Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

Análisis y Diseño de algoritmos

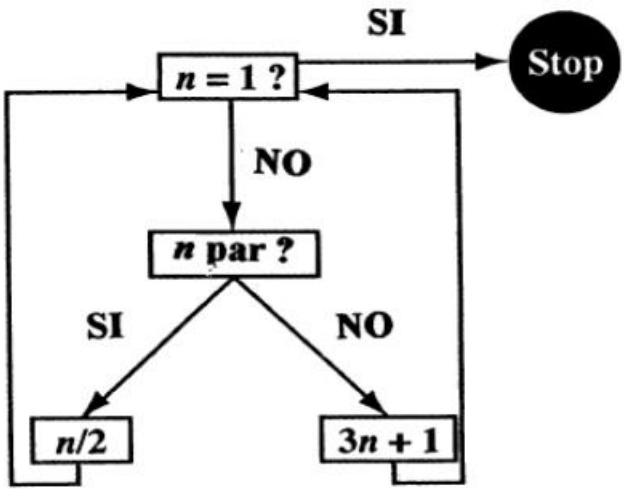
Examen

Primer parcial

Docente: Fernando Zacarias Flores

Integrantes	Matrículas
Diwsgen López Lozada	202043856
Juan Pablo Hernández Flores	202068295
Juan Martínez Díaz	202045427
Jesús Huerta Aguilar	202041509

1. **Dado el algoritmo mostrado en la figura siguiente, defina la serie que se genera si N=27 y la serie que se genera si N=15. De las series generadas deduzca la característica que aparece en estas y pruebe su conjetura con nuevas series.**



Este algoritmo se relaciona con la conjetura de Collatz ya que a partir de cualquier número **n**>=3 positivo y se le aplique esas condiciones, siempre se llegará a la secuencia final de números 4, 2 ,1.

Además, se encontró que para cualquier número par e impar por múltiplos de 2 sus pasos para llegar a 1 serán **pasos(anteriores)**+1.
Ejemplo para 10:
Pasos para llegar a 1= 6
Para 20 =7 …
Es decir:
n = pasos
2n = pasos (de n_anterior) +1
…

Código:

```
#Definir el valor de n

#Primer valor n=27

#Segundo valor n=15
#n=27
n=27
pasos=0

#Algoritmo
#Mientras n sea diferente de 1
while(n!=1):
    #Si n es par
    if(n%2==0):
```

```
        #Lo dividimos entre 2
        n=n/2
        print("N par valor:",n)
        pasos=pasos+1
    else:
        #SI no se multiplica por 3 mas 1
        n=3*n+1
        print("N  no es par:",n)
        pasos=pasos+1

print("Total de pasos",pasos)
```

Ejecución con número 27:

```
N  no es par: 160.0
N par valor: 80.0
N par valor: 40.0
N par valor: 20.0
N par valor: 10.0
N par valor: 5.0
N  no es par: 16.0
N par valor: 8.0
N par valor: 4.0
N par valor: 2.0
N par valor: 1.0
Total de pasos 111
```

Ejecución con número 15:

```
N  no es par: 160.0
N par valor: 80.0
N par valor: 40.0
N par valor: 20.0
N par valor: 10.0
N par valor: 5.0
N  no es par: 16.0
N par valor: 8.0
N par valor: 4.0
N par valor: 2.0
N par valor: 1.0
Total de pasos 17
```

Ejemplo con numero 69:

```
N  no es par: 208
N par valor: 104.0
N par valor: 52.0
N par valor: 26.0
N par valor: 13.0
N  no es par: 40.0
N par valor: 20.0
N par valor: 10.0
N par valor: 5.0
N  no es par: 16.0
N par valor: 8.0
N par valor: 4.0
N par valor: 2.0
N par valor: 1.0
Total de pasos 14
```

