Índice

- **EL ENFOQUE GREEDY**
- **ALGORITMOS GREEDY EN GRAFOS**
- **HEURÍSTICA GREEDY**
  - Introducción a la Heurística Greedy
  - El Problema de Coloreo de un Grafo
  - Problema del Viajante de Comercio
  - Problema de la Mochila
  - Problema de asignación de tareas

Heurísticas Greedy

SITUACIÓN QUE NOS PODEMOS ENCONTRAR

- Hay casos en los cuales no se puede conseguir una algoritmo voraz para que el que se pueda demostrar que encuentra la solución óptima

- Existen para distintos problemas NP-completos

Heurísticas

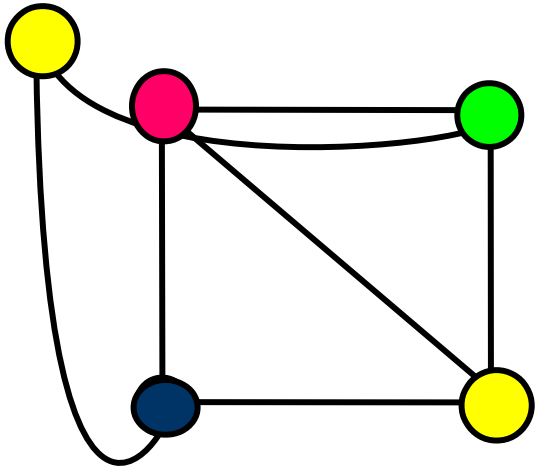
- Heurística: Son procedimientos que, basados en la experiencia, proporcionan buenas soluciones a problemas concretos
- Metaheurísticas de propósito general:
  - Enfriamiento Simulado, Búsqueda Tabu,
  - GRASP (Greedy Randomized Adaptive Search Procedures), Búsqueda Dispersa, Búsqueda por Entornos Variables, Búsqueda Local Guiada,
  - Computación Evolutiva (Algoritmos Genéticos, …), Algoritmos Meméticos, Colonias de Hormigas, Redes de Neuronas

Heurísticas Greedy

- **¿Satisfacer / optimizar?**
- El tiempo efectivo que se tarda en resolver un problema es un factor clave
- Los algoritmos greedy pueden actuar como heurísticas
  - El problema del coloreo de un grafo
  - El problema del Viajante de Comercio
  - El problema de la Mochila
  - ...
- Suelen usarse también para encontrar una primera solución (como inicio de otra heurística)

El Problema del Coloreo de un Grafo

- Planteamiento
  - Dado un grafo plano G=(V, E), determinar el mínimo numero de colores que se necesitan para colorear todos sus vértices, y que no haya dos de ellos adyacentes pintados con el mismo color
- Si el grafo no es plano puede requerir tantos colores como vértices haya
- Las aplicaciones son muchas
  - Representación de mapas
  - Diseño de páginas webs
  - Diseño de carreteras



El Problema del Coloreo de un Grafo

- El problema es NP y por ello se necesitan heurísticas para resolverlo.
- El problema reune todos los requisitos para ser resuelto con un algoritmo greedy.
- Del esquema general greedy se deduce un algoritmo inmediatamente.
- Teorema de Appel-Hanke (1976): Un grafo plano requiere a lo sumo 4 colores para pintar sus nodos de modo que no haya vértices adyacentes con el mismo color.

El Problema del Coloreo de un Grafo

- Suponemos que tenemos una paleta de colores (con más colores que vértices).
- Elegimos un vértice no coloreado y un color. Pintamos ese vértice de ese color.
- Lazo greedy: Seleccionamos un vértice no coloreado v. Si no es adyacente (por medio de una arista) a un vértice ya coloreado con el nuevo color, entonces coloreamos v con el nuevo color.
- Se itera hasta pintar todos los vértices.

