DEFINICIONES ALGORÍTMICA

CAPÍTULO 1

Ciencia de la Computación: Estudio de los Algoritmos, incluyendo sus propiedades, su hardware, sus aspectos lingüísticos (sintácticos y semánticos) y sus aplicaciones.

Algoritmo: Un método de solución (un algoritmo) para resolver un problema es un proceso iterativo que genera una sucesión de puntos, conforme a un conjunto dado de instrucciones, y un criterio de parada. Características: finitud, especificidad, input, output, efectividad.

La diferencia entre **algoritmo** y **programa** es que cualquier sistema de cómputo real tiene un límite sobre el tamaño de los casos que puede manejar. Sin embargo, este límite no puede atribuirse al algoritmo que vayamos a usar.

Peor caso: Es útil cuando necesitamos una garantía total acerca de lo que durará la ejecución de un programa. El tiempo del peor caso se calcula a partir del caso que tarda más, es decir, el caso que realiza el máximo número de etapas.

Mejor caso: Es el caso que tarda menos tiempo, en el que se realizan el mínimo número de etapas.

Caso promedio: Es la media de los tiempos de todos los posibles casos del mismo tamaño (hay que conocer la distribución de probabilidad asociada a los casos). El caso promedio no existe como tal.

Eficiencia empírica: Depende del agente tecnológico usado.

Eficiencia teórica: No depende del agente tecnológico usado sino de cálculos matemáticos.

Eficiencia híbrida: La forma de la función que describe la eficiencia del algoritmo se determina teóricamente, y entonces cualquier parámetro numérico que se necesite se determina empíricamente sobre un programa y una máquina particulares.

Principio de Invarianza: Dos implementaciones diferentes de un mismo algoritmo no difieren en eficiencia más que, a lo sumo, en una constante multiplicativa.

Constante oculta: Constante en la que se acumulan todos los factores relativos a los aspectos tecnológicos.

Fuerza Bruta: Es un enfoque directo para resolver un problema, que se basa exclusivamente en el planteamiento del problema y en las definiciones y conceptos que

intervienen en el mismo. No recurre a algoritmos, métodos, procedimientos o técnicas que no vayan incorporadas en el problema en sí mismo. Ventajas: gran aplicabilidad y simplicidad, son algoritmos estándar para realizar tareas simples de cálculo. Desventajas: raramente son eficientes, muy lentos, no es constructivo ni creativo.

Algoritmos de Búsqueda Exhaustiva (Algoritmos Combinatorios) : Son algoritmos de FB para problemas que suponen la búsqueda de algún elemento con una propiedad especial, generalmente entre conjuntos generados por permutaciones, combinaciones o subconjuntos de un conjunto. Son frecuentes en recorridos sobre grafos, problemas de optimización… Suelen ser muy lentos pero en algunos casos son la única posible solución.

Clase P: Algoritmos con tiempos polinomiales (su tiempo de ejecución es una función polinomial).

Clase NP: Algoritmos para los que la única forma de que tengan un tiempo polinomial es realizando una etapa aleatoria (no determinista). Típicamente, en una primera etapa se elige una posible solución y en etapas posteriores se comprueba si esa solución es correcta.

Clase P $\subseteq$ Clase NP pero, ¿P = NP? Hoy por hoy es un problema abierto.

CAPÍTULO 2

Enfoque DyV: Se divide el problema en subproblemas más sencillos para resolver cada uno por separado (vencer) y posteriormente combinar las soluciones. Suele utilizarse recursivamente y a menudo proporciona soluciones eficientes. Es idóneo para resolver la mayoría de los problemas de ordenación.

Problema Torres de Hanoi: Mover n discos del tubo A al B usando el tubo C como intermedio temporal.

Umbral: Si comparamos dos implementaciones distintas, el umbral es el tamaño del problema para el que los tiempos de ejecución de ambas coinciden, y a partir del cual una implementación es mejor que la otra.

Ordenación por mezcla (MergeSort): Partimos la lista en dos partes A y B, que ordenamos recursivamente. Luego combinamos las dos listas ordenadas A y B (mezcla). Tiene eficiencia O(nlogn).

