**ALGORITMOS GREEDY**
RELACIÓN DE EJERCICIOS Y PROBLEMAS

1. Se tiene un buque mercante cuya capacidad de carga es de k toneladas y un conjunto de contenedores $c_1$, …, $c_n$ cuyos pesos respectivos son $p_1$, …, $p_n$ (expresados también en toneladas). Teniendo en cuenta que la capacidad del buque es menor que la suma total de los pesos de los contenedores:

    a. Diseñe un algoritmo que maximice el número de contenedores cargados.

    b. Diseñe un algoritmo que intente maximizar el número de toneladas cargadas.

2. Sean n programas $P_1$, ..., $P_n$ que hay que almacenar en un disco. El programa $P_i$ requiere $S_i$ megabytes de espacio y la capacidad del disco es D megabytes, siendo $D < \Sigma S_i$. Resuelva las siguientes cuestiones:

    a. Diseñe un algoritmo greedy que maximice el número de programas almacenados en el disco (existe un algoritmo que proporciona una solución óptima).

    b. Diseñe un algoritmo greedy que intente aprovechar al máximo la capacidad del disco (demuestre que podemos utilizar un algoritmo greedy que seleccione los programas por orden no creciente para obtener la solución exacta o encuentre un contraejemplo de lo contrario).

    *[Brassard & Bratley, 1997, problema 6.21]*

3. Tenemos que completar un conjunto de n tareas con plazos límite. Cada una de las tareas consume la misma cantidad de tiempo (una unidad) y, en un instante determinado, podemos realizar únicamente una tarea. La tarea i tiene como plazo límite $d_i$ y produce un beneficio $g_i$ ($g_i > 0$) sólo si la tarea se realiza en un instante de tiempo $t \leq d_i$.

    a. Diseñe un algoritmo greedy que nos permita seleccionar el conjunto de tareas que nos asegure el mayor beneficio posible.

    b. Aplique su algoritmo a la siguiente instancia del problema:

| Tarea | i | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| Beneficio | $g_i$ | 50 | 10 | 15 | 30 |
| Plazo límite | $d_i$ | 2 | 1 | 2 | 1 |

    *[Brassard & Bratley, 1997, sección 6.6.2: "Planificación con plazo fijo"]*

4. Consideremos un grafo no dirigido G = (V, E). Un conjunto de nodos U se dice que es un recubrimiento de G si U es un subconjunto de V tal que cada arista en E incide en, al menos, un vértice de U. Un conjunto de nodos es un recubrimiento minimal de G si es un recubrimiento con el número mínimo posible de nodos.

    a. Diseñe un algoritmo greedy para obtener un recubrimiento de G.

    b. Indique si el algoritmo greedy devuelve un recubrimiento minimal para cualquier grafo si, como función de selección, se escoge el nodo con mayor grado de incidencia de entre los nodos aún no seleccionados.

    NOTA:
    El grado de incidencia de un vértice es el número de aristas que inciden en él.

5. Supongamos que un único camarero debe atender a n clientes de un restaurante y atender al cliente i-ésimo requiere un tiempo $t_i$. La satisfacción de un cliente es inversamente proporcional a su tiempo de espera y la propina que recibirá el camarero dependerá de la satisfacción del cliente (si el cliente i espera $w_i$ minutos, la propina que dará al camarero será $p_i=1/w_i$).

    a. Diseñe un algoritmo greedy para maximizar las ganancias del camarero y demuestre si el algoritmo diseñado devuelve siempre la solución óptima [o encuentre un contraejemplo que muestre que no lo hace].

    b. Nuestro restaurante toma la decisión de contratar a más camareros para reducir el tiempo de espera de sus clientes. Modifique el algoritmo anterior para maximizar las propinas de k camareros.

    *[Brassard & Bratley, 1997, problema 6.20]*

6. Supongamos que disponemos de n trabajadores y n tareas. Sea $c_{ij} > 0$ el coste de asignarle el trabajo j al trabajador i. Una asignación válida es aquélla en la que a cada trabajador le corresponde una tarea y cada tarea la realiza un trabajador diferente. Dada una asignación válida, definimos el coste de dicha asignación como la suma total de los costes individuales.

    a. Diseñe distintas estrategias greedy para asignar tareas.

    b. Encuentre contrajemplos que demuestren que ninguna de ellas nos permite encontrar la solución óptima con un algoritmo greedy.

7. Un autobús realiza una ruta determinada entre dos ciudades. Con el tanque de gasolina lleno, el autobús puede recorrer n kilómetros sin parar. El conductor dispone de un mapa de carreteras que le indica las gasolineras que hay en su ruta. Para minimizar el tiempo empleado en recorrer su ruta, el conductor desea pararse a repostar el menor número posible de veces. Diseñe un algoritmo greedy que determine en qué gasolineras tiene que repostar y demuestre que su algoritmo encuentra siempre la solución óptima.

