Examen de Programación 3
17 de julio de 2023

En recuadros con este formato aparecerán aclaraciones que cumplen una función explicativa pero que no eran requeridos como parte de la solución.

Ejercicio 1 (35 puntos)

Una hospital planifica el uso eficiente del tiempo en su área quirúrgica ambulatoria. La intervención quirúrgica de cada paciente, $i \in \{1, ..., n\}$, consiste en una cirugía y un proceso de seguimiento posterior (postoperatorio). Los tiempos de cirugía y postoperatorio son todos diferentes y denotados según c_i y p_i, respectivamente. Mientras que el postoperatorio es independiente entre los pacientes, es decir su puede realizar simultáneamente, la cirugía no lo es, debido a que solo se puede realizar, en el quirófano, una cirugía a la vez. Se requiere determinar en qué orden intervenir a los pacientes de modo de minimizar el tiempo total de cirugía y postoperatorio de todos los pacientes.

Observación: no se requiere determinar tiempos de comienzo o fin de las cirugías, ni de los postoperatorios.

(a) Dé un algoritmo eficiente para resolver el problema.

(b) Demuestre la corrección de su algoritmo. **Sugerencia**: utilice la técnica *exchange argument*, aplicada a inversiones entre soluciones factibles.

Solución:

(a) Entrada: un conjunto de tuplas (c_i, p_i) con $i \in \{1, ..., n\}$
Salida: una secuencia $O = (o_1, ..., o_n)$, permutación de $\{1, ..., n\}$
Ordenar las cirugías según orden decreciente de su tiempo de postoperatorio; es decir, para todo par (o_i, o_{i+1}), con $i \in \{1, ..., n-1\}$, se cumple $p_{o_i} > p_{o_{i+1}}$.

(b) El tiempo de intervención acumulado hasta la cirugía del paciente o_i es la suma de los tiempos de cirugía de los pacientes previos más sus tiempos de cirugía y postoperatorio,

$$t_{o_i} := \sum_{j=1}^{i-1} c_{o_j} + c_{o_i} + p_{o_i}.$$

El tiempo de intervención en una secuencia O es el máximo tiempo de intervención hasta la crujía de alguno de sus pacientes,

$$T_O := \max_{i \in \{1,...,n\}} t_{o_i}.$$

Una secuencia óptima, $O^* = (o_1^*, ..., o_n^*)$, minimiza el tiempo máximo de intervención entre las secuencias posibles ($\mathcal{O}$),

$$T_{O^*} := \min_{O \in \mathcal{O}} T_O.$$

Se demuestra, mediante *exchange argument*, que dada la secuencia obtenida por el algoritmo, $O^* = (o_1^*, ..., o_n^*)$, y para toda secuencia factible $O = (o_1, ..., o_n)$, se cumple que $T_{O^*} \leq T_O$.

Si O difiere de O^* entonces existe al menos una inversión entre dos posiciones contiguas de los índices en O con respecto a O^*. Es decir, en O existen pacientes o_i y o_{i+1}, con $i \in \{1, ..., n-1\}$, donde o_i es intervenido antes que o_{i+1} y $p_{o_i} < p_{o_{i+1}}$. Se busca transformar O revertiendo dicha inversión de forma que O transformada se asemeje a O^*, sin empeorar el tiempo de intervención.

Sean o_i e o_{i+1} los pacientes de dicha inversión, a partir de

$$O = (o_1, ..., o_{i-1}, o_i, o_{i+1}, o_{i+2}, ..., o_n)$$

se genera la secuencia transformada (O'), en base a los índices de O, donde se revierte la inversión,

$$O' = (o_1, ..., o_{i-1}, o_{i+1}, o_i, o_{i+2}, ..., o_n).$$

Se tiene que los tiempos de intervención acumulados, t_{o_j}, de los pacientes previos, $o_1, ..., o_{i-1}$, y posteriores, $o_{i+2}, ..., o_n$, a la inversión no cambian luego de revertirla.
Los tiempos de intervención de la inversión pueden cambiar. Estos tiempos según secuencia son

$$t_{o_i}(O) := \sum_{j=1}^{i-1} c_{o_j} + c_{o_i} + p_{o_i}$$

$$t_{o_{i+1}}(O) := \sum_{j=1}^{i-1} c_{o_j} + c_{o_i} + c_{o_{i+1}} + p_{o_{i+1}}$$

$$t_{o_{i+1}}(O') := \sum_{j=1}^{i-1} c_{o_j} + c_{o_{i+1}} + p_{o_{i+1}}$$

$$t_{o_i}(O') := \sum_{j=1}^{i-1} c_{o_j} + c_{o_{i+1}} + c_{o_i} + p_{o_i}.$$

Donde se tiene que $t_{o_{i+1}}(O')$ es menor o igual que $t_{o_{i+1}}(O)$.
Para que $t_{o_i}(O')$ sea menor o igual que $t_{o_{i+1}}(O)$ se requiere que $p_{o_{i+1}} \geq p_{o_i}$; lo cual es el criterio del algoritmo para generar su secuencia solución. Usando dicho criterio se tiene que $T_{O'} \leq T_O$.

La eliminación reiterada de las inversiones de cualquier secuencia O con respecto a O^* mediante el criterio del algoritmo no aumenta el tiempo total de intervención en O^*. Por lo que el algoritmo determina una secuencia solución que es al menos tan buena como cualquier otra secuencia.

