Universidad del Valle de Guatemala
Análisis y Diseño de Algoritmos - Sección 10
María Fernanda Estrada 14198
17/03/2020

Tarea 3
Complejidad

1. **Se define el problema LONGEST-PATH como un problema de decisión cuyas instancias son tuplas $\langle G,u,v,k\rangle$, donde G es un grafo no-dirigido, u y v son un par de vértices; y k es un número mayor o igual a cero. El problema pregunta si entre los vértices u,v existe un camino simple (sin ciclos) que pase por al menos k vértices. Luego se define el problema LONGEST-PATH-LENGTH como el problema con instancia $\langle G,u,v\rangle$, cuya solución es el número de vértices que hay en el camino simple más largo entre u,v. Demuestre que LONGEST-PATH-LENGTH puede ser resuelto en tiempo polinomial si y sólo si LONGEST-PATH pertenece a P.**
   Si LONGEST-PATH pertenece a P, el algoritmo se ejecutará como máximo n veces (tiempo polinomial al recorrer cada vértice). Si se utiliza este valor k para resolver LONGEST-PATH-LENGTH, nunca pasará de tiempo polinomial n.

2. **Considere el conjunto $GI=\{\langle G_1,G_2\rangle\}$ donde G_1 y G_2 son grafos isomorfos. Un par de grafos son isomorfos si existe un isomorfismo entre ellos, que es una función biyectiva $f{:}V(G_1)\rightarrow V(G_2)$ de los vértices de G_1 a los vértices de G_2 tal que $u,v \in G_1$ son adyacentes sí y sólo sí $f(u),f(v) \in G_2$ son adyacentes también (i.e., G_1 y G_2 tienen la misma cantidad de vértices, conectados de la misma forma). Demuestre que el problema de determinar si dos grafos son isomorfos es NP, proveyendo un algoritmo que permita verificar el isomorfismo en tiempo polinomial.**
   Primero, se listan las parejas no ordenadas de vértices del primer grafo G_1; se utiliza la fórmula de combinación para obtener estas parejas en una lista n. Este paso toma tiempo $O(n^2)$.
   Luego, para cada pareja de vértices, se debe revisar que sí tengan una imagen en el grafo G_2. Después de revisar que sí tenga imagen, se comprueba que haya conexión en ambos sentidos; este paso toma tiempo $O(n^2)$.
   Como ningún paso toma más que $O(n^2)$, se determina que la verificación toma tiempo polinomial.

3. **Sea X un problema nuevo e Y un problema demostrado NP-complete. ¿En qué sentido se debe hacer la reducción para demostrar que X es también NP-complete? ¿Por qué?**
   Se reduce Y a X. Como Y es NP-complete también es NP-hard, entonces los problemas NP pueden reducirse a Y. Si Y se reduce a X, también los problemas NP

pueden reducirse a X; esto implica que X ahora es NP-hard. Por último, como se mencionó para Y, si X es NP-hard también es NP-complete.

4. **Se puede determinar la satisfactibilidad de una fórmula booleana en tiempo polinomial si la fórmula está en forma normal disyuntiva. Investigue sobre forma normal disyuntiva (DNF) y forma normal conjuntiva (CNF) y responda: ¿por qué, pese a que determinar la satisfactibilidad de una fórmula DNF toma tiempo polinomial, SAT sigue siendo NP-complete? ¿Cómo es que determinar la satisfactibilidad de una fórmula en DNF toma tiempo polinomial?**
   La forma normal disyuntiva consiste de partes conjuntivas unidas por disyunciones. Entonces si una parte es satisfacible, toda la fórmula es satisfacible; esto toma tiempo polinomial porque solo se debe recorrer la parte en tiempo polinomial. SAT sigue siendo NP-complete porque transformar de CNF a DNF toma un tiempo no polinomial.

5. **El longest-simple-cycle problem es el problema de hallar el ciclo simple (visita cada uno de sus nodos una única vez) más largo que existe en un grafo dado. Demuestre que este problema es NP-complete.**
   El problema es NP porque se puede resolver en n iteraciones al mismo tiempo, se forma no-determinística en tiempo polinomial. También si nos dan un camino, se puede verificar en tiempo polinomial esa solución. Al revisar el ciclo Halmitoniano (que es NP-complete), se hace una reducción de LSC problem a ciclo Halmitoniano y se puede llegar a una solución. Entonces se concluye que, como el ciclo Halmitoniano es NP-complete, también lo debe ser LSC problem.

6. **Demuestre que el problema de las torres de Hanoi para tres torres y n discos es NP-hard pero no NP-complete.**
   Es NP-hard por la cantidad de movimientos que se deben hacer para trasladar los discos de la torre 1 a la torre 3. Sin embargo, no es NP-complete porque la verificación de que no quede un disco más pequeño abajo que uno más grande toma tiempo exponencial.