

Algorítmica

Hoja 5 de problemas – Vuelta atrás

1. Escribir el pseudocódigo de una versión recursiva del algoritmo que resuelve el *Problema de las 8 reinas*, estudiar su complejidad y realizar una estimación de su efectividad utilizando el método de Monte Carlo (al menos 3 estimaciones).
2. Escribir el pseudocódigo de una versión iterativa del algoritmo que resuelve el *Problema de las 8 reinas* que calcule sólo la primera solución. ¿Se pueden utilizar métodos de Monte Carlo para realizar una estimación ajustada del número de nodos explorados?
3. Escribir el pseudocódigo de una versión iterativa del algoritmo que resuelve el *Problema de la suma de subconjuntos*. Dar un ejemplo de ejecución con $W = (5, 10, 12, 13, 15, 18)$ y $M = 30$. Comparar el número de nodos explorados con la versión recursiva propuesta en clase.
4. Escribir el pseudocódigo de una versión recursiva del algoritmo que resuelve el *Problema de optimización de la suma de subconjuntos*. Este problema consiste en encontrar el subconjunto de un conjunto dado de números positivos W de modo que sumen una cantidad positiva M y *que tenga el menor cardinal posible*. Dar un ejemplo de ejecución con $W = (5, 10, 12, 13, 15, 18)$ y $M = 30$.
5. Dado el *Problema de optimización de la suma de subconjuntos*, como se enuncia en el problema anterior, mejorar la versión anterior de modo que haga uso de una *función de acotación* que elimina aquellos estados que sólo pueden dar lugar a una suma con mayor número de elementos que la mejor obtenida hasta el momento. ¿Cual es la diferencia entre esta función de acotación y las funciones de limitación? ¿Cuántos nodos se evita explorar si $W = (5, 10, 12, 13, 15, 18)$ y $M = 30$?
6. Escribir el pseudocódigo del algoritmo que resuelve el *Problema de decisión de colorabilidad*, consistente en decidir si es posible colorear un grafo con a lo sumo m colores distintos. Colorear un grafo consiste en asignar colores a todos sus vértices de modo que dos vértices adyacentes no tengan el mismo color. Encontrar un ejemplo en el cual no se alcance la decisión afirmativa en el primer camino explorado.
7. Escribir el pseudocódigo del algoritmo que resuelve el *Problema de optimización de colorabilidad*, consistente en encontrar el número mínimo de colores que permiten colorear un grafo. Se aconseja escribir el algoritmo de modo que se ponga como máximo de colores disponibles una cantidad adecuada (deducirla) y se exploren las posibilidades comparando con la mejor asignación en momento dado.
8. Escribir el pseudocódigo de una versión iterativa del algoritmo que resuelve el *Problema de los ciclos hamiltonianos*. Comparar el número de nodos explorados con la versión recursiva propuesta en clase.
9. Escribir el pseudocódigo de una versión recursiva del algoritmo que resuelve el *Problema del mínimo ciclo hamiltoniano*. Este problema consiste en encontrar el ciclo hamiltoniano de menor coste. ¿Se parece a algún problema ya estudiado? ¿Cuál es la solución más efectiva?

10. Escribir el pseudocódigo del algoritmo que resuelve el *Problema de la mochila entera* por vuelta atrás. Comparar la efectividad de este algoritmo con el algoritmo basado en programación dinámica.
11. El *Problema del rey* consiste en, dado un rey situado en la casilla (i,j) de un tablero de ajedrez, encontrar la secuencia de movimientos del mismo, si existe, que pasa por todas las casillas del tablero una sola vez. Escribir el pseudocódigo de un algoritmo de vuelta atrás que resuelva el *Problema del rey*. Estudiar la complejidad de dicho algoritmo. (Nota: el rey de ajedrez se puede mover una casilla en horizontal, vertical o diagonal.)
12. Sea un laberinto plano representado como una matriz cuadrada de 0's y 1's. Un 1 en una casilla representa la existencia de un bloque en esa posición, por la que no se puede pasar. El *Problema del laberinto* consiste en llegar a la casilla (n,n) desde la casilla $(1,1)$, que se suponen sin bloques, con movimientos horizontales y verticales. Escribir el pseudocódigo de un algoritmo de vuelta atrás que resuelva el *Problema del laberinto*. Estudiar la complejidad de dicho algoritmo.
13. Un Recorrido de Euler o euleriano es un recorrido de un grafo que pasa por todas las aristas del mismo una sola vez. Dado un grafo no dirigido, escribir el pseudocódigo de un algoritmo de vuelta atrás que decida si tal recorrido existe en el grafo dado.