Ejercicios. Algoritmos voraces

1. Antonio va a preparar un guiso de carne para invitar a todos sus amigos. Para ello, necesita cocinar 10 kilos de carne. En casa tiene los siguientes productos, cada uno con el precio que ha pagado por ellos:

- 3 kg de conejo – 16€
- 4 kg de pollo – 16€
- 6 kg de ternera – 45€
- 2 kg de cerdo – 9.5€
- 4 kg de pavo – 25€

Como es un poco tacaño, quiere que el guiso de carne le cueste el menor dinero posible. ¿Qué carnes tiene que añadir? ¿Cuánta cantidad de cada una?

2. Manuel tiene en casa varios bonos de gasolina gratis que ha ganado repostando en su gasolinera habitual. Cada uno de los bonos es equivalente a la gasolina consumida en un trayecto de un número fijo de kilómetros. En concreto, tiene los siguientes bonos:

- 3 bonos de 100 km
- 1 bono de 50 km
- 4 bonos de 40 km
- 2 bonos de 30 km
- 8 bonos de 5 km
- 6 bonos de 1 km

Mañana va a hacer un viaje a un pueblo de la provincia de Madrid, que está situado a 458 km. Suponiendo que tiene el depósito vacío, y que quiere gastar el menor número de bonos, ¿qué bonos debe emplear para llegar hasta su destino, si quiere terminar con el depósito vacío?

3. En el supermercado del barrio se forman grandes colas a la hora de pagar, lo que está generando malestar entre los clientes. Para solventar este problema, el gerente está estudiando cómo repartir los clientes entre las 3 cajas disponibles. Antes de que un cliente llegue a pagar, ya sabe el tiempo que va a estar en la caja, puesto que éste depende del número de productos (cuantos menos productos lleve, menos tiempo tarda). En un momento dado, se acercan a las cajas 10 clientes, que llevan los siguientes productos:

- cliente 1: 15 productos
- cliente 2: 6 productos
- cliente 3: 18 productos
- cliente 4: 1 producto
- cliente 5: 10 productos
- cliente 6: 3 productos
- cliente 7: 7 productos
- cliente 8: 9 productos
- cliente 9: 4 productos
- cliente 10: 2 productos

¿Cómo debe distribuirlos el gerente, para que la suma de los tiempos que están todos los clientes en las cajas sea mínima?

4. Un camionero conduce entre dos ciudades siguiendo una ruta dada con un camión que le permite, con el tanque de gasolina lleno, recorrer n kilómetros sin parar. El camionero dispone de un mapa de carreteras que le indica las distancias entre las gasolineras que hay en la ruta y, como tiene prisa, desea pararse a repostar el menor número posible de veces. Diseña un algoritmo voraz para determinar en qué gasolineras tiene que parar y demuestra que el algoritmo siempre encuentra la solución óptima.

5. Supongamos que disponemos de n trabajadores y n tareas. Cada trabajador tiene un coste asociado por realizar cada una de las tareas. Debemos realizar una asignación de tareas, de tal forma que a cada trabajador solo le corresponda una tarea, y cada tarea sea realizada solo por un trabajador. El coste de una asignación se calcula como la suma de los costes que cada trabajador tiene asociado a su tarea.
Diremos que una asignación es óptima si es de mínimo coste. A la hora de diseñar un algoritmo voraz para este problema, podemos pensar en dos estrategias distintas: asignar a cada trabajador la mejor tarea posible, o bien asignar a cada tarea el mejor trabajador posible. Sin embargo, ninguna de estas dos estrategias tiene por qué encontrar soluciones óptimas.
Piensa un ejemplo en el que haya 3 trabajadores y 3 tareas, en el que la primera estrategia no llegue el mejor resultado posible y otro ejemplo del mismo tamaño en el que la segunda estrategia no obtenga el resultado óptimo. ¿Se te ocurre alguna otra estrategia que obtenga buenos resultados? Mediante esa estrategia, ¿son los resultados siempre óptimos? Si es así, demuéstralo. En caso contrario, busca otro ejemplo para demostrarlo.