QuickSort (Algoritmo de Hoare): Es el algoritmo de ordenación más eficiente. Ordena un array A eligiendo un pivote v entre sus elementos. Organiza tres secciones: izquierda, pivote y derecha. Todos los elementos en la izquierda son menores que el pivote, mientras que todos los elementos en la derecha son mayores o iguales. Luego mediante recursividad ordena los elementos en la izquierda y en la derecha. Tiene eficiencia O(nlogn).

Problema del Skyline: Dadas las situaciones exactas (coordenadas) y las formas de n edificios rectangulares en una ciudad bidimensional, construir un algoritmo DyV que calcule eficientemente la línea del horizonte de esos edificios, eliminando las líneas ocultas.

CAPÍTULO 3

Enfoque Greedy: La idea básica consiste en seleccionar en cada momento lo mejor de entre un conjunto de candidatos, sin tener en cuenta lo ya hecho y si esa decisión será la mejor a largo plazo, hasta obtener una solución para el problema. Suele usarse para problemas de optimización. Características que debe reunir: conjunto de candidatos, lista de candidatos ya usados, criterio (función) solución, criterio de factibilidad, función de selección, y función objetivo. Suele proporcionar soluciones óptimas, pero no tiene por qué ser la mejor solución de todas.

Conjunto convexo: S es un conjunto convexo si dados dos puntos del conjunto S, todo el segmento lineal que los une está en el conjunto.

Grafo: Un grafo es un conjunto de vértices unidos por un conjunto de aristas.

Grafo dirigido: Cuando las aristas tienen origen y final (dirección): se llaman arcos.

Grafo ponderado: Cuando las aristas tienen pesos numéricos.

Grafo completo: Si cualquier par de vértices está unido por una arista.

Vértices adyacentes: Si existe una arista que los une.

Aristas adyacentes: Si comparten un vértice.

Grafo plano: Si puede pintarse en un plano o esfera sin que sus aristas se crucen.

Grado de un vértice: Número de aristas que pasan por ese vértice.

Camino: Sucesión de aristas distintas adyacentes (no se pueden repetir).

Circuito: Camino que comienza y termina en el mismo vértice.

Grafo conexo: Si cualesquiera dos vértices pueden unirse por un solo camino.

Camino Euleriano: Camino que pasa a través de cada arista del grafo una única vez.

Circuito Euleriano: Circuito que pasa a través de cada arista del grafo una única vez.

Teorema de Euler: Si todos los vértices de una grado son de grado impar, entonces no existen circuitos eulerianos. Si un grafo es conexo y todos sus vértices son de grado par, entonces existe al menos un circuito euleriano.

Circuito Hamiltoniano: Circuito que pasa a través de cada vértice una y solo una vez, y termina en el mismo vértice en el que comenzó.

Problema del Cartero Chino: Encontrar el circuito de longitud minimal que recorre cada arista de un grafo al menos una vez.

Problema del Viajante de Comercio: Hallar el circuito de longitud mínima que recorre todos los nodos de un grafo una y solo una vez, comenzando y terminando por el mismo vértice.

Árbol: Grafo que no tiene ciclos, es decir, grafo conexo y sin circuitos.

Árbol generador de un grafo: Subgrafo que es acíclico y conexo.

Árbol Generador Minimal (AGM): Árbol generador de un grafo ponderado en el que la suma de los pesos de sus arcos es mínima. El teorema de Cayley nos dice que si un árbol tiene n nodos, entonces tiene n^(n-2) árboles generadores.

Heurística: Procedimiento que, basado en la experiencia, proporcionan buenas soluciones a problemas concretos.

Heurística Greedy: "Es mejor satisfacer que optimizar". El tiempo efectivo es un factor clave.

Problema del Coloreo de un Grafo: Dado un grafo plano, determinar el mínimo número de colores que se necesitan para colorear todos sus vértices, y que no haya dos de ellos adyacentes pintados con el mismo color. Es un problema NP, por lo que se necesitan heurísticas para resolverlo. Con Greedy es O(n).

Teorema de Appel-Hanke: Un grafo plano requiere a lo sumo 4 colores para pintar todos sus nodos de modo que no haya vértices adyacentes con el mismo color.