Implementacion del algoritmo

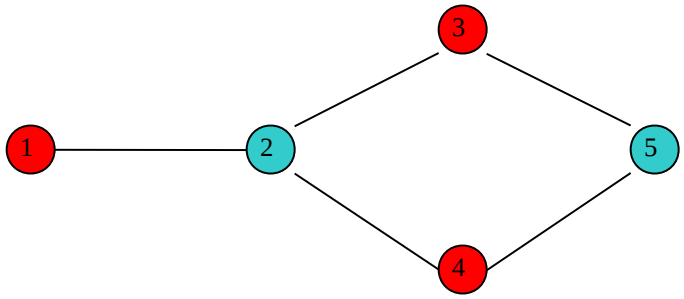
Funcion **COLOREO**

{COLOREO pone en NuevoColor los vertices de G que pueden tener el mismo color}

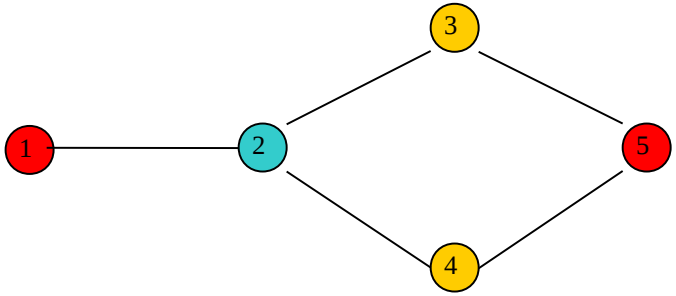
```
Begin
    NuevoColor = ∅
    Para cada vertice no coloreado v de G Hacer
      Si v no es adyacente a ningun vertice en NuevoColor
      Entonces
        Marcar v como coloreado
        Añadir v a NuevoColor
End
```

Se trata de un algoritmo que funciona en **O(n),** pero que no siempre da la solución óptima.

Ejemplo



El orden en el que se escogen los vértices para colorearlos puede ser decisivo: el algoritmo da la solución optimal en el grafo de arriba, pero no en el de abajo



Ejemplo: Diseño de cruces de semáforos

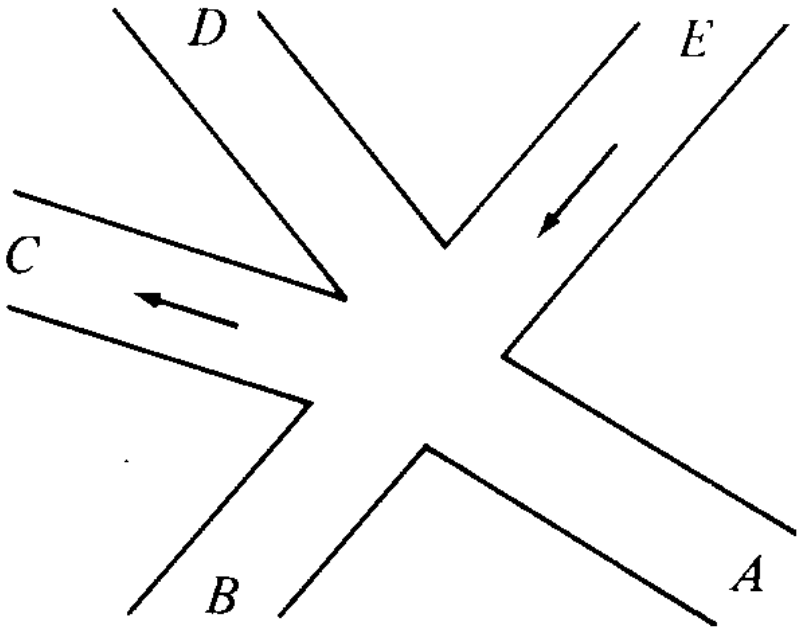
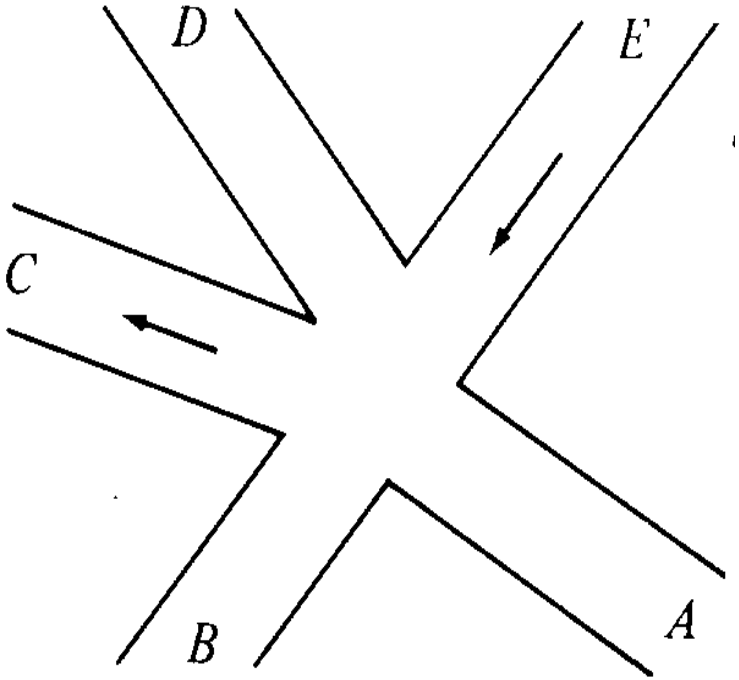


Fig. 1.1. An intersection.

