Teoría de Algoritmos (TB024) – Curso Buchwald - Genender - Podberezski

Examen parcial: 2R de curso regular; 1R de curso anual; primera instancia curso asincrónico – 19/12/2024

1. Imaginá que estamos organizando un torneo de guardias en un castillo. El castillo tiene un suelo dividido en una cuadrícula de tamaño `n x m`, y cada celda puede estar ocupada por un guardia o estar vacía. Los guardias tienen la habilidad de vigilar todas las celdas adyacentes a su posición, incluidas las diagonales, es decir, pueden ver las celdas vecinas que están justo al lado, arriba, abajo, a la izquierda, a la derecha o en las esquinas.

   Se nos pide colocar la mayor cantidad posible de guardias en el castillo sin que ninguno pueda vigilar a otro. Esto significa que no podemos colocar dos guardias en celdas adyacentes, ya que estarían vigilándose mutuamente.

   Implementar un algoritmo **greedy** que permita colocar el mayor número posible de guardias en el castillo sin que se vigilen entre sí. Indicar y justificar la complejidad del algoritmo. Indicar por qué se trata, en efecto, de un algoritmo greedy. El algorimto, ¿es óptimo? si lo es, justificar brevemente, sino dar un contraejemplo.

2. Implementar un algoritmo que, utilizando **backtracking**, resuelva el problema del cambio (obtener la forma de dar cambio en la mínima cantidad de monedas) con una nueva restricción: no se tiene una cantidad indefinida de cada moneda, sino una cantidad específica (y esto hace que pueda no haber solución). Suponer que la función a invocar es `cambio(n, monedas, cantidad_x_monedas)`, donde `n` sea el valor a devolver en cambio, `monedas` sea una lista ordenada de los valores de las monedas, y `cantidad_x_monedas` un diccionario.

3. Podemos definir al problema de *K-ciclo* como: "Dado un grafo y un valor natural K, ¿existe un ciclo dentro del grafo de al menos K vértices?".

   Demostrar que el problema de *K-ciclo* es un problema NP-Completo. Para esto, recomendamos utilizar el problema de *ciclo Hamiltoniano*.

4. Contamos con una lista de n coordenadas satelitales (latitud-longitud) que conforman un polígono convexo, ordenadas en sentido antihorario. Queremos mostrar toda el área interior del polígono con el mayor tamaño posible en nuestra pantalla rectangular de la computadora. El programa que muestra el mapa acepta como parámetros 2 coordenadas para construir el rectángulo a mostrar: los correspondientes a los límites inferior izquierdo y superior derecho, tal que toda la imagen quede dentro de los límites. Implementar un algoritmo que resuelva el problema con complejidad $\mathcal{O}(\log n)$. Justificar adecuadamente la complejidad del algoritmo implementado.

5. Dado un Grafo dirigido, acíclico y pesado, y dos vértices s y t, implementar un algoritmo que, por **programación dinámica**, permita encontrar el camino de peso **máximo**. Indicar y justificar la complejidad del algoritmo implementado. Escribir y describir la ecuación de recurrencia de la solución, y la complejidad del algoritmo implementado. Implementar el algoritmo de reconstrucción de la solución, indicando su complejidad.

Teoría de Algoritmos (TB024) – Curso Buchwald - Genender - Podberezski

Examen parcial: 2R de curso regular; 1R de curso anual; primera instancia curso asincrónico – 19/12/2024

1. Imaginá que estamos organizando un torneo de guardias en un castillo. El castillo tiene un suelo dividido en una cuadrícula de tamaño `n x m`, y cada celda puede estar ocupada por un guardia o estar vacía. Los guardias tienen la habilidad de vigilar todas las celdas adyacentes a su posición, incluidas las diagonales, es decir, pueden ver las celdas vecinas que están justo al lado, arriba, abajo, a la izquierda, a la derecha o en las esquinas.

   Se nos pide colocar la mayor cantidad posible de guardias en el castillo sin que ninguno pueda vigilar a otro. Esto significa que no podemos colocar dos guardias en celdas adyacentes, ya que estarían vigilándose mutuamente.

   Implementar un algoritmo **greedy** que permita colocar el mayor número posible de guardias en el castillo sin que se vigilen entre sí. Indicar y justificar la complejidad del algoritmo. Indicar por qué se trata, en efecto, de un algoritmo greedy. El algorimto, ¿es óptimo? si lo es, justificar brevemente, sino dar un contraejemplo.

2. Implementar un algoritmo que, utilizando **backtracking**, resuelva el problema del cambio (obtener la forma de dar cambio en la mínima cantidad de monedas) con una nueva restricción: no se tiene una cantidad indefinida de cada moneda, sino una cantidad específica (y esto hace que pueda no haber solución). Suponer que la función a invocar es `cambio(n, monedas, cantidad_x_monedas)`, donde `n` sea el valor a devolver en cambio, `monedas` sea una lista ordenada de los valores de las monedas, y `cantidad_x_monedas` un diccionario.

3. Podemos definir al problema de *K-ciclo* como: "Dado un grafo y un valor natural K, ¿existe un ciclo dentro del grafo de al menos K vértices?".

   Demostrar que el problema de *K-ciclo* es un problema NP-Completo. Para esto, recomendamos utilizar el problema de *ciclo Hamiltoniano*.

4. Contamos con una lista de n coordenadas satelitales (latitud-longitud) que conforman un polígono convexo, ordenadas en sentido antihorario. Queremos mostrar toda el área interior del polígono con el mayor tamaño posible en nuestra pantalla rectangular de la computadora. El programa que muestra el mapa acepta como parámetros 2 coordenadas para construir el rectángulo a mostrar: los correspondientes a los límites inferior izquierdo y superior derecho, tal que toda la imagen quede dentro de los límites. Implementar un algoritmo que resuelva el problema con complejidad $\mathcal{O}(\log n)$. Justificar adecuadamente la complejidad del algoritmo implementado.

5. Dado un Grafo dirigido, acíclico y pesado, y dos vértices s y t, implementar un algoritmo que, por **programación dinámica**, permita encontrar el camino de peso **máximo**. Indicar y justificar la complejidad del algoritmo implementado. Escribir y describir la ecuación de recurrencia de la solución, y la complejidad del algoritmo implementado. Implementar el algoritmo de reconstrucción de la solución, indicando su complejidad.