6. Tenemos un conjunto de n tareas a realizar, necesitando cada una de ellas un tiempo unidad para ser procesada. En cada instante t = 1, 2, … se puede realizar una sola tarea. Además, la tarea i, $1 \leq i \leq n$, aporta un beneficio $b_i > 0$ sólo si es realizada no más tarde del instante d_i. ¿Qué estrategia se debe seleccionar para elegir el orden de las tareas si queremos obtener un beneficio máximo? Esa estrategia, ¿obtiene siempre la solución óptima?
Ejemplo

Tarea	Tiempo máximo	Beneficio
1	2	4
2	2	3
3	2	5
4	3	6
5	3	3

7. Tenemos n paquetes p1, p2, …, pn con tamaños t1, t2, …, tn y una caja con capacidad c,de tal forma que no todos los paquetes caben en ella (c<t1+t2+…+tn). Queremos llenar la caja tanto como podamos, es decir, dejar el mínimo espacio vacío, teniendo en cuenta que cada uno de los paquetes se puede meter o no en ella, pero no se puede fraccionar. Diseñar un algoritmo voraz para resolver este problema. ¿Encuentra siempre la solución óptima?

8. Para colorear los vértices de un grafo no dirigido teniendo en cuenta que todo par de vértices unidos por un arco deben tener colores diferentes y que se pretende utilizar el mínimo número de colores distintos, se pueden utilizar diferentes técnicas. Estudiar dos estrategias voraces para resolver este problema. ¿Alguna de ellas encuentra siempre una solución óptima? ¿Cuál de las dos estrategias propuestas es mejor?

9. En este problema queremos calcular la matriz producto M de n matrices ($M = M_1 M_2 \dots M_n$). Por ser asociativa, la multiplicación de matrices, existen muchas formas posibles de realizar esa operación, cada una con un coste asociado (en términos del número de multiplicaciones escalares). Si multiplico las matrices M_i de dimensiones ($a \times b$) y M_j de dimensiones ($b \times c$) se requieren abc operaciones.

Por ejemplo, tenemos 4 matrices, con las siguientes dimensiones: $M_1(30 \times 1), M_2(1 \times 40), M_3(40 \times 10), M_4(10 \times 25)$. Hay 5 formas de multiplicarlas, con diferentes costes asociados

$((M_1 M_2)M_3)M_4 = 30 \cdot 1 \cdot 40 + 30 \cdot 40 \cdot 10 + 30 \cdot 10 \cdot 25 = 20700$

$M_1(M_2(M_3 M_4)) = 40 \cdot 10 \cdot 25 + 1 \cdot 40 \cdot 25 + 30 \cdot 1 \cdot 25 = 11750$

$(M_1 M_2)(M_3 M_4) = 30 \cdot 1 \cdot 40 + 40 \cdot 10 \cdot 25 + 30 \cdot 40 \cdot 25 = 41200$

$M_1((M_2 M_3)M_4) = 1 \cdot 40 \cdot 10 + 1 \cdot 10 \cdot 25 + 30 \cdot 1 \cdot 25 = 1400$

$(M_1(M_2 M_3))M_4 = 1 \cdot 40 \cdot 10 + 30 \cdot 1 \cdot 10 + 30 \cdot 10 \cdot 25 = 8200$

Estudiar al menos tres estrategias para resolver el problema mediante algoritmos voraces, y comprobar si alguna de ellas encuentra siempre la solución óptima.