- A la izquierda tenemos un cruce de calles.
- Se señalan los sentidos de circulación.
- La falta de flechas significa que podemos ir en las dos direcciones.
- Queremos diseñar un patrón de semáforos con el mínimo número de semáforos, lo que
- Ahorrara tiempo (de espera) y dinero.
- Suponemos un grafo cuyos vértices representan turnos, y cuyas aristas

unen esos turnos que no pueden realizarse simultáneamente sin que haya colisiones. El problema del cruce con semáforos se convierte en un problema de coloreo de los vértices de un grafo.

Cruce con semáforos: vértices



Los turnos (vértices del grafo) serían 13:

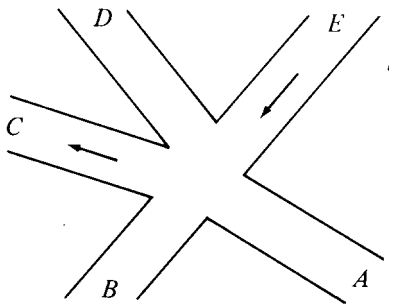
{AB, AC, AD,

BA, BC, BD,

DA, DB, DC,

EA, EB, EC, ED}

Cruce con semáforos: aristas



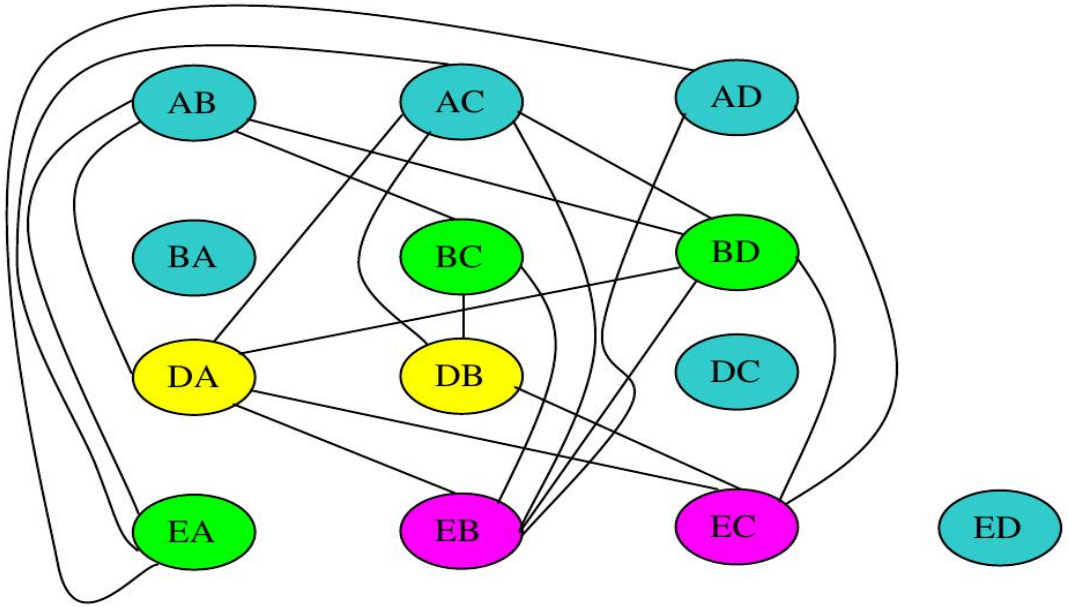
Los turnos (vértices del grafo) serían 13:

{AB, AC, AD,

BA, BC, BD,

DA, DB, DC,

EA, EB, EC, ED}



El Problema del Viajante de Comercio

- Un viajante de comercio que reside en una ciudad, tiene que trazar una ruta que, partiendo de su ciudad, visite todas las ciudades a las que tiene que ir una y sólo una vez, volviendo al origen y con un recorrido mínimo

- Es un problema NP, no existen algoritmos en tiempo polinomial, aunque si los hay exactos que lo resuelven para grafos con 40 vértices aproximadamente.

