Teoría de Algoritmos I (75.29, 95.06) – Curso Buchwald - Genender

2.^{do} recuperatorio de examen parcial – 04/03/2024

1. Un algoritmo sencillo para multiplicar matrices de $n \times n$ demora $\mathcal{O}(n^3)$. El algoritmo de Strassen (que utiliza División y Conquista) lo hace en $\mathcal{O}(n^{\log_2 7})$. La profesora Manterola quiere implementar un algoritmo de División y Conquista que sea aún más veloz, donde divida al problema en `A` subproblemas de tamaño de $\frac{n}{8}$, y que juntar las soluciones parciales sea $\mathcal{O}(n^2)$. ¿Cuál es el máximo `A` para que el orden del algoritmo sea menor que el del algoritmo de Strassen? Justificar.

2. El papá de Pepe le dió n monedas para repartir entre él y su hermanito. El padre puso las monedas formando una única fila. Cada moneda tiene con diferente valor v_i. El padre de Pepe le dice que primero debe elegir una para él, y que sólo puede elegir la primera o la última de la fila. Luego, debe elegir una para su hermano menor siguiendo la misma regla, luego otra para él, y así.

   Implementar un algoritmo que, utilizando programación dinámica, obtenga el valor máximo que pueda quedarse Pepe dadas estas condiciones (asumamos que usará parte de sus ganancias para comprarle un chocolate a su hermano). **Importante**: antes de escribir código, plantear y explicar la ecuación de recurrencia correspondiente.

3. Un set dominante (Dominating Set) de un grafo G es un subconjunto D de vértices de G, tal que todo vértice de G pertenece a D o es adyacente a un vértice en D. El problema de decisión del set dominante implica, dado un grafo G y un número k, determinar si existe un set dominante de a lo sumo tamaño k.

   Dominating Set es un problema NP-Completo. Ahora bien, queremos demostrar, nuevamente (pero de otra forma a la implementada probablemente en el tp), que *Hitting Set* es un problema NP-Completo. Demostrar que *Hitting Set* es un problema NP-Completo, utilizando *Dominating Set* para esto.

4. Implementar un algoritmo Greedy que busque aproximar la solución óptima al problema del mínimo Vertex Cover: dado un grafo, obtener el mínimo Vertex Cover del mismo. Indicar la complejidad del algoritmo implementado, dar un contraejemplo para el algoritmo implementado y justificar por qué el algoritmo implementado es un algoritmo greedy.

5. Indicar si las siguientes afirmaciones sobre Redes de Flujo son verdaderas o falsas, justificando detalladamente.

   a. Si aumentamos la capacidad de todas las aristas por una constante K, implicará que el flujo máximo aumente en $[K \times \min(\text{grado\_salida[fuente]}, \text{grado\_entrada[sumidero]})]$ unidades.
   b. En el caso del flujo máximo de la red, aumentarle la capacidad a una arista cuya capacidad no fue consumida no tienen ningún efecto sobre el flujo máximo.
   c. Eliminar una arista al azar del grafo puede no afectar el flujo máximo, pero si eliminamos una arista que es parte del corte mínimo, entonces obligatoriamente sí afectará al flujo máximo.

Teoría de Algoritmos I (75.29, 95.06) – Curso Buchwald - Genender

2.^{do} recuperatorio de examen parcial – 04/03/2024

1. Un algoritmo sencillo para multiplicar matrices de $n \times n$ demora $\mathcal{O}(n^3)$. El algoritmo de Strassen (que utiliza División y Conquista) lo hace en $\mathcal{O}(n^{\log_2 7})$. La profesora Manterola quiere implementar un algoritmo de División y Conquista que sea aún más veloz, donde divida al problema en `A` subproblemas de tamaño de $\frac{n}{8}$, y que juntar las soluciones parciales sea $\mathcal{O}(n^2)$. ¿Cuál es el máximo `A` para que el orden del algoritmo sea menor que el del algoritmo de Strassen? Justificar.

2. El papá de Pepe le dió n monedas para repartir entre él y su hermanito. El padre puso las monedas formando una única fila. Cada moneda tiene con diferente valor v_i. El padre de Pepe le dice que primero debe elegir una para él, y que sólo puede elegir la primera o la última de la fila. Luego, debe elegir una para su hermano menor siguiendo la misma regla, luego otra para él, y así.

   Implementar un algoritmo que, utilizando programación dinámica, obtenga el valor máximo que pueda quedarse Pepe dadas estas condiciones (asumamos que usará parte de sus ganancias para comprarle un chocolate a su hermano). **Importante**: antes de escribir código, plantear y explicar la ecuación de recurrencia correspondiente.

3. Un set dominante (Dominating Set) de un grafo G es un subconjunto D de vértices de G, tal que todo vértice de G pertenece a D o es adyacente a un vértice en D. El problema de decisión del set dominante implica, dado un grafo G y un número k, determinar si existe un set dominante de a lo sumo tamaño k.

   Dominating Set es un problema NP-Completo. Ahora bien, queremos demostrar, nuevamente (pero de otra forma a la implementada probablemente en el tp), que *Hitting Set* es un problema NP-Completo. Demostrar que *Hitting Set* es un problema NP-Completo, utilizando *Dominating Set* para esto.

4. Implementar un algoritmo Greedy que busque aproximar la solución óptima al problema del mínimo Vertex Cover: dado un grafo, obtener el mínimo Vertex Cover del mismo. Indicar la complejidad del algoritmo implementado, dar un contraejemplo para el algoritmo implementado y justificar por qué el algoritmo implementado es un algoritmo greedy.

5. Indicar si las siguientes afirmaciones sobre Redes de Flujo son verdaderas o falsas, justificando detalladamente.

   a. Si aumentamos la capacidad de todas las aristas por una constante K, implicará que el flujo máximo aumente en $[K \times \min(\text{grado\_salida[fuente]}, \text{grado\_entrada[sumidero]})]$ unidades.
   b. En el caso del flujo máximo de la red, aumentarle la capacidad a una arista cuya capacidad no fue consumida no tienen ningún efecto sobre el flujo máximo.
   c. Eliminar una arista al azar del grafo puede no afectar el flujo máximo, pero si eliminamos una arista que es parte del corte mínimo, entonces obligatoriamente sí afectará al flujo máximo.