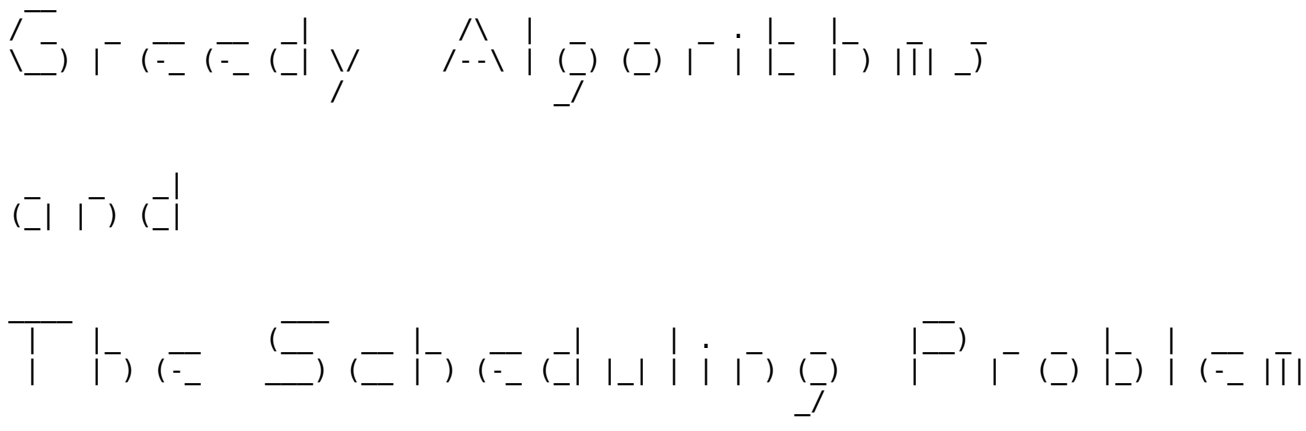
"Análisis de Algoritmos"
José Helo Guzmán



Contenido

▬▬▬▬ 1. Introducción.

Se presenta el “scheduling problem”, el cual es un problema para construir horarios. Se hablará de cómo resolver el problema de forma exhaustiva y de un método “greedy” para resolverlo.

▬▬▬▬ 2. The Scheduling Problem.

Tenemos un salón de clase, el cual debe utilizarse durante el día. A continuación se muestra una tabla con las lecciones que se pueden impartir en esa aula, donde se indica el nombre de la materia, la hora de inicio y la hora de finalización.

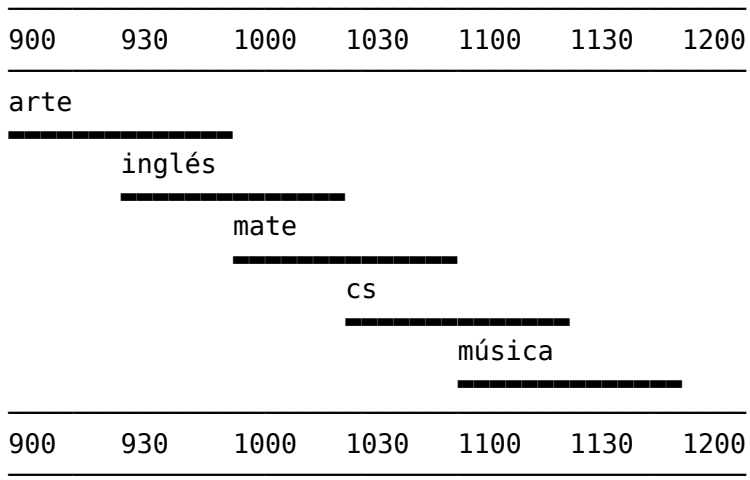
El objetivo deseado es tener el mayor número de lecciones en el aula, maximizando el tiempo de uso y minimizando el tiempo en que no se utiliza.

Clase	Inicio	Finaliza antes de	
arte	9:00	10:00	[9:00, 10:00[
inglés	9:30	10:30	[9:30, 10:30[
mate	10:00	11:00	[10:00, 11:00[
compu	10:30	11:30	[10:30, 11:30[
música	11:00	12:00	[11:00, 12:00[

Para no tener que preocuparnos por las conversiones de horas y minutos, podemos trabajar este problema utilizando hora continua, conocida como hora militar. Donde las 9:30am se escribe como 930.

Clase	Inicio	Finaliza antes de	
arte	900	1000	[900, 1000[
inglés	930	1030	[930, 1030[
mate	1000	1100	[1000, 1100[
compu	1030	1130	[1030, 1130[
música	1100	1200	[1100, 1200[

Evidentemente no se pueden ofrecer todas estas clases en el aula, pues tendríamos un choque de horarios.



