UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Algorítmica (11593)
Recuperación del Segundo Parcial
21 de enero de 2019

etsinf
Escuela Técnica Superior de Ingeniería Informática

NOMBRE:

1 **4 puntos**

Un viajero supersticioso quiere visitar una serie de ciudades. Cuando llega a una determinada ciudad en un medio de transporte, sus manías le impiden repetir el mismo tipo de transporte para salir de dicha ciudad. Así, por ejemplo, si llega a Mallorca en barco, no puede salir en barco pero sí en avión. No hay problema en usar un medio de transporte previamente utilizado siempre que no sea el último utilizado.

Debes diseñar un algoritmo de *Ramificación y Poda* que calcule la forma más económica de visitar una vez cada ciudad bajo las restricciones del "viajero supersticioso" empezando y terminando en su ciudad. El algoritmo recibe como datos: un conjunto V con el número que identifica cada ciudad (de 1 a N, se supone que el viajero empieza y termina su itinerario en la ciudad número 1); otro conjunto E que indica los enlaces existentes entre dos ciudades, el tipo de transporte y el precio del mismo. Hay T tipos de transporte diferentes, por tanto, entre una ciudad i y una ciudad j puede haber más de una forma de viajar, cada una de ellas con un medio de transporte diferente y un precio diferente, o incluso, no estar conectadas por ese medio de transporte. La función $E(i, j, t)$ dará el precio del transporte entre la ciudad i y j con el transporte t. Será $+\infty$ si no existe esa conexión.

Para diseñar el algoritmo, contesta a las siguientes cuestiones:

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.
   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).
   3) 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 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: Es el problema del viajante de comercio pero con la restricción adicional de, en la ramificación solo se sigue por aquellas ciudades que estén comunicadas con un tipo de transporte distinto

al último. Si hay más de un tipo de transporte entre la última ciudad visitada y la conectada, se ramifican varios nodos, pues son diferentes.

2 **4 puntos**

Disponemos de M mochilas en las que queremos cargar un conjunto de objetos. Hay N objetos, cada uno con un peso w_j y un valor v_j, para $1 \leq j \leq N$. Cada mochila soporta una carga máxima C_i, para $1 \leq i \leq M$. Deseamos conocer la selección de objetos que hemos de cargar en cada mochila de modo que el beneficio sea máximo y ninguna mochila vea superada su capacidad de carga.

Realiza una traza de un algoritmo de Ramificación y Poda basada en *poda explícita* que calcule el máximo beneficio, para la siguiente instancia: $M = 2$ mochilas (de capacidades $C_1 = 10$ y $C_2 = 20$) y $N = 4$ objetos de pesos 6, 8, 10 y 11, y valores 9, 3, 5 y 4, respectivamente. 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 de cada uno de ellos.
b) Explica brevemente la cota optimista que vas a utilizar.
c) Para la traza sobre la instancia presentada, ten en cuenta lo siguiente: 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.
d) Si utilizas una solución inicial para adelantar la poda, indica qué algoritmo utilizas para obtenerla y cuál es su coste.

3 **2 puntos**

Encuentra, si existe, un algoritmo voraz para resolver el *problema del cambio de monedas con un número limitado de monedas y billetes de cada tipo*, pero suponiendo que lo que se deseamos es *maximizar* el número de monedas y billetes a utilizar para la cantidad a cambiar. La cantidad a cambiar es Q y se dispone de N tipos de monedas y billetes, con un número n_i de cada tipo. Se pide realizar una función Python que reciba la cantidad a cambiar Q y la lista de la cantidad de monedas y billetes de cada tipo n_i, y que devuelva el máximo número de monedas y billetes a utilizar para la cantidad a cambiar Q. Calcula su coste temporal. Demuestra, en caso de encontrar tal algoritmo, si devuelve la solución óptima o no.

Solución: Se puede usar una estrategia voraz de ir seleccionando todas las monedas y billetes más pequeños primero: se ordenan los N tipos por valor con un coste $O(N \log N)$ y se van seleccionando mientras no alcancemos la cantidad a cambiar, con un coste $O(N)$. Por tanto, el algoritmo tiene un coste total $O(N \log N)$.

Este algoritmo no devuelve la solución óptima (ni tan siquiera nos asegura una solución, aunque exista). Para demostrarlo, basta con encontrar un contraejemplo. Supongamos que tenemos 2 tipos de monedas y billetes de valor 1 y 2, y 2 monedas de valor 1 y 1 moneda de valor 2, y una cantidad a cambiar 3. El algoritmo voraz no obtendría ninguna solución (seleccionaría todas las monedas de valor 1, y ya no podría seleccionar ninguna más, y así no puede alcanzar la cantidad a cambiar), siendo que la solución óptima es seleccionar una moneda de valor 1 y una moneda de valor 2.