UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Algorítmica (11593)
Voraces y RP–19/enero/2021

etsinf
Escuela Técnica Superior de Ingeniería Informática

NOMBRE: _____ **NÚM.:** _____

1 **2 puntos**

Eres el asistente personal de una famosa actriz y debes organizar su agenda de entrevistas para promocionar su última película. La actriz ha recibido una avalancha de propuestas de entrevistas, todas ellas en el único día que visita la ciudad. Cada propuesta de entrevista tiene estos campos:

- hora (entero entre 0 y 23) y minuto (entre 0 y 59) de inicio,
- duración de la entrevista (en minutos),
- nombre del periodista (para distinguir propuestas que puedan coincidir en el resto).

Lamentablemente, hay propuestas solapadas y te han encomendado seleccionar un conjunto que maximice el número de entrevistas que puede mantener. Como todas las entrevistas son en la misma habitación del hotel, el tiempo desde que finaliza una entrevista y comienza la siguiente es despreciable. En caso de que varias soluciones tengan el mismo número de entrevistas máximo, cualquiera de ellas es igualmente válida.

Se pide: diseñar una función Python que utilice una estrategia voraz para seleccionar las entrevistas. La función recibe una lista de propuestas (lista de tuplas `(hora,minuto,duracion,periodista)`) y devuelve la lista/conjunto de los periodistas a los que se acepta la entrevista. En el siguiente ejemplo, en el primer elemento de la lista, Jennifer propone realizar una entrevista a las 15:30 que dura 40 minutos, y de la misma forma se interpretan las otras solicitudes de la lista:

```
propuestas = [(15,30,40,"Jennifer"), (18,10,20,"Paul"), (16,00,30,"Mark"),
              (16,25,40,"Robert"),  (17,00,30,"Anna"), (17,40,50,"Mary")]
seleccionados = selecciona_entrevistas(propuestas)
```

¿Qué devuelve tu algoritmo para el ejemplo presentado?

Solución: Claramente este problema se corresponde al problema visto en clase de teoría de voraces llamado "selección de actividades". Para este problema se conoce una estrategia voraz óptima consistente en elegir las actividades que no solapen ordenadas por instante de finalización (elegir primero las que terminen antes). En este caso, las actividades son las entrevistas. Una posible solución es:

```
def selecciona_entrevistas(propuestas):
    C = [(hora*60+minuto, hora*60+minuto+duracion, tag)
         for (hora,minuto,duracion,tag) in propuestas]
    resul,t_prev = set(), min(s for (s,t,tag) in C)
    for s,t,tag in sorted(C, key=lambda x:x[1]):
        if t_prev <= s:
            resul.add(tag)
            t_prev = t
    return resul

propuestas = [(15,30,40,"Jennifer"), (18,10,20,"Paul"), (16,00,30,"Mark"),
              (16,25,40,"Robbert"),  (17,00,30,"Anna"), (17,40,50,"Maria")]
seleccionados = selecciona_entrevistas(propuestas)
print(seleccionados)
```

El algoritmo devuelve: `set(['Robert', 'Paul', 'Jennifer'])`

La empresa de transporte Paraná desea fletar un avión para transportar su mercancía, formada por N paquetes, de pesos $p_1, p_2, \ldots, p_N$. Cada paquete, adicionalmente, viene marcado con su beneficio $b_1, b_2, \ldots, b_N$. Hay que cargar los paquetes distribuidos entre las 2 bodegas, de capacidades A y B. Para asegurar la estabilidad del avión, la diferencia de los pesos cargados en cada bodega no puede superar un umbral Θ. Sabiendo que no se pueden transportar todos los paquetes, pues el peso total de estos, $\sum_{i=1}^{N} p_i$, supera la capacidad de carga del avión, $A + B$, se desea seleccionar el subconjunto de paquetes que maximice el beneficio total, que viene dado por la suma del beneficio de los paquetes cargados en el avión. Para resolver el problema por Ramificación y poda, se pide:

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

Solución: Este problema es una pequeña variante del problema de las múltiples mochilas, con 2 mochilas (bodegas), teniendo en cuenta adicionalmente que la diferencia de peso entre las dos bodegas no puede superar el umbral. Podemos representar una solución como una tupla de N elementos $(x_1, x_2, ..., x_N)$, donde $x_i = 0$ indica que el paquete i-ésimo no se carga en el avión, $x_i = 1$ indica que el paquete i-ésimo se carga en la bodega A y con $x_i = 2$ indicamos que se carga en la bodega B. Por ejemplo, si hay 6 paquetes, una solución sería $(0, 1, 1, 2, 0, 1)$, esto es, los paquetes segundo, tercero y sexto se cargan en la bodega A, el paquete cuarto en la bodega B, y los paquetes primero y quinto no se cargan. Sería una solución siempre que se cumplan las restricciones (caben los paquetes en cada bodega y no se supera la diferencia de peso Θ entre ambas). Formalizamos:

