Instituto Politécnico Nacional
Escuela Superior de Cómputo

Práctica 7: **Verificación en tiempo polinomial**

Análisis de algoritmos
3CV12

Reyes Rodríguez Enrique Abdiel
Chávez Hernández Juan Diego

abykings1@gmail.com
jdiegohdez0233@gmail.com

Diciembre 2021

Resumen: En el siguiente trabajo se presenta el análisis a el problema del Ciclo Hamiltoniano, observando su comportamiento con soluciones dadas, viendo como estas tienen relación con el problema P vs NP.
Palabras clave: Algoritmo, complejidad, Ciclo Hamiltoniano, P vs NP.

1. Introducción

Uno de los problemas que más intrigada tiene a la comunidad científica e informática, es el de P vs NP. Tiene el fin de determinar que problemas se pueden resolver en un tiempo razonable y cuales no, todo esto con el fin de ver si es viable su cómputo.
Este problema fue propuesto por el Instituto Clay de Matemáticas, en Cambridge, Massachusetts. Fue clasificado como uno de los problemas del milenio, de los cuales si es resuelto, tendría una gran importancia en el avance tecnológico, además de ofrecer un millón de dólares como incentivo.
Simplificando mucho, se entiende que la clase P, son aquellos problemas a los que le podemos encontrar solución en tiempo razonable; mientras que la clase NP contiene a los problemas que podemos verificar su solución en un tiempo razonable.
Podemos deducir que los problemas que están en P, están en NP; podemos encontrar una solución en un tiempo razonable y también comprobarla en un tiempo razonable.
El problema viene cuando preguntamos si ¿NP está en P?, es decir, que podamos verificar una respuesta, no implica que podamos encontrar solución. Es de gran interés analizar los NP, ya que muchos de sus problemas cuentan con un interés muy práctico.
Existe un problema en la teoría de grafos llamado Problema del Ciclo Hamiltoniano, cual tiene como objetivo de terminar si existe un ciclo hamiltoniano en un grafo. Un ciclo hamiltoniano, a grosso modo, es un camino, que recorre todos los vértices del grafo, siendo que termina en el vértice donde empezó; añadiendo que no se deben repetir vértices.
Este es un problema difícil de resolver, pero muy fácil de verificar, ya que solo hay que seguir el grafo por el camino de solución y ver si cumple con los requisitos. A este tipo de problemas se les llama NP, en el desarrollo de la práctica se profundizará un poco más de este tema.

2. Conceptos básicos

P

Se dedfine a P como el conjunto de problemas que se pueden resolver en tiempo polinómico, esto es, si existe un algoritmo que lo resuelve en tiempo $O(n^k)$ para alguna k, dónde n es el tamaño de la entrada del problema.[1]

NP

Se define a NP como el conjunto de problemas en los cuales, un certificado (posible solución) puede ser verificado en tiempo polinómico.[1]

Ciclo Hamiltoniano

Es un ciclo que pasa una y solo una vez por todos los vertices de un grafo dado.[2]

Grafo

Un grafo es el conjunto de vértoces y aristas conectadas entre sí, de las cuales pueden tener orientación.[3]

Pseudocódigo de la verificación de el ciclo hamiltoniano

```
function verificaHamilton(Grafo G,Certificado C)
   flag
    for i = 0, i < c.length , i++
        if i+1 >= c.length && C[i]==C[0] && i== G.vertex.length
            flag = true
        else if !( map.vertex[C[i]].contains(C(i+1)) )
            flag = false

    return flag
```

3. Experimentación y resultados

3.1. Verificación del certificado

En este caso, se tiene que recorrer todo el grafo, siguiendo la ruta del certificado. Analizando su comportamiento, se observa que crece linealmente respecto a el camino que se le otorga como certificado, ya que por cada elemento de este, en el peor de los casos, se recorrerá también por el numero de vértices que tiene el grafo, siendo ese valor constante, tendremos que la complejidad para este algoritmo de verificación es $T(n) \in O(n)$.

```
abdiel:practice7/ (main*) $ javac *.java && java Hamilton
Graph:
0: 1 2
1: 0 2
2: 0 1

[0, 1, 2]

No es un ciclo hamiltoniano
cycle length 3 steps: 10
Graph:
0: 1 2
1: 0 2
2: 0 1

[0, 1, 2, 0]

Es un ciclo hamiltoniano
cycle 4 steps: 13
Graph:
0: 1 3
1: 0 2 3
2: 1 3
3: 0 1 2

[0, 3, 2, 1]

No es un ciclo hamiltoniano
cycle length 4 steps: 12
Graph:
0: 1 3
1: 0 2 3
2: 1 3
3: 0 1 2

[0, 3, 2, 1, 0]

Es un ciclo hamiltoniano
cycle 5 steps: 15
```

