CC40A - Guía 8

24 de Julio de 1997

NP-Completitud

1. El uso de Máquinas de Turing (MT) para medir eficiencia acarrea el problema de que una MT no puede acceder directamente a una celda de memoria sino que debe avanzar linealmente. Es justificable entonces usarla como modelo de un computador real, que puede acceder a cada celda en tiempo constante? Explique.

2. Describa una MT determinística (MTD) que reconozca el lenguaje $(ab)^* = \{\epsilon, ab, abab, ababab, abababab...\}$.

3. Determinar si un número es primo o no es un problema para el cual no se ha encontrado solución polinomial. Sin embargo, es conocido que para saber si n es primo basta con ver si tiene algún divisor entre 2 y $\sqrt{n}$. Puede explicar qué es lo que pasa?

4. Cuando vimos arreglos de sufijos se dijo que se los podía construír en tiempo $O(n)$, mediante recolectar las hojas del árbol de sufijos. Cómo es esto posible, si el problema es equivalente al de ordenar?

5. Demuestre que *vertex cover* es NP-Completo. Dado un grafo G y un entero k se desea saber si existe un subconjunto S de k vértices de G tal que toda arista de G incide en algún vértice de S.

6. Demuestre que *set cover* es NP-Completo. Dada una familia de conjuntos S_i y un entero , se desea saber si se puede seleccionar k de ellos de modo que cubran todos los elementos de los conjuntos.

7. Sean P_1 y P_2 problemas en $\mathcal{P}$, N_1 y N_2 problemas en $\mathcal{NP}$, NC $\mathcal{NP}$-completo, y F no perteneciente a P. Para cada una de las sigu ientes preguntas, conteste "siempre", "nunca", o "depende". Si su respuesta es distinta según $\mathcal{P} = \mathcal{NP}$ o no, responda para ambos casos. Para un problema X, X' representa el problema complementario.

1. $P_1 \ \alpha \ P_2$	11. $P_1 \ \alpha \ N_1$
2. $N_1 \ \alpha \ P_1$	12. $F \ \alpha \ P_1$
3. $P_1 \ \alpha \ NC$	13. $NC \ \alpha \ P_1$
4. $N_1 \ \alpha \ NC$	14. $N_1 \ \alpha \ N_2$
5. $NC \in \mathcal{P}$	15. $P_1' \in \mathcal{P}$
6. 3SAT $\alpha \ N_1$	16. $N_1 \ \alpha$ 3SAT
7. si $X \ \alpha \ NC$, X es $\mathcal{NP}$-completo	17. $P_1 \in \mathcal{NP}$
8. 3SAT $\alpha \ NC$	18. $NC \ \alpha$ 3SAT
9. si $X \ \alpha \ N_1$, $X \in \mathcal{NP}$	19. si $N_1 \ \alpha \ X$, $X \in \mathcal{NP}$
10. si $X \ \alpha \ P_1$, $X \in \mathcal{P}$	20. P_1 es $\mathcal{NP}$-completo

8. Refute las siguientes demostraciones de que $P = NP$

   (a) Knapsack es NP-completo, incluso para enteros. Pero vimos un algoritmo de programación dinámica que lo resolvía en $O(nK)$, que es un polinomio. Por lo tanto, es polinomial y entonces todos los NP son polinomiales.

   (b) SAT es NP-completo si las fórmulas están en FNC (conjunción de disyunciones), pero es fácil de resolver si están en FND (disyunción de conjunciones). Hay reglas muy simples para convertir de una a la otra, por ejemplo $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$. Entonces, dado un problema de SAT en FNC, convirtámoslo a FND y resolvámoslo en tiempo polinomial.

   (c) Camino Hamiltoniano es NP-completo (saber si existe un camino simple de largo n en un grafo). Sin embargo, es fácil saber si existen caminos de largo $\leq k$, con Floyd. Entonces, apliquémoslo al grafo y veamos si todos los valores de la matriz resultante son $< n$, y lo solucionamos, ya que si todos los caminos son de largo $< n$ no puede haber un camino de largo n.