Problema del Viajante de Comercio: Un viajante de comercio que reside en una ciudad tiene que trazar una ruta que, partiendo de su ciudad, visite todas las ciudades a las que tiene que ir una y sólo una vez, volviendo al origen y con un recorrido mínimo. Es un problema NP. Suponemos un grafo no dirigido y completo, y una matriz de distancias. Se quiere encontrar un Circuito Hamiltoniano Minimal. Heurísticas posibles: vecino más cercano (los candidatos son los nodos), las aristas son los candidatos (hacer lo mismo que con Kruskal pero garantizando que se forme un ciclo).

Problema de la Mochila Fraccional: Tenemos n objetos y una mochila con capacidad M. Cada objeto i tiene un peso wi y un precio pi. Si metemos en la mochila la fracción xi del objeto i, se genera un beneficio de valor pi*xi. El objetivo es rellenar la mochila de tal manera que se maximice el beneficio que produce el peso total de los objetos que se transportan, con la limitación de la capacidad de valor M (la suma de los pesos no puede superar M). Vamos eligiendo los elementos que tengan mayor precio/peso, así se obtiene la solución optimal.

Algoritmo Húngaro: Se usa para resolver el problema de la asignación de tareas.

CAPÍTULO 4

Programación Dinámica: Es un método de diseño de algoritmos que se puede usar cuando la solución del problema puede verse como el resultado de una sucesión de decisiones. Se suele aplicar a problemas de optimización. Se aplica en cuatro fases:

1. Naturaleza n-etápica del problema,
2. Verificación del POB,
3. Planteamiento de una recurrencia,
4. Cálculo de la solución (enfoque adelantado o atrasado).

Principio de Optimalidad de Bellman (POB): Cualquier subsecuencia de decisiones de una secuencia óptima de decisiones que resuelve un problema también debe ser óptima respecto al subproblema que se resuelve. Una política óptima solo puede estar formada por subpolíticas óptimas, cualquiera que sea el estado inicial y la decisión inicial elegidos

Enfoque adelantado (resp. atrasado): La solución del problema se construye desde el primer (resp. último) estado hasta el último (resp. primero).

PD	**DyV**	**Greedy**
Se progresa etapa por etapa con sub-problemas que se diferencian en tamaño 1 ud. Es aplicable cuando los subproblemas comparten subproblemas.	Divide el problema en subproblemas independientes, los resuelve recursivamente y combina sus soluciones para obtener la solución total.	Se progresa etapa por etapa con subproblemas que no tienen por qué coincidir en tamaño.
Se generan muchas subsucesiones de decisiones.	Los subproblemas no tienen ninguna característica común de tamaño ni se sabe su número a priori.	Solo se genera una sucesión de decisiones.
Hay un gran uso de recursos (memoria).		La complejidad en tiempo suele ser baja.
Como los problemas están encajados, no se repiten cálculos.	Como los problemas no tienen que ser independientes, pueden repetirse cálculos.	
En cada etapa se comparan los resultados obtenidos con los precedentes: siempre obtienen la solución optimal.	Se obtienen soluciones óptimas, pero no se garantiza que sea la optimal.	Como en cada etapa se selecciona lo mejor de lo posible sin tener en cuenta las decisiones precedentes, no hay garantía de obtener el óptimo.

Problema de la Mochila 0-1: La fracción xi de cada objeto i es 0 o 1, es decir, o se añade el objeto entero, o no se añade. Los algoritmos conocidos que lo resuelven son exponenciales, es un problema NP completo. Recurrencia para rellenar la matriz:

$$g\_j(c) = \text{máx}\{ g\_j\text{-}1(c) , g\_j\text{-}1(c - pj) + bj \}$$

Problema del Camino Mínimo (Algoritmo de Floyd): El problema consiste en determinar el camino de longitud mínima que una cualquier par de nodos. Recurrencia: D_k es la matriz después de la k-ésima iteración.

$$D\_k(i,j) = \text{mín}\{ D\_k\text{-}1(i,j) , D\_k\text{-}1(i,k) + D\_k\text{-}1(k,j) \} \qquad i \neq j$$

Inicialmente: D = L donde L es la matriz de distancias.