Figura 1: Esta es la compilación y ejecución del código en consola, donde nos muestra el grafo, el certificado, los pasos que tuvo al ejecutarse y si es un ciclo hamiltoniano, o no

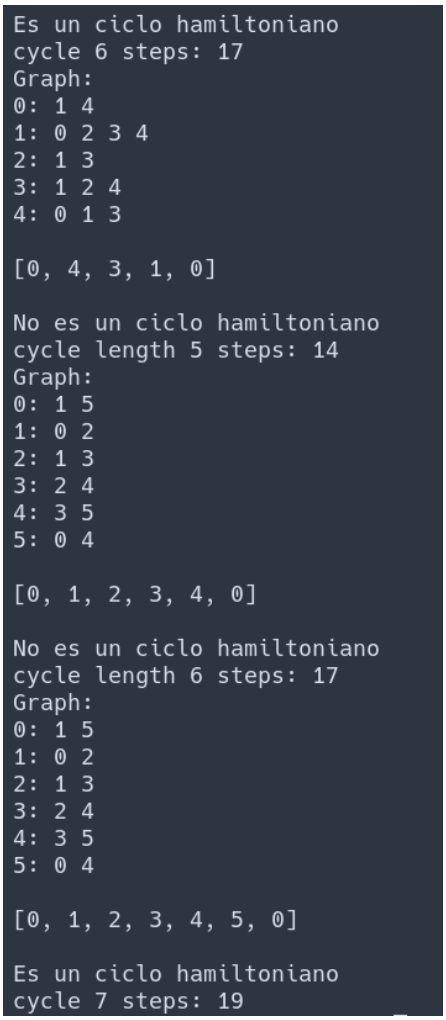


Figura 2: Esta es la compilación y ejecución del código en consola, donde nos muestra el grafo, el certificado, los pasos que tuvo al ejecutarse y si es un ciclo hamiltoniano, o no

n	c
3	10
4	13
4	12
5	15
6	17
5	14
6	17
7	19

Cuadro 1: Estos son los resultados de la ejecución del programa. En la primer columna, tenemos el tamaño del certificado con el cual se ejecutó, va del 3 al 7. En la segunda columna tenemos el valor del contador, que muestra el numero de instrucciones ejecutadas. Se observa como el crecimiento es constante

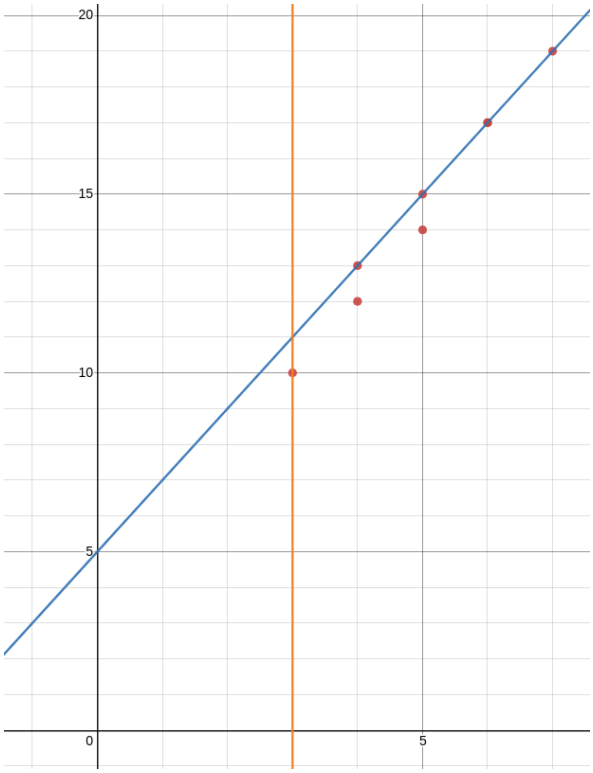
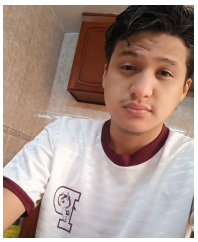


