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

Examen parcial: 2R de curso anual; 1R curso asincrónico – 10/02/2025

1. Dada una expresión representada por una cadena con aperturas y cierres de paréntesis, es sencillo implementar un algoritmo que, utilizando una pila, determine si la expresión se encuentra balanceada, o no (esto es un algoritmo sencillo de la materia anterior). Por ejemplo, la secuencia `()()`, se encuentra balanceada, así como `(())` también lo está, pero `((`, no lo está, ni `)()(`. Implementar un algoritmo greedy que reciba una cadena y determine el largo del **prefijo balanceado más largo** (es decir, el largo de la subsecuencia balanceada más larga que sí o sí comienza en el inicio de la cadena). Indicar y justificar la complejidad del algoritmo. Indicar por qué se trata, en efecto, de un algoritmo greedy. El algoritmo, ¿es óptimo? si lo es, justificar brevemente, sino dar un contraejemplo.

   Ejemplo: para `()())(())()((`, la respuesta es 4.

2. Resolver el problema anterior, pero esta vez para encontrar el largo de la **subsecuencia balanceada más larga** de una expresión (en este caso, no necesariamente comenzando en el inicio). Para esto, utilizar programación dinámica. La solución a este problema no dista mucho al planteo de la parte 2 del TP. Escribir y describir la ecuación de recurrencia de la solución, e indicar y justificar la complejidad del algoritmo implementado.

   Para el ejemplo anterior, la respuesta es 6 (comenzando en la posición 5).

3. Dado un arreglo de enteros ordenado, un elemento y un valor entero `k`, implementar una función que, usando división y conquista, encuentre los `k` valores del arreglo más cercanos al elemento en cuestión (que bien podría estar en el arreglo, o no). La complejidad de la función implementada debe ser menor a $\mathcal{O}(n)$, suponiendo que $k < n$. Justificar adecuadamente la complejidad del algoritmo implementado.

4. Realizar un modelo de programación lineal que resuelva el problema planteado en el ejercicio 3. OJO nos interesa que sean cercanos al elemento, y eso se ve con el valor absoluto de la diferencia, y el **operador módulo** no es un operador lineal. Si se incluye el **operador módulo** como parte del Modelo, el ejercicio estará Mal. Indicar la cantidad de restricciones definidas.

5. El problema de *Hamiltonian Completition* enuncia lo siguiente: Dado un Grafo G y un número K, ¿es posible hacer que el grafo G tenga un Ciclo Hamiltoniano agregándosele a lo sumo K aristas?

   Demostrar que el problema de *Hamiltonian Completition* es un problema NP-Completo. Valga la redundancia, recordar que el problema de *Ciclo Hamiltoniano* es un problema NP-Completo.

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

Examen parcial: 2R de curso anual; 1R curso asincrónico – 10/02/2025

1. Dada una expresión representada por una cadena con aperturas y cierres de paréntesis, es sencillo implementar un algoritmo que, utilizando una pila, determine si la expresión se encuentra balanceada, o no (esto es un algoritmo sencillo de la materia anterior). Por ejemplo, la secuencia `()()`, se encuentra balanceada, así como `(())` también lo está, pero `((`, no lo está, ni `)()(`. Implementar un algoritmo greedy que reciba una cadena y determine el largo del **prefijo balanceado más largo** (es decir, el largo de la subsecuencia balanceada más larga que sí o sí comienza en el inicio de la cadena). Indicar y justificar la complejidad del algoritmo. Indicar por qué se trata, en efecto, de un algoritmo greedy. El algoritmo, ¿es óptimo? si lo es, justificar brevemente, sino dar un contraejemplo.

   Ejemplo: para `()())(())()((`, la respuesta es 4.

2. Resolver el problema anterior, pero esta vez para encontrar el largo de la **subsecuencia balanceada más larga** de una expresión (en este caso, no necesariamente comenzando en el inicio). Para esto, utilizar programación dinámica. La solución a este problema no dista mucho al planteo de la parte 2 del TP. Escribir y describir la ecuación de recurrencia de la solución, e indicar y justificar la complejidad del algoritmo implementado.

   Para el ejemplo anterior, la respuesta es 6 (comenzando en la posición 5).

3. Dado un arreglo de enteros ordenado, un elemento y un valor entero `k`, implementar una función que, usando división y conquista, encuentre los `k` valores del arreglo más cercanos al elemento en cuestión (que bien podría estar en el arreglo, o no). La complejidad de la función implementada debe ser menor a $\mathcal{O}(n)$, suponiendo que $k < n$. Justificar adecuadamente la complejidad del algoritmo implementado.

4. Realizar un modelo de programación lineal que resuelva el problema planteado en el ejercicio 3. OJO nos interesa que sean cercanos al elemento, y eso se ve con el valor absoluto de la diferencia, y el **operador módulo** no es un operador lineal. Si se incluye el **operador módulo** como parte del Modelo, el ejercicio estará Mal. Indicar la cantidad de restricciones definidas.

5. El problema de *Hamiltonian Completition* enuncia lo siguiente: Dado un Grafo G y un número K, ¿es posible hacer que el grafo G tenga un Ciclo Hamiltoniano agregándosele a lo sumo K aristas?

   Demostrar que el problema de *Hamiltonian Completition* es un problema NP-Completo. Valga la redundancia, recordar que el problema de *Ciclo Hamiltoniano* es un problema NP-Completo.