Problema del Viajante de Comercio: Definimos g(i,S) para cada i como la longitud del camino más corto desde el nodo i al nodo 1 que pasa exactamente una vez a través de cada nodo de S. Con esto, g(1, N-{1}) es la longitud de un circuito optimal. Si L es la matriz de distancias, la recurrencia para formar el grafo es:

$$g(i,S) = \min\{ L\_ij + g(j , S\text{-}\{j\}) : j \in S \}$$

$$g(i, \varnothing) = L\_i1$$

Un ejemplo de inversiones: Se quiere asignar n personas a t tareas (n ≠ t). Para cada tarea i definimos:

f_i(n) = lo que produce la tarea i con n personas

$$F\_i(n) = \text{máx}\{f\_i(y) + f\_i\text{-}1(n\text{-}y) : y \leq n \}$$

La solución optimal la deducimos de F_t(n), pues la vamos sacando de los valores máximos que hemos obtenido de F_i para ese F_t(n).

Problema de la multiplicación encadenada de matrices: Dadas n matrices A1, A2, … An, con Ai de dimensión p_(i-1) x p_i. Se quiere determinar el orden de multiplicación para minimizar el número de multiplicaciones escalares. Suponemos que la multiplicación de una matriz p x q por otra q x r requiere pqr multiplicaciones escalares. Recurrencia para rellenar la matriz en la que cada casilla m[i,k] contiene el costo óptimo de multiplicar A_i x … x A_k:

$$m[i,j] = \text{mín}\{ m[i,k] + m[k+1,j] + p\_(i\text{-}1)*p\_k*p\_j : i \leq k \leq j \}$$

$$m[i,i] = 0$$

Problema del Play Off: Supongamos que dos equipos A y B juegan una final en la que se quiere saber cuál será el primero en ganar n partidos (como mucho se podrán ganar

2*(n-1) partidos). Se supone que ambos partidos tienen 1/2 de probabilidad de ganar un partido. Definimos P(i,j) como la probabilidad de que A gane la final si A necesita i partidos para ganar y B necesita j:

$$P(i,j) = \begin{cases} 1 & i = 0, j > 0 \\ 0 & i > 0, j = 0 \\ [\, P(i-1, j) + P(i, j-1) \,] / 2 & i,j > 0 \end{cases}$$

Problema de la Subsecuencia de Mayor Longitud: Tenemos dos secuencias X e Y y queremos calcular la subsecuencia común a ambas que sea de mayor longitud. Recurrencia para rellenar la matriz:

$$l(i,j) = \begin{cases} 0 & i = j = 0 \\ l(i-1, j-1) + 1 & xi = yj \\ \text{máx}\,(l(i-1, j), l(i, j-1) & xi \neq yj \end{cases}$$

CAPÍTULO 5

Grafo Y/O: Representa un problema que puede resolverse mediante la reolución de varios subproblemas, o mediante la resolución de otro problema aislado.

Generación de un grafo en anchura: Cada vez que llego a un nodo, desarrollo todos sus posibles descendientes simultáneamente.

Generación de un grafo en profundidad: Cuando llego a un nodo desarrollo su primer hijo (la rama entera), luego su segundo hijo… Inconveniente: la profundidad de un árbol puede ser infinita. Esto se evita restringiendo la búsqueda solo hasta una profundidad determinada.

Árbol de juego, o árbol topológico: Colección finita de vértices, conectados por arcos que constituyen una figura conexa que no incluye curvas cerradas simples. Cada nodo representa una configuración, y el nodo raíz sería la configuración inicial C(1). El grado de cualquier nodo en el árbol es igual, a lo más, al número de movimientos legales distintos que haya (este número es finito). Los árboles de juego con útiles para determinar el siguiente movimiento que un jugador debe hacer.

Juego: Un juego n-personal en forma extensiva se define como:

- Un árbol topológico T con un vértice distinguido A que suele llamarse punto de partida de T (punto de comienzo del juego).
- Una función de pagos que asigna un n-vector a cada vértice terminal de T. Esto da la forma en que se van a obtener recompensas.
- Una partición de los vértices no terminales de T en n+1 conjuntos llamados jugadores (S0, S1, … Sn). De esta manera se dividen los movimientos en movimientos de azar (i=0) y movimientos personales (i=1..n) correspondientes a los n jugadores.
- Una distribución de probabilidad definida en cada vértice. Se define así un esquema de aleatorización en cada movimiento de azar.
- Para cada i=1,…,n, una subpartición de Si en j conjuntos llamados conjuntos de información, tales que dos vértices en el mismo conjunto de información tienen el mismo número de vértices seguidores inmediatos, y ningún vértice puede seguir a ningún otro en el mismo conjunto de información.
- Para cada conjunto de información existe un conjunto de índices y una aplicación uno a uno del conjunto de índices en el conjunto de los vértices inmediatos seguidores de cada vértice.

