1) PROGRAMACIÓN DINÁMICA [5 puntos] Se sabe que el denominado perfil genético de cada persona es prácticamente único e irrepetible, lo cual es de gran utilidad en la identificación de personas a través de la comparación de perfiles. Para tal fin, se calcula el valor **RMP (Probabilidad de coincidencia aleatoria)**, mediante el producto de un conjunto de probabilidades, donde cada una de ellas se corresponde con la probabilidad de que un individuo tenga en su ADN una categoría p en una característica n, siendo N (N >= 0) el número total de características y P (P>=1) el número total de categorías dentro de cada característica. Además, se tendrá en cuenta otro factor denominado CPI que se define como la suma de las probabilidades elegidas para RMP, y que deberá ser inferior a una probabilidad C (C>=0 o C<0).

Ejemplo: Dada la siguiente matriz A, siendo C = 0,7.

	Categorías		
Característica	**Español**	**Italiano**	**Hispano**
c1	**0,1**	**0,2**	**0,6**
c2	**0,5**	**0,4**	**0,8**

Algunos RMP's de los 9 posibles junto con sus CPI's asociados serían los siguientes:

RMP = 0,1 * 0,5 = 0,05 (c1=1, c2=1) = (Español, Español)	CPI = 0,1+0,5 = 0,6 <= C Candidata
RMP = 0,2 * 0,8 = 0,16 (c1=2, c2=3) = (Italiano, Hispano)	CPI = 0,2 + 0,8 = 1 > C Rechazada

Si se realizasen los 9 MPI's se concluiría que lo más probable es que el individuo tuviese una alta probabilidad de tener los valores (c1= 2, c2=1) = (Italiano, Español), con un RMP de 0,1 (0,2*0,5) con CPI = 0,2+0,5 = 0,7.

Se pide diseñar un algoritmo basado en Programación Dinámica que calcule el máximo RMP, el valor para cada una de las características vinculadas a dicho RMP, y con la restricción de que el valor de CPI sea inferior o igual a un valor dado C.

Responder a cada uno de los siguientes apartados con claridad y concisión:

a) Secuencia de decisiones (número de decisiones y significado de las mismas) (10%)
b) Función objetivo y restricciones (20%)
c) Demostración del principio de optimalidad (15%)
d) Definición recursiva y cómo se efectúa la primera llamada a la función recursiva (25%)
e) Árbol de llamadas COMPLETO, correspondiente a la definición recursiva anterior, para el ejemplo anterior (10%).
f) Tipo de estructuras de almacenamiento elegidas y sus dimensiones y en qué orden se rellenan. Cómo se calculan dos posiciones concretas de las estructuras (una correspondiente a un caso trivial y a la otra a un caso no trivial). Cómo se obtiene la solución (valor y secuencia de decisiones). A este apartado f) se puede responder haciendo referencia al ejemplo del apartado e) o de forma genérica (20%)

SOLUCIÓN.-

a) SECUENCIA DE DECISIONES

< x_1, x_2, …, x_N > tupla de longitud fija, donde x_1 indicará el índice de la categoría asociada a la característica 1, x_2 indicará el índice de la categoría asociada a la característica 2, …, x_i indicará el índice de la categoría asociada a la característica i, …

b) FUNCIÓN OBJETIVO Y RESTRICCIONES

$$\text{maximizar} \prod_{i=1}^{N} A[i][xi]$$

sujeto a

$$\sum_{i=1}^{N} A[i][xi] \leq C$$

c) DEMOSTRACIÓN DEL PRINCIPIO DE OPTIMALIDAD

Sea < x_1, x_2, …, x_N > la solución óptima del problema de seleccionar una categoría p para cada una de las N características, con el objetivo de obtener la máxima probabilidad total para RMP. Denominaremos a dicho problema RMP(N, C). El valor (probabilidad total máxima) asociado a dicha secuencia de decisiones es

$$\prod_{i=1}^{N} A[i][xi]$$

cumpliéndose además que

$$\sum_{i=1}^{N} A[i][xi] \leq C$$

Supongamos que prescindimos de la última decisión, esto es, x_N. Esta decisión indica la categoría seleccionada para la característica N. La subsecuencia que queda, esto es < x_1, x_2, …, x_{N-1}>, es la solución óptima para el problema asociado, es decir, para el subproblema RMP(N-1, C-A[N][xN]), que es el subproblema de seleccionar una categoría para cada una de las N-1 características con el objetivo de obtener la máxima probabilidad y sin exceder el valor de (C-A[N][xN]). El valor (impacto total) asociado a dicha secuencia de decisiones es

