Práctica 2

Víctor Manuel Arroyo Martín

Técnicas de los Sistemas Inteligentes

Ejercicio 1.-

Este pedía resolver un puzzle cripto-aritmético. Para ello, he hecho un array “cifras” de tamaño el número de letras que hay (10) que puede tomar valores del 0 al 9. A cada letra le he hecho corresponder una posición del array de la siguiente forma:

T	E	S	F	D	I	N	K	R	A
1	2	3	4	5	6	7	8	9	10

Y para revolver el problema he movido cada letra a suposición, es decir, lo he multiplicado por 10000, 1000, 100, 10 o 1 para colocarlas en las centenas, decenas, etc. y las he sumado para componer el número. Por último, he impuesto la suma y su solución, y da como resultado que:

T	E	S	F	D	I	N	K	R	A
3	0	8	6	5	9	7	1	4	2

Ejercicio 2.-

Usando la función count, este problema se vuelve sencillo. Sólo hay que crear un array con la longitud del número que pide el enunciado, 10 del 0 al 9, e imponer que todas sus cifras (mediante un forall) sean el cout (número de apariciones) del número de posición en el que están. Da como solución 6210001000.

Ejercicio 3.-

Para resolver este he creado una variable por profesor en la que cada una puede tomar como valores las horas disponibles en las que pueden dar clase. Por último, he impuesto que las horas deben ser distintas entre todos ellos para que sólo haya un profesor en el aula cada vez, quedando el horario de la siguiente forma:

Profesor	Hora
Profesor 1	14
Profesor 2	12
Profesor 3	13
Profesor 4	10
Profesor 5	9

Ejercicio 4.-

Para resolver el problema he creado dos matrices de tamaño 4x4, una en la que pondré los grupos y asignatura y otra en la que pondré qué profesor da esa asignatura (lo pondré en la misma posición que la clase que da). Las filas son las aulas y las columnas las horas. Para ello, le he asignado a cada asignatura-grupo un identificador:

1	2	3	4	5	6	7	8	9	10	11	12
IA-G1	IA-G2	IA-G3	IA-G4	TSI-G1	TSI-G2	TSI-G3	TSI-G4	FBD-G1	FBD-G2	FBD-G3	FBD-G4

El 0 significa que no hay clase en ese aula. Así, he empezado con un all_different en cada columna para asegurar que la duración de las clases son 1 hora. Además he añadido para que me fuera más fácil, que el grupo i está en el aula i, lo cual lleva de igual forma a un resultado correcto como ahora veremos. Por último he puesto las restricciones para los profesores, es decir, un all_different en las filas para asegurar que el mismo profesor no está dando en dos aulas distintas a la misma hora y he impuesto que cada profesor de a los grupos correspondientes mediante el operador ‘ → ’ . El horario queda de la siguiente forma:

	Aula 1	Aula 2	Aula 3	Aula 4
9-10	IA-G1 prof1	FBD-G2 prof2	-	FBD-G4 prof3
10-11	-	IA-G2 prof1	IA-G3 prof4	TSI-G4 prof3
11-12	TSI-G1 prof1	-	FBD-G3 prof3	IA-G4 prof4
12-13	FBD-G1 prof2	TSI-G2 prof1	TSI-G3 prof3	-

Ejercicio 6.-

Para resolver este problema he creado 5 vectores: color casa, regiones, mascotas, profesiones y bebidas, que representarán lo que hay en cada casa. Además dos posiciones consecutivas en el array significa que esas dos casas están juntas.
Así, le he asignado un identificador a cada elemento que aparece en el enunciado:

1	Rojo	6	Vasco	11	Perro	16	Pintor	21	Té
2	Verde	7	Catalán	12	Caracoles	17	Escultor	22	Café
3	Blanco	8	Gallego	13	Zorro	18	Diplomático	23	Leche
4	Amarillo	9	Navarro	14	Caballo	19	Violinista	24	Zumo
5	Azul	10	Andaluz	15	Cebra	20	Médico	25	Agua

Primero he aplicado un all_different a todos los vectores ya que no se pueden repetir índices en ninguno y he ido apartado por apartado escribiendo las restricciones mediante '< - >' o '→' y también las restricciones implícitas que pudiera haber, quedando el siguiente resultado:

Casa 1	Casa 2	Casa 3	Casa 4	Casa 5
Amarillo	Azul	Rojo	Blanco	Verde
Andaluz	Navarro	Vasco	Catalán	Gallego
Zorro	Caballo	Caracoles	Perro	Cebra
Diplomático	Médico	Escultor	Violinista	Pintor
Agua	Té	Leche	Zumo	Café

Por tanto, la respuesta a la pregunta es: La cebra está en la casa 5 y bebe agua el de la casa 1.