Juego finito: Juego para el que no existen sucesiones válidas de longitud infinita. Todos los posibles casos de un juego finito pueden representarse por el árbol del juego, en el que cada nodo i representa una configuración C(i). Las configuraciones terminales son aquellas que representan una situación de ganar, perder o empate, y están en los nodos hoja. Suelen etiquetarse por el nombre del jugador que gana.

Backtracking: Es una técnica de exploración de grafos primero en profundidad. Supongamos que tenemos que tomar una serie de decisiones entre una gran variedad de opciones donde no tenemos suficiente información como para saber cuál elegir y cada decisión nos lleva a un nuevo conjunto de decisiones, el Backtracking es un método de busca de una sucesión que nos convenga. Para poder aplicarlo, la solución debe ser expresable como una n-tupla. Su eficiencia en el peor caso es O(p(n)2^n).

Ventaja: Si a partir de un vector parcial que hemos construido se deduce que no se puede construir una solución, entonces pueden ignorarse por completo todas las subdecisiones de ese vector.

Diferencia con otras técnicas:

- En PD la solución se construye por etapas a partir del POB y los resultados se almacenan y no se tienen que recalcular, lo que no es posible con Backtracking.
- En Greedy se construye la solución buscada aprovechando la posibilidad de calcularla a trozos, pero con Backtracking la elección de un sucesor en una etapa no implica su elección definitiva.
- DyV se usa para algoritmos de búsqueda y ordenación. Esto no se puede con Backtracking.

Nodo vivo: Nodo que ha sido generado, pero para el que no se han generado todavía sus hijos.

E-nodo: Nodo vivo cuyos hijos están siendo generados en ese momento (nodo en curso).

Nodo muerto: Nodo generado que o no se va a expandir, o que todos sus hijos ya han sido generados.

Restricciones explícitas: Son las reglas que restringen cada componente del vector a tomar valores solo en un conjunto dado: son condiciones frontera. Dicho de otra manera, las describe el conjunto de soluciones factibles. Pueden depender o no del caso particular del problema a resolver. Vienen dadas por el propio planteamiento del problema.

Restricciones implícitas: Determinan cuál de las tuplas en ese espacio de solución satisface la función de criterio (función de acotación). Por tanto, son las restricciones que impone el camino que define la solución, es decir, los puntos que pueden ser solución. No vienen especificadas como tal pero se deducen de lo que conocemos del problema.

Problema de las Ocho Reinas: Clásico problema combinatorio que consiste en colocar ocho reinas en un tablero de ajedrez de modo que no haya dos que se ataquen, es decir, que estén en la misma fila, columna o diagonal.

Problema de la Suma de Subconjuntos: Dados n+1 números positivos: w_i ($1 \leq i \leq n$) y M, se trata de encontrar todos los subconjuntos de números w_i cuya suma valga M.

Branch&Bound: Técnica muy similar a Backtracking (se puede ver como una generalización) que realiza un recorrido sistemático de ese árbol, que no tiene por qué ser necesariamente en profundidad. Se suele aplicar para resolver problemas de optimización y para jugar juegos. Normalmente estos algoritmos tienen orden exponencial o peor en su peor caso.

La principal **novedad** es que habrá una estrategia de ramificación: se elige de algún modo qué nodo se va a desarrollar. Las técnicas de poda cobran mucha importancia y se realizan estimando en cada nodo cotas del beneficio que podemos llegar a tener.

Diferencia BB con Backtracking: En Backtracking tan pronto como se genera un nuevo hijo del nodo en curso (E-nodo), este hijo pasa a ser el nodo en curso, además de que el test de comprobación nos dice directamente si es fracaso o no. En BB se general todos los hijos del nodo en curso antes de que cualquier otro nodo vivo pase a ser el nuevo nodo en curso, y la cota nos sirve para podar el árbol y para saber el orden de ramificación. En consecuencia, en Backtracking los únicos nodos vivos son los que están en el camino de la raíz al nodo en curso, mientras que en BB puede haber más, que se almacenan en la lista de nodos vivos (LNV).

- Problema de la mochila 0/1, Problema del Viajante de Comercio