[Backtracking 8 puntos] Una escuela universitaria debe realizar la planificación de un conjunto de E exámenes para un día concreto en la próxima convocatoria de julio. Para ello dispone de A aulas cuyas capacidades están recogidas en el vector Capacidad[1..A], siendo Capacidad[i] la capacidad del aula i-ésima. También se conoce el número de alumnos que se van a presentar a cada examen, dicha información está recogida en el vector NumAlumnos[1..E], donde NumAlumnos [i] es el número de alumnos que asistirán al examen i-ésimo.

A partir de estos datos se trata de diseñar un algoritmo utilizando la metodología *Backtracking* que indique a la escuela la forma de organizar los exámenes de ese día de tal modo que se minimice el número de aulas a utilizar. Habrá de tenerse en cuenta que no es posible dividir un examen en varias aulas y que en un aula se podrán meter varios exámenes si es que caben.

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

a) Secuencia de decisiones (nº de decisiones y significado de las mismas) (10%)
b) Expresar la función objetivo en lenguaje natural (5%)
c) Restricciones explícitas (10%)
d) Expresar las restricciones implícitas en lenguaje natural (10%)
e) Tipo de solución buscada (Todas las factibles/Una factible/Óptima) (5%)
f) Preparar_recorrido_nivel_k (5%)
g) Existe_hermano_nivel_k (5%)
h) Siguiente_hermano_nivel_k (5%)
i) Función Solución (5%)
j) Indicar qué hacen las funciones Correcto y/o Valor si es que fueran necesarias en tu solución. Escribir el pseudocódigo de dichas funciones (≡ cómo lo hacen) (20%)
k) Para el siguiente ejemplo, dibujar el árbol de búsqueda que se explora hasta localizar la primera solución factible. Numerar los nodos reflejando el orden en el que se visitan, indicar cuándo se realiza una poda y por qué: (20%):

A = 4, E = 3, Capacidad[1..4] = { 20, 10, 40, 10 } y NumAlumnos[1..3] = { 20, 20, 20 }

SOLUCIÓN.-

SECUENCIA DE DECISIONES

$<x_1,x_2,\ldots,X_E>$ donde xi indica el aula en la que se realizará el examen i-ésimo.

FUNCIÓN OBJETIVO

Minimizar el número de aulas utilizadas. Esto implica que deberemos contabilizar las diferentes aulas que se van a usar, no importando, claro está, que un aula pueda aparecer varias veces en la secuencia de decisiones. Esto únicamente indicará que en ese aula se realizan varios exámenes.
Ejemplos:

< 2, 4, 2, 1 > En este caso, se utilizan 3 aulas: aula 1, aula 2 y aula 4.
< 5, 5, 2, 5> En este caso, se usan 2 aulas: aula 5 y aula 2.

RESTRICCIONES EXPLÍCITAS

$$(\forall i)(x_i \epsilon \{1,2,\ldots,A\}: 1 \leq i \leq E)$$

RESTRICCIONES IMPLÍCITAS

La suma de los alumnos de los diferentes exámenes destinados a un mismo aula (mismo valor en la secuencia de decisiones) no tiene que exceder la capacidad de dicha aula.

TIPO DE SOLUCIÓN

La solución óptima.

PREPARAR_RECORRIDO_NIVEL_K

x[k] = 0

EXISTE_HERMANO_NIVEL_K

x[k] < A

SIGUIENTE_HERMANO_NIVEL_K

x[k] = x[k] + 1

SOLUCIÓN

k = E

FUNCIÓN CORRECTO

La función Correcto necesita la secuencia de decisiones (x), la posición hasta la que se ha completado la secuencia de decisiones (k) y los vectores Capacidad y NumAlumnos.
La función Correcto produce un valor booleano: Verdadero, si la suma de todos los alumnos que realizan su examen en el aula x[k] no superan la capacidad del aula; Falso, en caso contrario.

```
Funcion Correcto ( x : tupla, k : entero; Capacidad[1..A], NumAlumnos[1..E] : vector de enteros)
                                                                        retorna (b:booleano)
        var i : entero; fvar
        Total_alumnos = NumAlumnos[k];
        i = 0;
        mientras (i < k-1) hacer
                i = i + 1;
                si ( x[ i ] = x[ k ] ) entonces
                        Total_alumnos = Total_alumnos + NumAlumnos[i];
                fsi
        fmientras
        si Total_alumnos ≤ Capacidad [x[k]] entonces retorna Verdadero sino retorna Falso fsi
ffunción
```

