Algoritmos y Estructuras de Datos II - 1° cuatrimestre 2019

# Práctico 3 - Parte 1: Algoritmos voraces

1. Demostrar que el algoritmo voraz para el problema de la mochila *sin fragmentación* no siempre obtiene la solución óptima. Para ello puede modificar el algoritmo visto en clase de manera de que no permita fragmentación y encontrar un ejemplo para el cual no halla la solución óptima.

2. Considere el problema de dar cambio. Pruebe o dé un contraejemplo: si el valor de cada moneda es al menos el doble de la anterior, y la moneda de menor valor es 1, entonces el algoritmo voraz arroja siempre una solución óptima.

3. Se desea realizar un viaje en un automóvil con autonomía $A$ (en kilómetros), desde la localidad $l_0$ hasta la localidad $l_n$ pasando por las localidades $l_1, \ldots, l_{n-1}$ en ese orden. Se conoce cada distancia $d_i \leq A$ entre la localidad $l_{i-1}$ y la localidad $l_i$ (para $1 \leq i \leq n$), y se sabe que existe una estación de combustible en cada una de las localidades.

   Escribir un algoritmo que compute el menor número de veces que es necesario cargar combustible para realizar el viaje, y las localidades donde se realizaría la carga.

   Suponer que inicialmente el tanque de combustible se encuentra vacío y que todas las estaciones de servicio cuentan con suficiente combustible.

4. Ejecutar paso a paso, graficando las soluciones parciales, los algoritmos de Prim y Kruskal que computan el *árbol generador mínimo* sobre el grafo con nodos $\{1, 2, \ldots, 8\}$ y pesos dados por la función $w$:

$$\begin{array}{llll} w((1,2)) = 7 & w((2,3)) = 4 & w((3,6)) = 4 & w((5,6)) = 6 \\ w((1,6)) = 3 & w((2,4)) = 2 & w((3,8)) = 6 & w((6,7)) = 5 \\ w((1,7)) = 5 & w((2,5)) = 1 & w((4,6)) = 8 & w((8,5)) = 2 \\ w((1,3)) = 3 & w((3,4)) = 5 & w((5,4)) = 3 & w((8,7)) = 3 \end{array}$$

5. En numerosas oportunidades se ha observado que cientos de ballenas nadan juntas hacia la costa y quedan varadas en la playa sin poder moverse. Algunos sostienen que se debe a una pérdida de orientación posiblemente causada por la contaminación sonora de los océanos que interferiría con su capacidad de inter-comunicación. En estos casos los equipos de rescate realizan enormes esfuerzos para regresarlas al interior del mar y salvar sus vidas.

   Se encuentran $n$ ballenas varadas en una playa y se conocen los tiempos $s_1, s_2, \ldots, s_n$ que cada ballena es capaz de sobrevivir hasta que la asista un equipo de rescate. Dar un algoritmo voraz que determine el orden en que deben ser rescatadas para salvar el mayor número posible de ellas, asumiendo que llevar una ballena mar adentro toma tiempo constante $t$, que hay un único equipo de rescate y que una ballena no muere mientras está siendo regresada mar adentro.

6. Sos el flamante dueño de un teléfono satelital, y se lo ofrecés a tus $n$ amigos para que lo lleven con ellos cuando salgan de vacaciones el próximo verano. Lamentablemente cada uno de ellos irá a un lugar diferente y en algunos casos, los períodos de viaje se superponen. Por lo tanto es imposible prestarle el teléfono a todos, pero quisieras prestárselo al mayor número de amigos posible.

   Suponiendo que conoces los días de partida y regreso ($p_i$ y $r_i$ respectivamente) de cada uno de tus amigos, ¿cuál es el criterio para determinar, en un momento dado, a quien conviene prestarle el equipo?

   Tener en cuenta que cuando alguien lo devuelve, recién a partir del día siguiente puede usarlo otro. Escribir un algoritmo voraz que solucione el problema.

7. Para obtener las mejores facturas y medialunas, es fundamental abrir el horno el menor número de veces posible. Por supuesto que no siempre es fácil ya que no hay que sacar nada del horno demasiado temprano, porque queda cruda la masa, ni demasiado tarde, porque se quema.