$$\prod_{i=1}^{N-1} A[i][xi]$$

cumpliéndose además que

$$\sum_{i=1}^{N-1} A[i][xi] \leq \mathrm{C} - \mathrm{A[N][xN]}$$

Supongamos que < x_1, x_2, …, x_{N-1} > **NO** es la solución óptima para el subproblema RMP(N-1, C-A[N][xN]), sino que existe otra solución < y_1, y_2, …, y_{N-1} > que mejora su valor, lo cual significa que

$$\prod_{i=1}^{N-1} A[i][yi] > \prod_{i=1}^{N-1} A[i][xi] \quad [1]$$

cumpliéndose además que

$$\sum_{i=1}^{N-1} A[i][yi] \leq \mathrm{C} - \mathrm{A[N][xN]}$$

Si se multiplica el término A[N][xN] a ambos lados de la desigualdad [1], se cumplirá que:

$$\prod_{i=1}^{N-1} A[i][yi] * \mathrm{A[N][xN]} > \prod_{i=1}^{N-1} A[i][xi] * \mathrm{A[N][xN]}$$

lo que significa que la secuencia < y_1, y_2, …, y_{N-1}, x_N > mejora el resultado de la secuencia <x_1, x_2, …, x_N > para el problema RMP(N, C), lo cual contradice la hipótesis de partida, pues < x_1, x_2, …, x_N > era la solución óptima de dicho problema. En conclusión, se cumple el principio de optimalidad.

e) DEFINICIÓN RECURSIVA

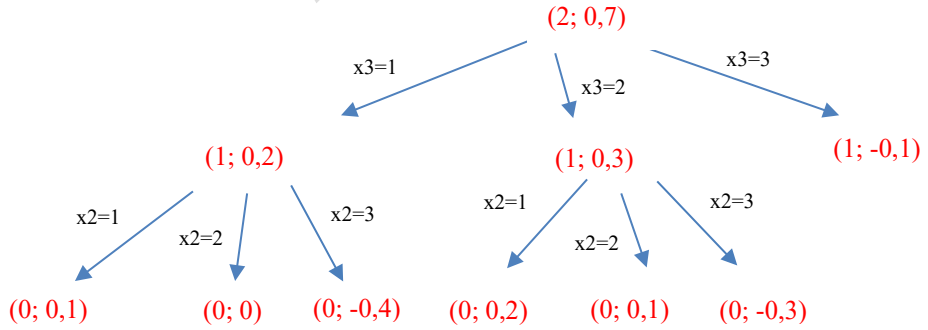
$$RMP\ (n,c) = \begin{cases} 1 & si\ n = 0\ y\ c \geq 0 \\ 0 & si\ n \geq 0\ o\ c < 0 \\ \max_{1 \leq x_i \leq P} \{RMP(n-1, c - A[n][xi]) * A[n][xi]\} & si\ n > 0\ y\ c > 0 \end{cases}$$

donde RMP(n,c), devuelve la máxima probabilidad total para un conjunto de características n y sin sobrepasar una probabilidad c. La llamada inicial a la función se producirá como RMP(N,C), siendo N el número total de características y C la máxima probabilidad que no puede ser excedida.

f) ÁRBOL DE LLAMADAS

N=2, P=3, C=0,7

$$A[1..2][1..3] = \begin{bmatrix} 0{,}1 & 0{,}2 & 0{,}6 \\ 0{,}5 & 0{,}4 & 0{,}8 \end{bmatrix}$$



g) ESTRUCTURAS DE ALMACENAMIENTO

Dado que nuestra función recursiva tiene dos parámetros, se precisan dos estructuras bidimensionales. Estas tendrán (N+1) filas, dado que 0 ≤ n ≤ N, y (C*10+2) columnas, donde la columna -1 representa los valores negativos para C, y dado que C es un número real perteneciente a [0, 1], se considera la columna 0 si C = 0, donde C puede ser un valor perteneciente a [0,1; 0,2; 0,3; 0,4; …; 1], es decir, para simplificar se considera que C solo puede tomar valores dentro de ese rango. Así por ejemplo, si C = 0,4 se necesitarían 6 columnas [-1; 0; 0,1; 0,2; 0,3; 0,4] (C*10+2=0,4*10+2 = 6). En una de las estructuras, RMPmax, se guardará el valor asociado a cada subproblema, esto es, la máxima probabilidad. Concretamente en RPMmax[n][c] estará recogido el valor correspondiente a RMP(n,c). En la otra estructura, Dec, se almacenará la alternativa que proporcione el máximo para cada subproblema.

