DECSAI
Departamento de Ciencias de la Computación e I.A.
Universidad de Granada

Algoritmos Greedy
Análisis y Diseño de Algoritmos

Algoritmos Greedy

- Características generales
- Elementos de un algoritmo greedy
- Esquema de un algoritmo greedy
- Ejemplos
  - Selección de actividades
  - Almacenamiento óptimo en cintas
  - Problema de la mochila fraccional
- Heurísticas greedy
  - Ejemplo: El problema de la mochila
- Aplicaciones

Características generales

- Se utilizan generalmente para resolver problemas de optimización (obtener el máximo o el mínimo).
- Toman decisiones en función de la información que está disponible en cada momento.
- Una vez tomada la decisión, ésta no vuelve a replantearse en el futuro.
- Suelen ser rápidos y fáciles de implementar.
- No siempre garantizan alcanzar la solución óptima.

Características generales

greedy (adj):
avaricioso, voraz, ávido, codicioso, glotón…

¡Cómete siempre todo lo que tengas a mano!

Características generales

NOTA IMPORTANTE

El enfoque "greedy" no nos garantiza obtener soluciones óptimas.

Por lo tanto, siempre habrá que estudiar la **corrección del algoritmo** para demostrar si las soluciones obtenidas son óptimas o no.

Elementos

Para poder resolver un problema usando el enfoque greedy, tendremos que considerar 6 elementos:

1. **Conjunto de candidatos** (elementos seleccionables).
2. **Solución parcial** (candidatos seleccionados).
3. **Función de selección** (determina el mejor candidato del conjunto de candidatos seleccionables).
4. **Función de factibilidad** (determina si es posible completar la solución parcial para alcanzar una solución del problema).
5. **Criterio que define lo que es una solución** (indica si la solución parcial obtenida resuelve el problema).
6. **Función objetivo** (valor de la solución alcanzada).

Esquema general

- Se parte de un conjunto vacío: S = ∅.
- De la lista de candidatos, se elige el mejor (de acuerdo con la función de selección).
- Comprobamos si se puede llegar a una solución con el candidato seleccionado (función de factibilidad). Si no es así, lo eliminamos de la lista de candidatos posibles y nunca más lo consideraremos.
- Si aún no hemos llegado a una solución, seleccionamos otro candidato y repetimos el proceso hasta llegar a una solución [o quedarnos sin posibles candidatos].

Esquema general

```
Greedy (conjunto de candidatos C): solución S

S = ∅
while (S no sea una solución y C ≠ ∅) {
      x = selección(C)
      C = C - {x}
      if (S∪{x} es factible)
         S = S∪{x}
}

if (S es una solución)
   return S;
else
   return "No se encontró una solución";
```

Ejemplo
Selección de actividades

Tenemos que elegir de entre un conjunto de actividades:

- Para cada actividad, conocemos su hora de comienzo y su hora de finalización.
- Podemos asistir a todas las actividades que queramos.
- Sin embargo, hay actividades que se solapan en el tiempo y no podemos estar en dos sitios a la vez.

Posibles objetivos

1. Asistir al mayor número de actividades posible.
2. Minimizar el tiempo que estamos ociosos.

Ejemplo
Selección de actividades

Problema de selección de actividades

Dado un conjunto C de n actividades, con

s_i = tiempo de comienzo de la actividad i

f_i = tiempo de finalización de la actividad i

encontrar el subconjunto S de actividades compatibles de tamaño máximo (esto es, actividades que no se solapen en el tiempo).

Ejemplo
Selección de actividades

Elementos del problema

- Conjunto de candidatos: C = {actividades ofertadas}.
- Solución parcial: S (inicialmente, S=∅).
- Función de selección: menor duración, menor solapamiento, terminación más temprana…
- Función de factibilidad: x es factible si es compatible (esto es, no se solapa) con las actividades de S.
- Criterio que define lo que es una solución: C=∅.
- Función objetivo (lo que tratamos de optimizar): El tamaño de S.

Ejemplo
Selección de actividades

Estrategias greedy alternativas

Orden en el que se pueden considerar las actividades:

- Orden creciente de hora de comienzo.
- Orden creciente de hora de finalización.
- Orden creciente de duración: $f_i - s_i$
- Orden creciente de conflictos con otras actividades (con cuántas actividades se solapa).

Ejemplo
Selección de actividades

Estrategias greedy alternativas

Contraejemplos (para descartar alternativas)

- Orden creciente de hora de comienzo.

- Orden creciente de duración: $f_i - s_i$

- Orden creciente de conflictos con otras actividades

Ejemplo
Selección de actividades

Algoritmo Greedy

```
SelecciónActividades (C: actividades): S
```

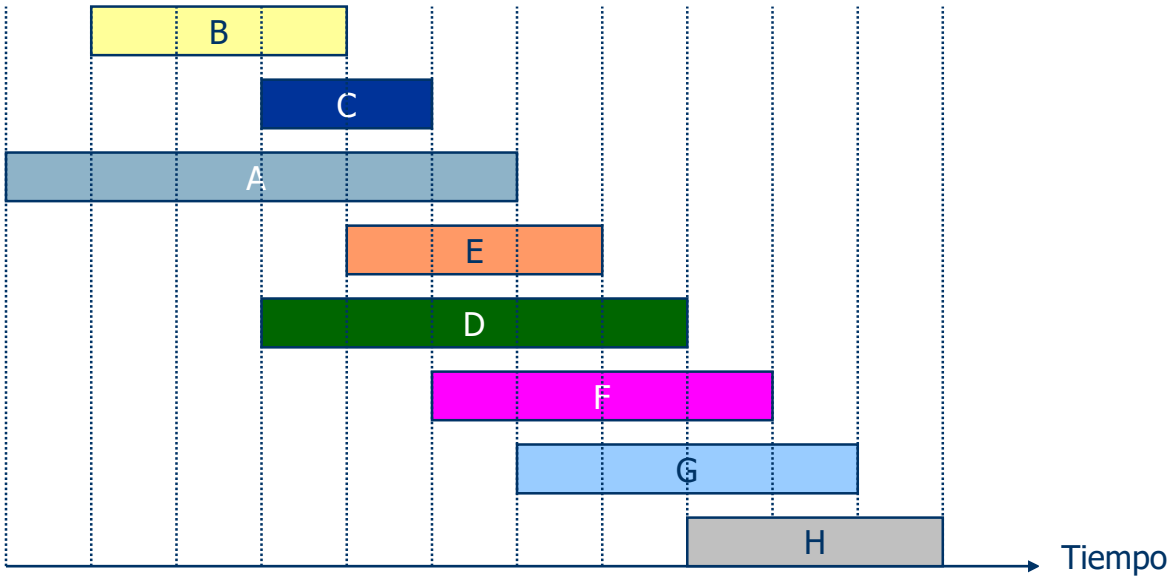
Ordenar C en orden creciente de tiempo de finalización.
Seleccionar la primera actividad de C
(esto es, extraerla del conjunto C y añadirla a S).
Repetir
    Extraer la siguiente actividad del conjunto ordenado:
    Si comienza después de que la actividad previa en S haya terminado, seleccionarla (añadirla a S).
hasta que C esté vacío.