10. Dada una secuencia de palabras $p_1, p_2, \dots, p_n$ de longitudes $l_1, l_2, \dots, l_n$ se desea agruparlas en líneas de longitud L. Las palabras están separadas por espacios cuya amplitud ideal es b, pero los espacios pueden reducirse o ampliarse si es necesario (aunque sin solapamiento de palabras), de tal forma que una línea $p_i, p_{i+1}, \dots, p_j$ tenga exactamente longitud L. Sin embargo, existe una penalización por reducción o ampliación en el número total de espacios que aparecen o desaparecen. El coste de fijar la línea $p_i, p_{i+1}, \dots, p\_j$ es la suma de cuanto se diferencia cada uno de los espacios con su longitud ideal b. No obstante, si $j = n$ (la última palabra) el coste será cero, a menos que se haya reducido algún espacio en esa línea.

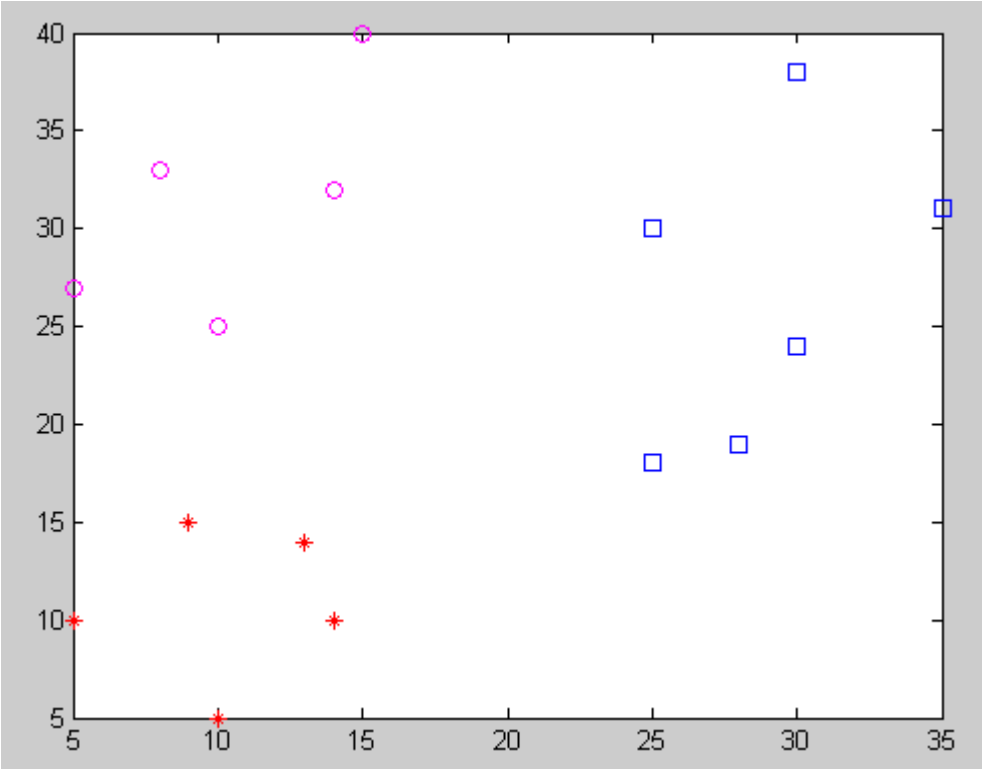
Plantear un algoritmo voraz para resolver el problema y presentar un ejemplo donde este algoritmo no encuentre la solución óptima, o bien demostrar que ese ejemplo no existe.

11. Diseñar un algoritmo voraz para resolver un problema de clustering. En un problema de clustering partimos de un conjunto de datos y queremos agruparlos en varios grupos, de tal forma que los datos más cercanos pertenezcan al mismo grupo. En este caso, todos los datos están situados en el mismo plano, por lo que se representan por sus coordenadas x e y, y la distancia entre dos datos se puede calcular mediante la distancia euclídea. Si partimos del conjunto de datos y del número de grupos que queremos obtener, diseñar un algoritmo voraz que resuelva el problema.

