Práctica 4

Programación dinámica y voracidad

ALGORÍTMICA Y COMPLEJIDAD
Grados en Ing. Informática y Matemáticas
Universidad de Cantabria

Camilo Palazuelos Calderón

Objetivos de la práctica

1. Resolver un problema mediante programación dinámica y voracidad
2. Comparar el coste espacial empírico de dos algoritmos
3. Discutir la aplicabilidad de diferentes estrategias algorítmicas

Duración de la práctica y fecha de entrega

- Las dos sesiones destinadas a la práctica son las de los días 16/17 y 23/24 de abril
- La práctica debe entregarse, a través de la plataforma Moodle, antes del 28 de abril a las 23:59

Qué debéis entregar

- Memoria con respuestas a las preguntas formuladas en este documento
- Código desarrollado y material suplementario que se considere oportuno

---

Cambio de monedas voraz…

Los algoritmos voraces son fáciles de plantear, pero rara vez son la solución a un problema computacional. Dependiendo del problema, es posible que desconozcamos la función de selección con que diseñar un algoritmo voraz que lo resuelva o que esta función ni siquiera exista. En este sentido, es paradigmático el problema de cambio de monedas, en que buscamos el número mínimo de monedas de ciertos valores que sumen una cantidad dada.

Una posible función de selección para el cambio de monedas es ir añadiendo a la solución la moneda más alta cuyo valor sea menor o igual que la cantidad aún por cambiar. Esta estrategia resuelve el problema para algunos sistemas monetarios, pero no para todos, como el antiguo inglés, en que los valores de las monedas eran 1 penique (p), 3 p, 6 p, 12 p (chelín), 24 p (florín) y 30 p (corona); por

ejemplo, si tuviéramos que cambiar 48 p mediante el algoritmo voraz, añadiríamos a la solución una corona (tras lo cual quedarían 48 – 30 = 18 p por cambiar), un chelín (y quedarían 18 – 12 = 6 p por cambiar) y una moneda de 6 p, es decir, lo haríamos con tres monedas, cuando la solución habría sido cambiar los 48 p con dos florines (24 + 24 = 48 p).

… y mediante programación dinámica

Formalmente, supongamos que queremos cambiar una cantidad de n unidades en un sistema monetario de m monedas de valores $V[1..m]$ ordenados de menor a mayor, donde $n \geq 0$, $m > 1$ y $V[1] = 1$. Para resolver el problema mediante programación dinámica, rellenamos un array $A[1..m, 0..n]$, donde cada **$A[i, j]$ representa el número mínimo de monedas de valor $V[i]$ o menor para cambiar una cantidad de j unidades** y, para todo i tal que $1 < i \leq m$, se calcula así:

```
if V[i] > j
    A[i, j] ← A[i – 1, j]
else
    A[i, j] ← min {A[i – 1, j], A[i, j – V[i]] + 1}
```

Una vez rellenado $A[1..m, 0..n]$, para construir la solución,

1. comenzamos en la última posición del array, es decir, $(i, j) \leftarrow (m, n)$;
2. **si $A[i, j] = A[i – 1, j]$,** la solución no puede incluir más monedas de valor $V[i]$, por lo que repetimos este paso con $A[i – 1, j]$;
3. **si $A[i, j] \neq A[i – 1, j]$,** la solución puede incluir una moneda más de valor $V[i]$, por lo que la añadimos al cambio y repetimos el paso 2 con $A[i, j – V[i]]$;
4. continuamos hasta que **$i = 1$,** en cuyo caso añadimos al cambio j monedas de $V[i]$ unidades y finalizamos.

Por ejemplo, el número mínimo de monedas para cambiar 8 unidades con monedas de valores 1, 4 y 6 es dos monedas (de 4 unidades cada una), ya que

$A[1..3, 0..8]$ =

0	1	2	3	4	5	6	7	8
0	1	2	3	1	2	3	4	2
0	1	2	3	1	2	1	2	2

y

1
4
6

= $V[1..3]$.

Preguntas en grupos de 2 o 3 alumnos

Pregunta 1 [1 PUNTO]. Diseñad el algoritmo voraz para el problema de cambio de monedas, cuya función de selección consiste en ir añadiendo a la solución la moneda más alta cuyo valor sea menor o igual que la cantidad aún por cambiar.

Pregunta 2 [1 PUNTO]. Diseñad, siguiendo la estrategia de programación dinámica de arriba abajo (*top-down*), el algoritmo que resuelve el problema de cambio de monedas.

Pregunta 3 [1 PUNTO]. Diseñad, siguiendo la estrategia de programación dinámica de abajo arriba (*bottom-up*), el algoritmo que resuelve el problema de cambio de monedas.

Pregunta 4 [1 PUNTO]. Calculad el coste espacial de vuestro diseño del algoritmo de la pregunta 3. ¿Qué relación hay entre este y el coste espacial de vuestro diseño del algoritmo de la pregunta 2?

Preguntas en grupos de 4 o 5 alumnos

Pregunta 5 [3 PUNTOS]. Implementad vuestros diseños de las preguntas 1, 2 y 3 en Python. Mostrad que las implementaciones son correctas proporcionando 4 o más ejemplos de entrada junto con sus salidas correspondientes.

Pregunta 6 [2 PUNTOS]. Dibujad una gráfica con los recursos de memoria utilizados por vuestras implementaciones de los algoritmos de las preguntas 2 y 3 tanto en el antiguo sistema monetario inglés como en el del euro en función de n (para, por ejemplo, n entre 10 y 1000 de 10 en 10). ¿Son coherentes las curvas obtenidas con el coste espacial calculado en la pregunta 4?

Pregunta 7 [1 PUNTO]. ¿Cuál de los tres algoritmos elegiríais para resolver el problema de cambio de monedas en cualquier sistema monetario con cantidades que cambiar muy elevadas? ¿Y en un sistema monetario como el del euro con cantidades que cambiar de hasta 1000 €? Justificad vuestras respuestas.