../../IPNlogo.jpg

Escuela Superior de Cómputo.
Instituto Politécnico Nacional, México.

../../logoesc

Práctica 10: Verificación en Tiempo Polinomial.

Blancas Pérez Bryan Israel
orionmunecaycanica@gmail.com

Resumen: Implementar un algoritmo que verifique un grafo C , para determinar si es o no un ciclo Hamiltoniano de un grafo G, además de mostrar que este algoritmo, se ejecuta en tiempo polinomial.

Palabras Clave: Ciclo Hamiltoniano, Grafo Hamiltoniano, Verificación en Tiempo Polinomial, Complejidad Computacional, Problemas NP y NP-completos.

1. Introducción

Verificación en Tiempo Polinomal.
En la Teoría de la Complejidad Computacional, se sabe que para resolver problemas pertenecientes a la clase de complejidad NP, se requiere de una Máquina de Turing no Determinista, ya que con la tecnología actual, el tiempo necesario para encontrar la solución de estos problemas es super-polinomal. Por el contrario, para los problemas que pertenecen a la clase de complejidad P, es posible hallarles una solución en tiempo polinomial. Sin embargo, una propiedad de los problemas NP, es que se puede verificar si una propuesta de solución lo es realmente, implementando un algoritmo que se ejecute en tiempo polinimial. Este artículo, pretende mostrar lo anterior dicho, verificando una solución para el problema del ciclo Hamiltoniano, problema que se sabe es NP-completo.

2. Conceptos Básicos

Problema del Ciclo Hamiltoniano.

En teoría de grafos, el problema del ciclo Hamiltoniano trata de determinar si un ciclo Hamiltoniano existe o no en un determinado grafo. Se sabe que este problema pertenece a la clase NP-completo, lo cual nos dice que aún no se encuentra un algoritmo polinomial que halle la solución.[1]
En esta práctica, se pretende demostrar una propiedad de los problemas de la clase NP-completos, la cual es verificar si una propuesta de solución dada, lo es o no para el problema del ciclo Hamiltoniano.

Ciclo Hamiltoniano.

Un ciclo Hamiltoniano, es un camino de un grafo (sucesión de aristas adyacentes), que visita todos los vértices del grafo una y sólo una vez, si además el último vértice visitado es adyacente al primero.[2]
En la figura 1, se muestra un ejemplo de un ciclo Hamiltoniano en un grafo arbitrario.

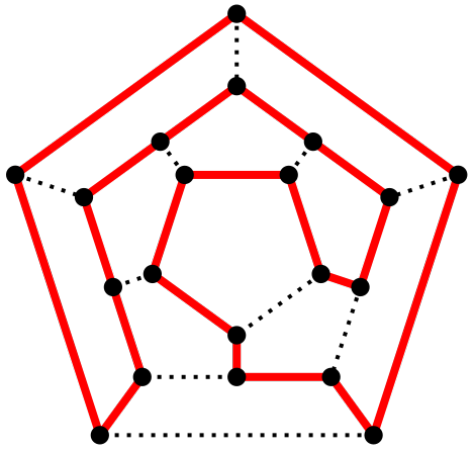

Figura 1: Ciclo Hamiltoniano.

3. Experimentación y Resultados

Ejercicio 1.
Implementar un algoritmo para verificar el problema de los ciclos Hamiltonianos, es decir, implementar un algoritmo **Verificacion_Hamilton(Grafo G, certificado C)** que verifique en tiempo polinomial (muestre esto mediante gráficas y analíticamente) que el certificado C es o no un ciclo Hamiltoniano del grafo G.

Pseudocódigo del algoritmo:

```
1  Verificacion_Hamilton(Grafo G, Certificado C)
2
3    #Verificar que los nodos de C, existan en G
4
5    for node in C:
6      if node.existIn(G)==0 :
7        print "Nodo inexistente"
8        return 0
9
10   #Verificar que las aristas usadas en C, existan en G
11
12   for arista in C:
13     if arista.existIn(G)==0:
14       print "Arista Inexistente"
15       return 0
16
17   #Verificar que se visite cada nodo una vez, y que  C sea un
         circuito
18
19   frecuencia_nodos[NodesIn(G)][2]
20
21   for i=0 to NodesIn(G):
22        frecuencia_nodos[i][1]=0
23
24   i=0
25   for node in G:
26     frecuencia_nodos[i++][0]=node
27
28   for node in C:
29     k=find(frecuencia_nodo,node)
30     frecuencia_nodo[k][1]++
31
32
```

```
33    for i=0 to NodesIn(G):
34      if(frecuencia_nodo[i][1]<1)
35        print "Grafo C no es un circuito , nodo i no visitado"
36        return 0
37      if(frecuencia_nodo[i][1]>1)
38        print "Nodo i visitado mas de una vez"
39        return 0
40
41    return 1
```