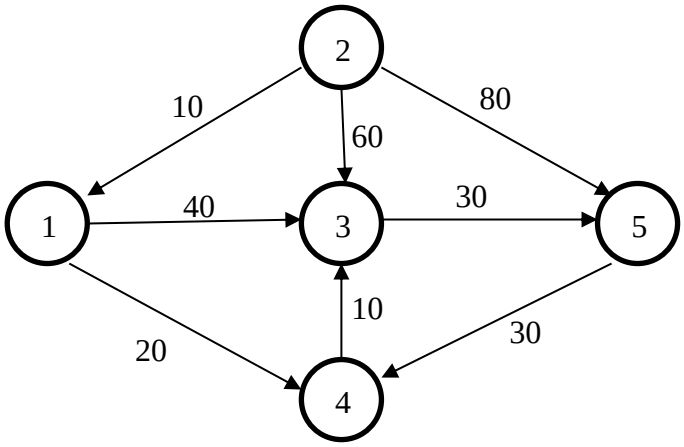
Ejemplo:

Tenemos el conjunto de datos (5,10) (5, 27) (10, 5) (9, 15) (10, 25) (8,33) (14,10) (13,14) (14,32) (15,40) (25,18) (25,30) (28,19) (30,24) (30,38) (35,31) y queremos dividirlo en 3

grupos. La solución del algoritmo voraz debe ser



12. Resolver el problema de los caminos de coste mínimo desde el vértice 1 con el algoritmo de Dijkstra, sobre el siguiente grafo:

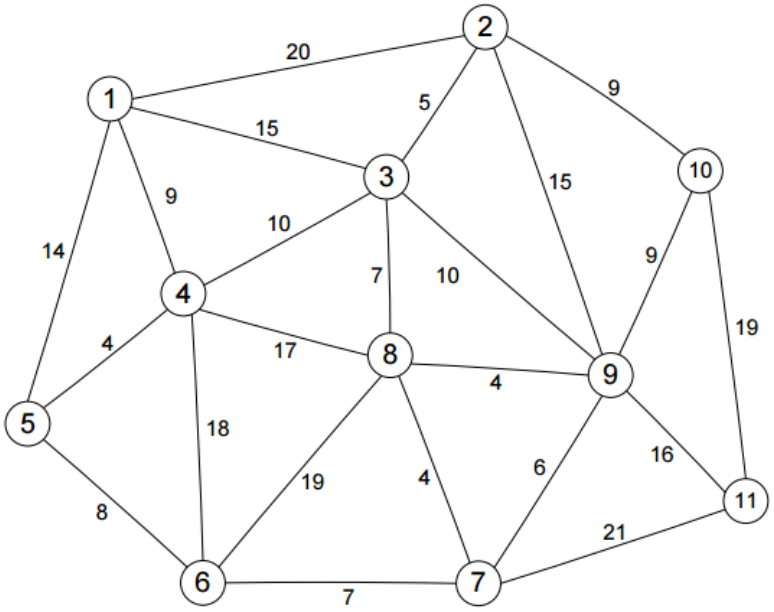


13. Consideramos un grafo no dirigido G=(V,A), donde cada arco $a \in A$ tiene un coste c_a. Suponemos que todos los arcos tienen un coste positivo y diferente entre ellos. Sea T el árbol recubridor mínimo de G y P el camino más corto entre el vértice s al vértice t. Ahora vamos a suponer que incrementamos el coste de cada arco en 1 unidad, es decir c_a+ 1. Llamamos a este nuevo grafo G'. ¿Cuál de las siguientes es cierta? Justifica tu respuesta.

