

Algoritmos Heurísticas  
y  
Algoritmos Aproximados

**Juan Manuel Ussher**  
**CI: 82.102.248**

## **Heurística:**

- Arte de Inventar.**
- Técnica de la indagación y del descubrimiento.**

*Algoritmos Heurísticos:*

*son un procedimiento que puede producir una solución para nuestro problema.*

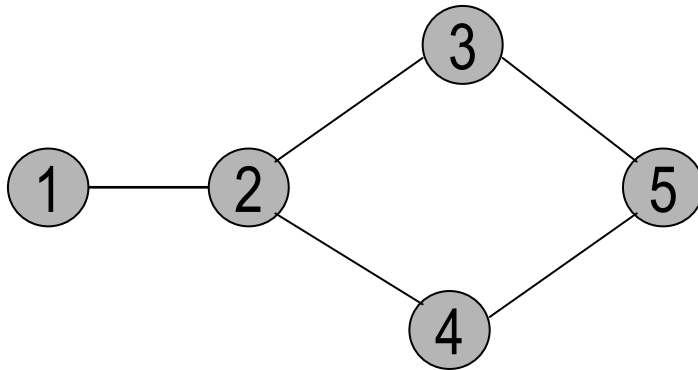
*Algoritmos aproximados:*

*son un procedimiento que siempre proporciona algún tipo de solución para el problema aún cuando probablemente no encuentre una solución óptima.*

# Algoritmos Heurísticos

---

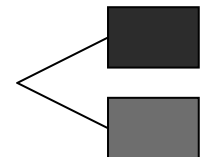
## ***Ejemplo 1*** **Coloreado de un grafo**



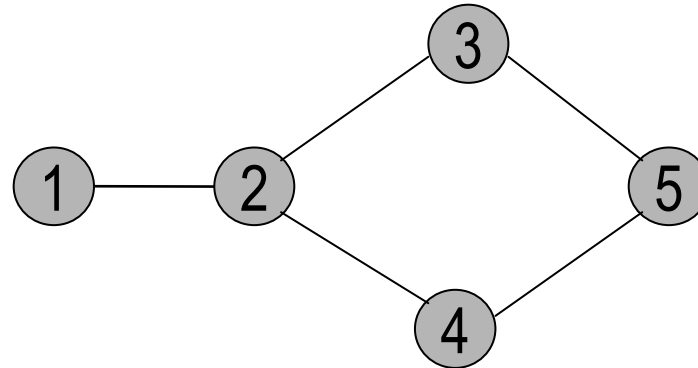
1. Colorear cada nodo, de manera que no haya dos nodos adyacentes del mismo color.
2. Cual es el número mínimo de colores

---

La solución óptima nos dice que el mínimo son dos colores



Secuencia: 1-2-3-4-5



Selecciona el nodo 1 como inicial pinta con un color al azar.

Pasa al 2, no puede pintar(1 ya esta pintado).

Pasa al 3, pinta.

Pasa al 4, pinta.

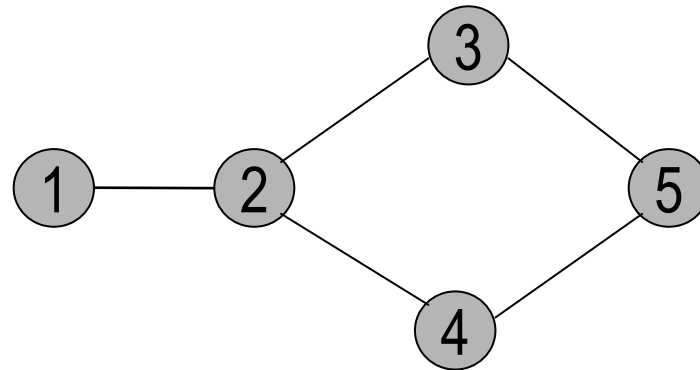
Pasa al 5, no puede pintar(3 y 4 ya están pintados).

Selecciona otro nodo y pinta con otro color.

Pasa al siguiente, etc.

Solución óptima

Secuencia: 1-5-2-3-4



Selecciona el nodo 1 como inicial pinta con un color al azar.

Pasa al 5, pinta

Pasa al 2, no puede pintar(1 ya esta pintado).

Pasa al 3, no puede pintar(5 ya esta pintado).

Pasa al 4, no puede pintar(5 ya esta pintado).

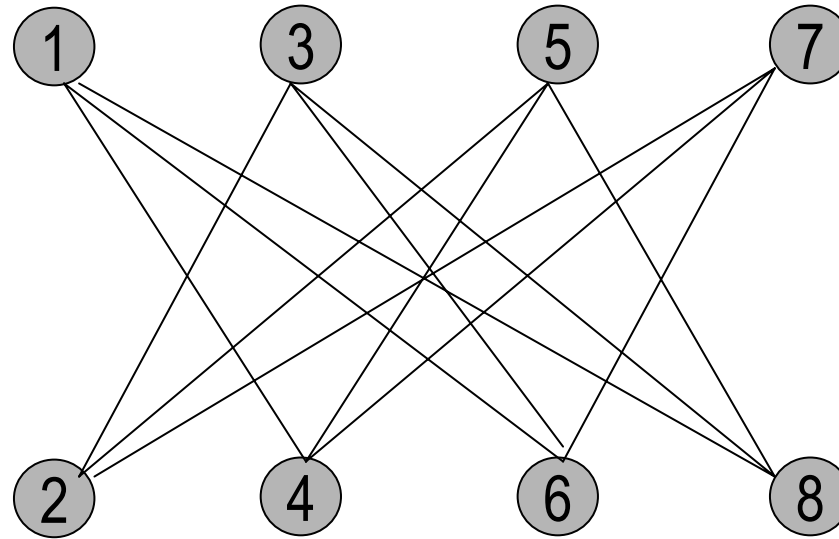
Selecciona otro nodo(2) y pinta con otro color.