   En el horno se encuentran $n$ piezas de panadería (facturas, medialunas, etc). Cada pieza $i$ que se encuentra en el horno tiene un tiempo mínimo necesario de cocción $t_i$ y un tiempo máximo admisible de cocción $T_i$. Si se la extrae del horno antes de $t_i$ quedará cruda y si se la extrae después de $T_i$ se quemará.

Asumiendo que abrir el horno y extraer piezas de él no insume tiempo, y que $t_i \leq T_i$ para todo $i \in \{1, \ldots, n\}$, ¿qué criterio utilizaría un algoritmo voraz para extraer todas las piezas del horno en perfecto estado (ni crudas ni quemadas), abriendo el horno el menor número de veces posible? Implementarlo.

8. Un submarino averiado descansa en el fondo del océano con $n$ sobrevivientes en su interior. Se conocen las cantidades $c_1, \ldots, c_n$ de oxígeno que cada uno de ellos consume por minuto. El rescate de sobrevivientes se puede realizar de a uno por vez, y cada operación de rescate lleva $t$ minutos.

   (a) Escribir un algoritmo que determine el orden en que deben rescatarse los sobrevivientes para salvar al mayor número posible de ellos antes de que se agote el total $C$ de oxígeno.

   (b) Modificar la solución anterior suponiendo que por cada operación de rescate se puede llevar a la superficie a $m$ sobrevivientes (con $m \leq n$).

9. Ejecutar paso a paso el algoritmo de Dijkstra que computa el *camino de costo mínimo* entre un nodo dado y los restantes nodos de un grafo, sobre el grafo dado en el ejercicio 4.

   Considerar 1 como el nodo inicial. Explicitar en cada paso el conjunto de nodos para los cuales ya se ha computado el costo mínimo y el arreglo con tales costos.

**ejercicios adicionales**

10. Implementar el algoritmo de Prim utilizando (para encontrar la mínima arista con origen en $T$ y destino fuera de $T$) dos arreglos en los que se guarde para cada vértice fuera de $T$ su adyacente en $T$ más cercano (si hay) y la longitud de tal arista. Analizar su eficiencia.

11. ¿Qué sucede si se ejecutan los algoritmos de Prim y Kruskal sobre un grafo no conexo? ¿De qué manera pueden adaptarse los algoritmos de Prim y Kruskal para encontrar una familia de árboles generadores de costo mínimo, en el caso de un grafo no conexo?

12. Variante del ejercicio 3, donde ahora se conoce que el precio del litro de combustible en cada una de las localidades es $p_0, \ldots, p_n$. Se asume que con un litro de combustible se recorren exactamente $k$ kilómetros.

    Se pide dar un algoritmo voraz que determine el menor costo posible del viaje a realizar, sin importar cuántas veces sea necesario cargar combustible. No es necesario llenar el tanque cada vez que se carga, puede no convenir. Justificar con claridad.

    Se asume que el tanque esta inicialmente vacío y que no es necesario volver a cargar combustible una vez que se ha finalizado el viaje. También se asume que todas las estaciones de servicio cuentan con suficiente combustible.

13. Se conoce el valor actual $v_1^0, \ldots, v_n^0$ de las acciones de $n$ empresas. Afortunadamente se dispone de una "bola de cristal" que permite conocer el valor que tendrán las acciones durante los $m$ días subsiguientes. Es decir, los valores $v_1^1, \ldots, v_1^m$ que tendrán las acciones de la empresa 1 dentro de 1 día, ..., dentro de $m$ días respectivamente; los valores $v_2^1, \ldots, v_2^m$ que tendrán las acciones de la empresa 2 dentro de 1 día, ..., dentro de $m$ días respectivamente, etcétera. En general, $v_i^j$ es el valor que tendrá una acción de la empresa $i$ dentro de $j$ días. Estos datos vienen dados en una matriz $V[0..n, 0..m]$.

    Dar un algoritmo voraz que calcule el máximo dinero posible a obtener al cabo de los $m$ días comprando y vendiendo acciones, a partir de una suma inicial de dinero $D$.

    Se asume que siempre habrá suficientes acciones para comprar y que no se cobra comisión alguna por la compra y venta. También se asume que puede comprar una fracción de acción si no le alcanza para comprar una entera, en este caso la ganancia es proporcional a la fracción que se compró. Recordar que no siempre las acciones incrementan su valor.