Orden en el que se rellenan:

a. Primero, los problemas triviales, esto es RMP(0,c) con (0 =< c <= C), que corresponde con rellenar la fila 0 de cada matriz, es decir, RMPmax[0][c] con 1, y la columna -1 que se corresponde con RMPmax[n][c] con (n>= 0) y (c < 0) con 0. En ambos casos, Dec[n][c] será igual a 0.
b. Dado que para solucionar el subproblema RMP(n,c) se precisa conocer la solución de los subproblemas del tipo RMP(n-1,c-A[n][p]) donde 1 ≤ p ≤ P, las matrices RMPmax y Dec se rellenan por filas en sentido creciente, desde la fila 1 hasta la fila N, y dentro de cada fila desde la columna 1 hasta la columna C, aunque también podría ser en sentido inverso de C a 1. (=> Dependencia de los subproblemas).

La probabilidad máxima estará en la posición [N][C] de la matriz RMPmax.

La secuencia óptima de decisiones se obtiene recorriendo determinadas posiciones de la matriz Dec. Comenzaríamos por la posición [N][C], el valor ahí almacenado correspondería a x_N. Seguidamente iríamos a Dec[N-1][C-A[N][x_N]], ahí se encontrará el valor correspondiente a x_{N-1}. Seguidamente iríamos a Dec[N-2][C-A[N][x_N] – A[N-1][x_{N-1}], ahí se encontrará el valor correspondiente a x_{N-2} . Y así sucesivamente.

2) VUELTA ATRÁS [2.5 puntos] Resolver el problema anterior aplicando la metodología de Vuelta atrás (Backtracking). Se pide responder con claridad y concisión a las siguientes cuestiones:

a) Secuencia de decisiones (número de decisiones y significado de las mismas) (5%)
b) Función objetivo (5%)
c) Restricciones explícitas e implícitas (20%)
d) Preparar_recorrido_nivel_k, Existe_hermano_nivel_k y Siguiente_hermano_nivel_k (15%)
e) Función Solución (5%)
f) Indicar qué hacen las funciones Correcto y/o Valor si es que fueran necesarias en la solución. Escribir el pseudocódigo de dichas funciones (≡ cómo lo hacen) (25%)
g) Para el mismo ejemplo del problema de dinámica, dibujar el árbol de búsqueda que se explora con todas las soluciones, e indicar cuáles no son factibles. Numerar los nodos reflejando el orden en el que se visitan e indicar cuándo se realiza una poda y por qué (25%).

SOLUCIÓN.-

a) SECUENCIA DE DECISIONES

$< x_1, x_2, \ldots, x_N >$, por consiguiente, tupla de longitud fija, y donde x_1 indicará la categoría para la característica 1, x_2 indicará la categoría para la característica 2, …, x_i indicará la categoría para la característica i.

b) FUNCIÓN OBJETIVO

$$maximizar \prod_{i=1}^{N} A[i][xi]$$

c) RESTRICCIONES EXPLÍCITAS E IMPLÍCITAS

Restricciones explícitas $(\forall i)(x_i \epsilon \{1, \ldots, P\}: 1 \leq i \leq N)$

Restricciones implícitas

$$\sum_{i=1}^{N} A[i][xi] \leq C$$

$$si\ k < N\ entonces \sum_{i=1}^{k} A[i][xi] \leq C$$

d) PREPARAR_RECORRIDO_NIVEL_K, EXISTE_HERMANO_NIVEL_K Y SIGUIENTE_HERMANO_NIVEL_K

preparar_recorrido_nivel_k x[k] = 0

existe_hermano_nivel_k x[k] < P

siguiente_hermano_nivel_k x[k] = x[k] + 1

e) FUNCIÓN SOLUCIÓN

k = N

f) FUNCIÓN CORRECTO Y FUNCIÓN VALOR

QUÉ HACE.- la función correcto, tras recibir la secuencia de decisiones x y el valor de k correspondiente, devuelve

- en el caso de una tupla incompleta: falso, si la suma de las probabilidades que suponen las categorías asignadas a las características analizadas supera el valor C; verdadero, en caso contrario.
- en el caso de una tupla completa: verdadero, si la suma de las probabilidades que suponen las categorías asignadas a las características analizadas es menor o igual al valor C; falso, en caso contrario.

