EntregaPracticasGrupoC.pdf

CharlsMars

Algorítmica

2º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

UNIVERSIDAD DE GRANADA
Departamento de Ciencias de la Computación e Inteligencia Artificial
/ UGR / decsai

Algorítmica
Grado en Ingeniería Informática

Prácticas: Ejercicio de entrega

Un profesor de algorítmica decide otorgar a sus estudiantes una calificación de P puntos (valor entero positivo entre 1 y 100) por la elaboración de ejercicios optativos. Hay un total de N ejercicios, y la valoración de cada ejercicio *i* es v_i (valor entero positivo entre 1 y 100). El objetivo es hallar cuál es el mínimo número de ejercicios que debe realizar el estudiante para obtener la **cantidad de puntos <u>exacta</u> P**. Por ejemplo, si se dispone de ejercicios con valoraciones v={1, 1, 2, 3, 4, 7, 9}, y la puntuación máxima es P=13, la solución sería realizar el ejercicio 5 (4 puntos) y el ejercicio 7 (9 puntos), con coste igual a 2.

Se pide:

1. (8 puntos) Diseñe un algoritmo de programación dinámica que resuelva el problema.
2. (2 puntos) Exponga un caso de ejemplo del problema y explique una traza del algoritmo sobre la instancia diseñada.

Criterios de evaluación:

1. Componentes de la técnica de diseño de algoritmos (4 puntos)
2. Algoritmo de cálculo del coste óptimo (2 puntos)
3. Algoritmo de recuperación de la solución (2 puntos)
4. Explicación del funcionamiento en un caso de ejemplo (hasta 2 puntos)

Ejercicio examen Grupo C

Programación Dinámica

Análisis

En el problema propuesto se asume que hay que devolver el mínimo número de ejercicios con mayor puntuación **P**. Esta puntuación se consigue a través de la realización de i ejercicios {1..n} cuyo valor es $\{v_1, v_2, \ldots v_n\}$. Adicionalmente, se pone la restricción de que no se sobrepase la cantidad de puntos P.

Diseño del algoritmo

- **Resolución por etapas:** El problema se puede resolver por etapas. En cada etapa se seleccionaría hacer o no el ejercicio de una puntuación dada. Supondremos que los ejercicios están ordenados de menor a mayor valor v_i para asegurar una solución óptima factible desde las etapas más tempranas.
- **Ecuación recurrente:**

  La solución depende de las etapas (puntuación de ejercicios a considerar hacer, que notaremos como *i*) y de la puntuación máxima que se puede obtener, que notaremos como *j*. Llamaremos al valor T(i,j) al mínimo número de ejercicios a hacer para obtener una puntuación *j*, supuesto que consideremos hacer o no el ejercicio 1, hacer o no el ejercicio 2, hacer o no el ejercicio 3, … hasta hacer o no el ejercicio i.

  En la etapa i consideramos hacer el ejercicio i. Caben dos posibles decisiones:

  - Realizar un ejercicio *i*: En tal caso, el mínimo número de ejercicios a devolver sería $1 + T(i, j-v_i)$; es decir, 1 (por el ejercicio que hemos hecho, y el mínimo número de ejercicios a realizar para una puntuación que resta el valor del ejercicio que hemos hecho.
  - No hacer el ejercicio *i*: En tal caso, el mínimo número de ejercicios a hacer será el mismo que cuando ni siquiera habíamos considera hacer el ejercicio *i*.

  Con estas dos decisiones posibles, y dado que el problema es de minimización, tendríamos que podríamos expresar la ecuación recurrente como:

  - $T(i,j) = \min\{T(i-1, j), 1 + T(i, j-v_i)\}$

  Los casos base para esta ecuación serían:

  - Sin puntuación restante, j = 0: T(i,0) = 0. Da igual el ejercicio a hacer que estemos considerando; si no quedan puntos por conseguir, el mínimo número de ejercicios por hacer es de 0.

        - Considerando hacer ejercicios de solo 1 punto. En tal caso, T(1,j) = j; es decir, para obtener j puntos restantes haciendo ejercicios de 1 punto, necesitamos hacer j ejercicios.

- **Valor objetivo:** Se desea conocer el valor T(n,P), el mínimo número de ejercicios a realizar para obtener P puntos, suponiendo que cada ejercicio tiene cualquier puntuación v_i desde {1..100}.
- **Verificación del cumplimiento de P.O.B:**

  Vamos a demostrar que cumple el P.O.B. mediante inducción:

  Tenemos los casos base T(i,0) = 0 y T(1,j) = j; los cuales son óptimos.

  Según va evolucionando el algoritmo y según vamos rellenando la memoria, tenemos que T(i,j) sería el valor mínimo de entre hacer un ejercicio o no hacerlo.

  $T(i,j) = \min\{T(i-1, j), 1 + T(i, j-v_i)\}$

  Cuando llegamos a la secuencia T(n,P), asumimos que es óptimo.

  Si llegamos a un caso general, T(i,j), estamos suponiendo que hacer ese conjunto de ejercicios es óptimo y si no fuera óptimo, existiría otra secuencia diferente de decisiones diferentes a las tomadas que fuesen todavía más óptimas que ellas; y esto no es posible.

