externo.



Tenga en cuenta la propiedad enunciada, y también que la diagonal principal de la matriz permanecerá invariante. Esto le permitirá ahorrar cálculos.

7. Dada una matriz binaria A de dimensión $n \times n$, podemos definir su *clausura reflexiva y transitiva*, A^*, de la siguiente forma:

$$A^* = I + A + A \cdot A + \cdots + A \overset{(i)}{\cdots} A + \cdots$$

donde + y $\cdot$ representan la *suma lógica matricial* y el *producto lógico matricial* (es decir, las operaciones matriciales de suma y producto teniendo en cuenta que con los elementos de las matrices se emplean operaciones $\vee$ y $\wedge$). Relajando la notación, simplemente diremos que $A^* = \sum_{i=0}^{\infty} A^i$.

a) Demuestre por inducción sobre m que:

$$(I + A)^m = \sum_{i=0}^{m} A^i$$

b) Se demuestra que bastan los n primeros términos de la serie para calcular A^*, es decir, que $A^* = \sum_{i=0}^{n-1} A^i$. Empleando este hecho y la propiedad anterior, diseñe un algoritmo que permita calcular A^*. Analice la eficiencia temporal de dicho algoritmo y compárela con la del algoritmo de *Warshall*.

algoritmo (A, m) → S

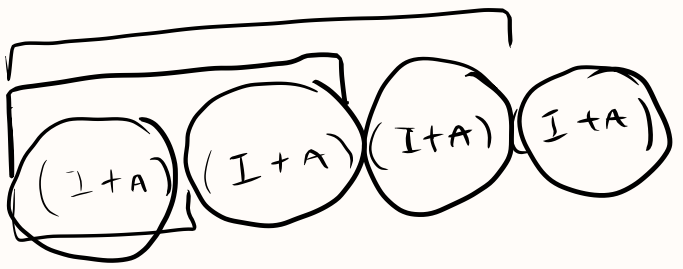
S ← 1

i ← 1

mientras i ≠ m

S ← (I ∧ A) ∨ S

i ← i + 1



8. Sea $m \in \mathbb{N}$, $m \geq 2$. Un *grafo multietapa* $G = (V, A)$ de m etapas $V_1, \ldots, V_m \subset V$ es un grafo orientado y ponderado, donde $V = \{v_1, \ldots, v_n\}$ es el conjunto de los vértices, $A \subseteq V \times V$ es el conjunto de las aristas, $p : V \times V \to \mathbb{R}^+ \cup \{0, \infty\}$ es la función de ponderación, y se verifica la siguiente propiedad:

$$\bigcap_{i=1}^{m} V_i = \varnothing \wedge \bigcup_{i=1}^{m} V_i = V \wedge |V_1| = |V_m| = 1 \wedge \forall (u, v) \in A \; \exists i \in [1, m) \; (u \in V_i \wedge v \in V_{i+1})$$

Sean v_1 y v_n los únicos vértices de V_1 y V_m, respectivamente, a los que denominaremos *vértice inicial* y *vértice final*. Diseñe un algoritmo para hallar el *camino mínimo* y su valor desde el vértice inicial al final. Dicho algoritmo realizará, en el peor de los casos, un número de operaciones elementales perteneciente a $O(n^2)$.

Algoritmo Dijkstra

9. Sea $G = (V, A)$ un grafo sin orientación, donde V es el conjunto de los vértices y $A \subseteq V \times V$ el de las aristas. Se define su *clausura reflexiva y transitiva* como el grafo $G^* = (V^*, A^*)$ dado por:

$$V^* = V \qquad A^* = \{(u, v) \in V \times V \mid u = v \vee \exists w \in V \; u \; A \; w \; A^* \; v\}$$

a) Demuestre que si A es la relación de adyacencia del grafo G y A^* la del grafo G^* se verifica que A^* es reflexiva y transitiva. ¿Cuál es el significado de la relación A^*?

b) Diseñe un algoritmo que calcule las *componentes conexas* de G a partir de A^*. Analice su eficiencia temporal.

10. Para modelar una *red de comunicaciones* utilizamos un grafo $G = (V, A)$ orientado y ponderado, donde V es el conjunto de los vértices, $A \subseteq V \times V$ el conjunto de las aristas y $p : V \times V \to [0, 1]$ la función de ponderación.

Los vértices representan *equipos de comunicaciones* y las aristas las *líneas de comunicaciones* que los unen. La función de ponderación es la *fiabilidad* de la línea que une dos equipos, es decir, la probabilidad de que la transmisión de una unidad de información desde un equipo a otro tenga éxito. Estas probabilidades se suponen independientes.

Diseñe un algoritmo que calcule el *camino más fiable* para transmitir una unidad de información entre dos equipos dados y la *probabilidad de error* de la transmisión. Dicho algoritmo realizará, en el peor de los casos, un número de operaciones elementales perteneciente a $O(n^3)$.

floyd modificado → calculando en vez de camino mínimo, probabilidad mí

si fiabil de 90%

Ponderacion(...)

10% probab deg pete

Floyd (p, n) → p

desde i ← 1 hasta n
  p[i,i] ← 0

desde k ← 1 hasta n
  desde i ← 1 hasta n
    desde j ← 1 hasta n
      p[i,j] ← min(p[i,j], p[i,k] + p[k,j])

Algoritm(matpond, 0, d, n) → Sol

mientras $i \leftarrow 1$ hasta n
  mientras $j \leftarrow 1$ hasta n
    si matpond[i,i] = 0
      matpond[i,i] ← ∞
    sino
      matpond[i,i] ← -1

11. Dado un grafo sin orientación, G, y un número natural, l, diseñe un algoritmo *eficiente* que determine si existe o no algún par de vértices tal que todo camino entre ellos tenga longitud superior a l. Realice el análisis temporal del algoritmo resultante.

12. Dada una *red de ordenadores* y la *probabilidad de fallo* de cada uno de sus enlaces, diseñe un algoritmo que para cada par de ordenadores indique si toda ruta entre ambos posee una probabilidad de fallo superior a f, $0 \leq f \leq 1$. Los fallos en enlaces distintos se suponen independientes.

No olvide definir adecuadamente la estructura de datos a emplear. Al analizar el algoritmo explique qué es lo que va a considerar como operación elemental. El algoritmo deberá realizar $O(n^3)$ operaciones elementales.