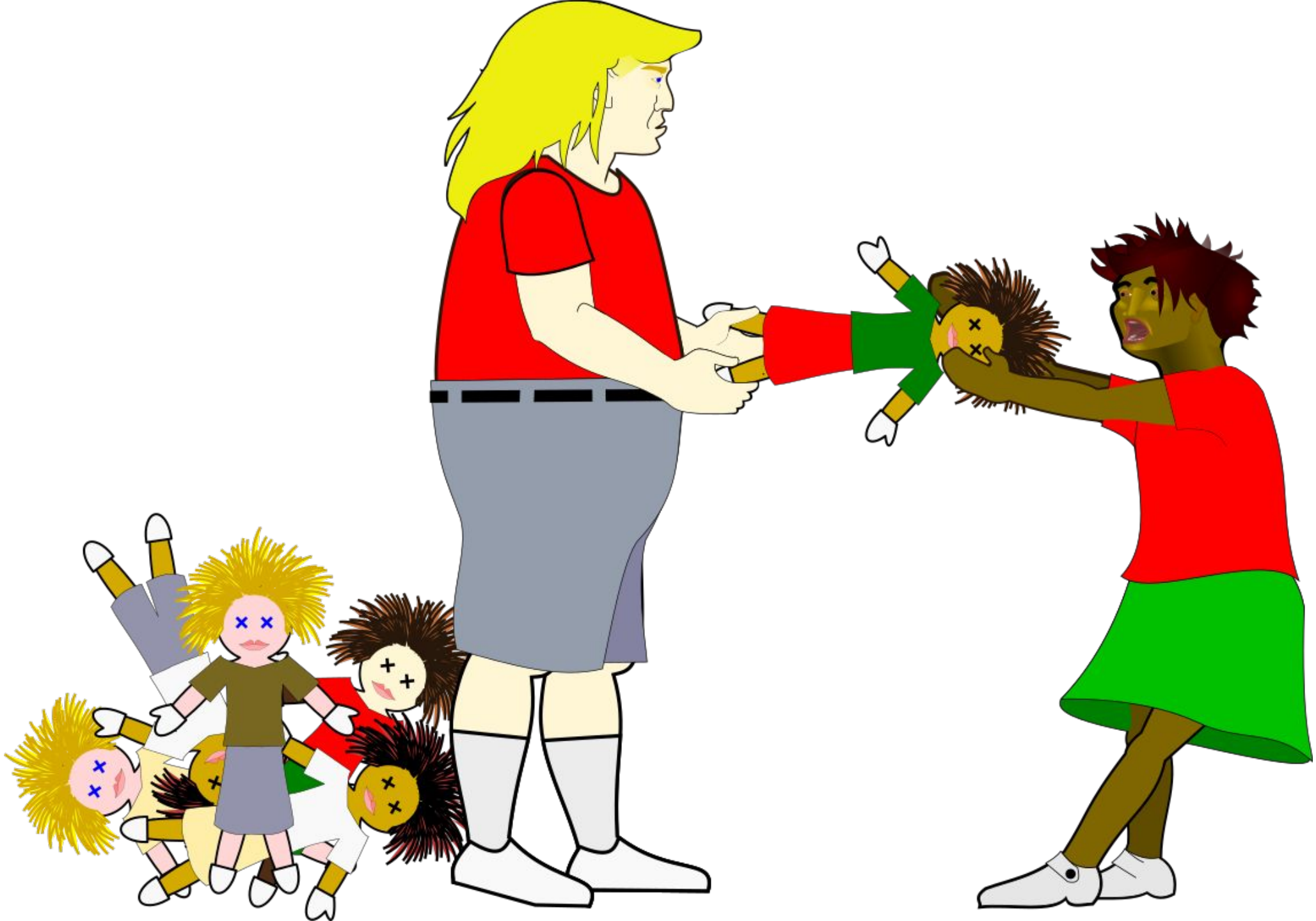
Algoritmos Greedy

MARKS CALDERÓN NIQUIN

UNIVERSIDAD ESAN



Algoritmos Greedy

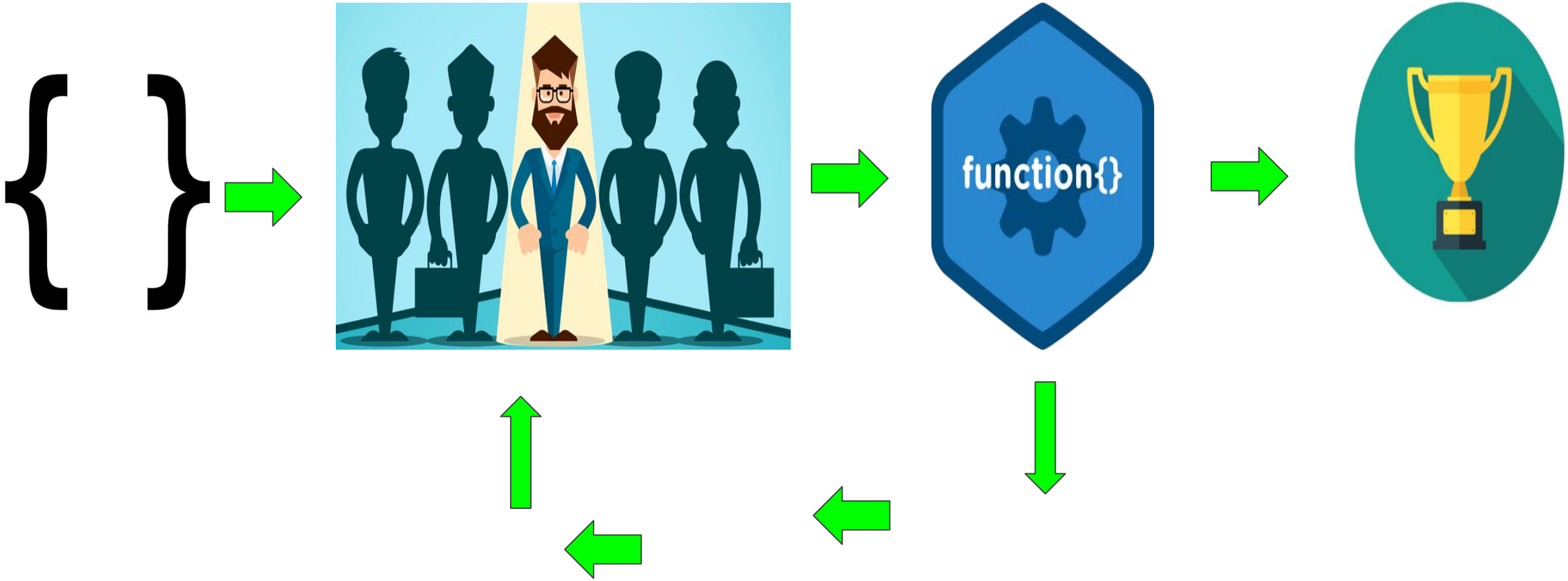
- Escoge lo que le parece más adecuado en cada momento
- Usado en problemas de optimización
- No siempre dan soluciones óptimas
- Una vez tomada la decisión, ésta no vuelve a replantearse en el futuro.
- Suelen ser rápidos y fáciles de implementar

