U N I V E R S I D A D N A C I O N A L

A U T Ó N O M A D E M É X I C O

F A C U L T A D D E I N G E N I E R Í A

Algoritmos y Estructuras de Datos I (2016)

Ing. Adrian Ulises Mercado Martínez

Algoritmos tipo P y NP

Hernández Rojas Mara Alexandra

Número de cuenta 317206876

Ciudad de México a 07 de junio de 2020

¿Cuál es la diferencia entre problema y algoritmo?

Solo hay un problema pero es posible que existan varios algoritmos que resuelven el mismo problema y cada uno de ellos tenga un grado de complejidad diferente dependiendo de las estrategias que utilicen.

¿Qué es la complejidad del algoritmo?

Estudia como incrementa el costo (en memoria y tiempo) al resolver un problema a medida que aumenta el tamaño de la entrada.

¿Qué significa tiempo razonable?

Entiéndase por tiempo razonable a la relación de eficiencia entre el crecimiento del número de entradas del problema y el crecimiento del número de operaciones necesarias para resolverlo. (Similar al análisis de algoritmos que revisamos en clase). Desde un punto de vista práctico si es de tipo polinomial o menor es una solución plausible, sin embargo, si es exponencial, factorial o peor es un despilfarro de recursos.

¿Qué es un problema P?

Un problema P (Polinomial) es parte de un conjunto de problemas de los cuales podemos ENCONTRAR una respuesta dentro de un tiempo razonable. Y por lógica también podemos COMPROBAR si dicha respuesta que encontramos es correcta o no lo es en un tiempo razonable. En otras palabras, hay un algoritmo que encuentra la solución y otro que comprueba si es correcta ambos de tipo polinomial.

¿Qué es un problema NP?

Un problema NP (No-Polinomial) es parte de un conjunto de problemas de los cuales podemos COMPROBAR si una respuesta dada es la solución correcta a ese problema en un tiempo razonable mas no somos capaces de encontrar dicha solución en un tiempo razonable (no es polinomial). Es decir, hay un algoritmo que nos dice si la solución es correcta en tiempo polinomial, pero no uno que la encuentra en tiempo p.

¿Qué es un problema NP-Completo?

Son los problemas de tipo NP considerados los más difíciles, cabe mencionar que podemos convertir cualquier problema NP en un NP-Completo.

¿Qué es el problema del milenio?

El problema es conocido como P vs NP. Sabemos que todos los problemas de P forman parte del conjunto NP porque si encontramos la solución fácilmente podemos verificar se es correcta fácilmente, entonces se hace la siguiente pregunta ¿NP = P? parece obvio que no lo son por la forma en la que conseguimos las respuestas, pero eso significa que NP no pertenece a P o que no hemos encontrado el algoritmo adecuado para resolver NP en tiempo polinómico.

¿Cómo podría resolverse?

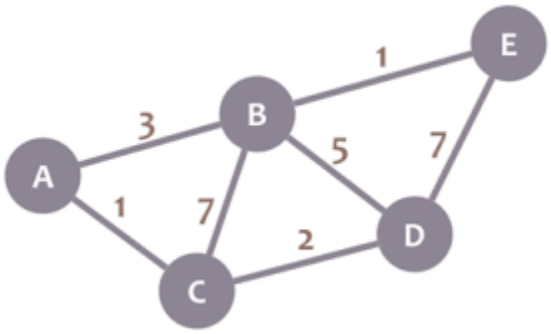
Una manera común de abordar el problema es tomar un problema de tipo NP-Complejo y tratar de resolverlo de una forma polinomial, si logras resolver uno resuelves todo y demuestras que P=NP.

¿Qué implican las posibles soluciones al problema?

La mayoría de las personas que estudian las ciencias de la computación sostienen que son conjuntos distintos, NP engloba a P, y la idea de que hay algoritmos más complicados que otros (qué no pueden ser fácilmente descifrados) ha sentado las bases para desarrollar todo el campo de la ciberseguridad y si pudiéramos hacer cualquier algoritmo sencillo una operación que nos habría llevado siglos podría llevarnos minutos, volveríamos inútil cualquier tipo de encriptación conocida.