Con su doble (138):

```
N par valor: 69.0
N  no es par: 208.0
N par valor: 104.0
N par valor: 52.0
N par valor: 26.0
N par valor: 13.0
N  no es par: 40.0
N par valor: 20.0
N par valor: 10.0
N par valor: 5.0
N  no es par: 16.0
N par valor: 8.0
N par valor: 4.0
N par valor: 2.0
N par valor: 1.0
Total de pasos 15
```

2. Resolver la siguiente ecuación de recurrencia: T(n) = 2nT(n/2) + n^n

$$T(n) = 2nT\left(\frac{n}{2}\right) + n^n$$

$a = 2^n, b = 2, d = \log_2(2^n)$

En caso 1:

$0 < \varepsilon \leq n - n \rightarrow 0 < \varepsilon \leq 0$

No existe epsilon que satisfaga la condición por lo que no se cumple.

En caso 2:

$f(n) \in \theta(N^d \log^k(N)), \forall k \geq 0$

$f(N) \in \theta(N^n \log^0(N))$

$f(n) \in \theta(N^n)$

$$\boxed{n^n \in \theta(N^n)}$$

3. **Supongamos que cada noche dispones de una hora de CPU para ejecutar cierto programa y que con esa hora tienes suficiente tiempo para ejecutar un programa con una entrada, a lo sumo, de tamaño n = 1 000 000. Ante esta situación, tu profesor te regala una máquina 100 veces más rápida que la vieja. Ahora ¿cuál es el mayor tamaño de entrada que podrá gestionar nuestro programa en una hora, si su complejidad T(n) fuera para alguna constante k_i?**

a) k1 = n b) k2 = n^2 c) k3 = 10^n

Utilizando :

$$\frac{T(n)}{100} \leq 1(hora)$$

$$k_i \bullet \frac{f(n)}{100} \leq 1$$

$$f(n) \leq \frac{100}{k_i}$$

Usando la complejidad T(n) = k1,

$$f(n) = n$$

$$f(n) \leq \frac{100}{k1}$$

$$n \leq \frac{100}{k1}$$

El tiempo de ejecución del programa sería lineal en el tamaño de entrada n, lo que significa que el programa tardaría n horas en ejecutarse para una entrada de tamaño n. Como n puede ser tan grande como 1,000,000, entonces el programa no se ejecutaría en una hora en la nueva máquina.

Usando la complejidad T(n) = k2

$$T(n) = k2 \bullet n^2$$

$$f(n) = n^2 \quad \rightarrow \quad f(n) \leq \frac{100}{k2}$$

$$n^2 \leq \frac{100}{k2} \quad \rightarrow \quad n \leq \sqrt{\frac{100}{k2}}$$

El tiempo de ejecución del programa sería cuadrático en el tamaño de entrada n, lo que significa que el programa tardaría n^2 horas en ejecutarse para una entrada de tamaño n. Como (100 * 1,000,000) ^2 = 10^18, que es mucho mayor que 0.01, el programa no se ejecutaría en una hora en la nueva máquina.

Usando la complejidad T(n) = k3

$$k = 100n \rightarrow k = n$$

Evaluando con el caso 2 del teorema maestro $\theta\left(n^d \log^{k+1}(n)\right)$:

$$T(n) = T\left(\frac{n}{100}\right) + 1000000$$

$$a = 1, b = 100, d = \log_{100} 1 = 0$$

$$f(n) = 1000000 \cdot n^0$$

$$f(n) \in \theta(1)$$

$$f(n) \in \theta\left(n^d \log^{k+1}(n)\right); \forall k \geq 0$$

Tomamos $k = 0$:

$$\theta(1) = \theta\left(n^0 \log(n)\right)$$

$\theta(1) = \theta(1)$ Esto es verdadero, por lo que:

$$T(n) = \theta\left(n^0 \log^{0+1}(n)\right)$$

$$T(n) = \theta\left(\log(n)\right)$$

Si tomamos una constante k:

$$\log(k) = \log_{10} k \Rightarrow k = 10^n$$

Por lo que el programa puede manejar una entrada de tamaño n = 1,000,000 en la nueva máquina en una hora.