Ejemplo
Selección de actividades

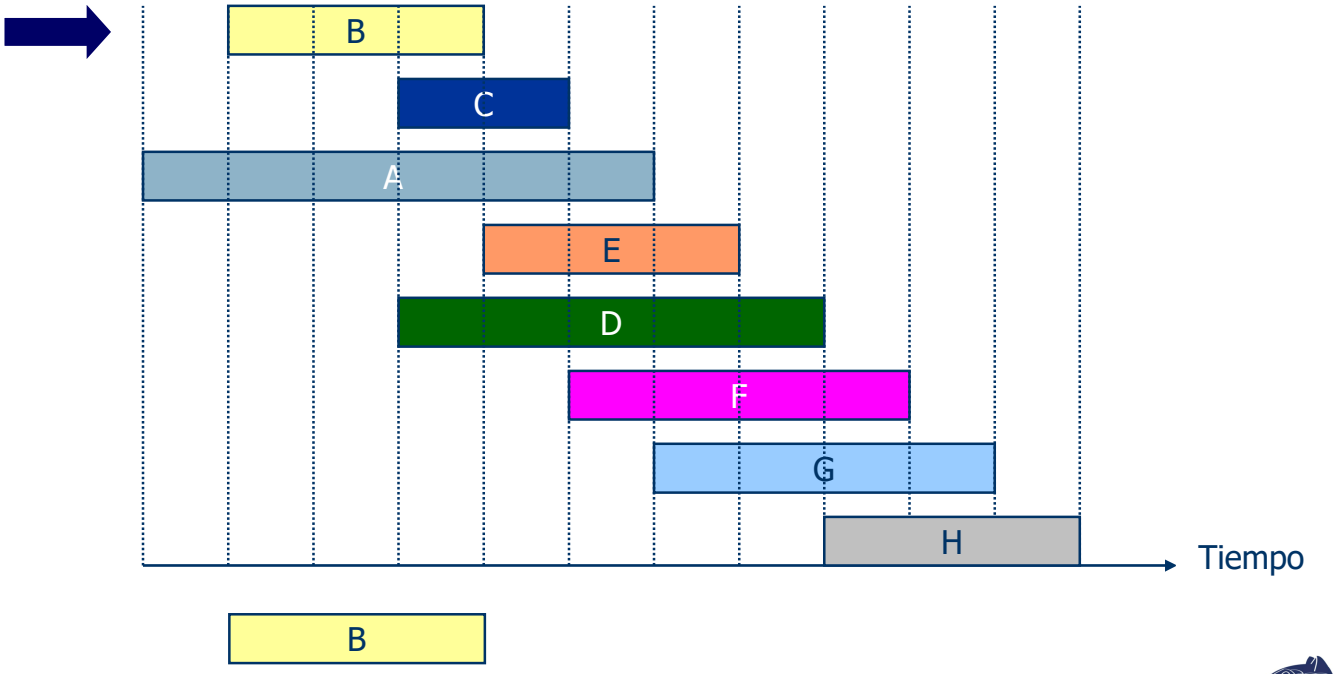
Algoritmo Greedy

```
SelecciónActividades (C: actividades): S
{
  sort(C);      // ordenar según tiempo de finalización
  S[0] = C[0]; // seleccionar la primera actividad
  i = 1; prev = 0;
  while (i<C.length) {                    // ¿solución(S)?
        x = C[i];                         // seleccionar x
        if (x.inicio >= S[prev].fin)      // ¿factible(x)?
           S[++prev] = x;                 // añadir x a S
        i++;
  }
}
```

