UNIVERSIDAD DE GRANADA

/ UGR / decsai

Departamento de Ciencias de la Computación e Inteligencia Artificial

Asignatura: Algorítmica
Grado en Ingeniería Informática
Relación de problemas: Tema 2

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. Divide y Vencerás: Conceptos

1. ¿Cuál es la idea general de DyV? ¿Con qué fin se aplica esta técnica?
2. ¿Qué requisitos es necesario que se cumplan para que se pueda aplicar la técnica DyV?
3. Escribe el procedimiento general de la técnica DyV.
4. Busca en el libro o en las diapositivas de clase el algoritmo MergeSort. ¿Qué problema resuelve este algoritmo? Escribe a continuación el procedimiento general MergeSort. ¿Qué eficiencia tiene este método?
5. Lee el procedimiento para combinar las soluciones resultantes de MergeSort. Interioriza qué hace cada una de estas líneas y aprende este procedimiento de memoria. Escríbelo y comenta qué es lo que se persigue con cada bloque de código. ¿Cuál es la eficiencia de este método?
6. ¿Cuál es la idea general del algoritmo QuickSort? ¿Cuál es la limitación principal de QuickSort? ¿Qué eficiencia tiene este algoritmo, en el caso peor y en el caso mejor? Escribe la ecuación en recurrencias para el caso peor y para el caso mejor. ¿A qué se deben estas diferencias tan grandes? Piensa y razona: ¿Qué situación sería la que haría que una ejecución de QuickSort estuviese en el peor de los casos?
7. Busca en las diapositivas el método general QuickSort, memorízalo y escríbelo a continuación, comentando qué hace cada línea.
8. Busca en las diapositivas de clase el código del procedimiento para calcular el pivote en QuickSort. Léelo detenidamente, intentando averiguar qué hace cada línea. Luego memorízalo, y explica a continuación cuál es la idea del procedimiento que hace el método de pivotar. Escribe, por último, el código que has memorizado.
9. MergeSort tiene una eficiencia $\Theta(n \cdot \log(n))$, mientras que QuickSort está en $\Omega(n \cdot \log(n))$ y en $O(n^2)$. Sin embargo, en promedio, QuickSort es mucho más rápido que MergeSort. Piensa y razona: ¿A qué se debe, en términos de eficiencia, este comportamiento?
10. ¿En qué consiste el problema del umbral, en algoritmos DyV? ¿Qué tratamos de evitar con este umbral? ¿Cómo se calcula? Pon un ejemplo del cálculo del umbral para un algoritmo DyV. Si es posible, con código fuente o pseudocódigo incluido.