Características generales

Greedy(adj):

- Avaricioso, voraz, ávido, codicioso, glotón

Esquema general



Esquema general

- Se parte de un conjunto vacio: $S = \emptyset$
- 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**). Caso contrario, 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

$S = \emptyset$

Mientras (S no es una solución y $C \neq \emptyset$){

$X = selección(C)$

$C = C - \{X\}$

Si ($S \cup \{X\}$ es factible)

$S = S \cup \{X\}$

}

Si (S es una solución)

- Retornar S;

Sino

Retornar "no existe una solución"

Ejemplo: selección de actividades

Tenemos que elegir de entre un conjunto de actividades:

- Para cada actividad, conocemos su hora de inicio y su hora de fin
- 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

- Asistir al mayor número de actividades posible.
- 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, $S = \emptyset$)
- 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= $\emptyset$
- 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

Algoritmo Greedy

SelecciónActividades (C: actividad)

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

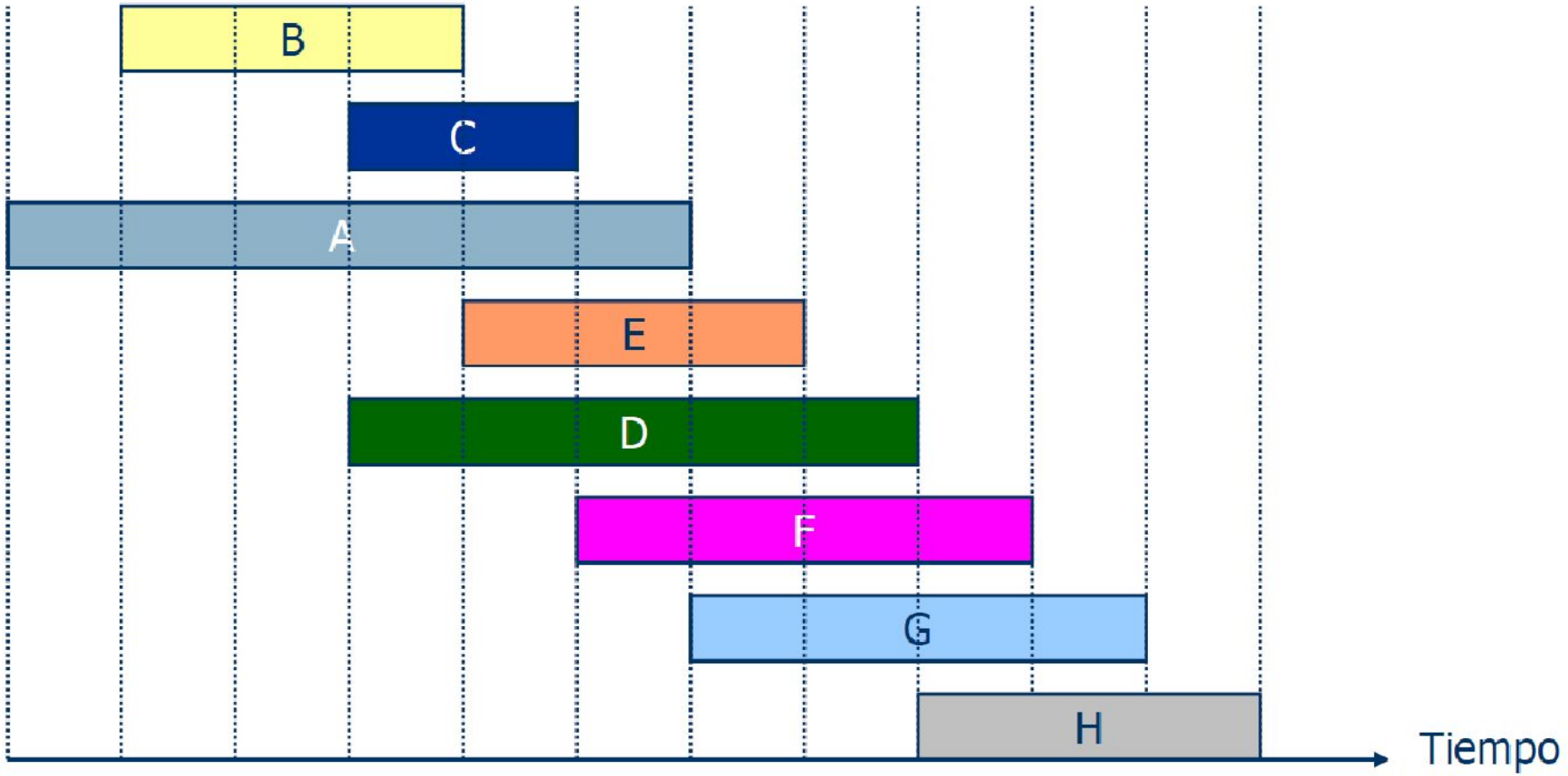
- Extraer la siguiente actividad del conjunto ordenado:
- Si comienza después de que la actividad previa en S haya terminado, seleccionarla (añadirla a S)