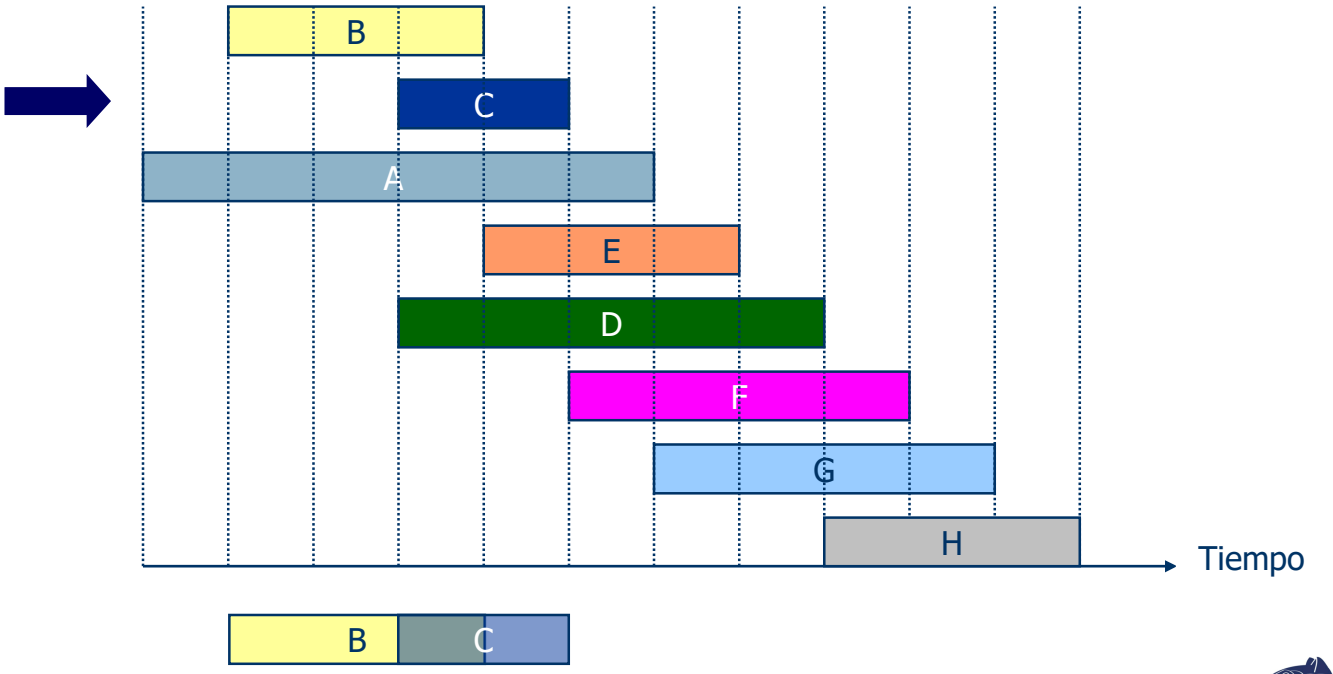
Ejemplo
Selección de actividades



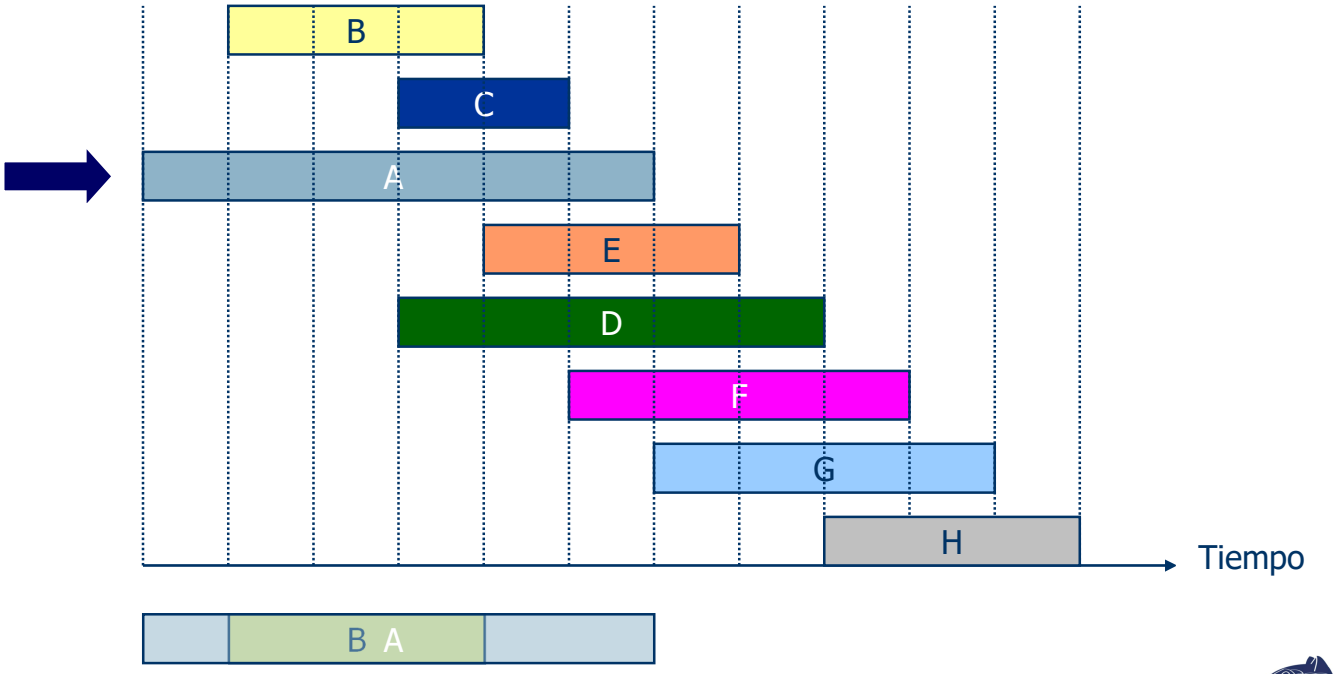
Ejemplo
Selección de actividades



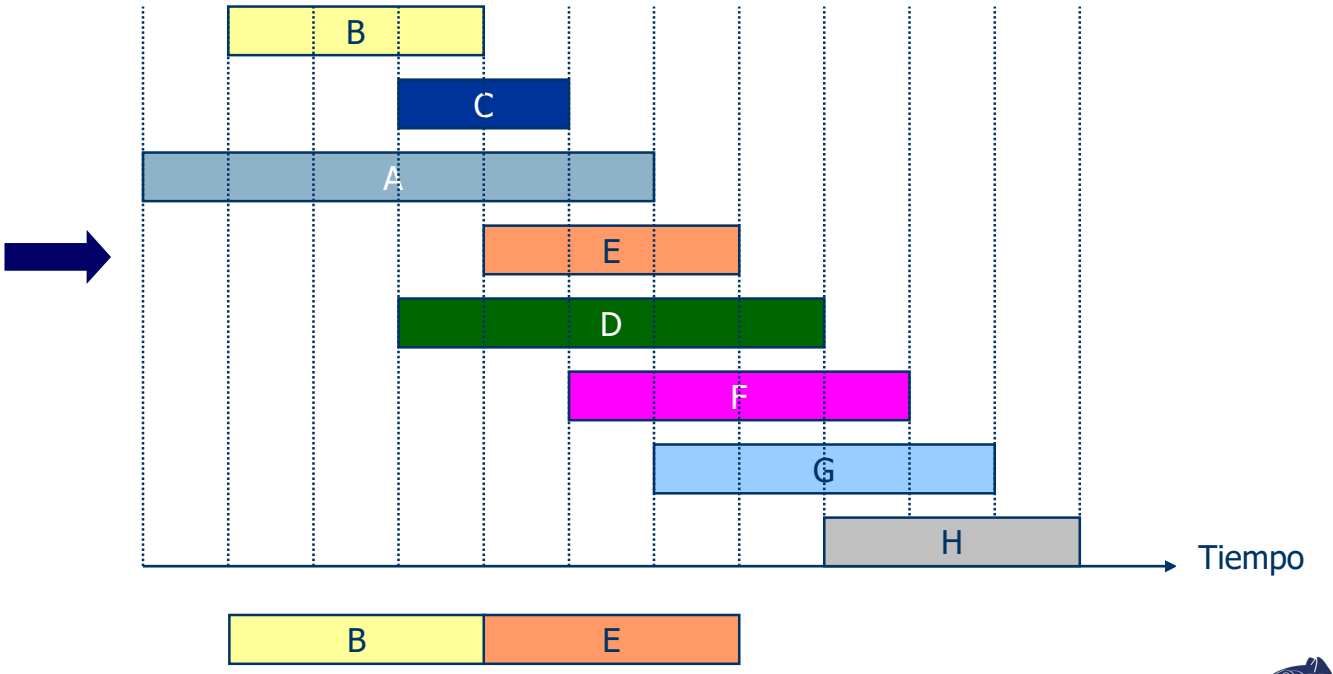
Ejemplo
Selección de actividades



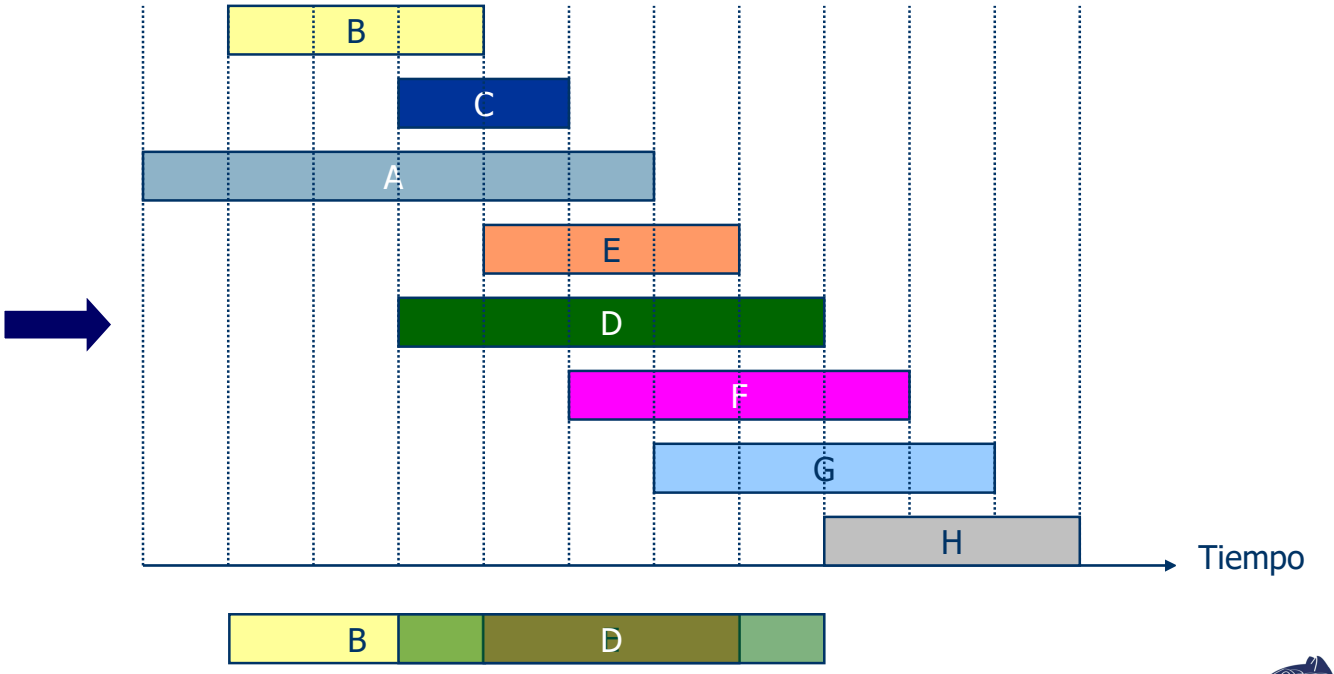
Ejemplo
Selección de actividades



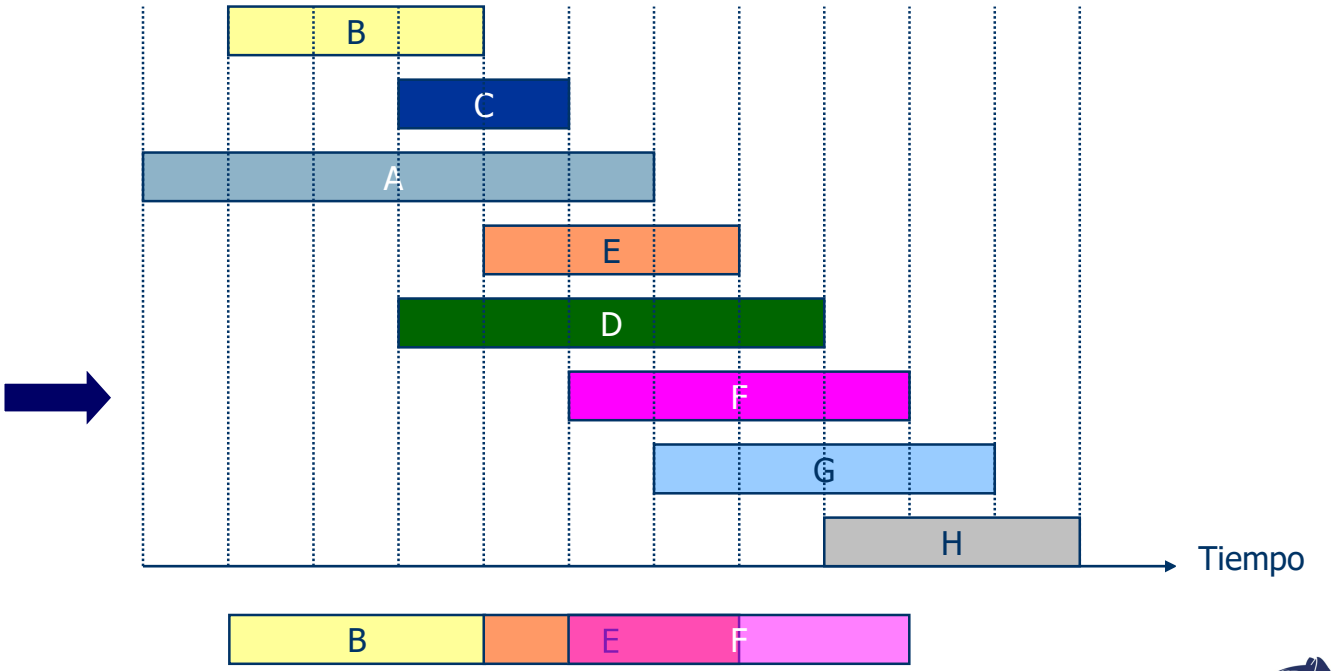
Ejemplo
Selección de actividades



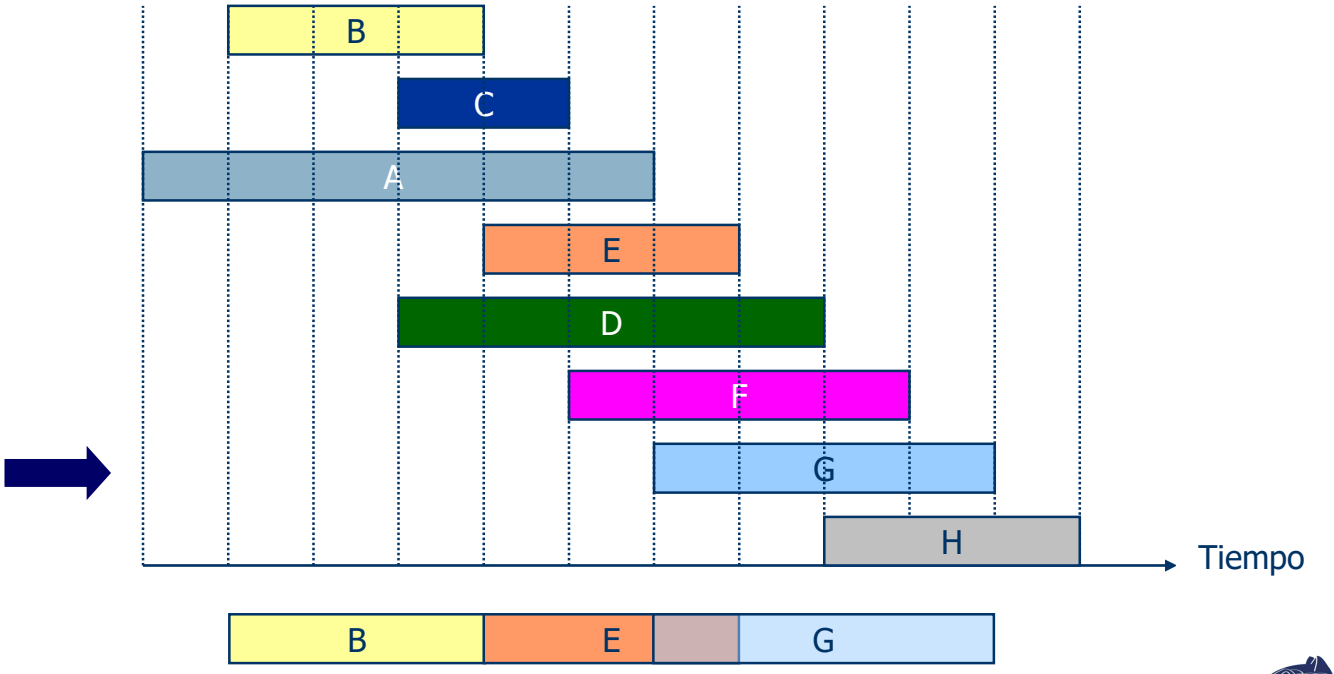