- Para más de 40, es necesario utilizar heurísticas, ya que el problema se hace intratable en el tiempo.

El Problema del Viajante de Comercio

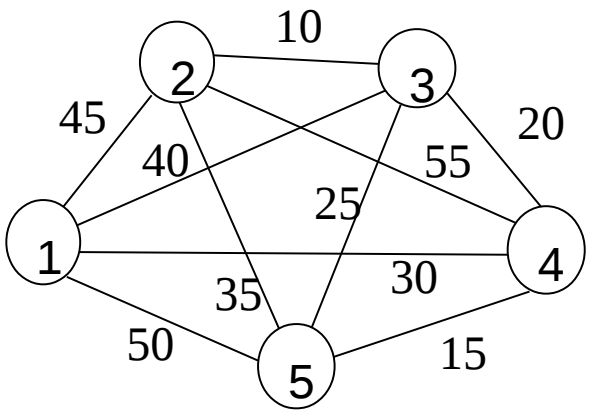
- Supongamos un grafo no dirigido y completo G = (N, A) y L una matriz de distancias no negativas referida a G. Se quiere encontrar un **Circuito Hamiltoniano Minimal**.

- Este es un problema Greedy típico, que presenta las 6 condiciones para poder ser enfocado con un algoritmo greedy.
- Puede plantearse con nodos como candidatos o con aristas como candidatos.

- Destaca de esas 6 características **la condición de factibilidad** (para el caso de aristas):
  - que al seleccionar una arista no se formen ciclos,
  - que las aristas que se escojan cumplan la condición de no ser incidentes en tercera posición al nodo escogido.

El Problema del Viajante de Comercio

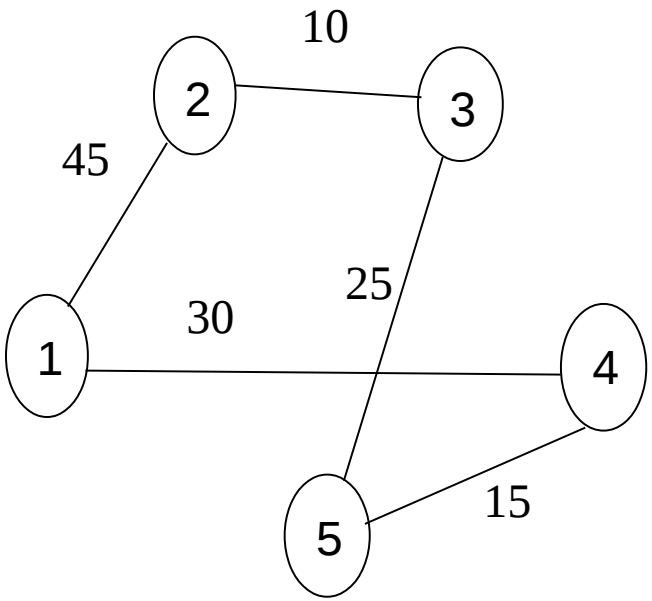
- Consideremos el siguiente grafo



- Posibilidades:
  - **Los nodos son los candidatos. Empezar en un nodo cualquiera y en cada paso moverse al nodo no visitado más próximo al último nodo seleccionado.**
  - **Las aristas son los candidatos. Hacer igual que en el algoritmo de Kruskal, pero garantizando que se forme un ciclo al final del proceso. Y de cada nodo no salgan más de dos aristas.**