Hasta que C esté vacío

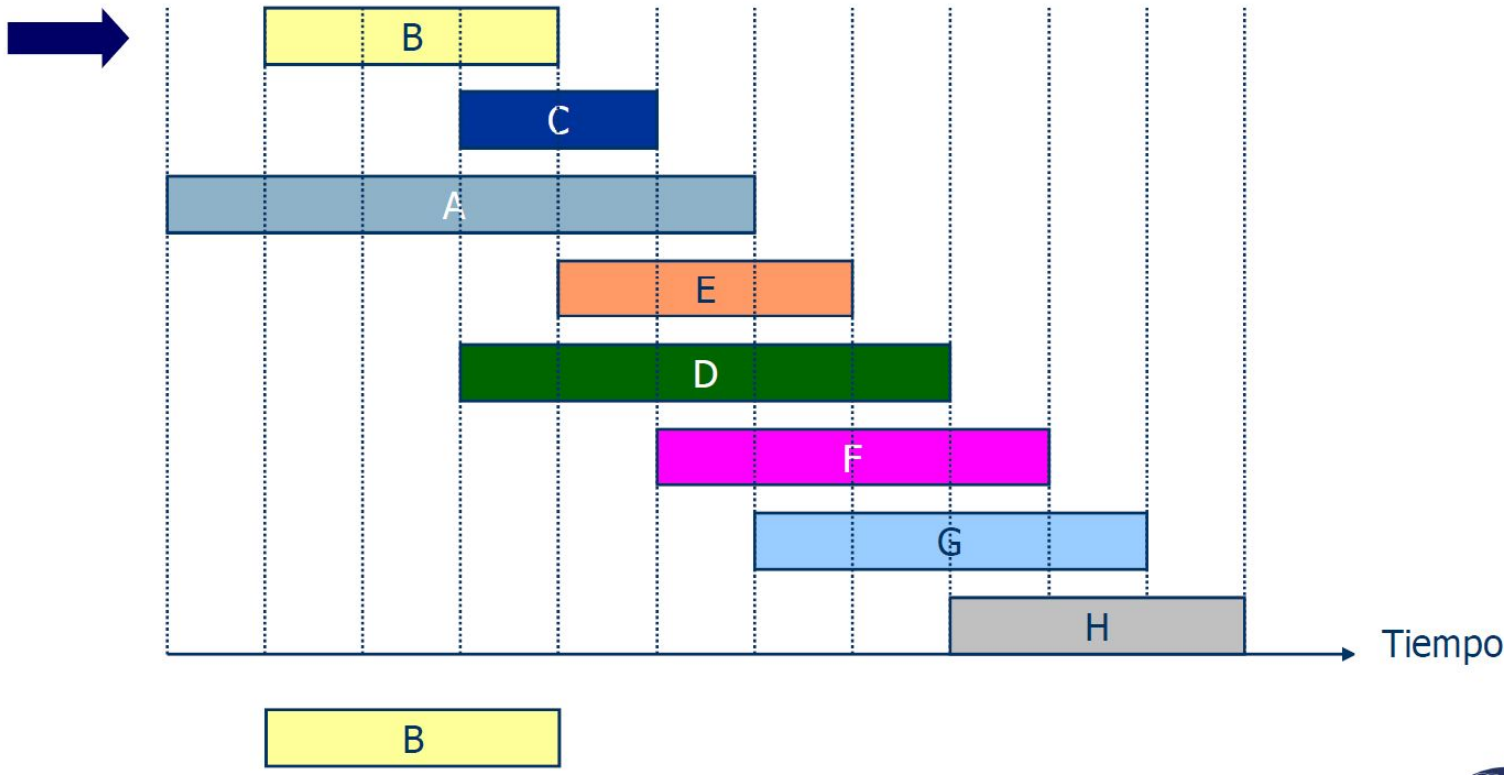
Algoritmo Greedy

```
SelecciónActividades (C: actividad): 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

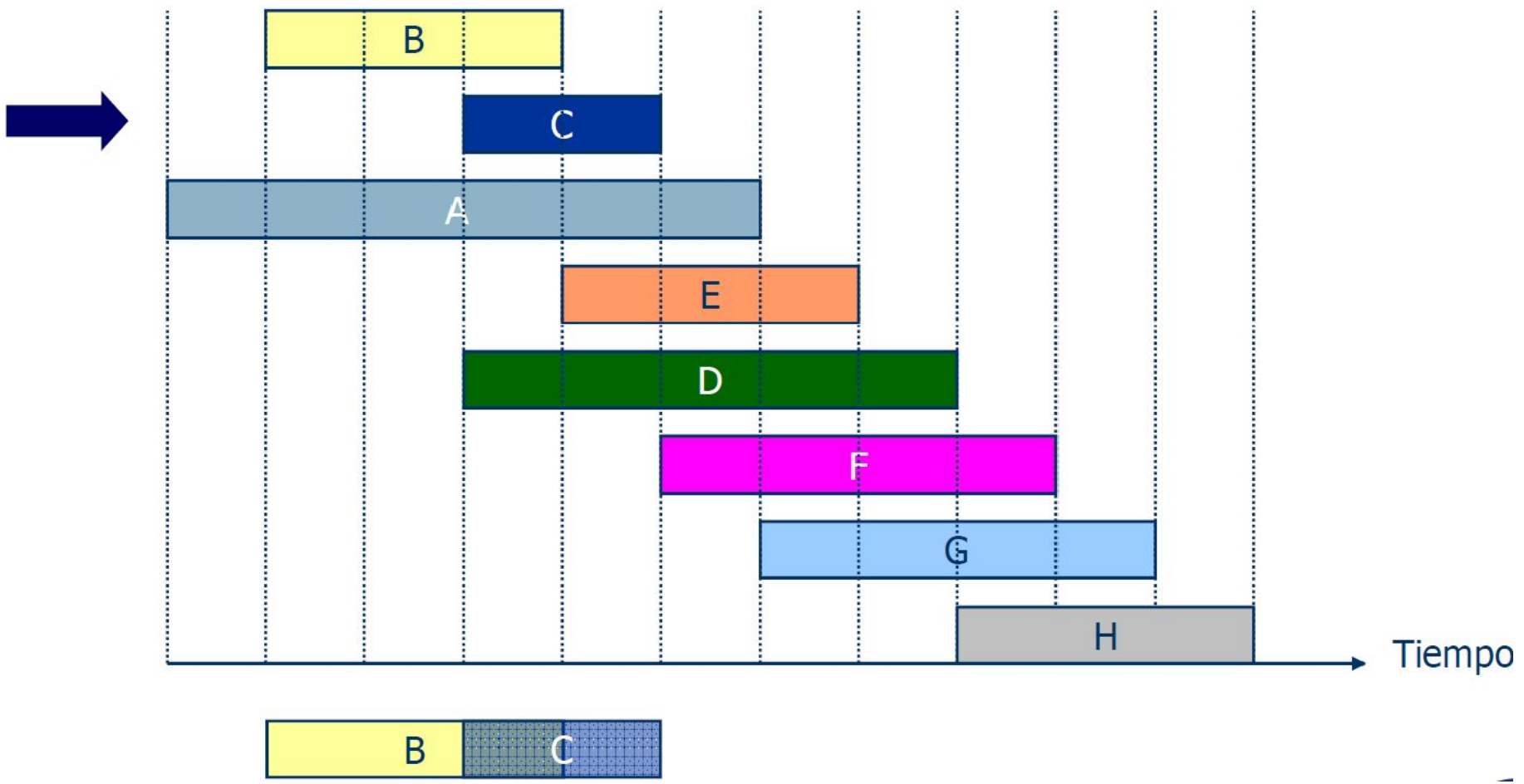


Ejemplo: Selección de actividades



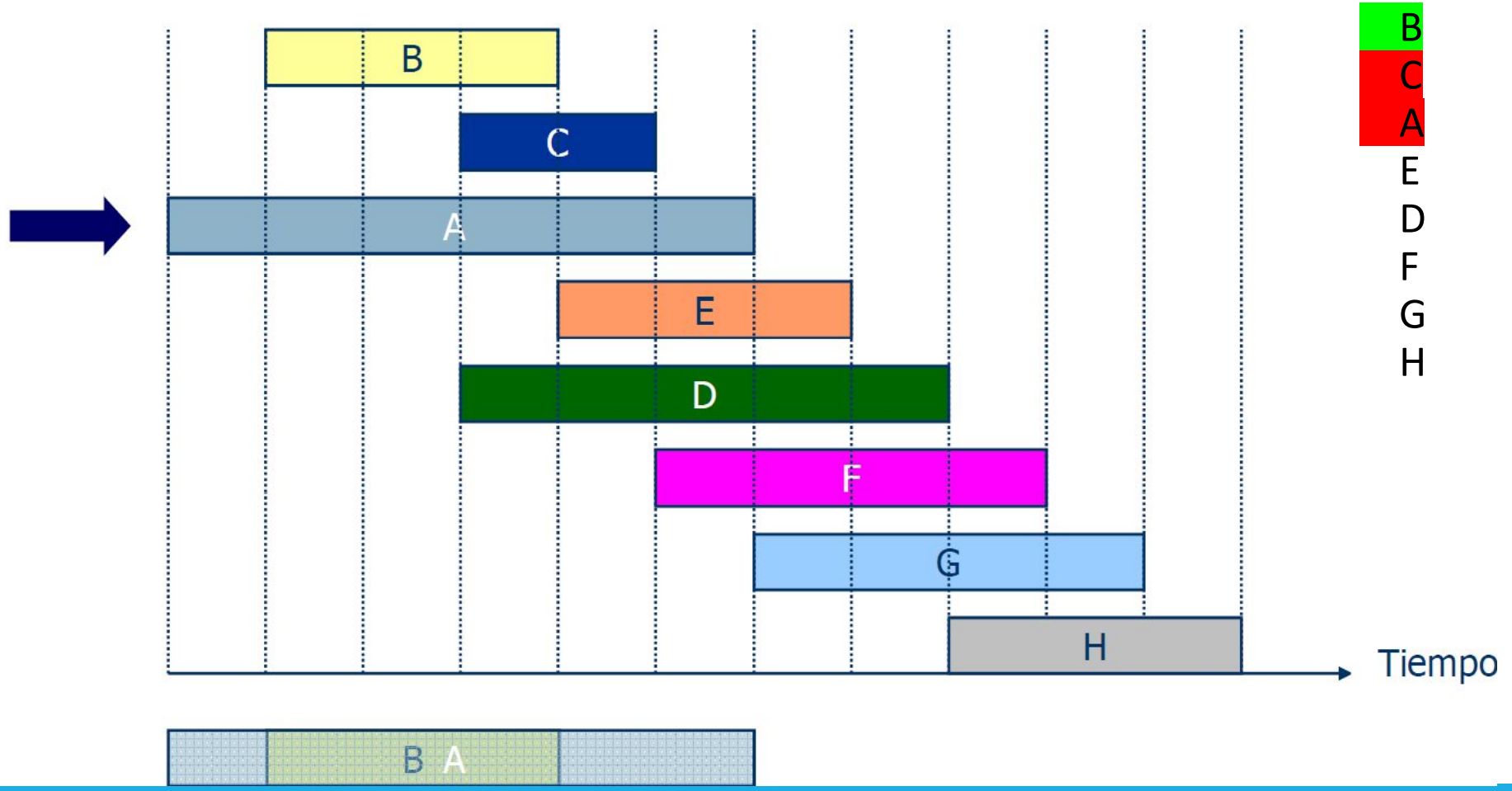
B
C
A
E
D
F
G
H

Ejemplo: Selección de actividades

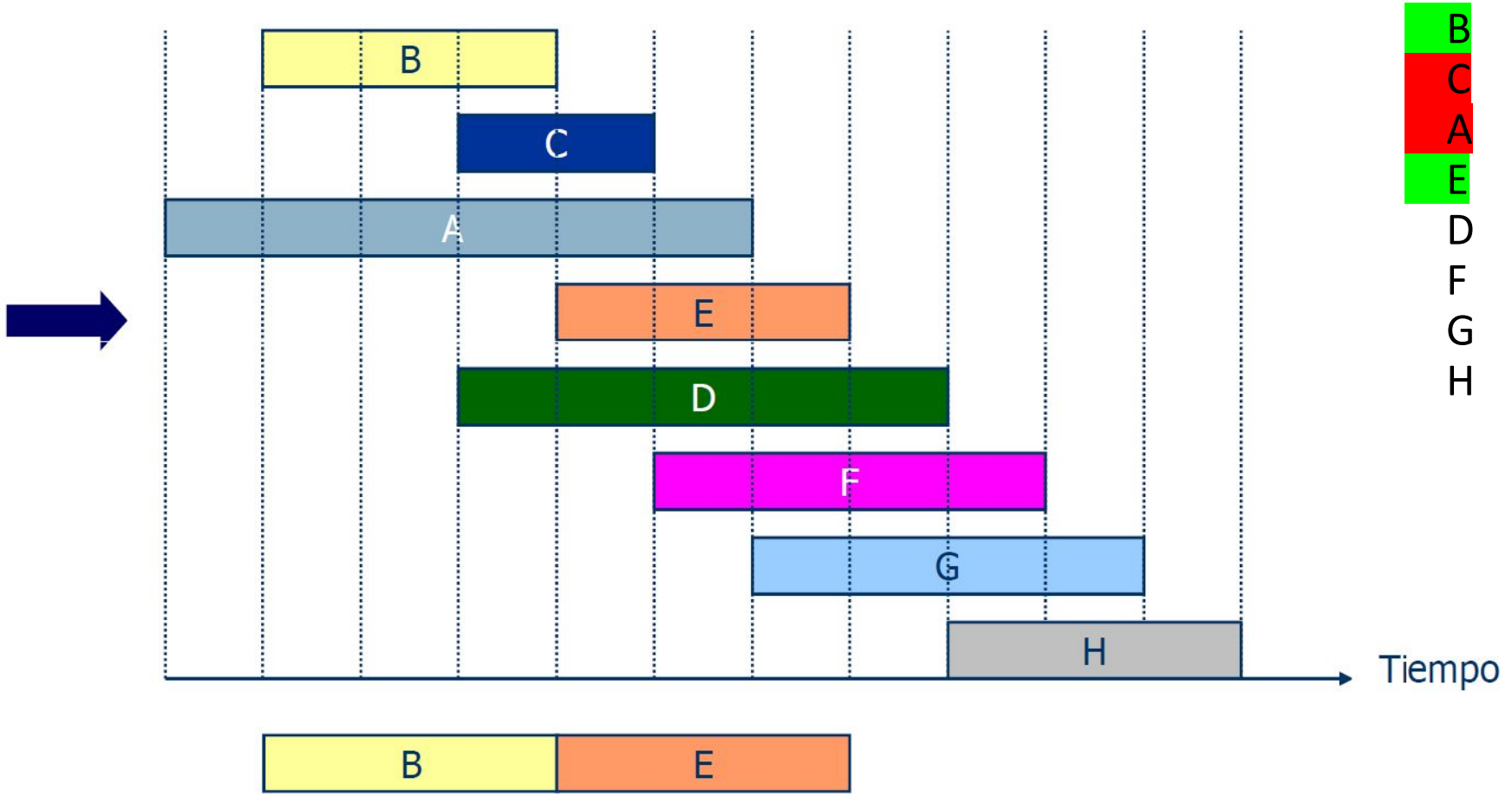


B
C
A
E
D
F
G
H

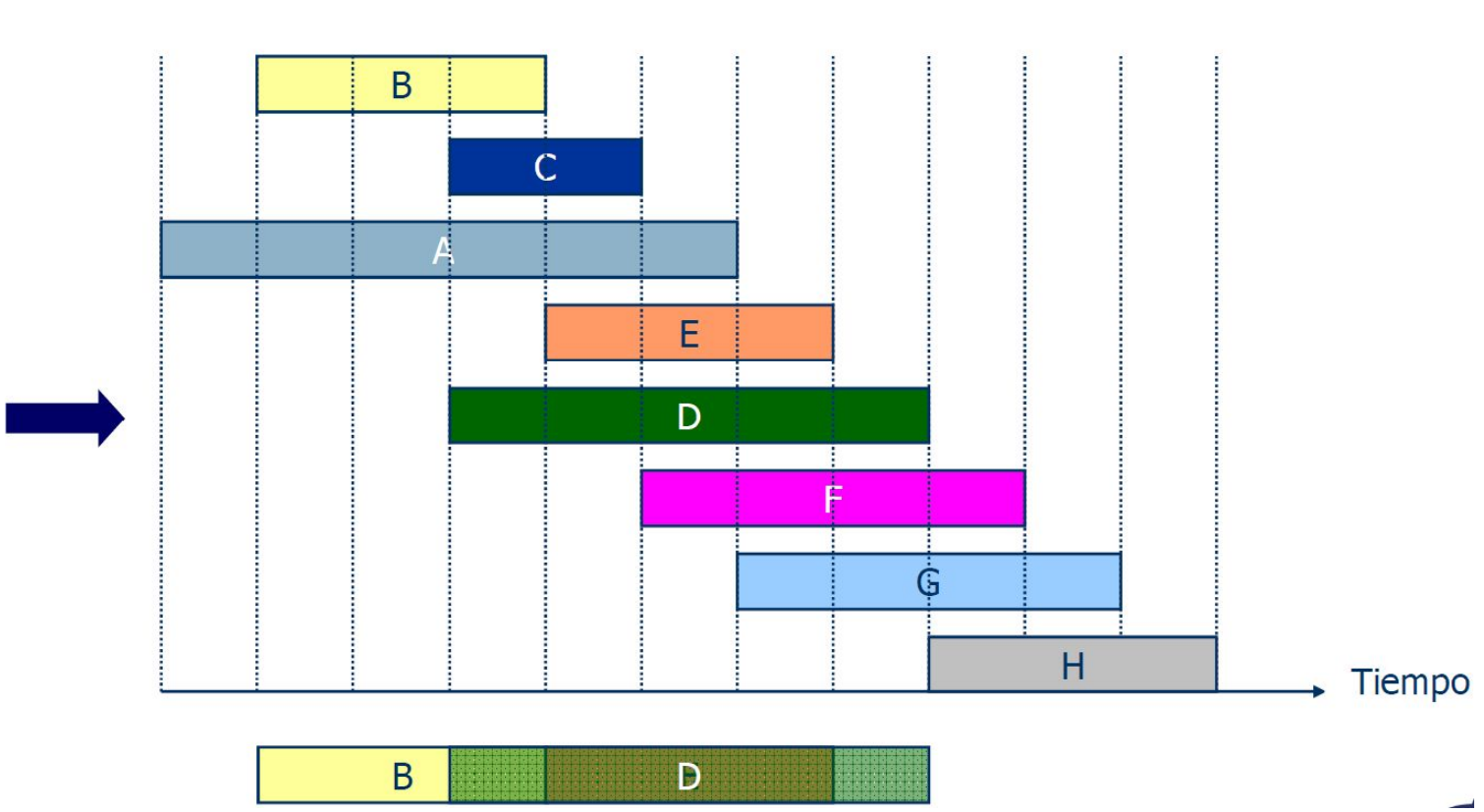
Ejemplo: Selección de actividades



