UNIVERSIDAD DE GRANADA

/ UGR / decsai

Departamento de Ciencias de la Computación e Inteligencia Artificial

Asignatura: Algorítmica
Grado en Ingeniería Informática
Relación de problemas: Tema 4

1. Introducción

Para elaborar los ejercicios es recomendable utilizar la siguiente bibliografía:

- G. Brassard, P. Bratley, “Fundamentos de Algoritmia”, Prentice Hall, 1997
- J.L. Verdegay: Lecciones de Algorítmica. Editorial Técnica AVICAM (2017)

Antes de realizar los ejercicios prácticos, se recomienda al estudiante revisar y comprender las diapositivas estudiadas en clase y complementarlas con la información procedente de la bibliografía básica y recomendada en la guía docente de la asignatura.

En la relación de problemas se presentarán, para cada temática, un conjunto de preguntas sobre conceptos teóricos que el alumno debe dominar antes de abordar los ejercicios prácticos.

2. Algoritmos basados en programación dinámica: Conceptos

1. Describe cuáles son las características de los algoritmos basados en programación dinámica, las similitudes con algoritmos greedy y DyV, y sus diferencias.
2. ¿Cuáles son las ventajas e inconvenientes de la técnica de programación dinámica? Pon un problema de ejemplo, con soluciones, donde se ilustren tanto las ventajas como los inconvenientes.
3. ¿Qué tipos de problemas resuelve la técnica de programación dinámica y qué requisitos deben cumplirse para poder aplicarla?
4. ¿En qué consiste el problema del cambio de monedas? ¿Se puede resolver mediante P.D.? Ilustra las diferencias con la versión greedy para solucionar este problema, remarcando ventajas e inconvenientes de cada técnica.
5. Plantea el problema del cambio de monedas desde la perspectiva de P.D.. Da un algoritmo para resolverlo en pseudo-código.
6. Demuestra que se cumple el P.O.B. para el algoritmo del cambio de monedas elaborado en el ejercicio anterior.
7. Pon un ejemplo, inventado por ti, del problema del cambio de monedas. Indica cómo actúa el algoritmo que has diseñado en ejercicios anteriores paso a paso para resolver el problema.
8. ¿En qué consiste el problema de la Mochila 0/1? ¿Se puede resolver mediante P.D.? Ilustra las diferencias con la versión greedy para solucionar este problema, remarcando ventajas e inconvenientes de cada técnica.
9. Plantea el problema de la mochila 0/1 desde la perspectiva de P.D.. Da un algoritmo para resolverlo en pseudo-código.
10. Demuestra que se cumple el P.O.B. para el algoritmo de la mochila 0/1 elaborado en el ejercicio anterior.
11. Pon un ejemplo, inventado por ti, del problema de la mochila 0/1. Indica cómo actúa el algoritmo que has diseñado en ejercicios anteriores paso a paso para resolver el problema.
12. ¿En qué consiste el problema de caminos mínimos? ¿Se puede resolver mediante P.D.? Ilustra las diferencias con la versión greedy para solucionar este problema, remarcando ventajas e inconvenientes de cada técnica.

13. Plantea el problema de caminos mínimos desde la perspectiva de P.D.. Da un algoritmo para resolverlo en pseudo-código.
14. Demuestra que se cumple el P.O.B. para el algoritmo de caminos mínimos elaborado en el ejercicio anterior.
15. Pon un ejemplo, inventado por ti, del problema de caminos mínimos. Indica cómo actúa el algoritmo que has diseñado en ejercicios anteriores paso a paso para resolver el problema.
16. ¿En qué consiste el problema de la división óptima? Plantea este problema utilizando la técnica de P.D. y proporciona un pseudo-código que lo resuelva.
17. Demuestra que se cumple el P.O.B. para el algoritmo que has diseñado en el ejercicio anterior.
18. Pon un ejemplo, inventado por ti, del problema de la división óptima. Resuelve el problema indicando, paso a paso, cómo actúa el algoritmo en cada instrucción.

3. Algoritmos basados en programación dinámica: Ejercicios prácticos

19. Multiplicación encadenada de matrices. Se desea multiplicar varias matrices A*B*C*D*…*Z. Supodremos, para que el problema tenga solución, que el número de columnas de una matriz es siempre igual al número de filas de la matriz siguiente. La pregunta es: ¿Cómo hacemos las multiplicaciones para que el número de operaciones sea mínimo? Por ejemplo, para 4 matrices $A_{f1,c1}$, $B_{f2,c2}$, $C_{f3,c3}$, $C_{f4,c4}$, la multiplicación A*B*C*D se puede realizar de 5 formas diferentes: ((A*B)*C)*D, (A*B)*(C*D), (A*(B*C))*D, A*((B*C)*D), A*(B*(C*D)). Sabemos que si tenemos **n** matrices, existen un total de combinaciones para multiplicarlas igual al siguiente valor:

$$\frac{1}{n}\binom{2n-2}{n-1}$$

La mejor forma de multiplicar las **n** matrices es calculando el número de operaciones de multiplicación simples que cada multiplicación de dos matrices requiere y agrupando las multiplicaciones de matrices de modo que se minimice este número. El número de multiplicaciones simples para multiplicar dos matrices se calcula como sigue: Para dos matrices de tamaño $A_{f1,c1}$, $B_{f2,c2}$, cada componente de la matriz resultante tiene exactamente c_1 multiplicaciones. Como la matriz resultante tendrá f_1 filas y c_2 columnas, esto hace que multiplicar A*B requiera $c_1*f_1*c_2$ multiplicaciones.

Se pide: Plantea este problema mediante la técnica de P.D. y proporciona un algoritmo para resolverlo. Demuestra que se cumple el P.O.B.

20. Sea un entero **N** y un conjunto de números enteros $\mathbf{A=\{a_1, a_2, \ldots, a_k\}}$. Se desea saber si existe algún subconjunto de **A** cuya suma sea exactamente **N**. Resolver este problema mediante P.D., haciendo el planteamiento, dando el algoritmo que lo resuelve y demostrando que se cumple el P.O.B.. Finalmente, pon un ejemplo del funcionamiento del algoritmo y explícalo paso a paso.
21. Se dispone de **n** objetos, con peso igual a $\mathbf{p_1, p_2, \ldots, p_n}$. Se desea repartir estos objetos en dos montones de modo que el peso quede repartido lo más equitativamente posible. Resuelve este problema mediante P.D., haciendo un planteamiento del mismo, dando el pseudo-código del algoritmo que lo resuelve y demostrando que se cumple el P.O.B.. Finalmente, expón un ejemplo de funcionamiento del mismo, explicando paso a paso cada instrucción que realiza el algoritmo.

22. En una agencia de viajes, se comenta que la planificación para volar entre dos ciudades diferentes cualesquiera A y B puede ser necesario coger trasbordo. Cada trasbordo realizado significa malestar para el cliente, por lo que se plantean elaborar un método para minimizar el número de trasbordos entre cualquiera de las ciudades que se desee viajar. Se pide: Dar una representación de este problema utilizando grafos. Seguidamente, plantee el problema desde la técnica de P.D. y resuélvalo, dando el pseudo-código del algoritmo. Demuestre que se cumple el P.O.B.. Finalmente, exponga un ejemplo de tamaño n=4 ciudades y describa cómo actuaría el algoritmo diseñado paso a paso.
23. Dada una tabla con números enteros de tamaño **n*n**, se pretende resolver el problema de viajar desde la casilla (1,1) hasta la casilla (n,n) de modo que la suma de los valores de las casillas por las que se viaje sea mínimo. Suponiendo que en cada paso sólo podemos movernos desde una casilla (i,j) hasta la adyacente (i+1,j) o la adyacente (i,j+1), plantee este problema desde la técnica de P.D.. Dé el pseudocódigo del algoritmo que lo resuelve y demuestre que cumple el P.O.B. Por último, exponga un ejemplo pequeño con **n=4** explicando cada paso realizado por el algoritmo para resolver el problema.

24. Sea V un vector con n valores enteros distintos. Diseñe un algoritmo eficiente basado en programación dinámica para encontrar la secuencia creciente de máxima longitud en V. Por ejemplo, si el vector de entrada es (11, 17, 5, 8, 6, 4, 7, 12, 3), la secuencia creciente de máxima longitud es (5, 6, 7, 12).

25. Se tienen n unidades de un recurso que deben ser asignadas a r proyectos. Si se asignan j unidades al proyecto i, 0<=j<=n, 0<=i<=r, se obtiene un beneficio N(i,j). Diseñe un algoritmo que asigne recursos a los proyectos maximizando el beneficio total obtenido.

26. Tenemos un conjunto de n eslabones, (e_1, e_2, … e_n), cada uno con un peso ((p_1, p_2, … p_n)) que nos permiten construir una cadena. La cadena la construimos seleccionando eslabones y creando subcadenas, para posteriormente ir construyendo cadenas más grandes hasta tener la cadena final deseada. El coste de unir dos subcadenas equivale a la suma de los pesos de los eslabones de sus extremos. Por ejemplo, para unir las cadenas c1= (ei, ei+1, …, ek) con la cadena c2(ek, ek+1, …, em), el coste se calcula como COSTE(c1, c2)= $p_{ei}+p_{ek}+p_{ek+1}+p_{em}$. Como hay n eslabones, al final será necesario realizar n-1 uniones. Diseñe un algoritmo de programación dinámica que nos permita conocer las uniones óptimas que hay que ir haciendo de modo que se minimice el coste total.