El algoritmo implementado, primero checa que todos los nodos de la solución C, existan en el grafo G, de lo contrario retorna 0 lo cual significa que no es solución al problema. EL siguiente paso que se realiza, es verificar que las aristas utilizadas en la solución C, existan en el grafo G. Y finalmente, se verifica que los nodos en la solución C, hayan sido visitados una y sólo una vez. Este último paso, se realiza utilizando una matriz que funciona como una tabla de frecuencia, en donde la primer columna contiene el nodo y la segunda columna contiene el número de veces que se visitó en el camino. Si la solución pasa las 3 verificaciones, el algoritmo retorna 1, lo cual significa que el certificado C, es en realidad un ciclo Hamiltoniano del grafo G.

Ejecución del Algoritmo.

Input.

Formato del archivo input.txt, (Grafo G)

```
5          //# nodos
7          //# aristas
1 2        //aristas nodoInicio nodoFin
1 3
1 4
2 4
2 5
3 4
4 5
```

Formato del archivo solución.txt, (Certificado G)

```
5          //# aristas
1 2        //aristas nodoInicio nodoFin
2 5
5 4
4 3
3 1
```

Las figuras 2 y 3, muestran la interpretación de los archivos de entrada.

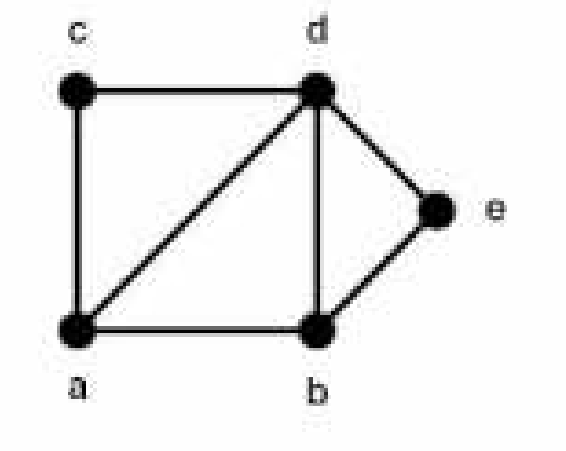

Figura 2: Interpretación de input.txt.

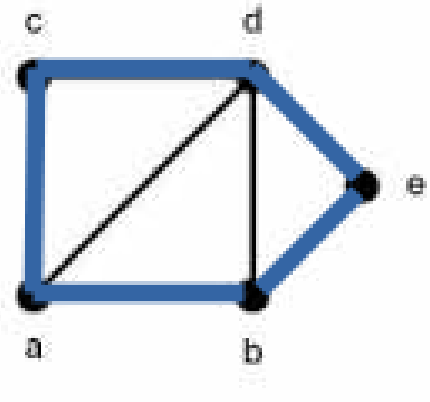

Figura 3: Interpretación de solución.txt.

Output.

```
bryan@bryan-ubuntu:~/Documentos/Análisis de Algoritmos/practica10/codigo$ ./hamilton<input.txt
Es un ciclo hamiltoniano
```

Figura 4: Output.

Si cambiamos solución.txt a:

```
1  5
2  1 2
3  2 5
4  5 4
5  4 3
6  3 5
```

```
bryan@bryan-ubuntu:~/Documentos/Análisis de Algoritmos/practica10/codigo$ ./hamilton<input.txt
Arista inexistente
No es un ciclo hamiltoniano
```

Figura 5: Output 2.

Demostración analítica del orden de complejidad de la verificación.

Orden de complejidad por verificación realizada (ver pseudocódigo página 3).
Sea nC y aC el número de nodos y de aristas respectivamente en el certificado C.
Sea nG y aG el número de nodos y de aristas respectivamente en el grafo G.
Verificación 1: $O(nC)$
Verificación 2: $O(aC)$
Verificación 3: $O(nC * nG)$

Como las verificaciones están en el mismo nivel, se concluye que el algoritmo tiene un orden de complejidad $O(nC * nG)$, pero para poder ser un ciclo Hamiltoniano es necesario que los nodos usados en el certificado C, sean los mismos que existen en el grafo G, entonces $nC = nG = n$.