   *[Kleinberg & Tardos, 2006, Solved Exercise 1]*

8. Supongamos que el coste de tender una red de fibra óptica entre dos ciudades es proporcional a la distancia euclídea entre ellas.

   a. Diseñe un algoritmo que permita interconectar un conjunto de ciudades minimizando el coste de la red de interconexión.

   b. Busque un ejemplo en el que se demuestre que puede resultar más económico instalar una centralita entre ciudades que utilizar solamente conexiones directas entre ciudades.

9. El departamento de marketing de nuestra empresa desea segmentar nuestro conjunto de clientes en K grupos para diseñar campañas de marketing dirigidas a cada grupo por separado. Si suponemos que podemos caracterizar a cada cliente mediante un vector de características ($c_1$, .., $c_n$) y medir la similitud entre dos clientes como la inversa de la distancia entre sus vectores de características, diseñe un algoritmo greedy que nos permita agrupar a nuestros clientes en K grupos diferentes de tal forma que se maximice la distancia entre clientes de distintos grupos.

   *[Kleinberg & Tardos, 2006, sección 4.7: "Clustering"]*

10. Nuestra empresa desarrolla software para terminales de puntos de venta [TPV] y desea que le añadamos la siguiente funcionalidad a nuestro sistema:

    a. En su versión para máquinas expendedoras, se pretende minimizar el número de monedas empleado para darle el cambio al cliente. Diseñe un algoritmo greedy que devuelva un número mínimo de monedas (de 0.01, 0.02, 0.05, 0.10, 0.20, 0.50 y 1 euro).

    b. Nuestra empresa también distribuye máquinas expendedoras de sellos de correos (de 0.54, 0.32, 0.17 y 0.01 euros) y decidimos reutilizar nuestro algoritmo greedy para decidir qué sellos ha de devolver la máquina. Demuestre que el algoritmo greedy no proporciona necesariamente una solución óptima, aun disponiendo de un suministro inagotable de sellos de cada valor.

11. Deseamos optimizar el uso de las aulas de un centro educativo. Dados un conjunto de aulas y un conjunto de clases con un horario preestablecido (la clase i empieza a la hora $s_i$ y termina a la hora $f_i$), diseñe un algoritmo greedy que minimice el número de aulas necesario para impartir toda la docencia del centro.

    *[Kleinberg & Tardos, 2006, sección 4.1: "Scheduling All Intervals"]*

12. Igual que antes, tenemos un problema de asignación de recursos en el que una tarea i requiere $t_i$ unidades de tiempo y ha de terminar, como muy tarde, en el instante de tiempo $d_i$. Ahora bien, podemos escoger la hora de inicio de la tarea $s_i$ de tal forma que $f_i=s_i+t_i$. Diseñe un algoritmo greedy que minimice el retraso máximo con el que se completa una tarea (esto es, $\max\{0,f_i-d_i\}$).

    *[Kleinberg & Tardos, 2006, sección 4.2: "Scheduling to Minimize Lateness"]*

13. Disponemos de una memoria caché con capacidad para almacenar k datos. Cuando la caché recibe una solicitud $d_i$, puede que el dato ya esté en la caché [hit] o que tengamos que buscarlo en memoria [miss], en cuyo caso reemplazaremos alguno de los datos ya almacenados en la caché por el dato que acabamos de obtener. Si conocemos de antemano el conjunto de solicitudes que deberemos atender $d_1..d_m$, diseñe un algoritmo greedy que minimice el número de fallos de caché.

    *[Kleinberg & Tardos, 2006, sección 4.3: "Optimal Caching"]*

14. Hemos decidido montar una empresa de desarrollo de software, para lo que debemos adquirir n licencias, cuyo precio actual es de 100 € cada una. Sin embargo, nuestro limitado presupuesto sólo nos permite adquirir una licencia mensualmente. El problema es que, aunque ahora mismo todas las licencias nos cuestan lo mismo (100€), el precio de la licencia i se incrementa mensualmente un tanto por ciento $\Delta p_i$. Esto es, dentro de k meses, la licencia i nos costará $100*(1+\Delta p_i)^k$ euros. Sabiendo que, al final, necesitaremos tener todas las licencias en regla, diseñe un algoritmo greedy que minimice el coste de adquisición de las n licencias a partir de sus tasas de incremento de precio $\Delta p_i$.

    Obsérvese que el problema no se puede resolver de forma óptima con un algoritmo greedy si, en vez de comprar licencias, nuestro objetivo fuese vender equipos que se deprecien con el tiempo ($\Delta p_i<0$).

    *[Kleinberg & Tardos, 2006, Solved Exercise 2]*

OBSERVACIONES:

- Al diseñar un algoritmo greedy, el problema determina el criterio que define lo que es una solución y la función objetivo que deseamos optimizar. Hemos de estudiar posibles conjuntos de candidatos para formar una solución y distintas funciones de selección y factibilidad.

- Si no conseguimos demostrar la corrección del algoritmo greedy, hemos de encontrar contraejemplos que nos demuestren que el algoritmo no siempre proporciona una solución óptima.

- Incluso cuando el algoritmo no nos asegure una solución óptima, hemos de razonar justificadamente los motivos que nos llevan a elegir la heurística que, para nosotros, resulta más prometedora.