Ejemplo: Selección de actividades

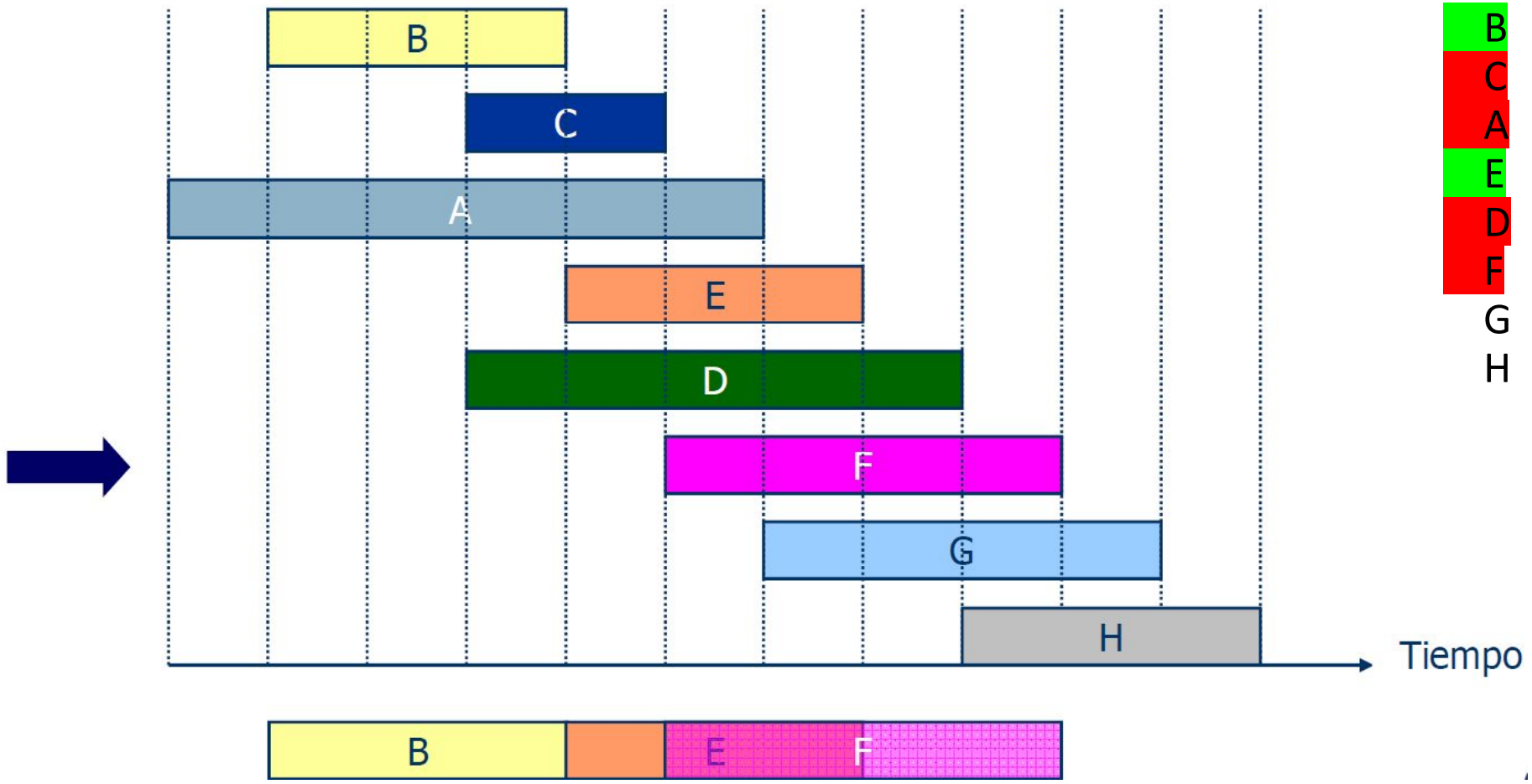


Ejemplo: Selección de actividades

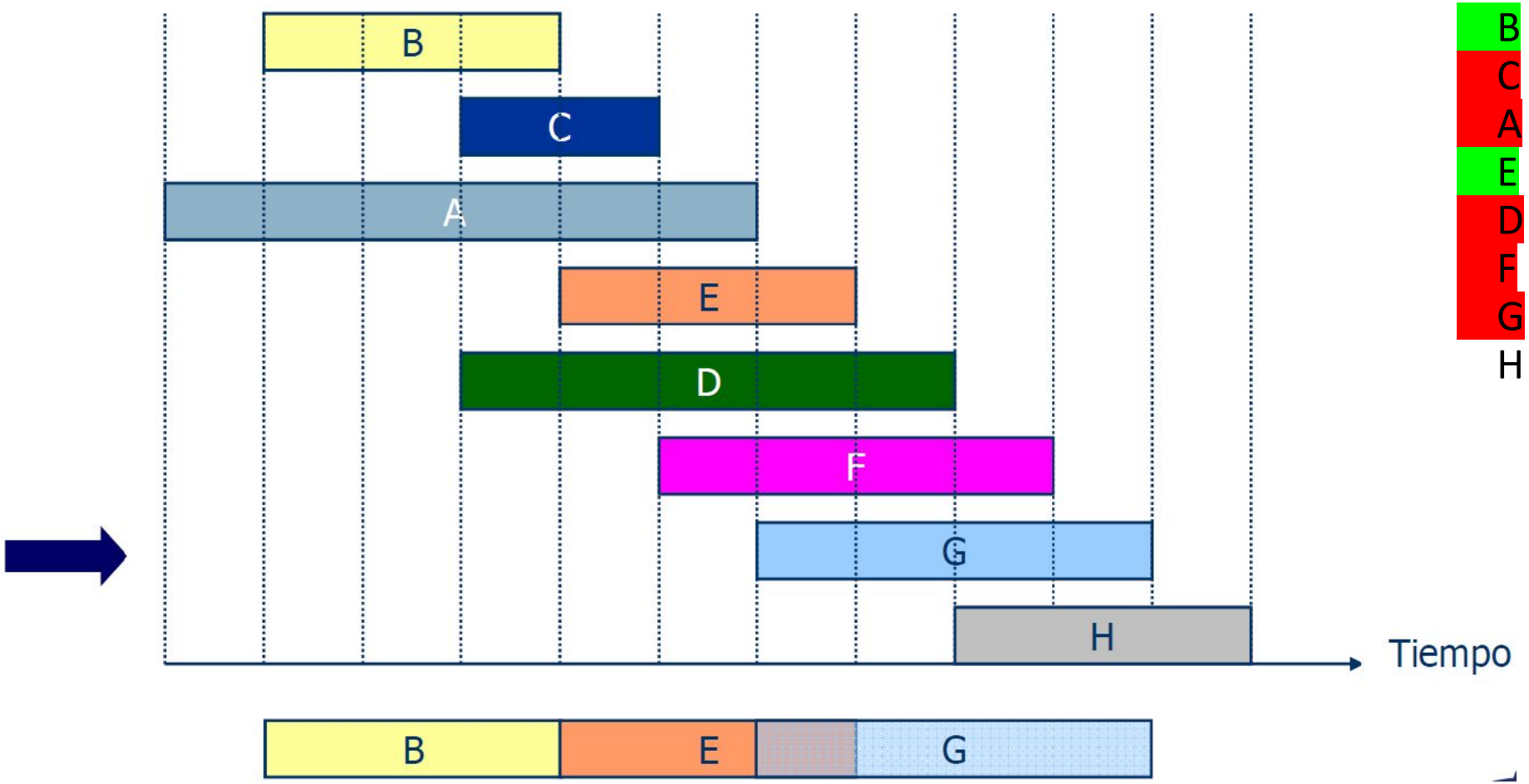


B
C
A
E
D
F
G
H

Ejemplo: Selección de actividades

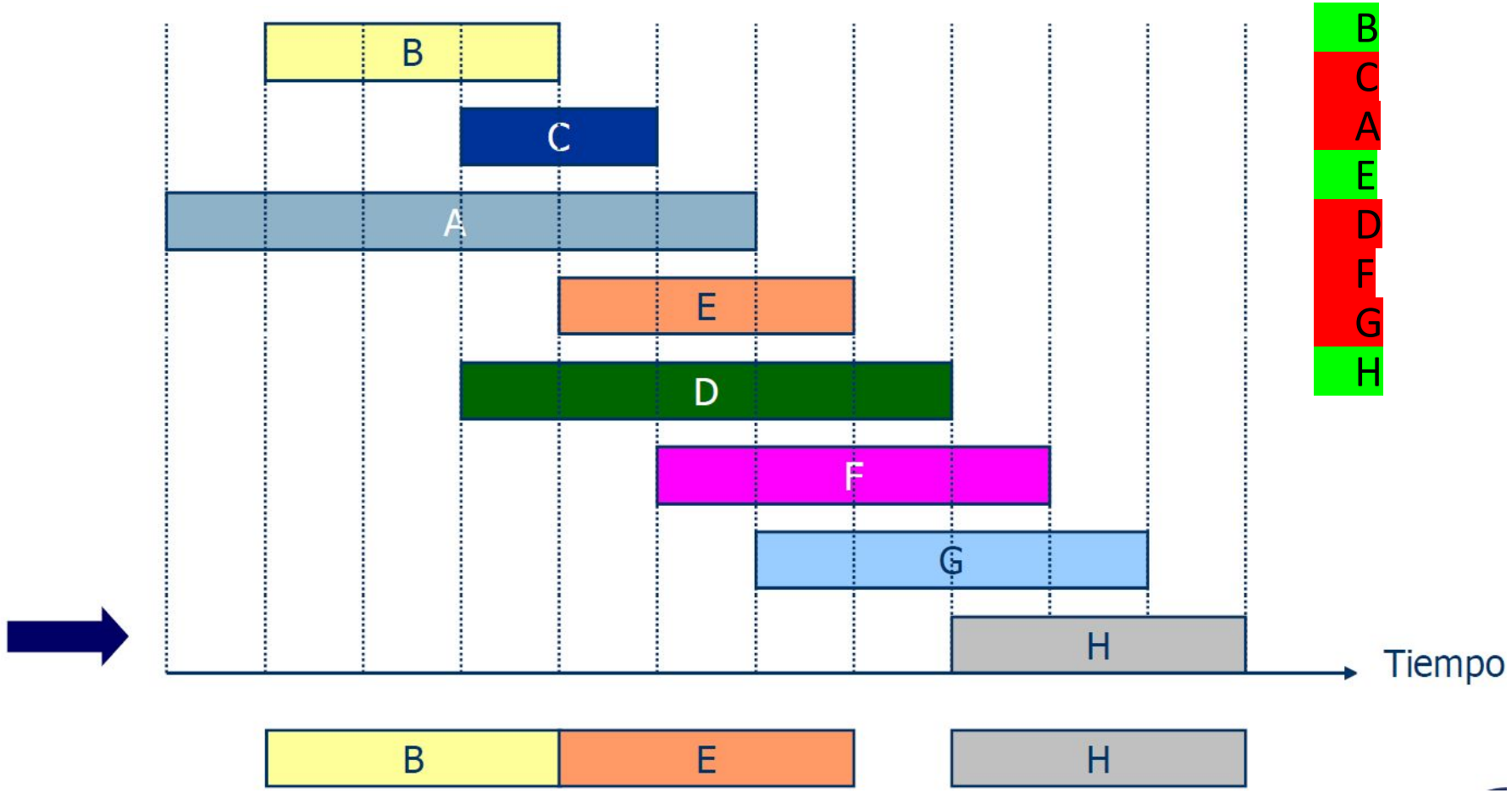


Ejemplo: Selección de actividades

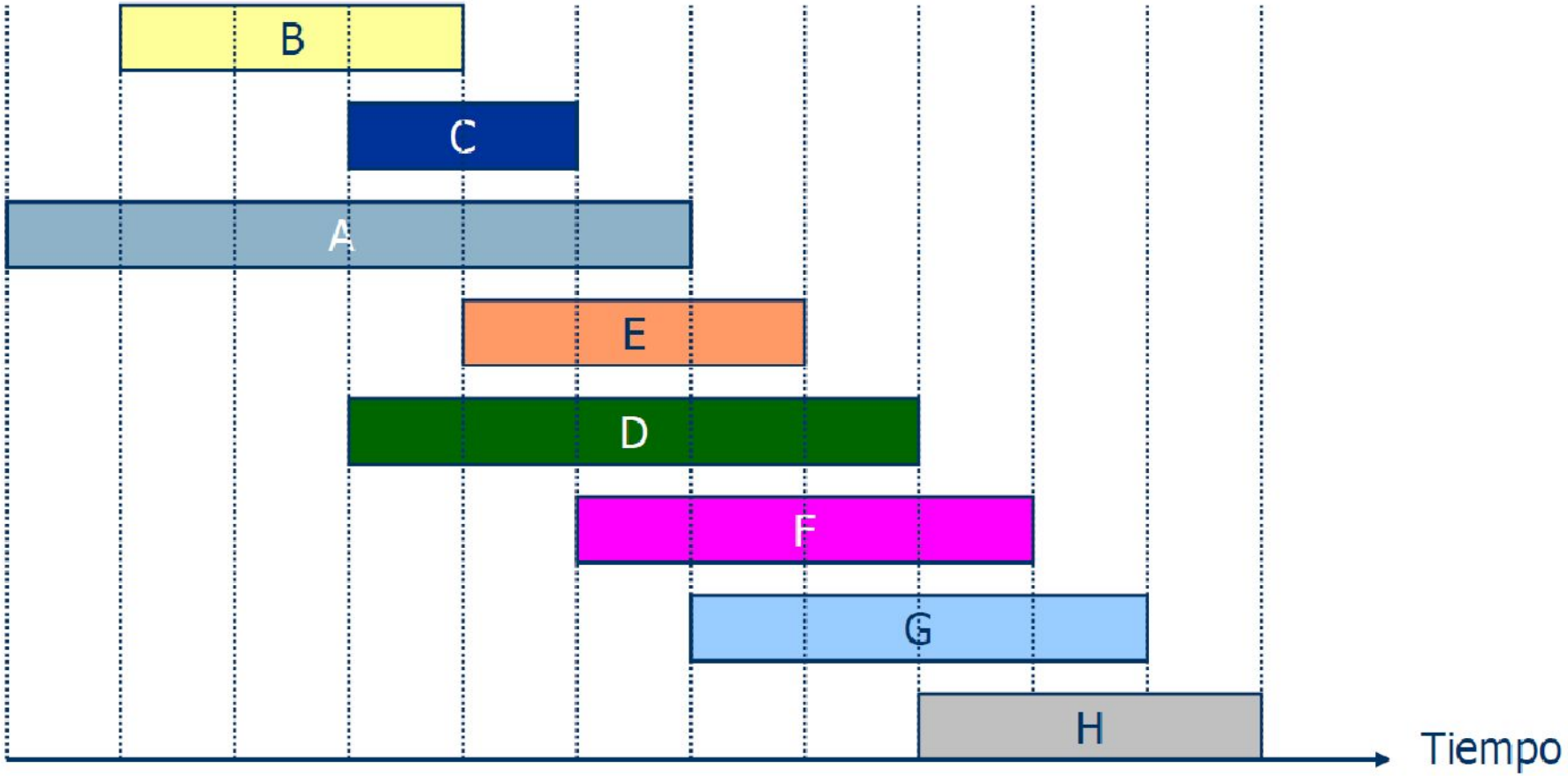


Ejemplo: Selección de actividades

Tiempo muerto: 2



Tiempo inicio



Tiempo muerto: 1

1.

Start	1	3	2	0	5	8
End	3	4	5	7	9	10
Act	A	B	C	D	E	F

