uniandes

Ingeniería de Sistemas y Computación

Diseño y análisis de algoritmos

Profesor: Mateo Sanabria Ardila

Examen 3

Fecha de entrega: Diciembre 02 | 2024-20 | Nota máxima: 120

---

1. **(+70 pts)** Eres un desarrollador en Facebook, y tu equipo está planeando implementar una nueva funcionalidad para analizar patrones de conectividad entre usuarios. En particular, necesitas determinar si toda la red de amigos de un usuario, representada como un grafo $G_B = (V_B, E_B)$, puede encontrarse como un subconjunto dentro de la red de amigos de otro usuario más grande, representada como un grafo $G_A = (V_A, E_A)$.

   El problema consiste en decidir si es posible asignar cada persona en G_B a una persona única en G_A de tal manera que:

   - Cada persona en G_B se mapea a una persona en G_A.
   - Cada amistad entre dos personas en G_B corresponde a una amistad entre sus contrapartes asignadas en G_A.

   Mientras exploras este problema, te das cuenta de que es computacionalmente complicado , y tu jefe te solicita justificar por qué ocurre esto. Para esto debes:

   1. **(+5 pts)** Mostrar el lenguaje aceptado del problema de decisión asociado.
   2. **(+15 pts)** Mostrar que el problema pertenece a NP.
   3. **(+50 pts)** Mostrar que el problema es NP-completo.

   Para cualquier algoritmo que proponga debe utilizar pseudocódigo o lenguaje de su preferencia. Una explicación en palabras no sera validad como algoritmo. Cualquier justificación de complejidad debe ser justificada sobre los algoritmos propuestos. Especifique las entradas y salidas de los algoritmos propuestos. **Si cualquiera de los anteriores puntos es incumplido no se otorgará puntaje**.

2. (**+20 pst**) Supongamos que generalizamos el problema de la cobertura de conjuntos de modo que cada conjunto S_i en la familia F tiene un peso asociado w_i y el peso de una cobertura C es $\sum_{S_i \in C} w_i$. La idea es determinar una cobertura de peso mínimo.

   Muestre cómo generalizar (**pseudocódigo**) de manera natural *GREEDY-SET-COVER* para proporcionar una solución aproximada a cualquier instancia del problema de cobertura de conjuntos cargados.

3. **(+20 pts)** Basado en técnicas vistas en clase, implemente una función (**CÓDIGO: Python, Java...**) para imprimir todas las secuencias válidas (es decir, correctamente abiertas y eventualmente cerradas) de n pares de paréntesis. **Noy hay puntos intermedios, para tener los puntos completos debe proporcionar código que ejecute correctamente la tarea, salvo errores de sintaxis.**

```
# Muestra por consola todas las secuencias válidas de n pares de paréntesis
    printParenthesis(n: int) -> None
```