11. Busca en las diapositivas o en el libro información sobre el problema de selección (en algunos libros puede aparecer como *problema de la Mediana*). Describe en qué consiste el problema de selección.
12. ¿Cómo podemos resolver el problema de selección reutilizando el procedimiento pivotar del algoritmo QuickSort? Escribe cuál es la idea general de esta solución DyV para resolver el problema. ¿Qué orden de eficiencia tiene la solución en el mejor de los casos? ¿Y en el peor? ¿Cómo se comporta en promedio?
13. Memoriza la solución recursiva para el problema de selección que se ha mostrado en las diapositivas de clase. A continuación, escríbela en el hueco que hay para ello debajo de esta pregunta. Escribe también las ecuaciones en recurrencias del algoritmo en los casos mejor y peor. ¿A qué se debe esta diferencia? Piensa y razona bien la respuesta.
14. Busca y memoriza la versión no recursiva del algoritmo para resolver el problema de selección que se ha mostrado en las diapositivas de clase. Seguidamente, escríbela y comenta qué hace cada línea. Analiza el código y extrae los órdenes de eficiencia $O(\cdot)$ y $\Omega(\cdot)$ del procedimiento.
15. Busca en las diapositivas o en el libro información sobre el problema de multiplicación de enteros largos. Estudia bien la solución a este problema. ¿En qué consiste el problema de la multiplicación rápida de enteros? Explica la idea básica de cómo podríamos plantear este problema para poder resolverlo con la técnica DyV.
16. Explica cómo un algoritmo DyV para multiplicar enteros largos es capaz de mejorar el orden de eficiencia del método clásico atendiendo al número total de multiplicaciones realizadas.
17. Escribe el procedimiento general del algoritmo DyV para multiplicar enteros largos. Asumiendo que la operación de suma es O(n), ¿cuál sería la ecuación en recurrencias que modela el tiempo de ejecución del mismo? ¿Cuál sería el orden de eficiencia? Calcúlalo a partir de la ecuación en recurrencias.
18. Busca en las diapositivas o en el libro información sobre el problema de multiplicación de matrices. Estudia bien la solución a este problema. ¿En qué consiste el problema de multiplicación rápida de matrices? Razona cómo se implementaría el algoritmo clásico para multiplicar dos matrices de tamaño f*k y k*c y escribe su pseudocódigo. ¿Cuál sería la eficiencia de este algoritmo? Para mayor comodidad, puedes definir el tamaño del problema como n= max{f, k, c}, para reducir el número de parámetros del que depende el problema a sólo uno.
19. Explica la idea general de cómo se podría aplicar DyV para multiplicar dos matrices.
20. Strassen descubrió una forma de ahorrar una multiplicación de matrices para poder aplicar DyV, de modo que se mejorase la eficiencia con respecto al algoritmo clásico. ¿Cómo este método ahorra una multiplicación? ¿Cómo calcula la multiplicación de las matrices?
21. El método de Strassen tiene unos requisitos para poder aplicarse. ¿Cuáles son estos requisitos? En el caso de que dos matrices A y B que deseásemos multiplicar no cumpliesen estos requisitos, ¿significa que no podríamos aplicar Strassen para multiplicarlas? Explica la solución a esta cuestión y pon un ejemplo a mano para multiplicar dos matrices A y B de tamaño 3x3.
22. En las diapositivas de clase no se ha proporcionado el esquema general DyV para solucionar el problema de multiplicación rápida de matrices. ¿Serías capaz de escribir tú el pseudo-código de este procedimiento? Para ello, puedes basarte en algunos pseudo-códigos presentes en las diapositivas, como por ejemplo el de multiplicación de enteros largos.

3. Divide y Vencerás: Ejercicios prácticos

23. Sea el vector v=(11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1). Explicar textualmente, ayudándote de un diagrama similar al que hay en las diapositivas para explicar el funcionamiento de QuickSort, cómo se ejecutaría el algoritmo MergeSort para ordenar este vector.

24. Para diseñar el algoritmo MergeSort con la técnica DyV, en las diapositivas de clase se ha explicado que el vector inicial se subdivide en dos vectores del mismo tamaño, se ordenan ambos vectores y finalmente se combina su solución para obtener el vector final ordenado. Se pide diseñar un algoritmo DyV que, en lugar de dividir el vector en 2 partes, lo divida en 3. Además, se pide dar la implementación del método MergeSortTernario y el método de combinación de subsoluciones.
25. Hay que organizar un torneo con *n* participantes. Cada participante tiene que competir exactamente una vez como todos los posibles oponentes. Además, cada participante tiene que jugar exactamente un partido cada día (con la posible excepción, en el caso de haber jugadores impares, de un solo día en el cual no juega). Por simplicidad, se asume que el número de participantes del torneo es potencia de 2, $n=2^k$. Como restricción, se impone que el número de días para jugar el campeonato sea de *n-1*.
    - Proporcione un método básico que permita calcular la tabla de emparejamientos.
    - Diseñar un algoritmo DyV que calcule la tabla con los emparejamientos de jugadores por día, asumiendo como caso base n=2.