PSEUDOCÓDIGO (CÓMO).-

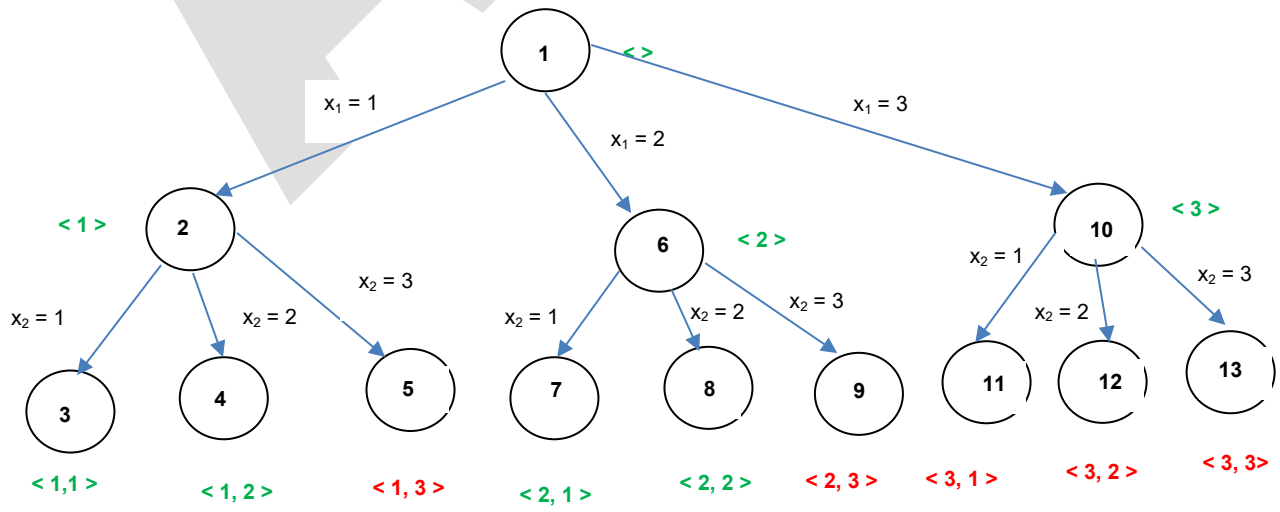
```
Funcion Correcto ( A[1..C, 1..P]: matriz de reales, C: doble, x:tupla, k:entero ) retorna ( b:booleano )
        var i: entero, total: real fvar
        total=0.0;
        para i=1 hasta k hacer
                total = total + A[ i ][x[ i ]]
        fpara
        si total ≤ C entonces retorna verdadero
                        sino retorna falso
        fsi
ffunción
```

QUÉ HACE.- la función Valor, tras recibir la secuencia de decisiones x y el valor de k, devuelve el valor de la función objetivo correspondiente a la secuencia de decisiones x, es decir, la máxima probabilidad total.

PSEUDOCÓDIGO (CÓMO).-

```
Funcion Valor ( A[1..N, 1..P]: matriz de reales, x:tupla, k:entero ) retorna ( b: real )
        var i: entero, total: real fvar
        total = 1.0;
        para i=1 hasta k hacer
                total = total * A[ i ][x[ i ]]
        fpara
        retorna total
función
```

g) ÁRBOL DE BÚSQUEDA



En los nodos 5, 9, 11, 12 y 13 se alcanza una secuencia de decisiones completa pero incorrecta ya que la suma de las probabilidades supera a C.

3) ESQUEMA VORAZ [2.5 puntos] El teatro Jovellanos está preparando la temporada primavera-verano 2022, y le han ofertado un conjunto de n actividades culturales diversas (obras de teatro, conciertos, …) que podría contratar para su programación de primavera-verano. Dado que dichas actividades estarán de gira por el país, para cada actividad se conoce en qué período puede estar en la ciudad. Por tanto, para cada actividad i, con $1 \leq i \leq n$, se conoce el día de comienzo (Ci) y el día de finalización (Fi), tales que Ci < Fi. De forma que la actividad i, si se contrata, debe hacerse durante el período [Ci, Fi). Dos actividades i y j se dicen compatibles si los intervalos [Ci, Fi) y [Cj, Fj) no se superponen, es decir, si Ci ≥ Fj ó si Cj ≥ Fi. Por simplicidad vamos a considerar que la temporada primavera-verano abarca un número de días, por ejemplo, 200 días (día 1, día 2, …, día 200) olvidándonos del formato fecha (dd/mm).