Ejemplos de problemas P:

UNO Dijkstra



En el siguiente diagrama (grafo) podemos ver un esquema en el que los vértices representan ciudades y las aristas son los caminos que las conectan al que se le asocian ciertos tiempos de trayecto. Lo que nosotros queremos encontrar es el camino más rápido (el que menos pesos acumule) entre dos ciudades, digamos A y E.

El algoritmo de Dijkstra permite encontrar los caminos más rápidos desde un vértice pivote hacia todos los demás vértices en el grafo. Funciona así:

INICIO

Seleccionar un nodo pivote (C) y le asignamos una distancia 0

Asignar a los demás nodos (nodos no visitados) una distancia mínima “X” o “∞”

Repetir mientras existan nodos no visitados

Seleccionar como nodo actual al nodo no visitado con la menor distancia al pivote.

Suma la distancia actual de A con los pesos en las aristas que lo conectan con el origen si el resultado de la suma es menor que la distancia del vértice actual se establece como la nueva distancia actual de A.

Marcamos el nodo Actual como visitado.

Fin repetir.

FIN

Su **complejidad** está expresada en tiempo **O(n^2)** por eso es parte de **P.**

DOS Ordenamiento por selección:

El problema aquí consiste en ordenar una serie de elementos de menor a mayor.

INICIO

Colocamos un auxiliar1 en la primera posición del conjunto de elementos a ordenar

Repetir hasta llegar a la última posición de la lista:

    Repetir hasta llegar a la última posición de la lista:

    Colocamos un auxiliar2 en la posición siguiente a auxiliar1.

        Si auxiliar2 es menor que auxiliar1 entonces:

            Intercambiamos la posición de auxiliar1 con la de auxiliar2.

        Sino auxiliar2 avanza una posición.

    Fin repetir.

    Auxiliar1 avanza una posición.

Fin repetir

FIN.

Su **complejidad** está expresada en tiempo **O(n^2)** por eso es parte de **P.**

Ejemplos de problemas NP:

UNO: Problema de las “N” reinas.

Problema: Dado un tablero de ajedrez de NxN casillas, se deben colocar N reinas de forma que ninguna de ellas se pueda comer a cualquiera de las otras. La reina se mueve vertical, horizontal y diagonalmente siempre y cuando las casillas sean continuas sin límites dentro del tablero.

No vamos a poder colocarlas en el mismo renglón ni columna ni diagonal.

La complejidad del problema es de orden factorial **O(n!)-O(n+1)** que significa que a medida que aumente N gastaríamos una cantidad titánica de recursos para resolver el problema y no es viable desde un punto de vista práctico.

FORMA DE COMPROBARLO

*Inicio

* Seleccionamos una reina

* Verificamos que no haya otra reina en ese renglón si ya hay una reina en ese renglón esa NO es la solución (n-1)

* Verificamos que no haya otra reina en esa columna si ya hay una reina en esa columna esa NO es la solución (n-1)

* Verificamos que no haya otra reina en esas diagonales si ya hay una reina en esas diagonales esa NO es la solución (n-1)

* Repetimos para las demás reinas (n).