Start	1	3	0	5	3	6
End	4	5	6	7	8	10
Act	A	B	C	D	E	F

2:

3:

Start	1	0	1	4	2	5
End	3	4	2	6	9	8
Act	A	B	C	D	E	F

3:

Start	1	0	1	4	2	5
End	3	4	2	6	9	8
Act	A	B	C	D	E	F

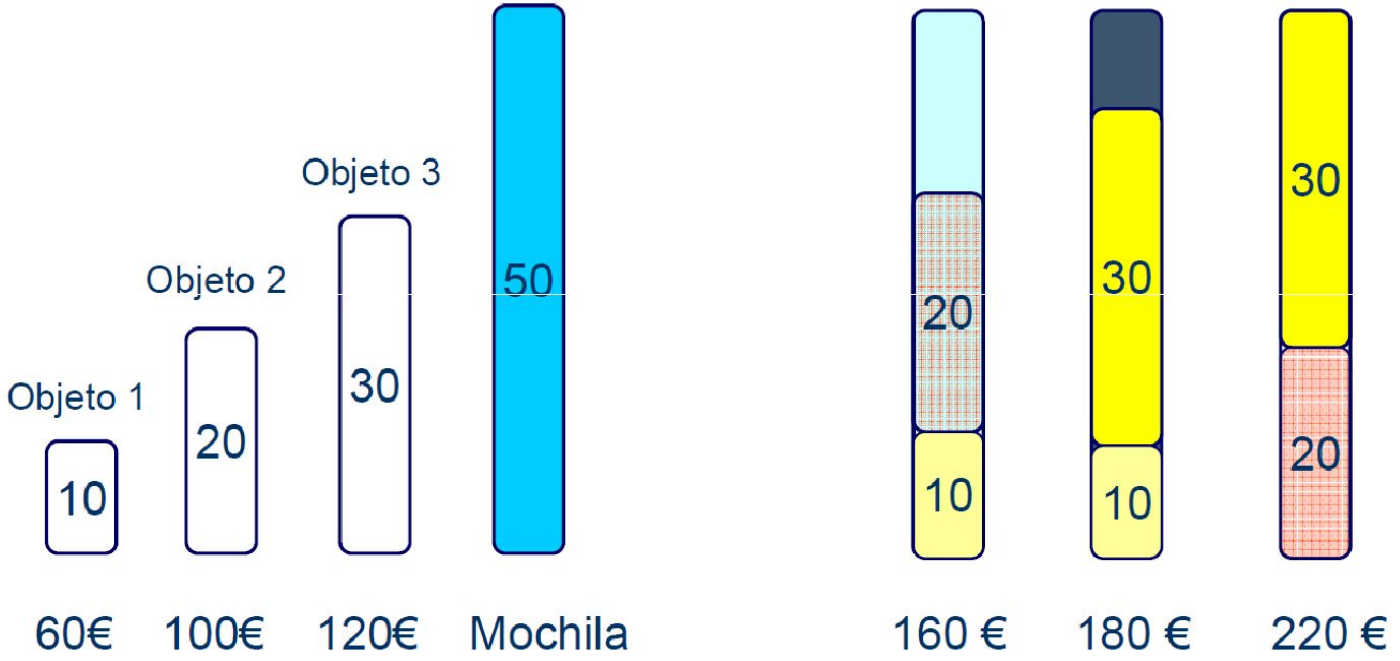
4:

Start	1	3	0	5	3	5
End	4	5	6	7	8	9
Act	A	B	C	D	E	F

5

Start	1	3	4	5	7	5
End	4	5	6	7	9	10
Act	A	B	C	D	E	F

Ejemplo: el problema de la mochila 0/1



¿Cómo seleccionamos los objetos? NP

Ejemplo: el problema de la mochila 0/1

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