$$X = \{(x_1, x_2, ..., x_N) \in \{0, 1, 2\}^N \mid \sum_{1 \leq i \leq N, x_i = 1} w_i \leq A, \sum_{1 \leq i \leq N, x_i = 2} w_i \leq B, \mid \sum_{1 \leq i \leq N, x_i = 1} w_i - \sum_{1 \leq i \leq N, x_i = 2} w_i \mid \leq \Theta\}$$

$$f((x_1, x_2, ..., x_N)) = \sum_{1 \leq i \leq N, x_i \neq 0} b_i$$

$$\hat{x} = \operatorname*{argmax}_{x \in X} \sum_{1 \leq i \leq N, x_i \neq 0} b_i$$

Un estado se representará con una secuencia incompleta $(x_1, x_2, ..., x_k)$, con $k < N$. Tendrá un coste espacial de $O(N)$. El estado inicial se puede representar como una tupla vacía (?). Para que un estado sea solución la tupla deberá tener una talla igual a N y además que se cumplan la restricciones:

la suma de los pesos de los paquetes cargados en la primera bodega no puede superar A, en la segunda bodega no puede superar B y la diferencia máxima de peso debe ser Θ.

La función de ramificación dará lugar a tres nuevos estados como mucho: el estado que añade la decisión de no cargar el siguiente paquete $k+1$ y, hasta dos nuevos estados hijo si la suma del peso de los paquetes cargados hasta el momento en cada bodega más el del nuevo paquete de peso p_{k+1} no sobrepasa la capacidad de esa bodega:

$$\begin{aligned} branch(x_1, x_2, ..., x_k, ?) = \{&(x_1, x_2, ..., x_k, 0, ?), \\ &(x_1, x_2, ..., x_k, 1, ?) \text{ si } \sum_{1 \le i \le k; x_i = 1} p_i + p_{k+1} \le A, \\ &(x_1, x_2, ..., x_k, 2, ?) \text{ si } \sum_{1 \le i \le k; x_i = 2} p_i + p_{k+1} \le B\} \end{aligned}$$

Para que la comprobación de la condición de no sobrepasar la capacidad de cada bodega sea eficiente, habrá que mantener en el estado el peso acumulado hasta el momento o, equivalentemente, la capacidad libre, en cada bodega.

La cota para un estado $(x_1, \ldots, x_k, ?)$, con $k < N$, estará compuesta por la suma de la función objetivo aplicada a la parte conocida, $\sum_{1 \le i \le k; x_i \neq 0} b_i$, más una estimación optimista de lo que queda por completar.

En pricipio, la cota optimista "cabe todo" es la trivial (es decir, añadir $\sum_{k+1 \le i \le N} b_i$). Esta cota se puede calcular de forma incremental

- En el estado inicial: $cota\_superior(?) = \sum_{1 \le i \le N} b_i$, con coste $O(N)$
- En un estado intermedio de forma incremental con coste $O(1)$:

$$cota\_superior(x_1, x_2, \ldots, x_k, 0, ?) = cota\_superior(x_1, x_2, \ldots, x_k, ?) - b_{k+1}$$

$$cota\_superior(x_1, x_2, \ldots, x_k, a, ?) = cota\_superior(x_1, x_2, \ldots, x_k, ?), \text{para } a \in [1, 2]$$

Para diseñar una cota superior más ajustada podemos utilizar, por ejemplo, el hecho de que las múltiples mochilas con fraccionamiento son equivalentes a una sola mochila de capacidad la suma de las capacidades. Por tanto, podríamos considerar como cota optimista para la parte desconocida el resultado de aplicar el algoritmo voraz de la mochila con fraccionamiento. En este caso, la capacidad libre de la "mochila" (las dos bodegas del avión) es $(A - \sum_{1 \le i \le k, x_i = 1} p_i) + (B - \sum_{1 \le i \le k, x_i = 2} p_i)$ en la que seleccionar paquetes a cargar desde $k+1$ hasta N. Para ello sería de utilidad un preproceso para ordenar los N paquetes según su beneficio unitario, con un coste $O(N \log N)$.