Por lo tanto, la opción correcta es **c) k3 = 10^n.**

4. **Sea f(n) = n y g(n) = n^3 funciones de N a R. Demostrar que g domina asintóticamente a f bajo la definición 1.**

Definición 1. Sean f y g funciones de N a R. Se dice que f domina asintóticamente a g o que g es dominada asintóticamente por f; si $\exists k \geq 0$, y $m \geq 0$ tales que: $|g(n)| \leq m\,|f(n)|$, $\forall n \geq k$

Demostración:

$$\left|f\left(n\right)\right| \leq m \bullet \left|g\left(n\right)\right|$$

*Ta*l que $f\left(n\right) = n$

$Y\ g\left(n\right) = n^3$

$\therefore$

$$\left|n\right| \leq m \bullet \left|n^3\right|$$

*D*ividimos ambos lados entre n^3

$$\left|\frac{1}{n^2}\right| \leq m$$

$$\lim_{n\to\infty}\left|\frac{1}{n^2}\right| = 0$$

$\therefore$

$$0 \leq m$$

Y por lo tanto lo cumple, en ejemplo:

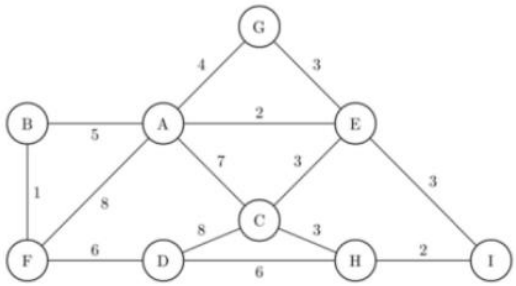
Sea m=2 y k=2

$$2 \leq 1 \bullet \left(2^3\right)$$

$$2 \leq 8$$

Por lo tanto, g domina asintóticamente a f bajo la definición 1.

5. **Obtener una representación del camino mínimo desde el vértice A en el siguiente grafo hacia todos los demás vértices, utilizando el algoritmo de BFS, DFS y A*. Volver a realizar el ejercicio, suponiendo que la arista de B a A ahora tiene un peso de 1.**



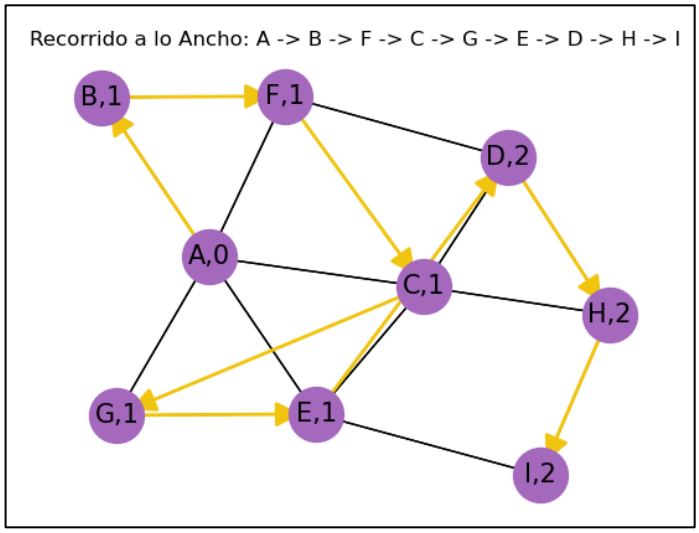
Para los códigos BFS, DFS se ocupó los dados en clase