Ejemplo
Selección de actividades



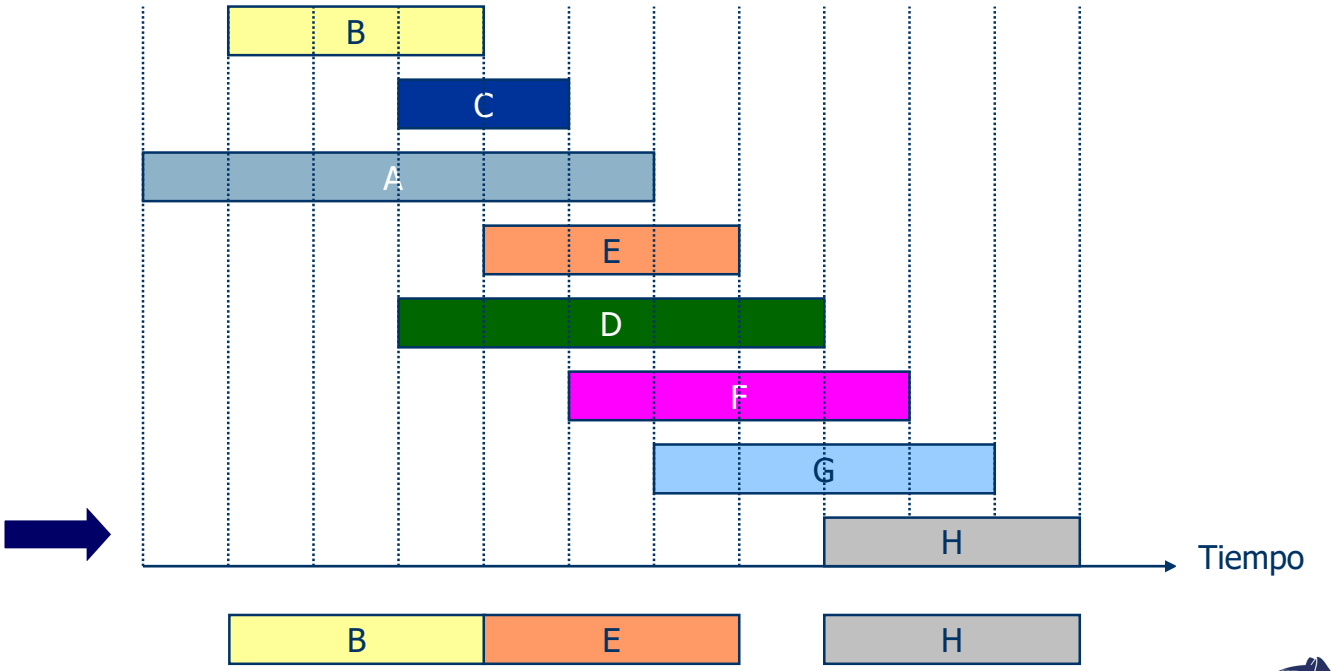
Ejemplo
Selección de actividades



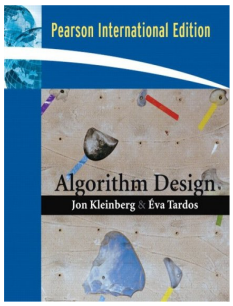
Ejemplo
Selección de actividades



Ejemplo
Selección de actividades



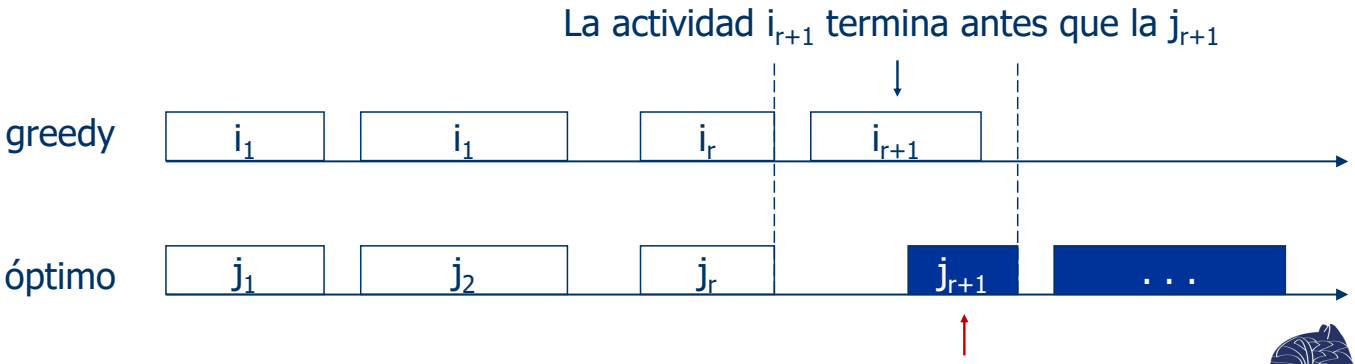
Ejemplo
Selección de actividades



Demostración de optimalidad

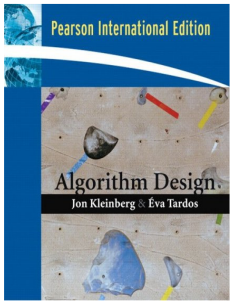
Por reducción al absurdo:

Suponemos que el algoritmo no calcula la solución óptima...



¿Por qué no reemplazar la actividad j_{r+1} con i_{r+1}?

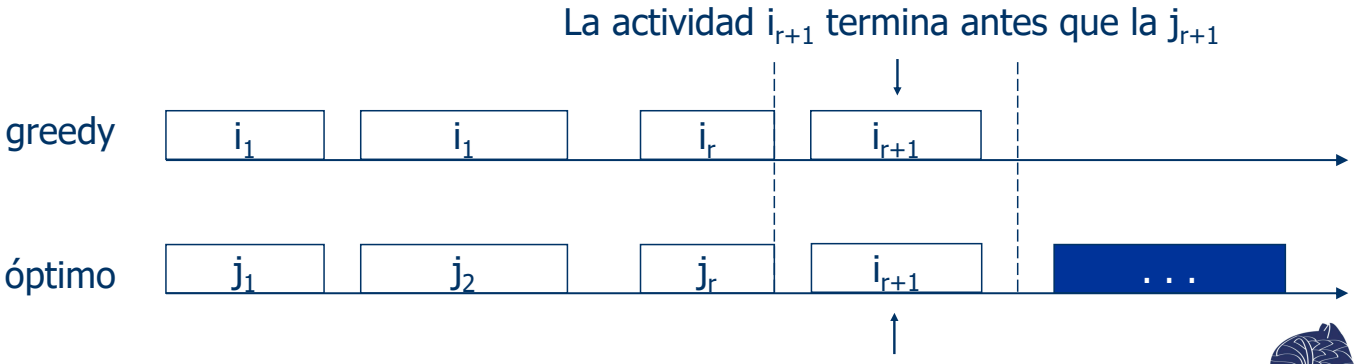
Ejemplo
Selección de actividades



Demostración de optimalidad

Por reducción al absurdo:

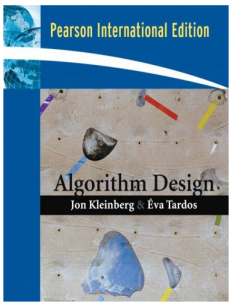
Suponemos que el algoritmo no calcula la solución óptima...

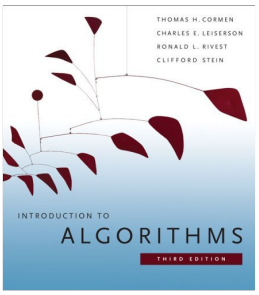


Pero entonces, la solución parcial del algoritmo greedy sigue siendo factible y, además, es óptima

Ejemplo
Selección de actividades

Demostración de optimalidad





- Jon Kleinberg & Eva Tardos: **Algorithm Design.** Sección 4.1 "Interval Scheduling: The greedy algorithm stays ahead".
- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest & Clifford Stein: **Introduction to Algorithms**. [2ª edición] Sección 16.1 "An activity-selection problem".