3 **4 puntos (*)**

Se han preparado N convoyes de material sanitario en las sedes donde se ha fabricado. Todos los convoyes son idénticos y deben ser traslados a los N centros de distribución del país, un convoy a cada centro de distribución. Debido a los recientes temporales, las rutas entre las sedes y los centros de distribución se han visto alteradas, por lo que se debe planificar el envío de forma que la suma de los tiempos de llegada a todos los centros de distribución sea el menor posible. Se ha hecho una estimación del tiempo de llegada de cada convoy a cada centro de distribución. En particular, hay $N = 5$ convoyes y centros de distribución, y la estimación del tiempo de llegada de cada convoy a cada centro de distribución viene dada en la matriz T:

	centro				
	1	*2*	*3*	*4*	*5*
convoy *1*	9	6	3	4	11
convoy *2*	8	5	5	1	5
convoy *3*	5	4	4	6	7
convoy *4*	10	3	2	2	2
convoy *5*	6	2	5	5	1

Realiza una traza de un algoritmo de Ramificación y poda que distribuya los convoyes entre los centros de forma que la suma de los tiempos de llegada a todos los centros de distribución sea la menor posible. Para ello:

- Sigue una estrategia por primero el mejor (en caso de empate, el estado más cercano a una solución) y usa poda explícita o implícita. Indica cuál usas.
- 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 implícita u otras podas en cada iteración que se produzca.

Responde, adicionalmente, 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 de cada tipo de estado.
b) Para el ejemplo de solución factible que has dado, calcula su valor de función objetivo.
c) Explica brevemente la cota optimista que vas a utilizar.
d) Explica cómo calculas la cota inferior para el estado inicial. ¿Qué coste tiene?
e) Explica cómo calculas la cota inferior para un estado incompleto y cómo puedes calcularla de forma incremental en un estado hijo. ¿Qué coste tiene?
f) Si para la traza utilizas el esquema que inicializa la variable mejor solución $\hat{x}$ a una solución factible o a una cota pesimista, describe qué algoritmo o método utilizas para calcularla y cuál es su coste temporal.

(*) La traza tendrá un valor de **3 puntos**, de los cuales, **1.5 puntos** evaluarán la parte más mecánica de la traza, que se corrresponde con la parte de la "Prueba objetiva de las prácticas". Las cuestiones tendrán un valor total de **1 punto**.

Solución: El conjunto de soluciones serán todas las posibles asignaciones de convoyes a centros. Podemos representarlas como N-tuplas $(x_1, x_2, \ldots, x_N)$ donde x_i representa el centro al que se ha enviado el convoy i-ésimo. Como cada convoy debe ir a un solo centro, las soluciones serán las permutaciones de los N centros. Así, un ejemplo de solución será $(2, 4, 5, 3, 1)$ que indica que el convoy 1 se ha enviado al centro de distribución 2, el convoy 2 al centro 4, el convoy 3 al centro 5, el 4 al centro 3 y el último convoy al centro 1. Un estado intermedio será una tupla incompleta, por ejemplo, $(2, 4, ?)$ y representaremos el estado inicial como la tupla vacía $(?)$. El valor objetivo de una solución será el tiempo total empleado en la distribución. Para la solución $(2, 4, 5, 3, 1)$ la matriz de tiempos nos indica que es 6+1+7+2+6=22.

Podemos usar como cota optimista la suma de los costes de la parte conocida (el tiempo ya usado en los convoyes ya asignados) y, para la parte desconocida, la estimación que los convoyes que faltan por enviar lo podrán hacer al centro al que lleguen en el menor tiempo posible, aunque a ese centro ya se le haya asignado un convoy. Hay una cota algo más ajustada que consiste en utilizar el centro disponible de menor coste sin considerar los ya asignados (pero tendría un coste mayor).

Para hacer eficiente el cálculo de la cota, podemos preprocesar el menor tiempo con que un convoy puede llegar a un centro y almacenarlo en un vector $minT$:

convoy	*1*	*2*	*3*	*4*	*5*
$minT$	3	1	4	1	2

Este preproceso tiene un coste $O(N^2)$, y el cálculo de la cota en el estado inicial requiere la realización de $O(N)$ sumas: $cota\_inferior((?)) = 3 + 1 + 4 + 1 + 2 = 11$. Por otra parte, cuando estamos calculando la cota en un estado intermedio lo podemos hacer de forma incremental con coste $O(1)$. Por ejemplo: $cota\_inferior((2, 4, 5, ?)) = cota\_inferior((2, 4, ?)) - minT(3) + T(3, 5)$, esto es, sustituyendo la estimación optimista del convoy elegido al ramificar por su valor real.