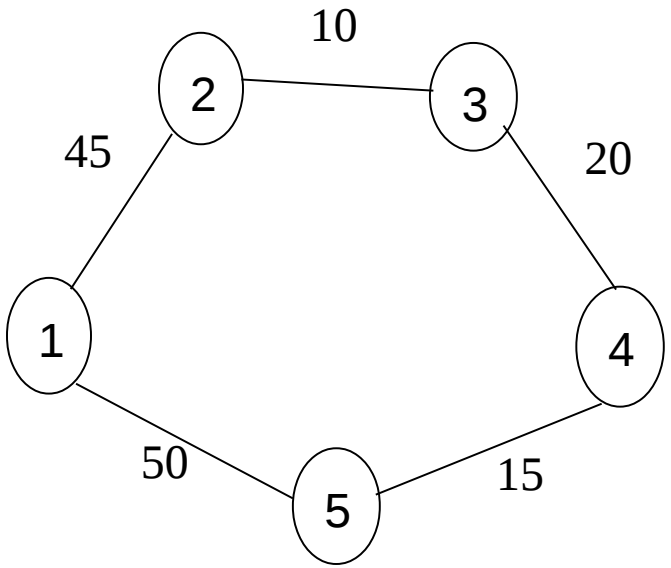
El Problema del Viajante de Comercio

- Solución con la primera heurística
- Solución empezando en el nodo 1



Solución: (1, 4, 5 ,3, 2),
125

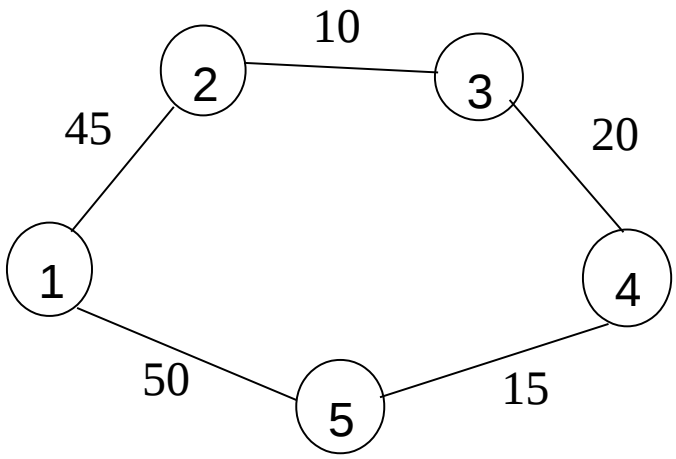
Solución empezando en el nodo 5



Solución: (5, 4, 3, 2, 1),
140

El Problema del Viajante de Comercio

- Solución con la segunda heurística



- Solución: ((2, 3), (4, 5), (3, 4), (1, 2), (1, 5))

  Coste = 10+15+20+45+50 = 140

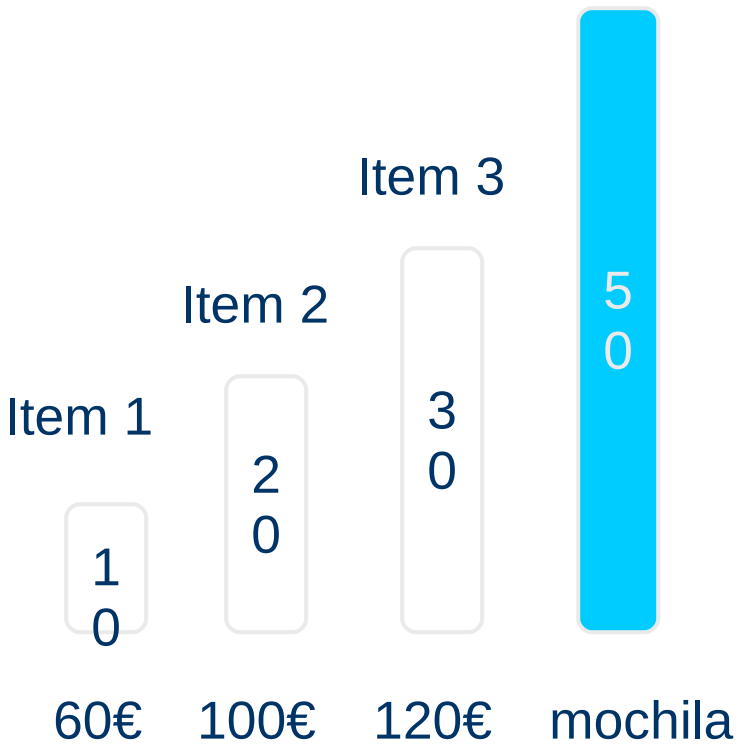
- En todos los casos la eficiencia es la del algoritmo de ordenación que se use

El problema de la Mochila

- Tenemos n objetos y una mochila. El objeto i tiene un peso w_i y la mochila tiene una capacidad M.
- Si metemos en la mochila el objeto i, generamos un beneficio de valor p_i
- El objetivo es rellenar la mochila de tal manera que se maximice el beneficio que producen los objetos que se transportan, con la limitación de la capacidad de valor M.

- Ya hemos visto que si los objetos se pueden fraccionar, el algoritmo greedy da la solución óptima.
- Pero si los objetos no se pueden fraccionar la estrategia greedy no es óptima, es solo una heurística.

