UNIVERSIDAD AUTÓNOMA DE CHIHUAHUA
Facultad de Ingeniería

Ingeniería en Ciencias de la Computación

TEORÍA DE LA COMPUTACIÓN
Problemas P vs NP y qué algoritmos existen para cada uno

Trabajo de: ADRIAN A. GONZÁLEZ DOMÍNGUEZ *[359834]*
Asesor: MARIO ANDRES CUEVAS GUTIERREZ

5 de noviembre de 2024

Teoría de la Computación | Noviembre 2, 2024

Problemas P vs NP y qué algoritmos existen para cada uno

Adrian A. González Domínguez 359834

Un problema de clase P, o polinómica es aquel que puede resolverse en tiempo polinómico con respecto a la entrada recibida, esto quiere decir que la complejidad de tiempo es del orden de $O(n^k)$, n es el tamaño de la entrada y k una constante.

Un problema NP es todo aquel cuya solución es verificable en un tiempo polinómico, pero esto no implica que sea resolvible o no en un tiempo polinómico. En sus peores casos, un problema NP tendrá una complejidad temporal exponencial $O(k^n)$.
La clase NP hace referencia a que la complejidad de tiempo es polinómica no-determinista.

Las clases P, NP, NP-completo y NP-difícil:

- P: Problemas resolvibles en tiempo polinómico.
- NP: Problemas de solución verificable en tiempo polinómico.
- NP-completo: Subconjunto de NP. Son problemas de decisión. Para cualquier entrada la salida es sí o no. La respuesta afirmativa es demostrable (verificable), en tiempo $O(n^k)$. Solo un algoritmo de fuerza bruta puede encontrar una solución. Dado esto, son los problemas más difíciles cuyas soluciones pueden ser verificables. De poder encontrar rápidamente soluciones de un problema NP-completo, se lograría para todos los problemas NP.

- NP-difícil: un problema computacional H es llamado NP-difícil si, para todo problema L resolvible en tiempo polinomial no determinístico, existe una reducción de tiempo polinomial de tiempo de L a H. En otras palabras, encontrar un algoritmo de tiempo polinomal para un problema NP-difícil, resultaría en algoritmos de tiempo polinomial para todos los problemas en la clase NP.

Para los problemas P existen muchos algoritmos
- De ordenamiento: quicksort, mergesort, counting sort, etc.
- De camino mínimo: dykstra
- Busquedas: busqueda binaria, busqueda lineal
- Flujo máximo: ford-fulkerson.

En el lado de problemas de clase NP, tenemos muchísimos ejemplos:
- 1-planaridad
- Emparejamiento tridimensional
- Problema de la banda
- Dimensión bipartita
- Problema Knapsack
- Problema del viajante de comercio
- Problema de partición
- Batalla naval
- Fillomino
- Corral
- Free Cell
- Nonograms
- Solitario Mahjong
- Consistencia de buscaminas
- Solución óptima para el cubo de Rubik de N×N×N
- Sudoku
- Problemas relacionados al Tetris

Y algunos algoritmos que se utilizan para problemas NP, específicamente para optimizarlos:

- Backtracking. Indaga recursivamente en el árbol de soluciones, y deshace sus pasos al encontrar que una solución parcial no puede llevar a una solución global.

- Programación dinámica. Almacenan resultados en los estados intermedios a fin de evitar redundancia de cálculos. Se utiliza para problemas de optimización combinatoria.

- Algoritmos genéticos. Se inspiran en los procesos de selección natural. Se utilizan para problemas de optimización de búsquedas en espacios complejos.

Fuentes bibliográficas.

- Colaboradores Wikipedia. (2024, Noviembre 5). P (complexity). Wikipedia. https://en.wikipedia.org/wiki/P_(complexity)
- Colaboradores Wikipedia. (2024, septiembre 19). NP(complexity). Wikipedia. https://en.wikipedia.org/wiki/NP_(complexity)
- Colaboradores Wikipedia. (2024, febrero 24). NP-Hardness Wikipedia. https://en.wikipedia.org/wiki/NP-hardness
- Colaboradores de Wikipeda. (2024, agosto 8). NP-Completness. Wikipedia. https://en.wikipedia.org/wiki/NP-completeness.
- Colaboradores de Wikipedia. (2024, noviembre 2). List of NP-complete problems. Wikipedia. https://en.wikipedia.org/wiki/List_of_NP-complete_problems