Por ejemplo, si hubiese 4 jugadores, un posible emparejamiento sería el siguiente:

	JUGADOR			
Día	**1**	**2**	**3**	**4**
1	2	1	4	3
2	3	4	1	2
3	4	3	2	1

Se pide: Indica claramente si el problema se puede resolver mediante DyV (en concreto, si se cumplen los requisitos para poder aplicar la técnica y porqué). Explicar cómo se diseñaría un algoritmo DyV para resolver este problema, asumiendo como caso base n=2. Dar el pseudocódigo que ilustre el diseño planteado.

26. La búsqueda de la posición de un elemento sobre un vector ordenado puede hacerse más eficiente utilizando la técnica DyV, con respecto a la clásica búsqueda secuencial de un elemento. Escriba el procedimiento clásico y calcule su eficiencia. Razone, explicando si el problema cumple con los requisitos, si se puede utilizar DyV para resolver este problema. Diseñe el algoritmo DyV que lo resuelva y proporcione su eficiencia. Escriba, por último, una implementación del algoritmo diseñado.

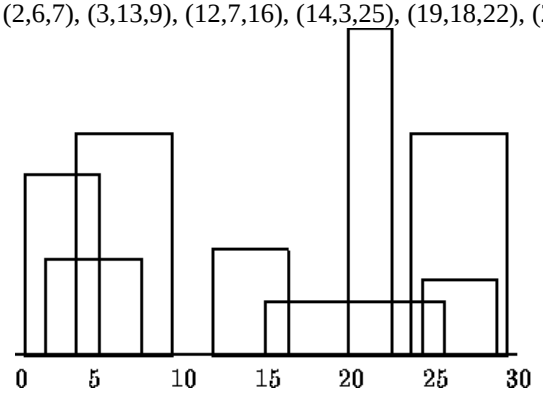
27. Nos encontramos en una convención con *n* participantes donde cada uno es de un partido político (en total hay *k* partidos políticos y $k<n$), aunque desconocemos cuál asistente es de qué partido. Si preguntamos directamente a cada persona cuál es su afiliación, éstas no nos responderán. Lo que sí sabemos es que cualquier par de personas que pertenecen al mismo partido, cuando se presenten entre sí, se saludarán amistosamente si pertenecen al mismo partido y se saludarán con enemistad si son de partidos distintos, y también que uno de los partidos políticos tiene mayoría de miembros (el número de asistentes a la convención que pertenecen a ese partido es mayor que para cualquier otro). Se desea confeccionar un algoritmo que pueda diseñar cómo han de saludarse los asistentes para identificar cualquier miembro del partido mayoritario en el menor número de pasos posible. Poner un ejemplo del correcto funcionamiento del algoritmo propuesto.

28. Supongamos un vector de enteros *v*, ordenado, de *n* componentes. Diseñar un algoritmo eficiente que permita encontrar (en el caso de existir) un número *i* tal que *v[i]=i* en el menor número de pasos posible. ¿Cuál es la eficiencia de este método?

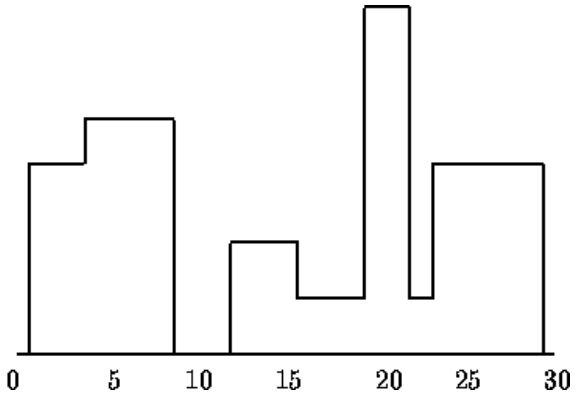
29. **La moneda falsa.** En un saco de monedas de oro se sabe que hay una única moneda falsa, que tiene un peso distinto de las demás. Disponemos de una balanza con dos platos para poder pesar conjuntos de monedas. Se pide: Hallar un método que permita descubrir la moneda falsa en tiempo mínimo $\log_3(2 \cdot n)$, cuando el número de monedas n>=3.