Ejemplo
Almacenamiento en cintas

Almacenamiento óptimo en cintas

- Tenemos n ficheros que hay que almacenar en una cinta de longitud L.
- Cada fichero i tiene una longitud l_i , $1 \le i \le n$
- Todos los ficheros se recuperan del mismo modo, siendo el tiempo medio de recuperación (TMR),

$$TMR = \frac{1}{n} \sum_{1 \le j \le n} t_j \qquad t_j = \sum_{1 \le k \le j} l_{i_k}$$

- Nos piden que encontremos una permutación de los n ficheros tal que, cuando estén almacenados en la cinta el TMR sea mínimo, lo que equivalente a minimizar

$$D(I) = \sum_{1 \le j \le n} \sum_{1 \le k \le j} l_{i_k}$$

Ejemplo
Almacenamiento en cintas

Almacenamiento óptimo en cintas

Ejemplo

n = 3

$(l_1,l_2,l_3) = (5,10,3)$

Orden **I**	Valor **D(I)**	
1,2,3	5 + 5 + 10 + 5 + 10 + 3	= 38
1,3,2	5 + 5 + 3 + 5 + 3 + 10	= 31
2,1,3	10 + 10 + 5 + 10 + 5 + 3	= 43
2,3,1	10 + 10 + 3 + 10 + 3 + 5	= 41
3,1,2	3 + 3 + 5 + 3 + 5 + 10	= 29
3,2,1	3 + 3 + 10 + 3 + 10 + 5	= 34

Ejemplo
Almacenamiento en cintas

Almacenamiento óptimo en cintas

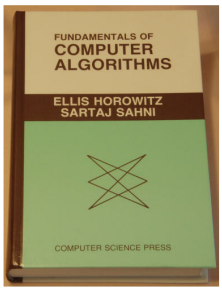
Algoritmo greedy

```
Partiendo de la cinta vacía…

Mientras queden ficheros por guardar
   Elegir el fichero más pequeño
   Añadirlo a continuación en la cinta
```

- El algoritmo escoge la opción más inmediata sin tener en cuenta si esa decisión será la mejor a largo plazo...

Ejemplo
Almacenamiento en cintas



Almacenamiento óptimo en cintas

Teorema

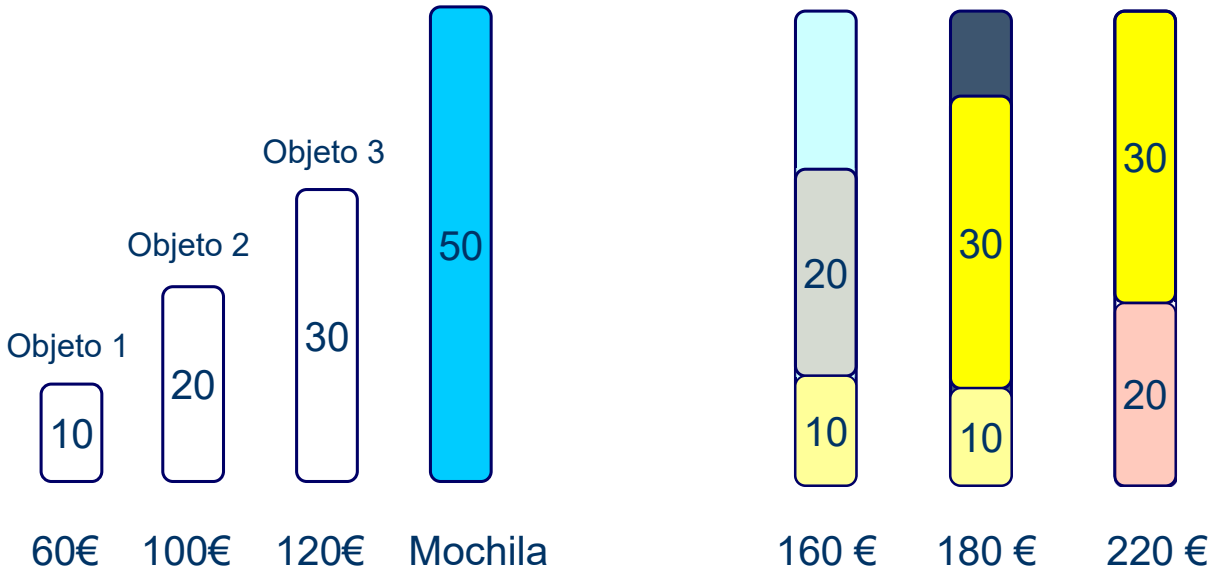
Si $l_1 \leq l_2 \leq \ldots \leq l_n$,
entonces el orden de colocación $i_j = j$, $1 \leq j \leq n$
minimiza

$$\sum_{k=1}^{n}\sum_{j=1}^{k} l_{i_j}$$

para todas las posibles permutaciones i_j

Demostración: Ellis Horowitz & Sartaj Sahni: **Fundamentals of Computer Algorithms**, 1978

Ejemplo
Problema de la mochila 0/1



¿Cómo seleccionamos los objetos de la mochila? NP

Ejemplo
Problema de la mochila fraccional

El problema consiste en llenar una mochila:

- La mochila puede soportar como máximo un peso P.
- Tenemos n objetos **fraccionables**.
- Cada objeto i tiene un peso p_i y proporciona un beneficio b_i

Objetivo:

Maximizar el beneficio de los objetos transportados.

$$\max \sum_{1 \le i \le n} x_i b_i \quad sujeto\ a \sum_{1 \le i \le n} x_i p_i \le P$$

Ejemplo
Problema de la mochila fraccional

Ejemplo

Mochila de 100kg

Beneficio (€)	20	30	65	40	60
Peso (kg)	10	20	30	40	50

¿Cómo seleccionamos los objetos?