Por lo tanto $verificacionHamilton \epsilon O(n^2)$.

Demostración mediante gráficas del orden de complejidad de la verificación.

La figura 6, muestra la gráfica $T(n)$ vs n, del algoritmo de verificación de los ciclos Hamiltonianos. La función roja, es $T(n) = n^2$, mientras que los puntos son los datos obtenidos de la ejecución del algoritmo, demostrando así, que el algoritmo tiene orden de complejidad $\emptyset(n^2)$, y más aún demostrando que el algoritmo se ejecuta en tiempo polinomial.

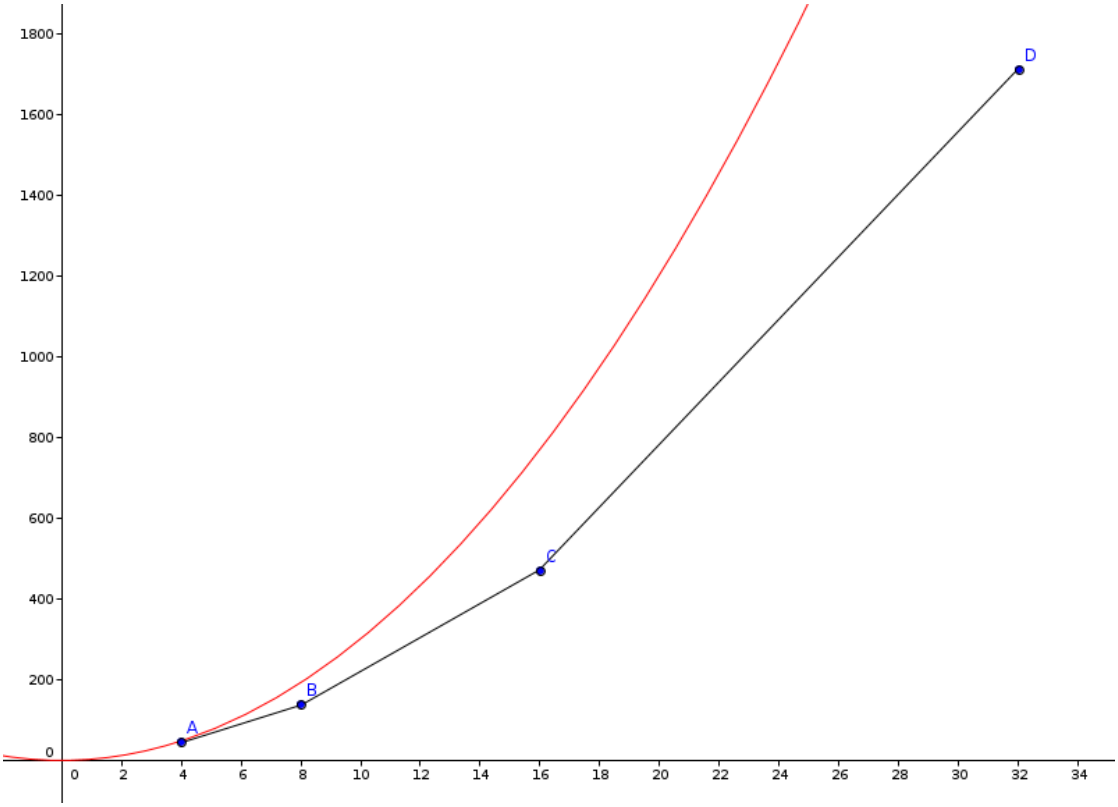

Figura 6: Output.

4. Conclusiones

Esta práctica es muy útil para demostrar que los problemas NP pueden verificarse en tiempo polinomial, pero a su vez nos da una perspectiva de lo difícil que es resolverlos. Tuve problemas al demostrar el orden de complejidad de manera analítica, sin embargo después de mucho analizar, pude dar con el resultado. También se me complico la implementación del algoritmo planteado, puesto que se tenían que hacer muchos ajustes para implementarlo en un lenguaje de programación como C.
A mi parecer esta última práctica termina con un muy buen curso en donde las prácticas definitivamente han sido lo mejor de éste.

5. Bibliografía

[1] https://es.wikipedia.org/wiki/Problema_del_ciclo_hamiltoniano

[2] https://es.wikipedia.org/wiki/Camino_hamiltoniano