  Sabemos que no es posible ya que en cada caso voy seleccionando siempre el mínimo entre hacer un ejercicio o no hacerlo. Como siempre se selecciona el mínimo, no puede existir otra subsecuencia que me devuelva una cantidad de ejercicios a realizar más pequeña.
- **Diseño de la memoria:**
  - Para resolver el problema, T(i,j) se representará como una matriz.
  - Esta matriz tendrá n filas, Cada fila i está asociada a una puntuación v.
  - Esta matriz tendrá n+1 columnas. Cada columna estará asociada a cada a la puntuación P que se puede obtener entre {0..P}.
  - Cada celda de la matriz T(i,j) contendrá el mínimo número de ejercicios a realizar para una puntuación P, suponiendo que estamos considerando realizar ejercicios con puntuaciones desde 1 hasta 100.
  - La memoria se rellenará de la siguiente forma:
    - En primer lugar, se rellenan las celdas correspondientes a los casos base.
    - En segundo lugar, se rellenarán las filas {2, 3, … , n}, en orden secuencial.
    - Cada fila se rellenará en orden creciente de columnas {1..100}.

Con este diseño, construimos el algoritmo de Programación Dinámica como sigue:

ALGORITMO T, V = Ejercicios(n, {v_1, v_2, … v_n})

T ← matriz de n filas indexadas {1..n} y n+1 columnas indexadas {0...n+1}

Para cada fila i en {1..n}, hacer:

T(i,0) = 0

Para cada columna j en {1..n}, hacer:

T(1,j) = j

Para cada fila i en {2..n}, hacer:

Para cada columna j en {1..n+1}, hacer:

T(i,j) = mín{T(i-1, j), 1 + T(i, j-v_i)}

V ← T(n,P)

Devolver T, V

- **Recuperación de la solución:** La solución se recuperará desde el valor objetivo de la matriz calculada previamente, recorriendo hacia atrás la recurrencia. El siguiente algoritmo devuelve la solución S (el conjunto de ejercicios a hacer).
    - Se debe llevar un contador de cada puntuación de los ejercicios que se han hecho {e_1, e_2, … , e_n}. Cada valor está inicialmente inicializado a 0 ya que no se ha hecho ningún ejercicio.
    - Cuando estamos evaluando T(i,j), se podría considerar realizar un ejercicio solo si se cumple que $e_i < m_i$. En caso contrario, solo se debe considerar la decisión T(i-1, j) para recorrer hacia atrás la solución hasta un caso base.
    - Si se decide hacer un ejercicio i, de debe actualizar el contador $e_i \leftarrow e_i + 1$
    - Si se consigue llegar al caso base T(i,0), se devuelven los ejercicios realizarlos.
    - Si se llega al caso base T(1, j), con j > 0, y no quedan ejercicios de puntuación 1, entonces se decide que no hay solución (en este caso el algoritmo no es óptimo) porque podría haber alguna decisión en algún T(i,j), distinta a la tomada, que sí devolviese el conjunto de ejercicios a realizar.

El algoritmo de recuperación de la solución es como sigue:

ALGORITMO S = RecuperaEjercicios(T(1..n, 1..n+1): Tabla resultante del algoritmo Ejercicios))

E ← {e_1, e_2, … , e_n}

Para cada i en {1..n}, hacer:

e_i ← 0

S ← ∅

i ← n, j ← n+1

Mientras j <> 0, hacer:

Si T(i,j) <> T(i, j-v_i) ó e_i >= m_i, hacer:

i ← i - 1

En otro caso, hacer:

j ← j – v_i

Añadir ejercicio con puntuación i a S

Si i < 1, hacer:

Devolver “No hay solución”

Devolver S

Caso de ejemplo del problema

Vamos a usar la información aportada en el enunciado.

m	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1_2	0	1	2	2	2	2	2	2	2	2	2	2	2	2
2_1	0	1	1	2	2	2	2	2	2	2	2	2	2	2
3_1	0	1	1	1	2	2	2	3	4	4	4	4	4	4
4_1	0	1	1	1	1	2	2	2	3	3	4	4	4	4
7_1	0	1	1	1	1	1	1	1	2	2	2	2	3	3
9_1	0	1	1	1	1	1	1	1	1	1	2	2	2	2

$$T(9,13) = \text{mín}\{\underbrace{T(i-1,j)}_{3},\ \underbrace{1+T(i,j-v_i)}_{1+T(9,4)=2}\}$$

$$T(9,4) = \text{mín}\{\underbrace{T(i-1,j)}_{1},\ \underbrace{1+T(i,j-v_i)}_{1+T(9,3)=2}\}$$

$$T(7,4) = \min\{\underbrace{T(i-1,j)}_{1},\ \underbrace{1+T(i,j-v_i)}_{1+T(7,3)=2}\}$$

$$T(4,4) = \text{mín}\{\underbrace{T(i-1,j)}_{2},\ \underbrace{1+T(i,j-v_i)}_{1+T(4,0)=1}\}$$

Como hemos llegado a un caso base, paramos la recurrencia y obtenemos las i que son los ejercicios realizados: 4, 9