FUNCIÓN VALOR

La función Valor necesita la secuencia de decisiones (x) la cual estará completa, lo que quiere decir que k=E.
La función Valor produce el número de aulas diferentes que se han utilizado en la organización de los exámenes. Para ello se utilizará un vector auxiliar (Aula_usada) en el que se almacenará un 0 en la posición i, en el caso que el aula i no se utilice y se almacenará un 1 en la posición i, en el caso que el aula i sí se utilice en la planificación de exámenes.

```
Funcion Valor ( x : tupla, k : entero) retorna (t : entero)
        var
           i, Total_aulas: entero;
           Aula_usada[1..A]: vector de enteros
        fvar
        Total_aulas = 0;
        para i=1 hasta A hacer
                Aula_usada[i]=0;
        fpara
        para i=1 hasta k hacer
                Aula_usada[x[i]]=1;
        fpara
        para i=1 hasta A hacer
                Total_aulas = Total_aulas  +  Aula_usada[i];
        fpara
        retorna Total_aulas
ffunción
```

Complejidad temporal: $T(A,E) \in \theta(max(A,E))$
Complejidad espacial: $T(A) \in \theta(A)$

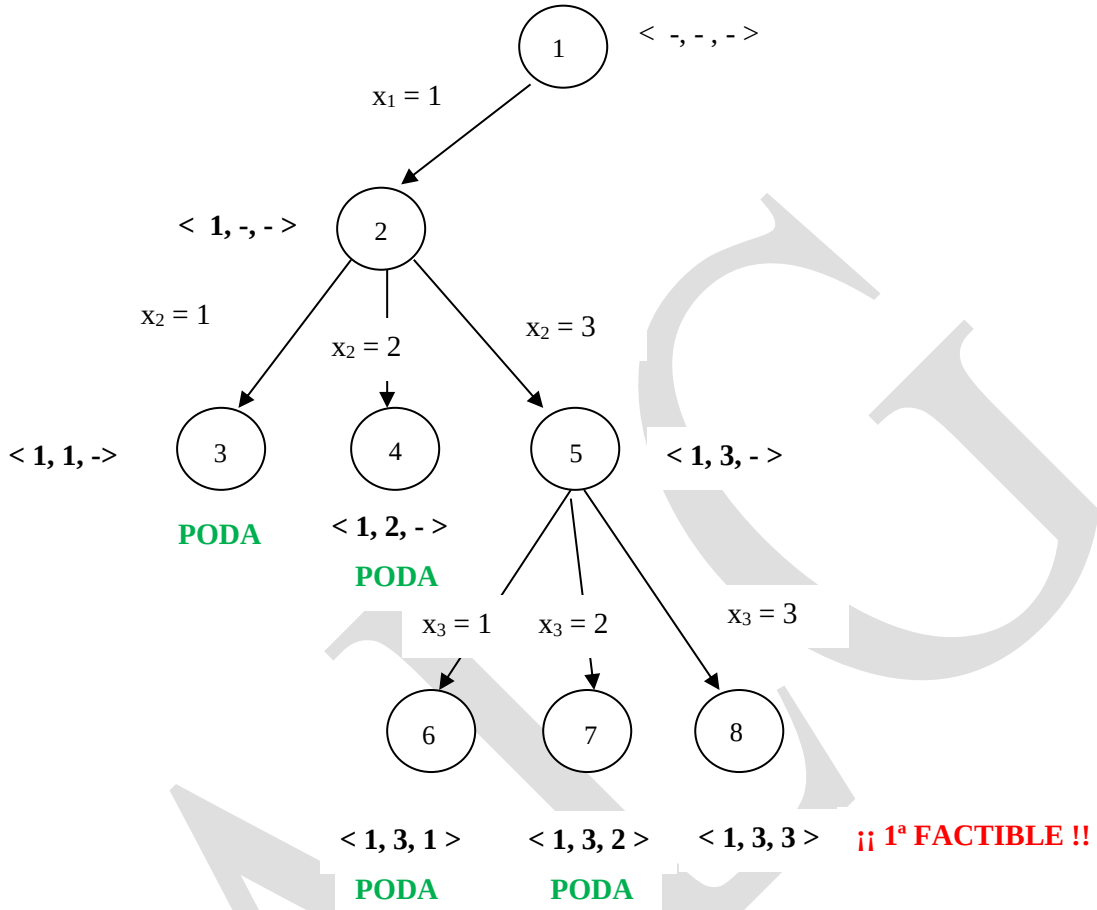
Otra posible solución para la función Valor, sin utilizar un vector de tamaño A, sería contabilizar únicamente la última aparición en el vector x de cada aula utilizada. Esta solución implica el uso de dos bucles anidados.

```
Funcion Valor ( x : tupla, k : entero) retorna (t : entero)
        var
           i, Total_aulas: entero;
        …..ultima_aparicion : booleano
        fvar
        Total_aulas = 0
        para i = 1 hasta k hacer
              unica_aparicion = Verdadero;
              j = i
              mientras j < k y unica_aparicion hacer
                   j = j + 1
                   si ( x[i] = x[j] ) unica_aparicion = Falso fsi
              fmientras
              si ( unica_aparicion ) Total_aulas = Total_aulas  +  1  fsi
        fpara
        retorna Total_aulas
ffunción
```