30. El algoritmo básico para elevar un número real *r* a un número entero *n*, r^n, consiste en multiplicar *n* veces el valor *r*. ¿Qué eficiencia tiene este método? Diseña, si es posible, un algoritmo DyV para mejorar la eficiencia de calcular r^n. Comenta si se cumplen los requisitos para aplicar DyV en este problema, qué casos podemos encontrar, y proporciona un pseudocódigo que solucione el problema, tanto si *n* es par como si *n* es impar. Escribe la ecuación en recurrencias de este método y, por último, calcula la eficiencia del mismo. Por último, proporciona la fórmula que nos permitiría calcular el umbral para aplicar el algoritmo básico en lugar de seguir dividiendo el problema con DyV.

31. **El problema de la línea del horizonte.** Un arquitecto necesita diseñar un programa para dibujar la línea del horizonte de una ciudad en un tiempo eficiente. Para simplificar el problema, supondremos que todos los edificios son rectangulares y la ciudad se puede ver en 2 dimensiones. Un edificio *i* se representa mediante la terna (I_i, A_i, D_i), donde I_i es la coordenada izquierda donde comienza la silueta del edificio, A_i, es la altura del mismo y D_i es la coordenada de la parte derecha, donde finaliza la silueta edificio. En el diagrama de la siguiente figura, los edificios se representarían como el vector ((1,11,5), (2,6,7), (3,13,9), (12,7,16), (14,3,25), (19,18,22), (23,13,29), (24,4,28)).



La línea del horizonte de la ciudad se representa mediante un vector bidimensional, donde las componentes v(i,1) contienen las coordenadas x de la línea del horizonte donde hay un cambio de altura, y las componentes v(i,2) la altura correspondiente. Para el ejemplo de la figura anterior, la línea del horizonte resultado sería el vector v= ((1, 11), (3, 13), (9, 0), (12, 7), (16, 3), (19, 18), (22, 3), (23, 13), (29, 0)). Este vector representa la línea del horizonte de la siguiente figura:



Sabiendo que puede haber un máximo de *n=5000* edificios y que la línea del horizonte estará comprendida entre los números enteros 0 y 50000, diseñe un algoritmo DyV que resuelva el problema en tiempo O(n·log(n)). Justifique si el problema cumple con los requisitos para poder aplicar esta técnica, explica cómo harías la división del problema en subproblemas más pequeños y comente cómo sería el método de combinación de soluciones para que este sea, como mucho, O(n). Por último, exponga un esquema del algoritmo resultante en pseudo-código, explicando cada uno de sus pasos.

32. Dos amigos juegan al siguiente juego: Uno piensa un número entero positivo en un rango [X, Y], y el otro debe adivinarlo únicamente preguntando si es menor o igual que otros números. Diseñe un algoritmo básico para hallar este número y otro basado en DyV. Calcule la eficiencia de ambos.

33. El problema del par más cercano consiste en encontrar dos puntos dentro de un conjunto de puntos cuya distancia sea menor que la que existe entre cualquier otro par de puntos del conjunto. Suponiendo que los puntos vienen dados por sus coordenadas (x, y), y que han sido ordenados en orden ascendente de la coordenada x, obtener un algoritmo DyV para solucionar este problema.

34. Diseñe un algoritmo "divide y vencerás" que permita calcular el k−ésimo menor elemento de un vector.

35. Dado un vector de n elementos, de los cuales algunos están duplicados, diseñe un algoritmo O(n log n) que permita eliminar todos los elementos duplicados.

36. Dado un vector ordenado de números enteros X, diseñe un algoritmo "divide y vencerás" que permita determinar si existe un índice i tal que X[i] = i.