Por la descripción del problema, se desean impartir el mayor número de clases posibles en el salón. Qué algoritmo debemos utilizar para seleccionar el mayor número posible de clases.

▬▬▬ 3. Solución por Fuerza Bruta.

Para resolver el problema de forma exhaustiva, debemos de ver todos los posibles subconjuntos de horarios que se pueden formar.

En este caso como tenemos 5 grupos, debemos revisar:

$2^5 = 32$ subgrupos.

A continuación se enumeran estos subgrupos:

Grupo Inicial:

{arte,inglés,mate,cs,música}

Subgrupos:

{},

{arte },
{inglés},
{mate },
{cs   },
{música}

{arte,   inglés},
{arte,   mate },
{arte,   cs   },
{arte,   música},
{inglés, mate },
{inglés, cs   },
{inglés, música},
{mate,   cs   },
{mate,   música},
{cs,   música},

{arte,   inglés, mate },
{arte,   inglés, cs   },
{arte,   inglés, música},
{arte,   mate,   cs   },
{arte,   mate,   música},
{arte,   cs,   música},
{inglés, mate,   cs   },
{inglés, mate,   música},
{inglés, cs,   música},
{mate,   cs,   música},

{arte,   inglés, mate, cs   },
{arte,   inglés, cs,   música},

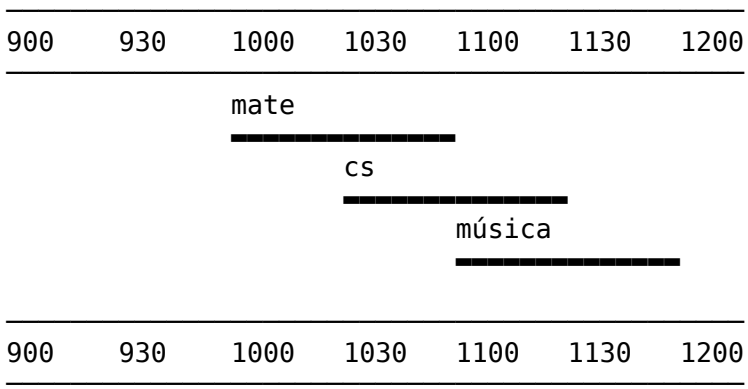
{arte,   mate,   cs,   música},
{arte,   inglés, mate, música},
{inglés, mate,   cs,   música},

{arte, inglés, mate, cs, música},

Algunas de estas respuestas tienen choques de horario, por lo que deben ser eliminadas.

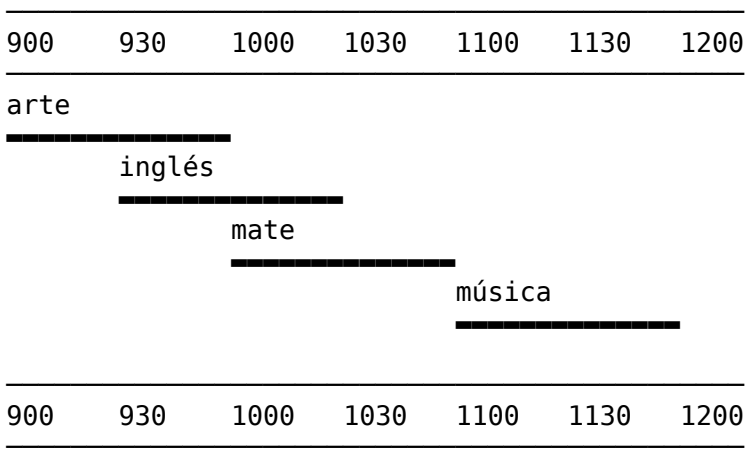
Por ejemplo, el conjunto: {cs, mate, música} debe ser eliminada.

cs   [1030, 1130[
mate,   [1000, 1100[
música   [1100, 1200[



Por ejemplo, el conjunto: {arte, inglés, mate, música} debe ser eliminada.

arte   [900, 1000[
inglés, [930, 1030[
mate,   [1000, 1100[
música   [1100, 1200[



●● Se eliminan los choques de horario.

A continuación se muestran los conjuntos de clases que no tienen choques de horario.