Se puede encontrar una solución inicial, subóptima, siguiendo una estrategia voraz, para adelantar la poda. Esta solución se obtendría "eligiendo el menor tiempo para cada convoy, en orden incremental, siempre que el centro aún no estuviera abastecido", que tiene un coste $O(N^2)$. Para la instancia dada, la solución voraz obtenida será: $(3, 4, 2, 5, 1)$, con un tiempo total de 16:

```
$ python3 voraces.py
convoy 0 elijo centro 2 coste 3
convoy 1 elijo centro 3 coste 1
convoy 2 elijo centro 1 coste 4
convoy 3 elijo centro 4 coste 2
convoy 4 elijo centro 0 coste 6
(16, (3, 4, 2, 5, 1))
```

CÓDIGO:

```
m = [[9,  6,  3,  4, 11],
     [8,  5,  5,  1,  5],
     [5,  4,  4,  6,  7],
     [10, 3,  2,  2,  2],
     [6,  2,  5,  5,  1]]

def voraz_por_convoy(m,N):
    resul = []
    total = 0
    centros = set()
    for cv,lista in enumerate(m):
        p,cn = min((p,c) for c,p in enumerate(lista) if c not in centros)
```

```
        total += p
        resul.append(cn+1)
        print(f"convoy {cv} elijo centro {cn} coste {p}")
        centros.add(cn)
    return total,tuple(resul)

print(voraz_por_convoy(m,5))
```

Se hace la traza comenzando con la variable mejor solución inicializada a $x = (3, 4, 2, 5, 1)$, con un tiempo total de $fx = 16$, la matriz de tiempos T y el vector $minT$ donde se ha precalculado el mínimo valor para cada convoy:

	centro					
	1	*2*	*3*	*4*	*5*	$minT$
convoy *1*	9	6	3	4	11	3
convoy *2*	8	5	5	1	5	1
convoy *3*	5	4	4	6	7	4
convoy *4*	10	3	2	2	2	2
convoy *5*	6	2	5	5	1	1

- $A_0 = \{\underline{(?)^{11}}\}$
- $A_1 = \{(2,?)^{11-3+6=14}, \underline{(3,?)^{11-3+3=11}}, (4,?)^{11-3+4=12}\}$
  El estado $(1,?)$ tiene una cota de $11 - 3 + 9 = 17$, no puede mejorar la solución que ya tenemos y no se guarda. Lo mismo el estado $(5,?)$ con cota de $11 - 3 + 11 = 18$.
- $A_2 = \{(2,?)^{14}, (4,?)^{12}, (3,2,?)^{11-1+5=15}, \underline{(3,4,?)^{11-1+1=11}}, (3,5,?)^{11-1+5=15}\}$
  El estado $(3,1,?)$ tiene una cota de $11 - 1 + 8 = 18$ y se descarta.
- $A_3 = \{(2,?)^{14}, (4,?)^{12}, (3,2,?)^{15}, (3,5,?)^{15}, (3,4,1,?)^{11-4+5=12}, \underline{(3,4,2,?)^{11-4+4=11}}, (3,4,5,?)^{11-4+7=14}\}$
- $A_4 = \{(2,?)^{14}, (4,?)^{12}, (3,2,?)^{15}, (3,5,?)^{15}, (3,4,1,?)^{12}, (3,4,5,?)^{14}, \underline{(3,4,2,5,?)^{11-2+2=11}}\}$
  El otro estado que resulta de la ramificación, $(3,4,2,1,?)$ tiene una cota de $11 - 2 + 10 = 19$ y se descarta.
- En la siguiente iteración se ramifica el estado seleccionado y se obtiene la solución $(3, 4, 2, 5, 1)$, que no mejora la solución almacenada en x (es de hecho, la misma solución). Nos queda en el conjunto de estados activos:
  $A_5 = \{(2,?)^{14}, (4,?)^{12}, (3,2,?)^{15}, (3,5,?)^{15}, \underline{(3,4,1,?)^{12}}, (3,4,5,?)^{14}\}$
- $A_6 = \{(2,?)^{14}, (4,?)^{12}, (3,2,?)^{15}, (3,5,?)^{15}, (3,4,5,?)^{14}, (3,4,1,2,?)^{12-2+3=13}, \underline{(3,4,1,5,?)^{12-2+2=12}}\}$
- Obtenemos otra solución: $(3, 4, 1, 5, 2)$, con un valor de función objetivo de 13. Nos queda en el conjunto de estados activos:
  $A_6 = \{(2,?)^{14}, \underline{(4,?)^{12}}, (3,2,?)^{15}, (3,5,?)^{15}, (3,4,5,?)^{14}, (3,4,1,2,?)^{12-2+3=13}\}$
- Todos los estados que resultan de ramificar $(4,?)$ tienen una cota superior a 13, por lo que no se guardan:
  $A_7 = \{(2,?)^{14}, (3,2,?)^{15}, (3,5,?)^{15}, (3,4,5,?)^{14}, (3,4,1,2,?)^{12-2+3=13}\}$
  En la siguiente iteración se seleccionaría como mejor de los que quedan, el estado $(3,4,1,2,?)$, que tiene una cota de 13. No puede mejorar la solución que ya tenemos y el algoritmo termina, devolviendo como solución óptima $(3, 4, 1, 5, 2)$, con un valor de función objetivo de 13.