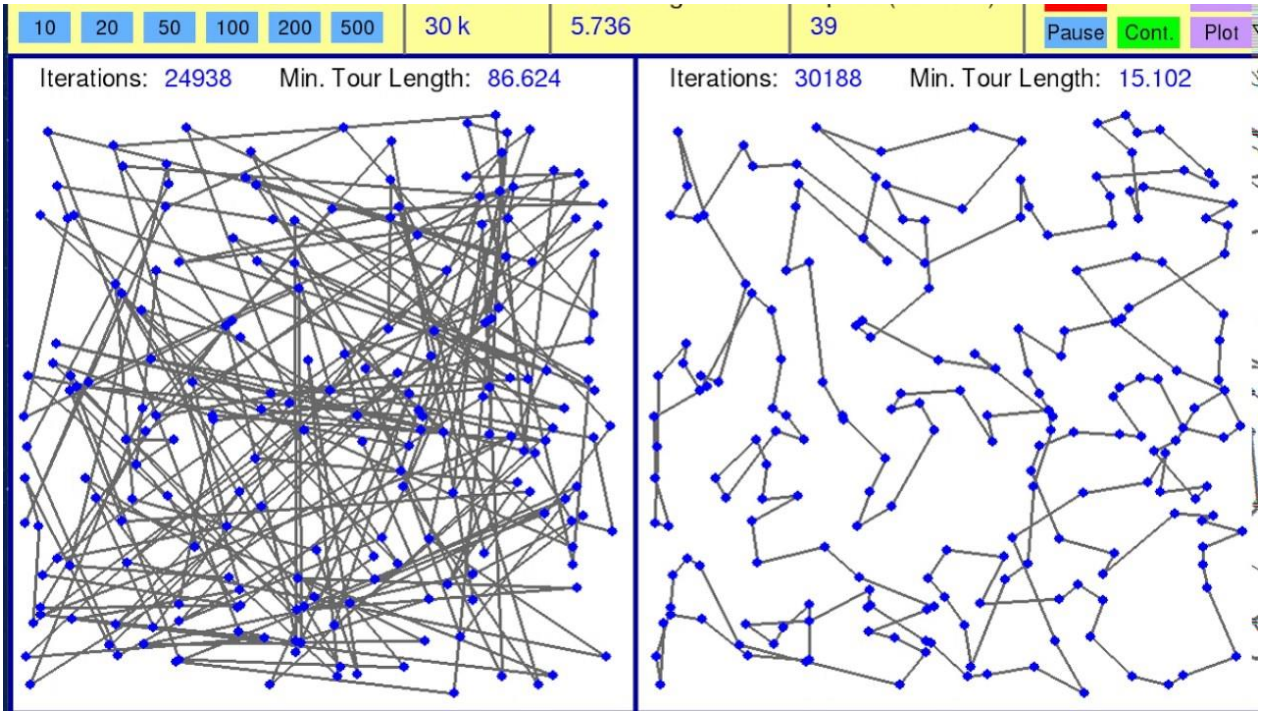
*Fin

Que nos deja una solución polinomial **O(n^4)** que significa que todavía es posible resolver el problema de manera práctica.

DOS:. Travelling Salesman (Agente Viajero/Vendedor Amblante)

Problema: Un vendedor tiene que viajar visitando todas las ciudades del país una sola vez y regresar antes de regresar a su casa (la misma ciudad de la que parte) si conocemos todos los caminos entre las ciudades y asumimos que recorre la misma distancia de ida y de regreso no importa la dirección en la que se desplace. Nuestro objetivo es hacer que el vendedor viaje la menor distancia posible sin regresar sobre sus pasos, (visitar una ciudad dos veces).

Este problema también se ayuda de la teoría de grafos. Y es de orden **O(n!)** utilizando fuerza bruta, utilizando programación dinámica **O($2^n * n^2$)**



FORMA DE COMPROBARLO (AYUDADOS DE UN DIAGRAMA)

Tenemos que seguir las aristas desde el punto de partida hasta regresar.

Referencias

- CodinGame. *Los caminos más cortos con el algoritmo de Dijkstra.* Recuperado de:*https://www.codingame.com/playgrounds/7656/los-caminos-mas-cortos-con-el-algoritmo-de-dijkstra (06/06/2020)*
- Guillen, P. *Clases P (Polynomial) y NP (Nondeterministic Polynomial)*. Recuperado de: https://pier.guillen.com.mx/algorithms/02-analisis/02.6-clases.htm (06/06/2020)
- Itaim, P. & Spading, A. *El problema de las n-reinas.(2005).* Recuperado de: https://www.cs.buap.mx/~zacarias/FZF/nreinas3.pdf (07/06/2020)
- LWH. *Ordenamiento por selección.* Recuperado de: http://lwh.free.fr/pages/algo/tri/tri_selection_es.html *(06/06/2020)*
- Ma, S. *Understanding the Travelling Salesman Problem (TSP)* Recuperado de: https://blog.routific.com/travelling-salesman-problem (07/06/2020)