UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Algorítmica (11593)

Voraces y RP–08 de enero de 2018

etsinf Escuela Técnica Superior de Ingeniería Informática

NOMBRE: _____ **NÚM.:** _____

1 — 4 puntos

Dada una mochila de capacidad W y una serie de N objetos de peso w_i para i entre 1 y N, se desea **llenar completamente** la mochila maximizando el número de objetos que se introducen. Para ello:

a) Expresa el problema en términos de optimización: expresa formalmente el conjunto de soluciones factibles, la función objetivo a maximizar y la solución óptima buscada.

b) Describe los siguientes conceptos sobre los estados que serán necesarios para el algoritmo:
   1) Representación de un estado (no terminal) y su coste espacial.
   2) Condición para que un estado sea solución.
   3) Identifica el estado inicial que representa todo el conjunto de soluciones factibles.

c) Define una función de ramificación. Contesta a las siguientes cuestiones:
   1) Explica la función.
   2) Define la función (en python o en lenguaje matemático). Calcula el coste temporal.

d) Diseña una cota optimista no trivial. Contesta a las siguientes cuestiones:
   1) Explica la cota (caso general, de un estado intermedio).
   2) Explica la cota del estado inicial.
   3) Define la cota (en python o en lenguaje matemático). Calcula el coste temporal.
   4) Estudia si se puede mejorar el cálculo de la cota, haciéndolo incremental. Define la cota así definida (en python o en lenguaje matemático). Calcula el coste temporal.

2 — 3 puntos

Aplicar la técnica de ramificación y poda con poda explícita al siguiente problema: sean N trabajadores y N tareas. Cualquier trabajador puede realizar cualquier tarea, con una duración en unidades de tiempo que puede variar dependiendo de la asignación trabajador-tarea. Se necesita realizar todas las tareas asignando exactamente una tarea a cada trabajador y un trabajador a cada tarea, de forma que el tiempo total de realizar todas las tareas se minimice. Muestra la ejecución sobre la siguiente instancia:

	Tarea1	Tarea2	Tarea3	Tarea4
p1	9	6	7	7
p2	6	4	3	7
p3	6	8	4	8
p4	7	5	9	6

Por ejemplo, la posición (1,2) de la matriz de costes indica que el trabajador p1 necesita 6 unidades de tiempo para ejecutar la tarea 2.

a) Explica brevemente cómo vas a representar un estado incompleto y un estado solución. Pon un ejemplo.

b) Explica qué cota optimista vas a emplear.

c) La traza se debe mostrar como el *conjunto de estados activos* de cada iteración del algoritmo indicando la cota optimista de cada estado y marcando el estado que se selecciona para la siguiente iteración. La traza debe indicar también si se actualiza la variable mejor solución y la aplicación, cuando corresponda, de la poda explícita. En caso de inicializar la mejor solución inicial a una solución arbitraria, debes indicar el algoritmo que usas y su coste.

3 — 3 puntos

Queremos empaquetar N objetos de pesos $w_1, \ldots, w_N$ gastándonos el menor dinero en las cajas. La tienda de cajas nos puede proveer de un lote de cajas siempre que **todas sean del mismo tipo** y nos ofrece una lista con la capacidad y el precio de cada tipo de caja. Un ejemplo de lista sería `[(25,3.5),(50,6.8),(70,9.3)]` que indica que hay cajas de 25Kg de capacidad que valen 3.5 euros cada una, otras de 50Kg a 6.8 euros la unidad y, finalmente, cajas de 70Kg a 9.3 euros.

Se pide realizar una función Python que reciba la lista de objetos y los datos del proveedor de cajas. La función debe devolver una tupla con el número de cajas necesarias y el tipo de caja. Si no hay solución deberá devolver (-1,None), mientras que para 0 cajas no hace falta especificar el tipo de la caja, con lo que puede devolver (0,None). Indica también el coste del algoritmo desarrollado y si resuelve o no este problema de manera óptima. Un ejemplo:

```
tipos, W = [(25,3.5),(50,6.8),(70,9.3)], [10,30,20,40,40,30,25,25,40]
print(empaquetar(W,tipos))
```

Una posible solución a esta instancia sería el valor `(4,70)` que corresponde a gastarse 37.2 euros en 4 cajas de 70Kg de capacidad.