Escuela Politécnica Nacional

Nombre: Marco Marcillo

Tema: [Actividad extracurricular 3b] Complejidad computacional y el problema PvsNP

Repositorio GIT: https://github.com/Alejandro0122/MN_PERSONAL

Objetivo

- Conocer y entender sobre la complejidad computacional.
- Familiarizarse con el problema de P vs NP.

Indicaciones

a) Investigue sobre la notación big *O*. Esta medida es ¿para el peor escenario, mejor escenario, o el promedio?

Para comenzar la notacion Big *O* se utiliza para describir el limite superior asintotico del crecimiento de una funcion, por ende a esto se lo tomaria como una medicion del peor escenario que puede presentar una funcion o un algoritmo.

b) ¿Cuál es la diferencia con little o, y las otras funciones big Omega Ω, big Theta Θ?

- Little o

En resuemen comparandolo con Big *O* f(n) = O(g(n)), este crece a lo sumo tan rapido como g(n), es decir, que f(n) es *O* de g(n).

Mientras que el Little o, f(n)=o(g(n)), es que f(n) es *O* de g(n) pero no *O* de g(n).Este crece **ESTRICTAMENTE MAS LENTO** que g(n).

- Big Omega Ω

Esta notacion es la inversa de la notacion Big *O*, describe el limite inferior de la funcion f(n) respecto a g(n).

Esto **garantiza el limite al minimo de la funcion** f(n) respecto a g(n).

- Big Theta Θ

Es una notacion de un limite ajustado, si un algoritmo es Θ(f(n)), significa que su tiempo de ejecución crece al mismo ritmo que f(n).

Esta notación se usa cuando **los límites superior e inferior son iguales**, es decir, el mejor y el peor escenario tienen la misma complejidad.

c) Investigue sobre el problema P vs NP, describa ejemplos de algoritmos en cada caso.

Se centra en la eficiencia computacional de resolver problemas versus verificarlos.

Se sabe que:

P: Representa los problemas resolubles eficientemente (ordenamiento, búsqueda).

NP: Representa los problemas verificables eficientemente, pero no necesariamente resolubles

Conociendo eso el problema P vs NP, se enfoca en la pregunta de si todos los problemas verificacbles pueden resolverse eficientemente.

Ejemplos

Ejemplo 1 [P]

Búsqueda en un árbol binario balanceado

Algoritmo: Búsqueda binaria.

Complejidad: O(log n).

Este algoritmo es eficiente para resolver problemas de búsqueda en un árbol binario balanceado, pero no es eficiente para resolver problemas de ordenamiento.

Ejemplo 2 [P]

Camino más corto en un grafo (sin pesos negativos)

Algoritmo: Dijkstra.

Complejidad: O((V+E)logV).

Este algoritmo es eficiente para resolver problemas de búsqueda en un grafo, donde los pesos de las aristas son positivos y el tiempo de ejecución es proporcional al número de vértices y aristas del grafo.

Ejemplo 3 [NP]

Problema de la mochila (Knapsack Problem)

Descripción: Dados objetos con pesos y valores, ¿se puede llenar una mochila con valor total ≥V sin exceder el peso máximo?

Verificación: Si alguien te da un subconjunto de objetos, puedes sumar su valor en O(n).

No se conoce un algoritmo polinómico determinista.

Ejemplo 4 [NP]

Problema del clique (Clique Problem)

Descripción: Dado un grafo, ¿existe un subgrafo completo (todos conectados entre sí) de tamaño k?

Verificación: Si alguien te da un conjunto de k nodos, puedes comprobar en O(k2) si todos están conectados.

No se conoce un algoritmo polinómico determinista