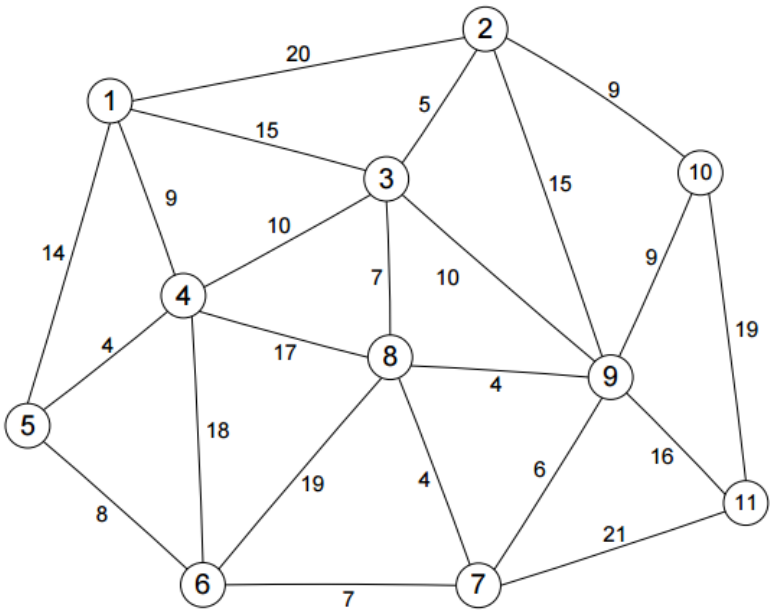
a) T es siempre el árbol recubridor mínimo y P es siempre el camino más corto entre s y t.
b) T puede que no sea el árbol recubridor mínimo pero P si es el camino más corto entre s y t.

c) T es el árbol recubridor mínimo y P puede que no sea el camino más corto entre s y t.
d) T puede que no sea el árbol recubridor mínimo y P puede que no sea el camino más corto entre s y t.

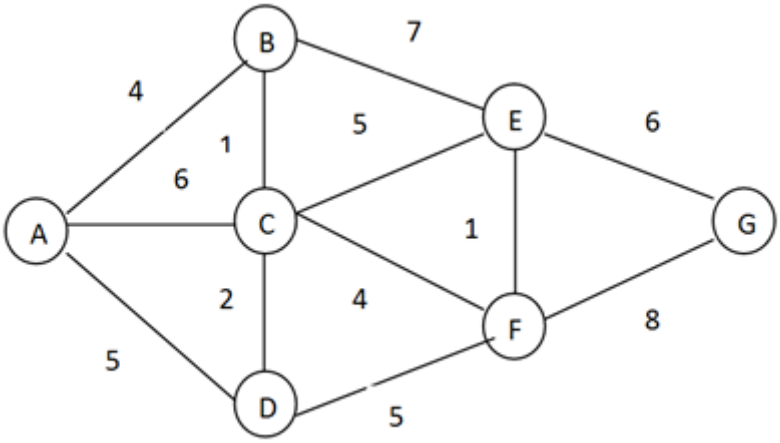
14. Ejecuta el algoritmo de Kruskal sobre el siguiente grafo, para encontrar el árbol recubridor de coste mínimo.