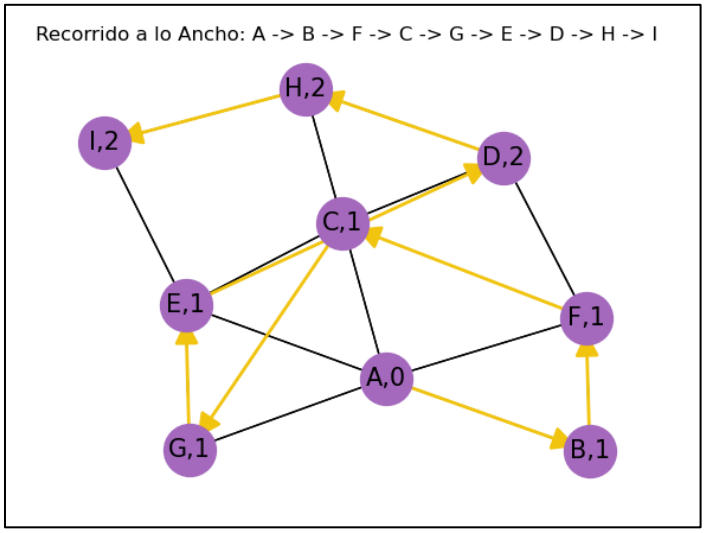
BFS:

Link del Código en Google Colaboratory:

https://colab.research.google.com/drive/1YmGJWF5qr7Xwmqe5aCBvYUPGdoUdAfGX?usp=sharing

Ejecuciones:





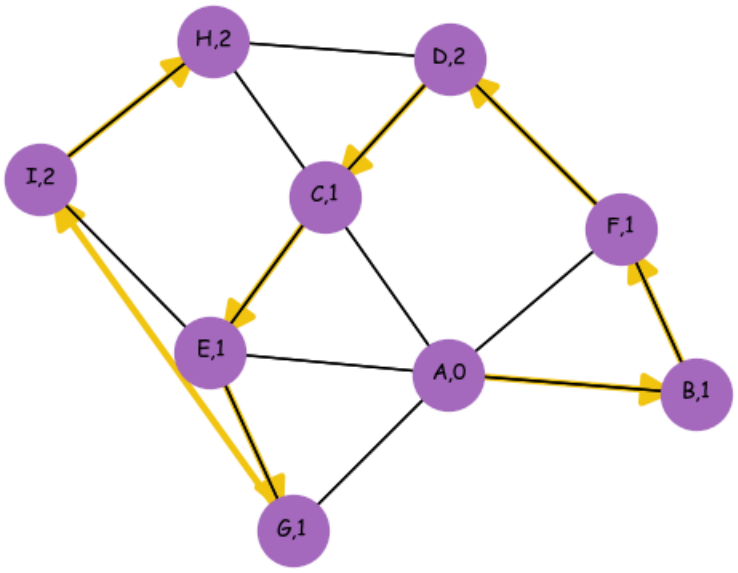
DFS:

Link del Código en Google Colaboratory:

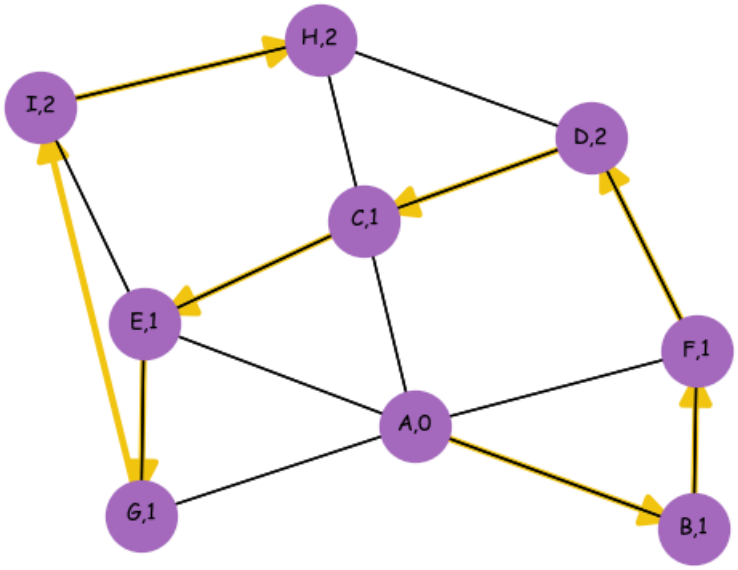
https://colab.research.google.com/drive/1q6-w2pkPk0WwYOTM_NS7OEa-76NH78Il?usp=sharing