Ejercicio 7.-

En este ejercicio se pedía minimizar el tiempo de construcción de una casa. Lo he resuelto creando un array del tamaño el número de tareas llamado t_inicios que guarda el momento de inicio de cada tarea. Los índices los he puesto según el orden de las letras, es decir, 1-A, 2-B, etc. Y para las restricciones he puesto que el momento de inicio de una tarea debe ser posterior al de sus predecesores más la duración de estos, empezando en el momento 1 la tarea A que es la primera. Y por último he creado una variable que será la que minimicemos y es la suma de la última tarea de todas (la I) más su duración. La solución queda de la siguiente forma:

Tiempo de construcción de la casa=19.

Tarea	A	B	C	D	E	F	G	H	I
Inicio	1	8	11	8	16	16	16	8	17

Ejercicio 8.-

Para este ejercicio, con respecto a los tiempos de inicio he usado lo mismo que en el 7. Esta vez lo que cambia son las restricciones posteriores:
He creado una matriz que me servirá para hacer las comprobaciones de los trabajadores y comprobar que en cada momento no hay más de 3. Tiene 28 columnas ya que tras ejecutarlo he visto que se completa en menos de 28. También tengo en el array nTrabajadores el número de trabajadores necesarios para cada tarea. Los índices de las tareas las he mantenido con respecto al ejercicio 7.

En primer lugar, para cada momento de inicio debe haber el número de trabajadores necesarios y esto durante toda la duración del trabajo, lo he hecho mediante un forall. También hay que comprobar que en cada momento no hay más de tres trabajadores. Así, los tiempos de inicios que obtengo son:

Tiempo total=22

Trabajo	A	B	C	D	E	F	G	H	I
Inicio	1	16	19	8	20	20	22	8	21

Ejercicio 9.-

Repetimos de nuevo lo del ejercicio 7 para la parte de satisfacción y tiempos de inicio en tareas consecutivas. La diferencia es que ahora el vector duración ahora es variable y al principio establecemos sus restricciones. Para ello usamos el array trab que indicará qué trabajador hace cada tarea y la matriz trabajadores donde guardamos qué trabajador está haciendo cada tarea, que tiene 15 columnas porque el programa ocupa menos y hacerla más grande solo lleva a más tiempo de ejecución. La primera restricción sobre el vector duración es que la duración para todas las tareas debe ser la de alguno de los trabajadores y todas tardan más de 1, como es lógico.

Para la siguiente aprovechamos que los trabajadores tienen tiempos distintos de ejecución de cada tarea y establecemos que si la duración de la tarea i es la misma que la del trabajador j en esa misma tarea, la tarea es del trabajador j.
También hay que asegurarse de que un trabajador está sólo en una tarea cada vez y de que el trabajo que inicia un trabajador en concreto debe llevarlo hasta el final de su duración. Así, queda la siguiente solución:

Tiempo mínimo en construir: 12

Trabajo	A	B	C	D	E	F	G	H	I
Inicio	1	5	8	8	10	10	10	5	11

Ejercicio 10.-

Un clásico problema de la mochila. Para resolverlo he asociado un índice a cada objeto en el orden en el que aparecen en la tabla y he creado dos arrays peso y preferencia donde guardo estos valores en orden. El array objetos lo he establecido a tamaño 8 para recortar en tiempo de ejecución porque de todas formas el óptimo ocupa menos de 8 casillas.

Así, he impuesto las restricciones: primero que el peso de la mochila (sumpeso) sea menor que el peso máximo del enunciado, que todos los objetos sean diferentes y que la preferencia sea la máxima posible. Con ello queda como resultado una mochila de:

Peso=274kg
Suma de preferencias=705

Indice	11	9	5	4	3	2	1
Objeto	Protector solar	Queso	Azúcar	Sandwich	Agua	Compás	Mapa