Ejercicio : Programación Lineal Entera (PLE)

Los profesores del departamento LSI deben repartirse la docencia de varias asignaturas. Cada profesor tiene asignado un número de créditos (ECTS) que deberá cubrir con su docencia. Así mismo, cada uno de los profesores ha indicado cuáles son las asignaturas que le gustaría impartir (recogido en la columna "Preferencias" de la Tabla 2, donde las preferencias aparecen como una lista de tamaño el número de asignaturas, de forma que el elemento j-ésimo de dicha lista representa la preferencia de dicho profesor por la asignatura j.). Las preferencias se expresan como un valor entero [0,3]. Al realizar el reparto de los créditos de las asignaturas entre los distintos profesores, se desea maximizar las preferencias, teniendo en cuenta que cada profesor puede impartir un número determinado de créditos en distintas asignaturas, y que no se permite que un profesor imparta una asignatura para la que presente preferencia igual a 0. Además, hay ciertas asignaturas que se imparten en la misma franja horaria que otras, por lo que un mismo profesor no podrá tener asignada la docencia de asignaturas de la misma franja. Por último, debemos considerar que cada profesor deberá cubrir todos sus créditos de docencia, por lo que no podrá quedar ningún profesor con créditos sobrantes.

Ejemplo: Para los siguientes datos de entrada de la tabla siguiente, la solución óptima sería el reparto que se indica a continuación, con una suma total de preferencias igual a 252 (84ECTS, todos ellos asignados con preferencia 3).

ADDA: Prof_0 (12ECTS) y Prof_1 (12ECTS)

FP: Prof_2 (6ECTS), Prof_3 (12ECTS) y Prof_4 (6ECTS)

PL: Prof_6 (12ECTS)

SO: Prof_5 (6ECTS) y Prof_7 (12ECTS)

SIE: Prof_2 (6ECTS)

ID de la asignatura	Asignatura	ECTS a cubrir	Franja horaria
0	ADDA	24	0
1	FP	24	2
2	PL	12	1
3	SO	18	0
4	SIE	6	0

Tabla 1: Información sobre las asignaturas

ID del profesor	ECTS	Preferencias
0	12	[3,2,2,0,0]
1	12	[3,3,3,3,3]
2	12	[0,3,1,2,3]
3	12	[2,3,2,2,1]
4	6	[0,3,2,1,0]
5	6	[1,2,1,3,2]
6	12	[1,1,3,2,2]
7	12	[1,1,2,3,2]

Tabla 2: Información sobre los profesores

Suponga que existe la clase DatosPAP, con los siguientes métodos implementados:

```
class DatosPAP {
	static Integer getNumAsignaturas() // Número de asignaturas a repartir.
	static Integer getNumProfesores() // Número de profesores en el reparto.
	static Integer getECTSAsignatura(Integer j) // Número créditos a cubrir de la
	asignatura j-ésima.
	static Integer getECTSProfesor(Integer i) // Número de créditos a impartir por el
	profesor i-ésimo..
	static Integer getFranjaHoraria(Integer j) // Franja horaria en la que se imparte
	la asignatura j-ésima.
	static Integer getPreferencia(Integer i, Integer j) // Preferencia del profesor i-
	ésimo para impartir la asignatura j-ésima.
	static Integer getNumFranjasHorarias() // Número de franjas horarias distintas.
}
```

SE PIDE:

a) Defina las variables que necesitará para implementar el modelo descrito en el enunciado y explique qué valores podrán tomar.

Usaremos variables enteras x[i,j] que representarán el número de créditos ECTS que el profesor i impartirá de la asignatura j. El valor 0 significará que ese profesor no imparte esa asignatura.

b) Implemente el modelo lsi para resolver el problema anterior mediante Programación Lineal Entera (PLE).

//SOLUCIÓN USANDO getNumFranjasHorarias()

```
head section

Integer getNumProfesores()

Integer getNumAsignaturas()

Integer getECTSAsignatura(Integer j)

Integer getECTSProfesor(Integer i)

Integer getPreferencia(Integer i, Integer j)

Integer getFranjaHoraria(Integer j)

Integer getNumFranjasHorarias()


Integer n = getNumProfesores()

Integer m = getNumAsignaturas()

Integer y[i,j] = 1 | x[i,j] >= 1, i in 0 .. n, j in 0 .. m

Integer o = getNumFranjasHorarias()
```

```
goal section
max sum(getAfinidad(i,j) x[i,j], i in 0 .. n, j in 0 .. m)

constraints section
sum(x[i,j], i in 0 .. n) = getECTSAsignatura(j), j in 0 .. m
sum(x[i,j], i in 0 .. n | getAfinidad(i,j)=0) = 0, j in 0 .. m
sum(x[i,j], j in 0 .. m) = getECTSProfesor(i), i in 0 .. n
sum(y[i,j], j in 0 .. m | getFranjaHoraria(j)=k) <= 1, i in 0 .. m, k in 0 ..
o

int
x[i,j], i in 0 .. n, j in 0 .. m

//SOLUCION SIN USAR getNumFranjasHorarias()
head section
Integer getNumProfesores()
Integer getNumAsignaturas()
Integer getECTSAsignatura(Integer j)
Integer getECTSProfesor(Integer i)
Integer getPreferencia(Integer i, Integer j)
Integer getFranjaHoraria(Integer j)

Integer n = getNumProfesores()
Integer m = getNumAsignaturas()
Integer y[i,j] = 1 | x[i,j] >= 1, i in 0 .. n, j in 0 .. m

goal section
max sum(getPreferencia(i,j) x[i,j], i in 0 .. n, j in 0 .. m)

constraints section
sum(x[i,j], i in 0 .. n) = getECTSAsignatura(j), j in 0 .. m
```

```
sum(x[i,j], i in 0 .. n | getPreferencia(i,j)=0) = 0, j in 0 .. m

sum(x[i,j], j in 0 .. m) = getECTSProfesor(i), i in 0 .. n

sum(y[i,j], j in 0 .. m | getFranjaHoraria(k)=getFranjaHoraria(j)) <= 1, i in 0 .. n, k in 0 .. m


int

x[i,j], i in 0 .. n, j in 0 .. m
```

EJEMPLO DE MODELO LSI (Este ejemplo no tiene nada que ver con lo que se pide en el enunciado. Es sólo para recordar la sintaxis de la gramática LSI):

```
1 head section
2
3 Integer getN()
4 Integer getM()
5 Double costes(Integer i, Integer j)
6 Integer n = getN()
7 Integer m = getM()
8
9 goal section
10
11 min sum(costes(i,j) x[i,j], i in 0 .. n, j in 0 .. m)
12
13 constraints section
14
15 sum(x[i,j], i in 0 .. n) = 1, j in 0 .. m
16 sum(x[i,j], j in 0 .. m) <= 1, i in 0 .. n
17
18 bin
19
20 x[i,j], i in 0 .. n, j in 0 .. m
```

Tiempo estimado: 30 min.