Ejecuciones:

Recorrido a lo DFS: A -> B -> F -> D -> C -> E -> G -> I -> H



Recorrido a lo DFS: A -> B -> F -> D -> C -> E -> G -> I -> H

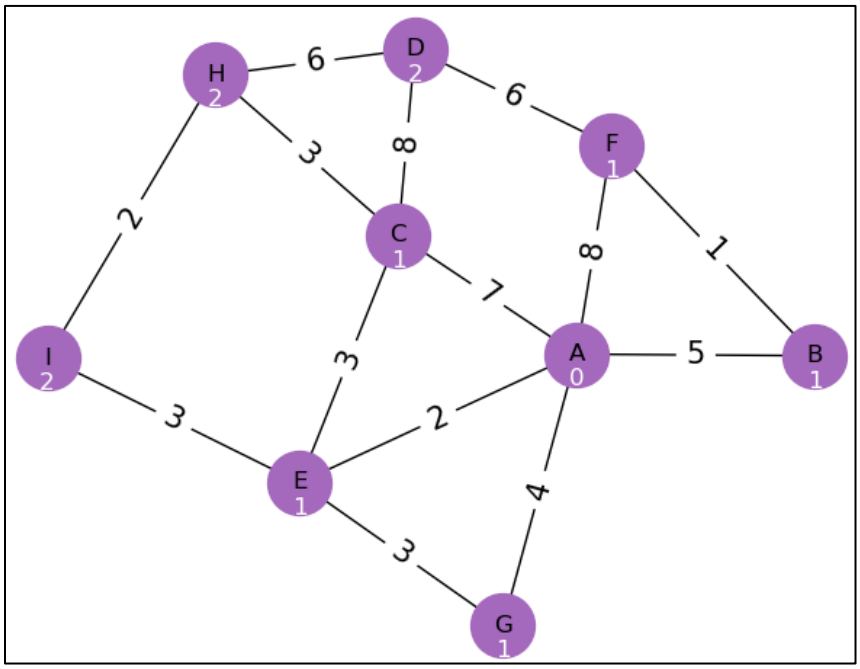


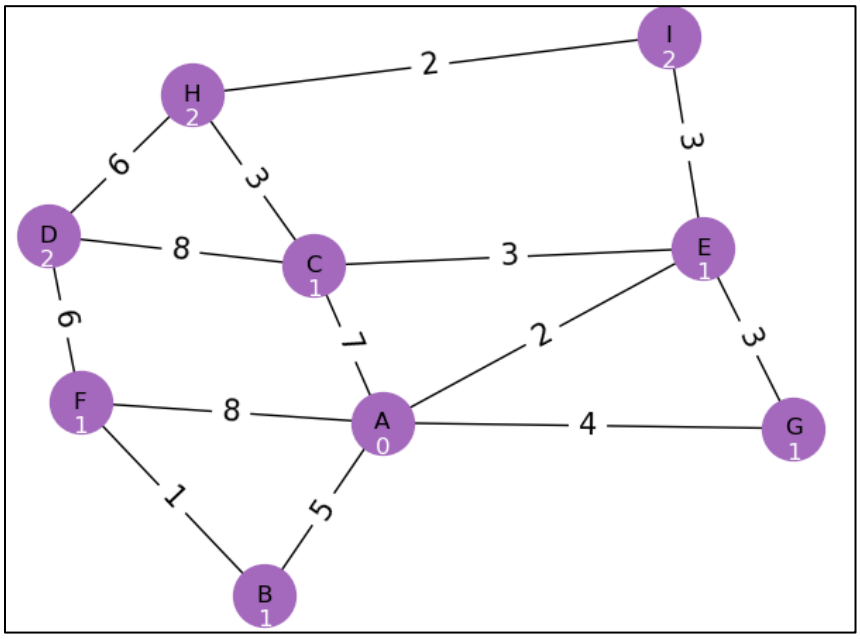
A*:

