UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Algorítmica (11593)
Voraces y RP–9 de enero de 2019

etsinf
Escuela Técnica Superior de Ingeniería Informática

NOMBRE: _____ **NÚM.:** _____

1 — 4 puntos

Una empresa de reciclaje debe separar los residuos en N tipos (por ejemplo, para $N = 8$ los tipos de residuos podrían ser: vidrio, papel y cartón, plásticos, tetrabrik, textiles, maderas, metales y componentes electrónicos).

Como restricción, la tecnología de separación utilizada solamente permite realizar el proceso de forma **secuencial**. Según en qué orden se clasifique el material, el tiempo que se requiere cambia, y esa cantidad viene determinada por la matriz t donde $t[n, i]$ indica el tiempo que se necesita para clasificar el material de tipo n, para $0 \leq n < N$, cuando en el proceso de clasificación este tipo de residuo se separa en la posición i-ésima, para $0 \leq i < N$.

La empresa debe determinar el orden en que deben separarse los N tipos de residuos para reducir el tiempo total del proceso. Diseña un algoritmo de Ramificación y Poda que proporcione el orden de separación de mayor eficiencia, calculado como la suma del tiempo de separación de cada uno de los N tipos de residuos. Para ello:

a) Expresa el problema en términos de optimización: expresa formalmente el conjunto de soluciones factibles X, la función objetivo a optimizar f y la solución óptima buscada. Explica brevemente cómo expresas una solución x del conjunto X.

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. Pon un ejemplo para la instancia presentada.
   2) Condición para que un estado sea solución. Pon un ejemplo para la instancia presentada.
   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).
   3) Calcula el coste temporal.
   4) Pon un ejemplo para la instancia presentada.

d) Diseña una cota optimista no trivial. Contesta a las siguientes cuestiones:

   1) Explica la cota (caso general, de un estado intermedio).
   2) Explica el cálculo de 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.

e) ¿Se te ocurre alguna forma de inicializar la variable "mejor solución vista hasta el momento" a un valor diferente al valor pésimo? Explica cómo y el coste de esa inicialización.

Solución: Este problema es una instanciación del problema de "ensamblaje" visto en clase.

Una empresa de transporte dispone de una flota de C camiones para transportar una carga compuesta de P paquetes, cada uno de ellos con un peso p_i y marcado con una urgencia u_i, para i entre 1 y P. El operario ha estimado que no puede transportar todos los paquetes, pero desea saber qué paquetes se pueden transportar en cada camión de forma que se maximice la suma de los niveles de urgencia de la carga total, sabiendo que cada camión soporta una carga máxima de c_j, para $1 \leq j \leq C$.

Realiza una traza de un algoritmo de Ramificación y Poda basada en *poda explícita* que calcule la selección de paquetes a transportar en cada camión de modo que el nivel de urgencia sea el máximo, dada la siguiente instancia: 3 camiones que pueden cargar hasta 10, 6 y 4 unidades de peso, y 6 paquetes con los siguientes valores de peso y urgencia:

	1	*2*	*3*	*4*	*5*	*6*
p	4	2	5	8	3	6
u	10	30	100	30	10	5

Responde a las siguientes cuestiones:

a) Explica brevemente cómo vas a representar: el estado inicial, un estado incompleto y un estado solución. Pon un ejemplo sobre la instancia.
b) Explica brevemente la cota optimista que vas a utilizar.
c) Para la traza sobre la instancia presentada, ten en cuenta lo siguiente: La traza se debe mostrar el *conjunto de estados activos* de cada iteración del algoritmo indicando la cota optimista de cada estado (ej: como superíndice) y subrayando el estado que se selecciona para la siguiente iteración. Hay que indicar también si se actualiza la variable mejor solución y la poda explícita u otras podas. Si utilizas una solución inicial para adelantar la poda, indica qué algoritmo utilizas para obtenerla y cual es su coste.

Solución: Este problema es una instanciación del problema de las múltiples mochilas visto en clase. Os remitimos a los apuntes para el modelado del problema y la función de ramificación. Para la traza vamos a usar como cota superior el asumir que los camiones no tienen limitación de carga. En caso de empate de la cota siempre elegiremos el estado que más cerca esté de la solución.

Además, haremos un preproceso para ordenar los paquetes de forma descendiente de acuerdo a su nivel de urgencia por unidad de peso. Según este proceso, que supone un coste $O(N \log N)$, los paquetes quedarían ordenados de la forma siguiente:

	1	*2*	*3*	*4*	*5*	*6*
u/p	20	15	3.75	3.3	2.5	0.8
p	5	2	8	3	4	6
u	100	30	30	10	10	5

Para intentar adelantar la poda, obtendremos una solución inicial con el siguiente algoritmo voraz: ir cargando paquetes (en el orden anterior) en el camión que primero quepa, ordenados estos de menor a mayor capacidad de carga. Este algoritmo tiene un coste $O(N \log N)+O(C \log C)$. Así obtendremos la solución $x = (2, 3, 1, 0, 0, 0)$ con un valor de función objetivo $fx = 160$.