- Primero el más ligero
  - Peso = 10+20+30+40 = 100 kg
  - Beneficio = 20+30+65+40 = 155 €
- Primero el más valioso
  - Peso = 30 + 50 + 20 = 100 kg
  - Beneficio = 65 + 60 + 20 = 145 €
- Primero el que tenga más valor por unidad de peso
  - Peso = 30 + 10 + 20 + 40 = 100 kg
  - Beneficio = 65 + 20 + 30 + 48 = 163 €

Ejemplo
Problema de la mochila fraccional

- Definimos la densidad del objeto O_i como b_i/p_i.
- **Algoritmo greedy:** Seleccionamos los objetos en orden decreciente de densidad.

$$b_i/p_i \geq b_{i+1}/p_{i+1} \text{ para } 1 \leq i < n$$

- Se añade a la mochila todo lo que se pueda:
  Si un objeto O_i no cabe entero, se rellena el espacio disponible con la fracción del objeto que quepa hasta completar la capacidad de la mochila.

Heurísticas greedy

Hay situaciones en las cuales no podemos encontrar un algoritmo greedy que proporcione una solución óptima…

En muchas ocasiones, se podrían obtener mejores soluciones reconsiderando alternativas desechadas por un algoritmo greedy (cuando, a partir de una solución óptima local no se puede alcanzar una solución óptima global).

Pese a ello, resultan útiles los algoritmos greedy que proporcionan una solución rápida a problemas complejos, aunque ésta no sea óptima.

Heurísticas greedy

Heurística: Procedimiento que proporciona una solución aceptable a un problema mediante métodos que carecen de justificación formal (p.ej. por tanteo, usando reglas empíricas…).

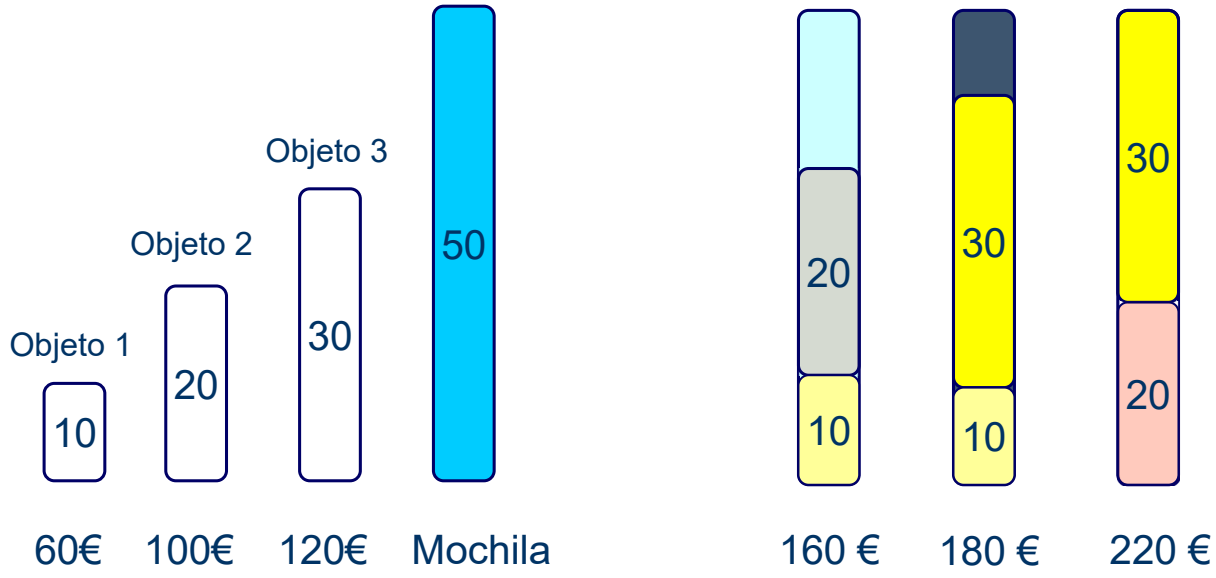
- Heurísticas para problemas concretos (p.ej. NP)
- Metaheurísticas (heurísticas de propósito general):
  - Enfriamiento simulado
  - Búsqueda tabú
  - GRASP [Greedy Randomized Adaptive Search Procedures]
  - Algoritmos bioinspirados: algoritmos genéticos, algoritmos meméticos, colonias de hormigas…

Heurísticas greedy

Satisfacer vs. optimizar

- Cuando el tiempo que se tarda en resolver un problema es un factor clave, un algoritmo greedy puede utilizarse como criterio heurístico.
- También puede utilizarse un algoritmo greedy para encontrar una primera solución (como punto de partida para otra heurística).

Ejemplo
Problema de la mochila 0/1



¿Cómo seleccionamos los objetos de la mochila?

Ejemplo
Problema de la mochila 0/1

Un algoritmo greedy proporciona la solución óptima del problema de la mochila cuando se pueden fraccionar los objetos. Sin embargo, el problema es más difícil de resolver cuando no se pueden fraccionar los objetos…

$$\max \sum_{1 \le i \le n} x_i b_i \quad sujeto\ a \sum_{1 \le i \le n} x_i p_i \le P \quad con\ x_i \in \{0,1\}$$

Ejemplo
Problema de la mochila 0/1

Heurística greedy

Ordenar los objetos por densidad no creciente:

$b_i/p_i \geq b_{i+1}/p_{i+1}$ para $1 \leq i < n$

Con un número ilimitado de objetos de cada tipo, si M es el valor máximo de los objetos que se pueden llevar en la mochila, la heurística garantiza obtener, al menos, un valor de M/2 (George Dantzig, 1957).

Sin embargo, la heurística puede proporcionar resultados mucho peores (muy alejados del óptimo) para instancias particulares del problema.

Aplicaciones

- Planificación de tareas.
  - Minimización del tiempo de espera = Almacenamiento en cintas.
  - Planificación de tareas a plazo fijo = Selección de actividades.
- Cajero (devolver un número mínimo de monedas/billetes [pero no sellos]).
- Caminos mínimos en grafos (algoritmo de Dijkstra).
- Árbol generador minimal (algoritmos de Prim & Kruskal).
- Códigos Huffman y compresión de datos.

Heurísticas greedy

- Construcción de árboles de decisión (IA/ML).
- Estrategias de inversión basadas en momentos.