UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Algorítmica (11593)

Primer Parcial
8 de noviembre de 2016

etsinf
Escuela Técnica Superior de Ingeniería Informática

NOMBRE:

1 — 2.5 puntos

Estamos organizando una excursión en canoa. Debemos embarcar un total de N pasajeros y queremos utilizar el menor número de canoas. Todas las canoas son iguales y admiten una o dos plazas, pero el peso máximo que soportan es de K kilos. Para facilitarnos la tarea, tenemos a los pasajeros ordenados por su peso $w_1 \leq w_2 \leq \cdots \leq w_N \leq K$. Por ejemplo, si tenemos canoas que soportan hasta 150Kg y 4 pasajeros de pesos 52Kg, 74Kg, 91Kg y 120Kg, podemos utilizar 3 canoas.

Nos piden abordar este problema con una estrategia voraz. Para ello nos piden definir la función *canoas* que podría utilizarse como en esta llamada: `canoas([52,74,91,120],150)` que devolvería `3`.

¿Cuál es el coste temporal del algoritmo que has desarrollado?

2 — 2 puntos

Cómo llenar una caja con exactamente W Kgs. de peso (ni más ni menos) utilizando pesas de $w_1, \ldots, w_N$ Kgs. es un problema clásico conocido como *"Suma del subconjunto"*. Este problema se puede resolver con la llamada $F(N, W)$ a la siguiente ecuación recursiva:

$$F(i,c) = \begin{cases} c = 0 & \text{si } i = 0 \\ F(i-1,c) & \text{si } i > 0 \text{ y } w_i > c \\ F(i-1,c) \vee F(i-1,c-w_i) & \text{si } i > 0 \text{ y } w_i \leq c \end{cases}$$

Podemos obtener el siguiente algoritmo iterativo que nos dice si es posible o no llenar la caja.

```
def subset_sum(W,w):
  N = len(w)
  F = { (0,c):(c==0) for c in range(W+1) }
  for i in range(1,N+1):
    for c in range(W+1):
      F[i,c] = F[i-1,c]
      if not F[i,c] and c>=w[i-1]:
        F[i,c] = F[i-1,c-w[i-1]]
  return F[N,W]
```

Modifica la función anterior para que, en caso de haber solución, nos diga la lista de objetos que pueden llenar completamente la caja (en otro caso devolvería el valor `None`).

3 — 5.5 puntos

Deseamos asfaltar los K kilómetros de carretera que separan dos ciudades. Una empresa constructora nos ofrece un catálogo de N tipos de tramo con los que construir la carretera: cada uno tiene una longitud de L_i kilómetros y un coste de C_i euros, para $1 \leq i \leq N$. Como los tramos deben ser utilizados enteros (no se pueden dejar a medias), nos piden conseguir comprar tramos que sumen exactamente los K kilómetros minimizando el coste total. Se pide:

1. Especificar formalmente el conjunto de soluciones factibles X, la función objetivo a minimizar f y la solución óptima buscada $\hat{x}$.
2. Una ecuación recursiva que resuelva el problema y la llamada inicial.
3. El algoritmo iterativo asociado a la ecuación recursiva anterior para calcular *el menor coste* (no hace falta determinar la secuencia de tramos a utilizar).
4. El coste temporal y espacial del algoritmo iterativo.