Esta segunda versión de la función Valor presenta mejor y peor caso. Si contabilizamos solamente las operaciones básicas que suceden en la instancia peor (cada aula aparece una única vez), la complejidad temporal es:

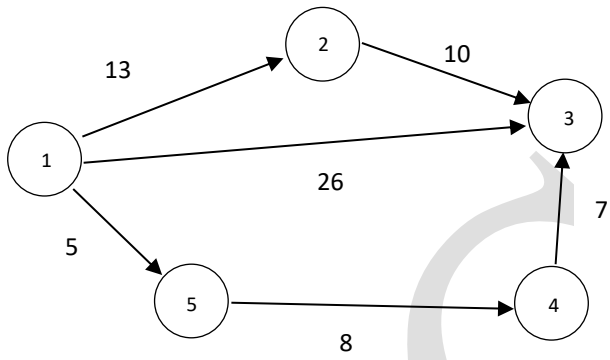
Complejidad temporal: $T_{PC}(E) \in \theta(E^2)$

ÁRBOL DE BÚSQUEDA



Nodo 3: Los alumnos de los exámenes 1 y 2 no caben en el aula 1
Nodo 4: Los alumnos del examen 2 no caben en el aula 2
Nodo 6: Los alumnos de los exámenes 1 y 3 no caben en el aula 1
Nodo 7: Los alumnos del examen 3 no caben en el aula 2

[Esquema voraz 2 puntos] Dado el siguiente grafo, rellenar la tabla adjunta indicando paso a paso cómo el algoritmo de Dijkstra encuentra los caminos mínimos desde el nodo 1 a los restantes nodos del grafo.



etapa	Nodo seleccionado	Nodos ya seleccionados	Nodos aún no seleccionados	Vector de distancias (D) D[2…5]	Vector de predecesores (P) P[2…5]
inicial	---	{ 1 }	{ 2, 3, 4, 5 }	[13, 26, ∞, 5]	[1, 1, -, 1]
1	5	{ 1, 5 }	{ 2, 3, 4 }	[13, 26, 13, 5]	[1, 1, 5, 1]
2	2 (*)	{ 1, 5, 2 }	{ 3, 4 }	[13, 23, 13, 5]	[1, 2, 5, 1]
3	4	{ 1, 5, 2, 4 }	{ 3 }	[13, 20, 13, 5]	[1, 4, 5, 1]

A partir de la solución obtenida, explicar con detalle cuál es el coste del camino mínimo desde el nodo 1 al nodo 3 y por donde discurre dicho camino mínimo.

La longitud del camino mínimo desde el nodo 1 hasta el nodo 3 está recogida en la posición 3 del vector D, esto es, 20. Para saber por donde discurre dicho camino mínimo hay que empezar acudiendo a la posición 3 del vector P. Dicho valor nos indica cuál es el nodo predecesor al nodo 3 (en dicho camino mínimo), esto es, 4.

4 → 3

Seguidamente habría que acudir a la posición P[4], que nos indica cuál es el nodo predecesor al nodo 4, es decir, 5.

5 → 4 → 3

Y por último habría que ir a la posición P[5], que nos indica cuál es el nodo predecesor al nodo 4, siendo 1.

1 → 5 → 4 → 3

(*) En la etapa 2 se podría haber seleccionado el nodo 4 en lugar del nodo 2, ya que en ambos el coste del camino desde el nodo 1 es 13. La solución final sería la misma.