La cota del estado inicial (?) sería la suma de los niveles de urgencia de todos los paquetes, con un valor 185. Pondremos en cada estado tanto el valor de la cota como el espacio libre que queda por ocupar en cada camión:
$A_0 = \{(?, 185, (10, 6, 4)))*\}$

$A_1 = \{(1?, 185, (5, 6, 4))*, (2?, 185, (10, 1, 4))\}$
En esta iteración, no se inserta el estado (0?) porque tiene una cota optimista (85) menor que la mejor solución en curso (160). El estado (3?) no se crea porque el peso del paquete 1 sobrepasa la capacidad de carga del camión 3.

$A_2 = \{(2?, 185, (10, 1, 4)), (11?, 185, (3, 6, 4))*, (12?, 185, (5, 4, 4)), (13?, 185, (5, 6, 2))\}$

$A_3 = \{(2?, 185, (10, 1, 4)), (12?, 185, (5, 4, 4))*, (13?, 185, (5, 6, 2))\}$
Los estados $(111?), (112?), (113?)$ no se crean porque el peso del paquete 3 sobrepasa la capacidad de carga del espacio disponible en los camiones.

$A_4 = \{(2?, 185, (10, 1, 4)), (13?, 185, (5, 6, 2))*\}$
Los estados $(121?), (122?), (123?)$ no se crean porque el peso del paquete 3 sobrepasa la capacidad de carga del espacio disponible en los camiones.

$A_5 = \{(2?, 185, (10, 1, 4))*\}$
Los estados $(131?), (132?), (133?)$ no se crean porque el peso del paquete 3 sobrepasa la capacidad de carga del espacio disponible en los camiones.

$A_6 = \{(21?, 185, (8, 1, 4))*, (23?, 185, (10, 1, 2))\}$
El estado $(22?)$ no se crea porque el peso del paquete 3 sobrepasa la capacidad de carga del espacio disponible en el camión 2.

$A_7 = \{(23?, 185, (10, 1, 2)), (211?, 185, (0, 1, 4))*\}$
Los estados $(212?), (213?)$ no se crean porque el peso del paquete 3 sobrepasa la capacidad de carga del espacio disponible en los camiones 2 y 3.

$A_8 = \{(23?, 185, (10, 1, 2)), (2110?, 175, (0, 1, 4)), (2113?, 185, (0, 1, 1))*\}$

$A_9 = \{(23?, 185, (10, 1, 2))*, (2110?, 175, (0, 1, 4)), (21130?, 175, (0, 1, 1))\}$

$A_{10} = \{(2110?, 175, (0, 1, 4)), (21130?, 175, (0, 1, 1)), (231?, 185, (2, 1, 2))*\}$

$A_{11} = \{(2110?, 175, (0, 1, 4)), (21130?, 175, (0, 1, 1))*, (2310?, 175, (2, 1, 2))\}$

$A_{12} = \{((2110?, 175, (0, 1, 4))*, (2310?, 175, (2, 1, 2))\}$
Al ramificar el estado $(21130?)$ obtenemos la solución (211300) con un valor de 170. Se actualiza la mejor solución y se entra en el proceso de poda explícita que no elimina ningún estado, pues sus cotas son mayores que 170.

$A_{13} = \{((21103?, 175, (0, 1, 0))*, (2310?, 175, (2, 1, 2))\}$
Al ramificar el estado $(2110?)$ obtenemos el estado $(21100?)$ con cota 165, que no se guarda.

$A_{14} = \{(2310?, 175, (2, 1, 2))*\}$
Al ramificar el estado $(21103?)$ obtenemos la solución $((211030)$ con un valor de 169, que no se guarda.

En la siguiente iteración se ramifica $(2310?)$, obteniéndose: $(23100?)$, de cota 165, que no se guarda, por lo que el conjunto de estados activos se vacía y el algoritmo termina, devolviendo como mejor solución (211300) con un valor de 170.

3 **2 puntos**

Nos hemos sacado un abono de 1 día para un festival de cine que nos permite asistir a todas las sesiones que deseemos. Hay N películas a concurso y deseamos ver el mayor número posible de ellas. Tenemos el programa y cada película i se exhibe en un horario h_i, con una duración d_i. Todas las películas se exhiben en una sala multicine, así que el tiempo de desplazamiento de una sala a otra es despreciable. Eso sí, no podemos entrar en una proyección ya comenzada.

Se pide realizar una función Python que reciba la lista de las N películas:

$$P = \{(h_1, d_1), (h_2, d_2), \ldots, (h_N, d_N)\}$$

y que devuelva una lista de las películas seleccionadas a las que podemos asistir. Las horas tienen un formato `m(hh,mm)`, con

```
def m(hor,min): return hor*60+min
```

La duración viene expresada en minutos. Sigue una estrategia voraz e indica el coste del algoritmo desarrollado y si resuelve o no este problema de manera óptima.

```
P = [m(18,00),126), (m(22,30),120), (m(16,00),90), (m(17,00),110), (m(20,30),140),
(m(16,30),210), (m(19,00), 95), (m(21,00), 110), (m(18,00),110) ]
print("Los peliculas a ver son:")
print(seleccionar(P))
```

Solución: Este problema es una instanciacón del problema de "selección de actividades" visto en clase.