Práctica 2: Satisfacción de Restricciones:

UNIVERSIDAD DE GRANADA

Luis González Romero, XXXXXXXXX, luisgonromero@correo.ugr.es
Grupo 2: Viernes 17:30-19:30

Escuela Técnica Superior de Ingeniería informática y Telecomunicaciones

3 de mayo de 2020

Práctica 2: Satisfacción de Restricciones

Memoria sobre la primera práctica de la asignatura Técnicas de los Sistemas Inteligentes cursada en la ETSIIT, de la UGR.

Luis González Romero, XXXXXXXXX, luisgonromero@correo.ugr.es

Abril de 2020

Índice

1. Ejercicio 1 **4**

1.1. Enunciado 4

1.2. Solución 4

2. Ejercicio 2 **5**

2.1. Enunciado 5

2.2. Solución 5

3. Ejercicio 3 **6**

3.1. Enunciado 6

3.2. Solución 6

4. Ejercicio 10 **7**

4.1. Enunciado 7

4.2. Solución 7

1. Ejercicio 1

1.1. Enunciado

El siguiente problema plantea un problema criptoaritmético, de forma que cada letra codifica un único dígito (es decir, un número entero en [0,9]) y cada dígito está asignado a una única letra. Se pide encontrar una asignación de dígitos a letras que satisfaga la siguiente suma (una posible traducción del alemán es "Prueba a fondo tus fortalezas"):
TESTE
+ FESTE
+ DEINE
========
KRAFTE

1.2. Solución

Para poder realizar el problema he recreado la suma como si fueran numeros, cada uno dependiendo de su posición le corresponden x ceros. Para la posición 1, 0 ceros; para la posición 2, 1 cero; para la posición 3, 2 ceros...

Cada número será representado con una variable, la cual puede ser *DIGITO* o bien *PRIMER_DIGITO* para determinar su rango de valores. La última restricción sería que cada letra represente un único valor, para ello uso *all_different()*.

2. Ejercicio 2

2.1. Enunciado

Encontrar un número X de 10 dígitos que satisfaga que el primer dígito de X representa el número de 0s en X, el segundo dígito de X representa el número de 1s en X, etc...

Por ejemplo, el número $X = 6210001000$ satisface dicha condición.

2.2. Solución

Para este problema he creado un array y comprobado que para todas las posiciones el número de ocurrencias de i es el mismo que el valor del número de la posición_i. Esto se hace de forma sencilla haciendo uso de *count()*:

3. Ejercicio 3

3.1. Enunciado

Encontrar una asignación de horarios que satisfaga las siguientes condiciones:

- Existe una única aula, disponible entre las 9:00 y las 15:00.
- El aula sólo puede estar ocupada por un único profesor/a al mismo tiempo.
- Cada profesor/a debe impartir una clase de 1h de duración.
- Cada profesor/a tiene las siguientes restricciones de horarios:

Profesor	Horario disponible
Prof-1	11:00 – 15:00
Prof-2	11:00 – 13:00
Prof-3	10:00 – 14:00
Prof-4	10:00 – 13:00
Prof-5	11:00 – 13:00
Prof-6	09:00 – 15:00

3.2. Solución

Para este problema he creado un rango de valores para cada profesor dependiendo de su horario disponible. Cada hora de la asignación de horario será una variable con rango dentro del horario disponible de cada profesor. Para la restricción he usado un array *horario* con las seis horas, una por profesor y he puesto una restricción para que todas sean distintas.

4. Ejercicio 10

4.1. Enunciado

Un turista desea llenar su mochila con varios objetos para hacer un viaje. El peso máximo que aguanta la mochila son 275kg, y cada objeto tiene distinto grado de preferencia para el turista (la preferencia se mide en una escala de 0 a 200, donde los números más altos representan los objetos más deseados por el turista). En la tabla siguiente se muestran los objetos, su peso y su preferencia. Encontrar el conjunto de objetos cuya suma de preferencias sea máxima sin exceder el peso máximo que aguanta la mochila.

Objeto	Peso (Kg)	Preferencia
Mapa	9	150
Compás	13	35
Agua	153	200
Sandwich	50	160
Azúcar	15	60
Lata	68	45
Plátano	27	60
Manzana	39	40
Queso	23	30
Cerveza	52	10
Protector Solar	11	70
Cámara	32	30

4.2. Solución

Para este problema lo he enfocado como el problema clásico de optimización combinatoria de la mochila. He creado tres array:

- x: array que determina si un objeto i es seleccionado.
- *peso*: array con los pesos mostrados en la tabla para cada objeto i
- *pref*: array con las preferencias mostradas en la tabla para cada objeto i

Una de las restricciones será que la suma(peso) de cada objeto escogido sea menor que el peso máximo. Y ya nos quedará maximizar que cada objeto i que sea seleccionado maximice la suma del total escogido(preferencia).