{},

{arte },
{inglés},
{mate },
{cs   },
{música},

{arte,   mate },
{arte,   cs   },
{arte,   música},
{inglés, cs   },
{inglés, música},
{mate,   música},

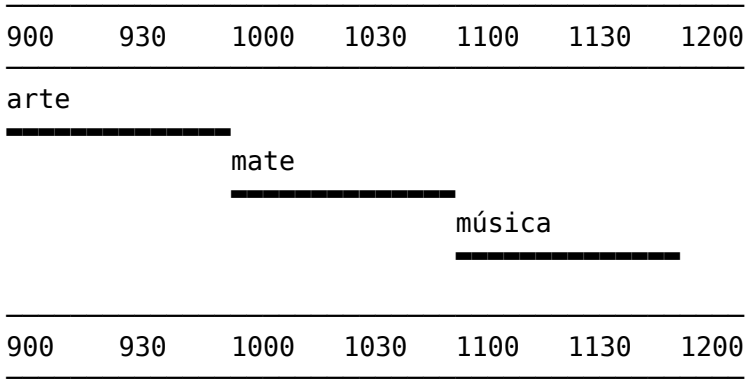
{arte, mate, música},

Y de estas opciones se selecciona la que tenga el mayor número de cursos.
En este caso la respuesta está dada por:

{arte,mate,música}, con 3 cursos.

Veamos la respuesta de manera gráfica:

arte,   [900, 1000[
mate,   [1000, 1100[
música, [1100, 1200[



▬▬▬ 4. Complejidad del Algoritmo.

Como es evidente, para este algoritmo se deben generar 2^n opciones las cuales tienen que ser revisadas, por lo tanto el algoritmo tiene un eficiencia de:

$O(2^n)$

▬▬▬▬ 5. Solución con un Algoritmo Greedy.

El objetivo principal es el de seleccionar el mayor número de clases para impartir en un aula. Veamos un posible algoritmo “greedy”.

1. Ordenamos las clases
   con base en la hora de finalización:

2. Seleccione la clase que termina lo antes posible.
   Esta es la primera clase que se va a impartir.

3. Selecciones una clase que inicie justo después de esa clase,
   y que termine lo antes posible.

Si seguimos este algoritmo tendremos la siguiente selección.

>>> Se ordenan las clases con base en la hora de finalización:

Clase	Inicio	Finaliza antes de		
arte	900	1000	[900, 1000[	
inglés	930	1030	[930, 1030[	
mate	1000	1100	[1000, 1100[	
compu	1030	1130	[1030, 1130[	
música	1100	1200	[1100, 1200[	

>>> De las clases posibles escogemos la que termina primero

Clase	Inicio	Finaliza antes de		
arte	900	1000	[900, 1000[	✔
inglés	930	1030	[930, 1030[	
mate	1000	1100	[1000, 1100[	
compu	1030	1130	[1030, 1130[	
música	1100	1200	[1100, 1200[	

>>> Se selecciona una clase que inicie después de esa clase.
    Y que termine lo antes posible.

Clase	Inicio	Finaliza antes de		
arte	900	1000	[900, 1000[	✔
inglés	930	1030	[930, 1030[	X
mate	1000	1100	[1000, 1100[	✔
compu	1030	1130	[1030, 1130[	
música	1100	1200	[1100, 1200[	

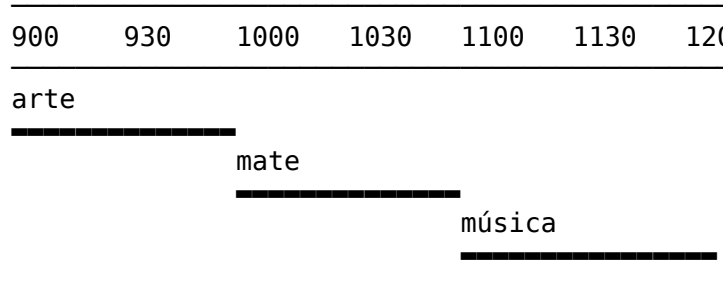
>>> Seguimos el proceso.

Clase	Inicio	Finaliza antes de		
arte	900	1000	[900, 1000[	✔
inglés	930	1030	[930, 1030[	X
mate	1000	1100	[1000, 1100[	✔
compu	1030	1130	[1030, 1130[	X
música	1100	1200	[1100, 1200[	✔

Al final obtenemos la respuesta:

```
[
[arte   [ 900, 1000]],
[mate   [1000, 1100]],
[música [1100, 1200]]
]
```

Podemos ver que es una respuesta adecuada, al igual que antes.



900	930	1000	1030	1100	1130	1200

▬▬▬ 6. Otro Ejemplo de Scheduling.