$$\max \sum_{1 \le i \le n} x_i b_i$$

- Sujeto a

$$\sum_{1 \le i \le n} x_i p_i \le P$$

Ejemplo: el problema de la mochila 0/1

Ejemplo:

Mochila de 100kg

¿Cómo seleccionamos los objetos?

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

2	1.5	2.1	1	1.2

- 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 + 30 = 155$
- Primero el que tenga más valor por unidad de peso
  - Peso = 30 + 10 + 20 + 40 = 100 kg
  - Beneficio = 65 + 20 + 30 + 40 = 155$

Ejemplo: el problema de la mochila 0/1

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

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

- Se añade a la mochila todo lo que se puede:
  - 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

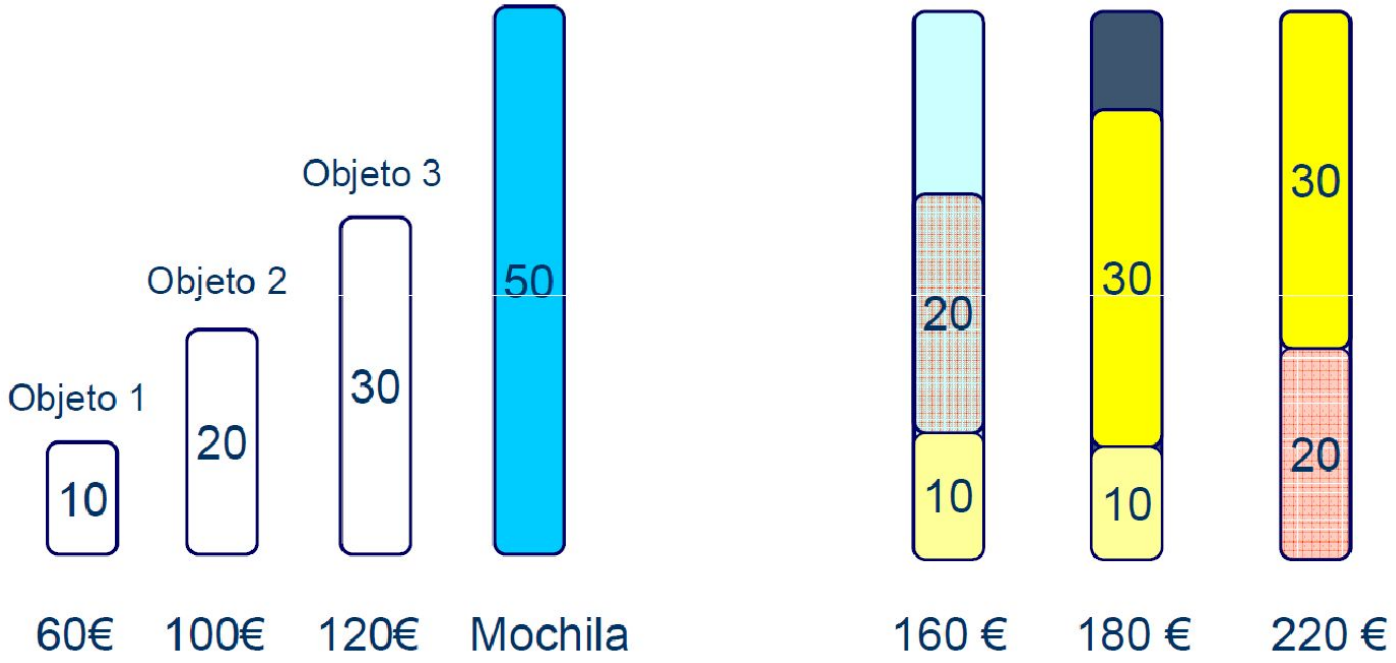
Heurísticas greedy

- 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 brinda una solución aceptable a un problema mediante métodos que carecen de justificación formal (ejemplo: por tanteo, usando reglas empíricas…)

Ejemplo: el problema de la mochila 0/1



¿Cómo sel

Ejemplo: el 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 …

$$\max \sum_{1 \le i \le n} x_i b_i$$

Sujeto a

$$\sum_{1 \le i \le n} x_i p_i \le P \qquad con\ x_i \in \{0,1\}$$

Ejemplo: el problema de la mochila 0/1

Heurística greedy

Ordenar los objetos por densidad no creciente:

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

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