15. Ejecuta el algoritmo de Prim sobre el siguiente grafo, para encontrar el árbol recubridor de coste mínimo.

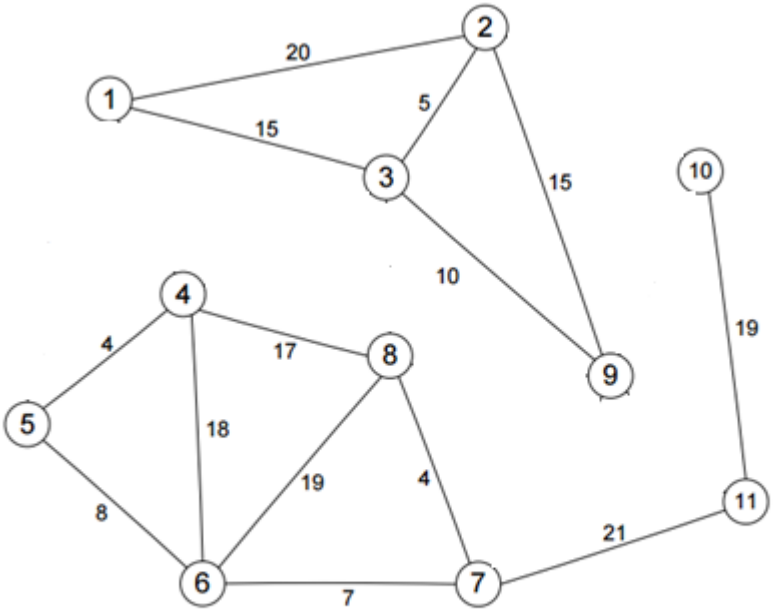


16. Existen muchos problemas donde los pesos de los arcos representan beneficios en lugar de costes. En estos casos, necesitamos encontrar el árbol recubridor de coste máximo. Calcula dicho árbol para el siguiente grafo. ¿Qué algoritmo has adaptado para ello?

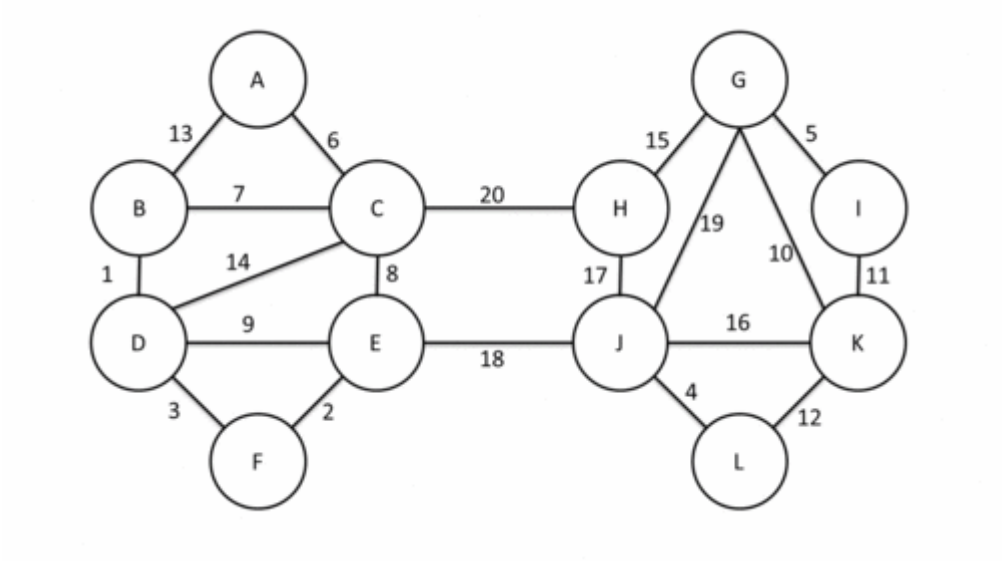


17. Tenemos un problema que hemos modelado con un grafo en el que existen pesos positivos y negativos. Necesitamos calcular su árbol recubridor de coste mínimo. ¿Podemos utilizar el algoritmo de Kruskal? ¿Y el de Prim?

18. Los algoritmos de Kruskal y Prim sirven para solucionar el problema del árbol recubridor de coste mínimo en grafos no dirigidos y conexos. Sin embargo, los problemas que nos encontramos no siempre se pueden representar como grafos conexos. En estos casos, se trata de hallar el bosque recubridor de coste mínimo. ¿Cómo podemos hacerlo? Aplica tu solución al siguiente ejemplo.



19. En este caso queremos un hallar un árbol recubridor de coste mínimo sobre el siguiente grafo. Sin embargo, por requerimientos del problema, hay algunos arcos que tienen que pertenecer a nuestro árbol (independientemente de que de esta forma el árbol no tenga el menor coste posible). Soluciona este problema para que el árbol final sea aquel cuyo coste sea mínimo incluyendo los arcos obligatorios.



Los arcos obligatorios son
(A - B), (C - D), (G - H), (J – L)