Supongamos que tenemos la lista de cursos siguiente:

Clase	Inicio	Finaliza antes de	
danza	800	1100	[800, 1100[
arte	900	1000	[900, 1000[
inglés	930	1030	[930, 1030[
física	1000	1200	[1000, 1200[
mate	1000	1100	[1000, 1100[
compu	1030	1130	[1030, 1130[
música	1100	1200	[1100, 1200[

>>> Inicialmente ordenamos las clases
con base en la hora de finalización:

Clase	Inicio	Finaliza antes de	
arte	900	1000	[900, 1000[
inglés	930	1030	[930, 1030[
danza	800	1100	[800, 1100[
mate	1000	1100	[1000, 1100[
compu	1030	1130	[1030, 1130[
física	1000	1200	[1000, 1200[
música	1100	1200	[1100, 1200[

>>> Utilizamos el Algoritmo:

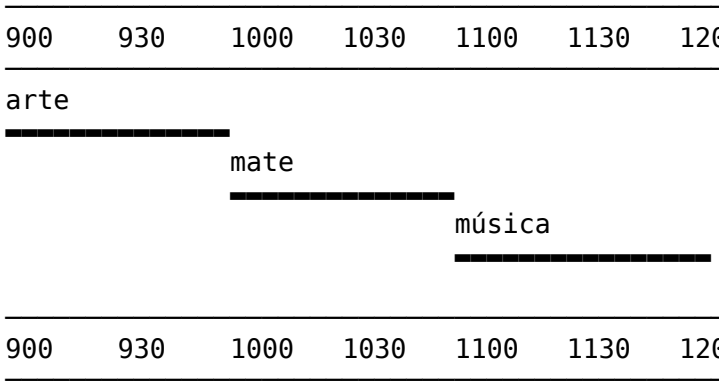
Clase	Inicio	Finaliza antes de		
arte	900	1000	[900, 1000[	✔
inglés	930	1030	[930, 1030[	X
danza	800	1100	[800, 1100[	X
mate	1000	1100	[1000, 1100[	✔
compu	1030	1130	[1030, 1130[	X
física	1000	1200	[1000, 1200[	X
música	1100	1200	[1100, 1200[	✔

Con lo que obtiene la respuesta final de:

arte, [900, 1000[✔

mate, [1000, 1100[✔

música,[1100, 1200[✔



▬▬▬ 7. Análisis de la Eficiencia.

Vemos como el algoritmo “greedy” es muy sencillo. En cada paso, seleccione la mejor opción. En términos más exactos, en cada paso se selecciona el óptimo local, y al final, esperamos estar cerca del óptimo global.

Si tenemos “n” clases para impartir en el aula y se tratan de resolver el problema, sin hacer un ordenamiento, el tiempo de este algoritmo es:

$O(n^2)$

Sin embargo, si se ordenan las clases, iniciando con la que inicia primero y termina primero, y así sucesivamente por orden de terminación, tal como lo vimos en el ejemplo, la eficiencia del algoritmo, sin tomar el cuenta el ordenamiento, está dado por:

$O(ordenar) + O(n)$

$O(n * \log_2(n)) + O(n)$

$= O(n * \log_2(n))$

▬▬▬ 8. Comentario Final.

En general, los algoritmos greedy son sencillos. En cada paso, se selecciona la opción óptima. Es decir, la mejor opción de las disponibles. En el caso del scheduling, cada vez que se selecciona una clase, escoge la clase que termine lo antes posible.

Es decir, en cada paso, se escoge un óptimo local. Y se espera que al escoger óptimos locales, se pueda llegar al óptimo global.

En general, los algoritmos greedy no encuentran la solución óptima global en todos los casos. En este ejemplo, este sencillo algoritmo si encuentra la respuesta óptima del problema de scheduling propuesto.

▬▬▬ 9. Resumen.

• Los algoritmos de scheduling son muy útiles en la vida real y se utilizan con mucha frecuencia.

• Este algoritmo se puede resolver por medio de fuerza bruta, para revisar todas la soluciones se necesita un tiempo de $O(2^n)$.

• Si se utiliza un algoritmo “greedy” sin hacer un ordenamiento de los horarios, se tardará un tiempo de $O(n^2)$.

• Si se utiliza un algoritmo “greedy” y se hace un ordenamiento de los horarios, se tardará un tiempo de $O(n*\log_2(n))$.

▬▬▬ 10. Ejercicios.

●● Ejercicio 1.

Programe este problema de scheduling utilizando fuerza bruta.

●● Ejercicio 2.

Programe este problema de scheduling utilizando un algoritmo greedy.

●● Ejercicio 3.

Existe una extensión del algoritmo de scheduling, llamado scheduling con pesos. Busque cómo es que se formula este problema.