Diseñar un algoritmo voraz que seleccione el mayor conjunto de actividades mutuamente compatibles.

Se pide responder con claridad y concisión a las siguientes cuestiones:

a) Secuencia de decisiones (número de decisiones y significado de las mismas) (10%)
b) Función objetivo y restricciones (10%)
c) Quiénes son los candidatos (5%)
d) Dar un criterio de selección, razonado y razonable, para determinar el candidato más prometedor (10%)
e) Dar el criterio para determinar cuándo un candidato es factible (10%)
f) ¿En qué consiste la función es_solución? (10%)
g) Mostrar, etapa a etapa, cómo dicha estrategia voraz construye la solución para el siguiente ejemplo (20%)

	1	2	3	4	5	6	7	8	9	10	11
C	1	5	12	5	8	3	6	8	3	2	1
F	4	9	16	7	12	5	10	11	9	14	6

h) Escribir el algoritmo voraz resultante (25%)

SOLUCIÓN.-

a) $<x_1, x_2, \ldots, x_k>$ donde x_1 es la primera actividad a realizar, x_2 es la segunda actividad a realizar, …, x_i es la i-ésima actividad a realizar. Esto es, la solución se puede expresar como una secuencia de decisiones, concretamente se devolverá como solución un subconjunto del total de las actividades de entrada.
b) Se trata de maximizar k y las restricciones son

$$(\forall i)(\forall j)(i \neq j \rightarrow C[xi] \geq F[xj] \vee C[xj] \geq F[xi]: 1 \leq j \leq k): 1 \leq i \leq k)$$

Esto es, se trata de elegir el mayor número de actividades mutuamente compatibles.

c) Los candidatos son las n actividades culturales

d) Criterio de selección: la actividad cultural más prometedora en cada etapa es aquella con el día de finalización más temprano. Esta elección maximiza la cantidad de tiempo no organizado restante. Para facilitar la aplicación del criterio, ordenaremos las actividades por Fi en sentido creciente.
e) Esa actividad cultural elegida (actividad i), se incorporará finalmente a la solución siempre y cuando su período de actividad, esto es, [Ci,Fi) no colisione con ninguno de los intervalos de las actividades que ya forman parte de la solución. Para cumplir con la restricción del problema basta comprobar que el día de comienzo de la actividad elegida (Ci) sea mayor o igual que el día de finalización de la última actividad incorporada a la solución.
f) La solución se alcanza cuando hayamos tratado todas las actividades.
g) Mostrar, etapa a etapa, cómo dicha estrategia voraz construye la solución:

Las actividades se ordenarán en sentido creciente por Fi, de modo que antes de comenzar el bucle voraz tendremos:

	1	2	3	4	5	6	7	8	9	10	11
C	1	3	1	5	3	5	6	8	8	2	12
F	4	5	6	7	9	9	10	11	12	14	16

Antes de comenzar el bucle voraz inicializaremos el conjunto solución con la primera actividad, la que abarca el período [1,4).

etapa	actividad seleccionada	Aceptada(A)/Rechazada(R)	Seleccion[1..k]	k
inicialización	1	A	[1]	1
1	2	R porque C2< F1	[1]	1
2	3	R porque C3 < F1	[1]	1
3	4	A porque C4 ≥ F1	[1,4]	2
4	5	R porque C5 < F4	[1,4]	2
5	6	R porque C6 < F4	[1,4]	2
6	7	R porque C7 < F4	[1,4]	2
7	8	A porque C8 ≥ F4	[1,4,8]	3
8	9	R porque C9 < F8	[1,4,8]	3
9	10	R porque C10 < F8	[1,4,8]	3
10	11	A porque C11 ≥ F8	[1,4,8,11]	4

En conclusión, el número de actividades seleccionadas es 4 y son las siguientes: 1, 4, 8 y 11.

h) Algoritmo

```
Función Jovellanos_Primavera_Verano_2022(C[1..n], F[1..n]:vector de enteros)
                                        retorna ( k:entero y Seleccion[1..n] )
    var i, k: entero; Seleccion[1..n]: vector de enteros fvar

    /* y=prepara(x) */
    ordenar los vectores C y F en sentido creciente por Fi

    /* inicializaciones */
    para i=1 hasta n hacer
        Seleccion[ i ] = 0;
    fpara
    Seleccion[1] = 1;
    k = 1
    i = 1
    mientras ( i < n ) hacer  /* bucle voraz */
        i = i + 1
        si C[ i ] ≥ F[ k ] entonces   /* si factible entonces */
            /* anadir */
            k = k + 1
            Seleccion[ k ] = i
        fsi
    fmientras
    retorna (k, Seleccion)
ffuncion
```