Pasa al 3 y al 4, no puede pintar(2 ya esta pintado).

Pasa al siguiente(3), y selecciona otro color.

Pasa al 4, pinta.

Solución NO óptima



2 colores

Secuencia:

1, 3, 5, ..., 2n-1, 2, 4, 6, ..., 2n

Solución óptima

4 colores

Secuencia:

1, 2, 3, 4, 5, ..., 2n-1, 2n teatro lírico

Solución no óptima

**n**



## ***Ejemplo 2***

### **Problema del viajante**

El viajante sale de una ciudad para visitar todas las demás pasando exactamente una vez por cada una y volver al punto de partida habiendo recorrido la menor distancia posible.

El problema se representa como un grafo no dirigido con  $n$  nodos.

Un ciclo del grafo que pase por todos los nodos exactamente una vez se denomina ciclo hamiltoniano.

Por lo tanto para resolver el problema hay que hallar el ciclo hamiltoniano más corto del grafo.

Por ejemplo:

	<b>Hacia</b>					
<b>Desde</b>	Ciudad	2	3	4	5	6
	1	3	10	11	7	25
	2		8	12	9	26
	3			9	4	20
	4				5	15
	5					18

En este caso el recorrido óptimo tiene una longitud de 58. Esta distancia se logró haciendo el recorrido 1,2,3,6,4,5,1.

Nuestro algoritmo heurístico comenzará en un nodo arbitrario y ahí decidirá visitar el nodo más cercano que no haya sido visitado.

Así, si comienzo en 1, recorriendo las distancias más cortas será: 1-2-3-5-4-6-1, dando un total de 60.

Aunque este algoritmo heurístico no es tan óptimo, tampoco está mal y puede ser utilizado.

# Algoritmos Aproximados

---

## ***Ejemplo***

### **Problema de la mochila**

Tenemos  $n$  objetos y una mochila. Cada objeto  $i$  tiene un peso  $w_i$  y un valor  $v_i$ . La mochila tiene capacidad para cargar un peso total  $W$ .

El problema consiste en llenar la mochila de tal forma que se maximice el valor de los objetos cargados.

Función *mochila-voraz* ( $w[1..n]$ ,  $v[1..n]$ ,  $W$ )

se ordena el caso de modo que  $v[i] / w[i] \geq v[j] / w[j]$  para todo  $1 \leq i \leq j \leq n$

*Peso* = 0

*Valor* = 0

Para  $i=1$  hasta  $n$  hacer

si  $\text{peso} + w[i] \leq W$  entonces

$\text{valor} = \text{valor} + v[i]$


$\text{peso} = \text{peso} + w[i]$

Devolver *valor*

Este algoritmo puede ser bastante malo.

Por ejemplo tomemos un entero  $x > 2$  tan grande como se desee y consideramos el caso de que la capacidad de la mochila  $W=x$ .

Hay dos objetos cuyos pesos y valores son  $w_1=1$ ,  $v_1=2$ ,  $w_2=x$ ,  $v_2=x$



Los valores ya están ordenados en orden no creciente de valores por unidad de peso ( $v_1 / w_1 = 2 \geq v_2 / w_2 = 1$ )

El algoritmo coloca en la mochila primero al objeto 1, luego cuando a insertar el objeto 2, éste no entra.

El algoritmo envía como solución el 2.

Claro está, que ésta no es la solución óptima. La solución óptima sería dejar el objeto uno y cargar el objeto 2 cuya solución devuelta sería  $x$ .

Este algoritmo proporciona un valor que esta dentro de un factor  $x / 2$  del óptimo, lo cual puede llegar a ser muy malo.

Consideremos ahora el algoritmo aproximado:

Función *mochila-aproximada* (  $w[1..n]$ ,  $v[1..n]$ ,  $W$  )

$máximo = \max \{ v[i] \mid 1 \leq i \leq n \}$

devolver  $\max (máximo, mochila-voraz ( w, v, W )$

Ahora si éste es nuestro algoritmo aproximado y partiendo del hecho de que:

$$\max ( x, y ) \geq ( x + y ) / 2$$

Y llamando  $opt$  a la solución óptima y  $opt'$  a la solución dada por nuestro algoritmo aproximado.

Sabiendo también que  $mochila-voraz ( w, v, W )$  es mayor o igual que la suma de todos los valores de los objetos desde el primero hasta el  $n - 1$

Es decir:

$$\text{mochila} - \text{voraz}(w, v, W) \geq \sum_{i=1}^{n-1} v_i$$

Entonces:

$$\text{opt}' = \max(\text{máximo}, \text{mochila} - \text{voraz}(w, v, W))$$

$$\text{opt}' \geq \frac{(\text{máximo} + \text{mochila} - \text{voraz}(w, v, W))}{2}$$

$$\text{opt}' \geq \frac{(v_n + \sum_{i=1}^{n-1} v_i)}{2}$$

$$\text{opt}' \geq \frac{\sum_{i=1}^n v_i}{2}$$

$$\text{opt}' \geq \text{opt} / 2$$

$$\text{opt} \leq 2\text{opt}'$$

Lo cual demuestra que la solución aproximada está dentro de un factor de 2 de la solución óptima y por supuesto esto es mejor a que los solución hallada con el primer algoritmo



# Gracias

**Juan Manuel Ussher**

**CI: 82.102.248**