Link del Código en Google Colaboratory:

https://colab.research.google.com/drive/1PTppZcgr1QVIB1KIuepY0SVQTzuRJ7aZ?usp=sharing

Ejecuciones:





```
Ruta de A a D: ['A', 'B', 'F', 'D']
Suma de distancias g(n): 4
Heurisitica de la ruta h(n): 12
f(n)=g(n)+h(n): 16

Ruta de A a F: ['A', 'B', 'F']
Suma de distancias g(n): 2
Heurisitica de la ruta h(n): 6
f(n)=g(n)+h(n): 8

Ruta de A a G: ['A', 'G']
Suma de distancias g(n): 1
Heurisitica de la ruta h(n): 4
f(n)=g(n)+h(n): 5

Ruta de A a B: ['A', 'B']
Suma de distancias g(n): 1
Heurisitica de la ruta h(n): 5
f(n)=g(n)+h(n): 6

Ruta de A a E: ['A', 'E']
Suma de distancias g(n): 1
Heurisitica de la ruta h(n): 2
f(n)=g(n)+h(n): 3

Ruta de A a I: ['A', 'E', 'I']
Suma de distancias g(n): 3
Heurisitica de la ruta h(n): 5
f(n)=g(n)+h(n): 8

Ruta de A a A: ['A']
Suma de distancias g(n): 0
Heurisitica de la ruta h(n): 0
f(n)=g(n)+h(n): 0

Ruta de A a C: ['A', 'E', 'C']
Suma de distancias g(n): 2
Heurisitica de la ruta h(n): 5
f(n)=g(n)+h(n): 7

Ruta de A a H: ['A', 'E', 'I', 'H']
Suma de distancias g(n): 5
Heurisitica de la ruta h(n): 7
f(n)=g(n)+h(n): 12
```

```
Ruta de A a H: ['A', 'E', 'I', 'H']
Suma de distancias g(n): 5
Heurisitica de la ruta h(n): 7
f(n)=g(n)+h(n): 12

Ruta de A a A: ['A']
Suma de distancias g(n): 0
Heurisitica de la ruta h(n): 0
f(n)=g(n)+h(n): 0

Ruta de A a E: ['A', 'E']
Suma de distancias g(n): 1
Heurisitica de la ruta h(n): 2
f(n)=g(n)+h(n): 3

Ruta de A a I: ['A', 'E', 'I']
Suma de distancias g(n): 3
Heurisitica de la ruta h(n): 5
f(n)=g(n)+h(n): 8

Ruta de A a F: ['A', 'B', 'F']
Suma de distancias g(n): 2
Heurisitica de la ruta h(n): 6
f(n)=g(n)+h(n): 8

Ruta de A a D: ['A', 'B', 'F', 'D']
Suma de distancias g(n): 4
Heurisitica de la ruta h(n): 12
f(n)=g(n)+h(n): 16

Ruta de A a B: ['A', 'B']
Suma de distancias g(n): 1
Heurisitica de la ruta h(n): 5
f(n)=g(n)+h(n): 6

Ruta de A a G: ['A', 'G']
Suma de distancias g(n): 1
Heurisitica de la ruta h(n): 4
f(n)=g(n)+h(n): 5

Ruta de A a C: ['A', 'E', 'C']
Suma de distancias g(n): 2
Heurisitica de la ruta h(n): 5
f(n)=g(n)+h(n): 7
```