OTRA POSIBLE SOLUCIÓN.-

a) $<x_1, x_2, \ldots, x_n>$ donde x_i indica si la actividad i se va a contratar o no, valor 1 ó 0 respectivamente.
b)

$$maximizar \sum_{i=1}^{n} x_i$$

sujeto a ($\forall$i)($\forall$j)($i \neq j \wedge x_i = 1 \wedge x_j = 1 \rightarrow Ci \geq Fj \vee Cj \geq Fi: 1 \leq j \leq n): 1 \leq i \leq n$)

esto es, se trata de elegir el mayor número de actividades mutuamente compatibles.

c) Los candidatos son las n actividades culturales
d) Criterio de selección: la actividad cultural más prometedora en cada etapa es aquella con el día de finalización más temprano. Esta elección maximiza la cantidad de tiempo no organizado restante.

e) Esa actividad cultural elegida (actividad i), se incorporará finalmente a la solución siempre y cuando su período de actividad, esto es, [Ci,Fi) no colisione con ninguno de los intervalos de las actividades que ya forman parte de la solución. Para cumplir con la restricción del problema basta comprobar que el día de comienzo de la actividad elegida (Ci) sea mayor o igual que el día de finalización de la última actividad incorporada a la solución.
f) La solución se alcanza cuando hayamos tratado todas las actividades.
g) Mostrar, etapa a etapa, cómo dicha estrategia voraz construye la solución:

	1	2	3	4	5	6	7	8	9	10	11
C	1	5	12	5	8	3	6	8	3	2	1
F	4	9	16	7	12	5	10	11	9	14	6

etapa	actividad seleccionada	Aceptada(A)/Rechazada(R)	Seleccion[1..n]	k
inicialización	---	---	[0,0,0,0,0,0,0,0,0,0,0]	--
1	1	A	[1,0,0,0,0,0,0,0,0,0,0]	1
2	6	R porque C6 < F1	[1,0,0,0,0,0,0,0,0,0,0]	1
3	11	R porque C11 < F4	[1,0,0,0,0,0,0,0,0,0,0]	1
4	4	A porque C4 ≥ F1	[1,0,0,1,0,0,0,0,0,0,0]	4
5	9	R porque C9 < F4	[1,0,0,1,0,0,0,0,0,0,0]	4
6	2	R porque C2 < F4	[1,0,0,1,0,0,0,0,0,0,0]	4
7	7	R porque C7 < F4	[1,0,0,1,0,0,0,0,0,0,0]	4
8	8	A porque C8 ≥ F4	[1,0,0,1,0,0,0,1,0,0,0]	8
9	5	R porque C5 < F8	[1,0,0,1,0,0,0,1,0,0,0]	8
10	10	R porque C10 < F8	[1,0,0,1,0,0,0,1,0,0,0]	8
11	3	A porque C3 ≥ F8	[1,0,1,1,0,0,0,1,0,0,0]	3

h) Algoritmo

```
Función Jovellanos_Primavera_Verano_2022(C[1..n], F[1..n]:vector de enteros)
                                                        retorna ( Seleccion[1..n] )
    var i, k: entero; Seleccion[1..n]: vector de enteros fvar

    /* inicializaciones */
    para i=1 hasta n hacer
        Seleccion[ i ] = 0;
    fpara
    k = 0
    i = 0
    mientras ( i < n ) hacer  /* bucle voraz */
        i = i + 1
        actividad_elegida = selecciona_actividad(C, F, Seleccion)
        si ( i = 1 || C[actividad_elegida] ≥ F[ k ] ) entonces   /* si factible entonces */
            /* anadir */
            k = actividad_elegida
            Seleccion[k] = 1
        fsi
    fmientras
    retorna Seleccion
ffuncion
```

Otro posible criterio de selección sería que la actividad más prometedora fuese aquella que tuviera una menor duración (Fi-Ci), esta elección busca ocupar el menor número de días con la idea de dejar hueco para más actividades. En este caso, la condición de factibilidad pasaría por confirmar que la actividad elegida en una etapa no colisione con ninguna de las actividades que ya forman parte de la solución.