Ejemplo: Mochila 0/1



En este ejemplo la solución óptima es incluir los items 2 y 3, con valor 220.

La estrategia greedy elegiría primero el item 1 (densidad 6) y luego el item 2 (densidad 5). El item 3 (densidad 4) ya no cabe. Valor: 160.

La asignación de tareas

- Supongamos que disponemos de n trabajadores y n tareas. Sea $b_{ij} > 0$ el coste de asignarle el trabajo j al trabajador i.
- Una asignación de tareas puede ser expresada como una asignación de los valores 0 ó 1 a las variables x_{ij}, donde
  - $x_{ij} = 0$ significa que al trabajador i no le han asignado la tarea j, y
  - $x_{ij} = 1$ indica que sí.
- Una asignación válida es aquella en la que a cada trabajador sólo le corresponde una tarea y cada tarea está asignada a un trabajador.

La asignación de tareas

- Dada una asignación válida, definimos el coste de dicha asignación como:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} b_{ij}$$

- Diremos que una asignación es óptima si es de mínimo coste.

- Cara a diseñar un algoritmo ávido para resolver este problema podemos pensar en dos estrategias distintas: asignar cada trabajador a la mejor tarea posible, o bien asignar cada tarea al mejor trabajador disponible.
- Sin embargo, ninguna de las dos estrategias tiene por qué encontrar siempre soluciones óptimas. ¿Es alguna mejor que la otra?

La asignación de tareas

		Tarea		
		1	2	3
Trabajador	1	16	11	17
	2	20	15	1
	3	18	17	20

- La estrategia greedy para trabajadores le asignaría la tarea 2 al trabajador 1, la tarea 3 al trabajador 2, y la tarea 1 al trabajador 3, con un coste de 11+1+18=30. En este caso ese es el óptimo.
- La estrategia greedy para tareas asignaría el trabaj. 1 a la tarea 1, el trabaj. 2 a la tarea 2 y el trabaj. 3 a la tarea 3, con un coste de 16+15+20=51.

La asignación de tareas

- Esas estrategias dependen del orden en que se procesan los trabajadores o las tareas.
- Si en vez de usar el orden 123 como antes usamos el 321 obtenemos:
- Para la estrategia para trabajadores: tarea 2 a trabaj. 3, tarea 3 a trabaj. 2 y tarea 1 a trabaj. 1, con coste 17+1+16=34.
- Para la estrategia para tareas: trabaj. 2 a tarea 3, trabaj. 1 a tarea 2 y trabaj. 3 a tarea 1, con coste 1+11+18=30

La asignación de tareas

- Otra estrategia (que no depende de ningún orden) es elegir en cada paso el par trabajador-tarea con menor coste, de entre los compatibles con las decisiones ya tomadas.
- En el ejemplo esa estrategia escogería los pares (trabajador,tarea) (2,3), (1,2) y (3,1), con costo 1+11+18=30, que en este caso es la solución óptima.
- Esta estrategia tampoco es óptima en general.

Asignación de tareas: no optimalidad

Cada uno de cuatro laboratorios, A,B,C y D tienen que ser equipados con uno de cuatro equipos informáticos. El costo de instalación de cada equipo en cada laboratorio lo da la tabla. Queremos encontrar la asignación menos costosa.

	1	2	3	4
A	48	48	50	44
B	56	60	60	68
C	96	94	90	85
D	42	44	54	46

No optimalidad del algoritmo greedy para asig. tareas

	1	2	3	4
A	48	48	50	44
B	56	60	60	68
C	96	94	90	85
D	42	44	54	46

La estrategia greedy asignaría el equipo 1 al laboratorio D (42), el equipo 4 al laboratorio A (44), el equipo 2 (o el 3) al laboratorio B (60), y el equipo 3 al laboratorio C (90), con un costo de 236.

Pero la estrategia D-2 (44), A-4 (44), B-1 (56), C-3 (90) tiene un costo menor, 234.

Algoritmica

Tema 1. La Eficiencia de los Algoritmos

Tema 2. Algoritmos “Divide y Vencerás”

Tema 3. Algoritmos Voraces (“Greedy”)

Tema 4. Algoritmos para la Exploración de Grafos (“Backtraking”, “Branch and Bound”)

Tema 5. Algoritmos basados en Programación Dinámica

Tema 6. Otras Técnicas Algorítmicas de Resolución de Problemas