Figura 3: Esta es la gráfica que obtenemos de la tabla anterior. Los puntos son las coordenadas dadas por la tabla, la recta azul es la función $y = 2x + 5$ obtenida ajustando la ecuación de la recta y $n0$ está en el punto $(3, 0)$

4. Conclusiones

Enrique Abdiel Reyes Rodríguez:
Lo más dificil a mi parecer fué el manejo de grafos, pensamos en usar C, pero nuestra habilidad con apuntadores no fué muy buena, por lo que optamos en usar Java.Logramos hacer nuestro Grafo usando un HashMap para evitar iterar de más en nuestra implementación. Se me hizo interesante el analisis de la verificación para concluir que es un problema NP, ya que muchas soluciones que observamos eran no tratables, siendo de orden exponencial.

Juan Diego Chávez Hernández:
Como práctica final pienso que juntar el tema de P vs NP fué una buena idea, sobretodo para comprender que tan viables son los problemas para computarse. Fué de mucha ayuda el comprender bien el problema, para pensar que era NP al inicio, ya que no logramos hallar soluciones que fueran polinómicas.



5. Anexo

5.1. Análisis a priori del algoritmo utilizado

Por medio del análisis por bloques de código vamos analizando el orden de complejidad de adentro hacia afuera. Empezamos por dentro del for, y vemos que todas sus instrucciones son constantes, el segundo if puede generar confusión, pero debemos recordar que k tiende como máximo a el numero de vértices que tiene el grafo, siendo constante; por lo cual queda $O(n*k) = O(n)$. La instrucción del nivel superior también es constante, así que $O(n+1) = O(n)$.
Vemos que para este caso, se tiene que $T(n) \in O(n)$

Pseudocódigo de la verificación de el ciclo hamiltoniano

```
function verificaHamilton(Grafo g,Certificado c)
   flag --------------------------O(1)
    for i = 0, i < c.length , i++ ----------------------------------------O(n*k)--O(n)
        if i+1 >= c.length && C[i]==C[0] && i== G.vertex.length----------O(1)
            flag = true ------------------------------------------O(1)-|
        else if !( map.vertex[C[i]].contains(C(i+1)) ) -----------O(k)
            flag = false-------------------------------------O(1)-|

    return flag
```

Recordando que, la definición de NP menciona que para pertenecer a este, el certificado (en este caso, el camino que seguirá el grafo) puede ser verificado en tiempo polinómico. Nuestra verificación nos dió como resultado una complejidad de $O(n)$ por lo que concluimos que el Problema del Ciclo Hamiltoniano es **NP**.

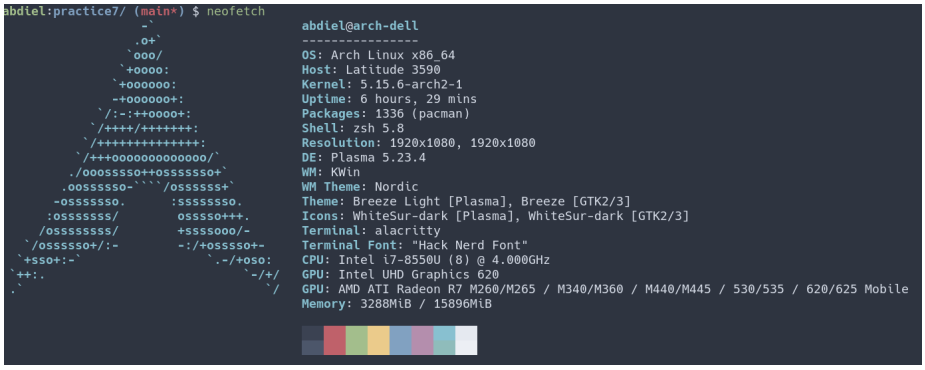


Figura 4: Estas son las especificaciones del equipo donde fueron ejecutados los experimentos

Referencias

[1] Lasker, R. (2018, 21 enero). P NP - Rashid Lasker. Medium. Recuperado 17 de diciembre de 2021, de https://medium.com/@rashidlasker/p-np-an-introduction-to-computer-sciences-most-interesting-problem-47869a34601a

[2] Finding Hamiltonian Cycle in Polynomial Time. (s. f.). Science Alert. Recuperado 17 de diciembre de 2021, de https://scialert.net/fulltext/?doi=itj.2006.851.859

[3] GRAFOS. (s. f.). CCIA. Recuperado 17 de diciembre de 2021, de https://ccia.ugr.es/