Ejercicio 2 (35 puntos)

La red de comunicación de datos *EasyData* consta de n routers y m enlaces bidireccionales (links) entre pares de routers. Se sabe que hay camino entre todo par de routers de la red (todo router es alcanzable desde cualquier otro en un número finito de saltos o "hops").
Los operadores de la red quieren determinar el tiempo mínimo de trasmisión de un paquete de datos desde un router origen s hasta un router destino t. El tiempo que demora un paquete de datos en ser enviado desde un cierto router u hasta un router vecino v con el que tiene enlace es la suma de dos tiempos conocidos:

- El tiempo de procesamiento del paquete en el router u, $TP(u)$,
- El tiempo de trasmisión del paquete desde u a v sobre el enlace (u, v), $TT(u, v)$.

El equipo de *EasyData* se plantea trabajar con la función $OPT(i, v)$, que representa el **tiempo mínimo** necesario para enviar un paquete desde el router origen s hasta el router v, recorriendo **a lo sumo** i enlaces (haciendo a lo sumo i saltos o "hops").

(a) Modele el problema con un grafo dirigido y especifique una relación de recurrencia para $OPT(i, v)$ que permita calcular el tiempo mínimo para enviar un paquete de datos desde el router origen s hacia el router destino t.

(b) Escriba un algoritmo **iterativo** usando la técnica de **programación dinámica** para calcular la recurrencia de la parte anterior, que retorne el valor del tiempo mínimo de envío de un paquete desde s a t. El algoritmo debe admitir una implementación con tiempo de ejecución polinomial en n y m.

(c) Calcule el orden del tiempo de ejecución del algoritmo anterior.

(d) Dé un algoritmo que devuelva una estrategia de tiempo mínimo, es decir una lista de los routers que atraviesa el paquete en su camino de tiempo mínimo desde el origen s hasta el router destino t (podría haber más de una estrategia óptima, se requiere sólo una de ellas). Puede asumir como disponibles estructuras auxiliares obtenidas en partes anteriores.

Solución:

(a) Modelamos a la red de comunicación como un grafo dirigido etiquetado $G = (V, E)$, en dónde el conjunto de nodos V de tamaño n representa a los routers de la red y por cada enlace entre dos routers vecinos u y v se tienen 2 aristas de E, (u, v) y (v, u) (2m aristas en total). La etiqueta de una arista (u, v) es la suma $TP(u) + TT(u, v)$, que es la demora total en el envío de un paquete desde u a v.
Como hay n nodos y los tiempos son todos positivos, el camino de tiempo mínimo entre el nodo fuente s y cualquier otro nodo v tiene a lo sumo largo $n - 1$. Teniendo esto en cuenta, la recurrencia para $OPT(i, v)$ queda:

$$\text{OPT}(0, v) = \begin{cases} 0 & v = s \\ +\infty & v \neq s \end{cases}$$

$$\text{OPT}(i, v) = \text{mín} \left\{ \text{OPT}(i-1, v), \min_{(u,v) \in E} \{\text{OPT}(i-1, u) + TP(u) + TT(u, v)\} \right\} 1 \leq i \leq n-1$$

(b) Para el cálculo de la función $OPT(i, v)$ se empleará una matriz OPT_{nxn} cuyas cuyas filas $0 \ldots n-1$ representan la cantidad de enlaces recorridos y cuyas columnas identifican a los n nodos de la red. El elemento $\text{OPT}(i, j)$ contiene el tiempo mínimo para enviar un paquete desde s a j recorriendo a lo sumo i enlaces.
La matriz OPT se va completando por filas, desde la fila 0 que implementa el paso base de la recurrencia hasta la fila $n - 1$. Los valores de cada fila a partir de la fila 1 se calculan en base a los de la fila anterior.

```
1  Algorithm TiempoMinimo (G = (V,E),s,t)
2      foreach u ∈ V, u ≠ s do
3          OPT[0, u] = +∞
4      end
5      OPT[0, s] = 0
6      for i=1 to n-1 do
7          foreach v ∈ V do
8              costo_min = +∞
9              foreach u ∈ V, (u, v) ∈ E do
10                 if OPT[i-1,u]+TP(u)+TT(u,v) <costo_min then
11                     costo_min = OPT[i − 1,u]+TP(u)+TT(u,v)
12                 end
13             end
14             if OPT[i-1,v] ≤ costo_min then
15                 OPT[i,v] = OPT[i − 1,v]
16             end
17             else
18                 OPT[i,v] = costo_min
19             end
20         end
21     end
22     return OPT[n − 1,t]
23 end
```

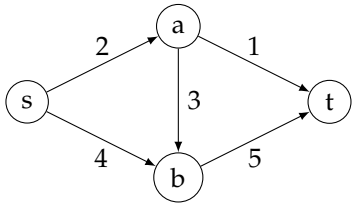
(c) El bucle de la línea 2 del algoritmo, de inicialización de la recurrencia, es $\mathcal{O}(n)$. El bucle *for* de la línea 6 se ejecuta $n-1$ veces. Con la representación del grafo dirigido mediante listas de adyacencia, el bucle *foreach* entre las líneas 7 y 20 visita una única vez cada arista, pues para cada nodo se visitan en su turno todas las aristas que inciden sobre él. Como el grafo tiene $2m$ aristas, resulta entonces que este bucle interior es $\mathcal{O}(m)$, lo que implica que el tiempo de ejecución del bucle *for* entre las líneas 6 y 21 es $\mathcal{O}(nm)$ y éste resulta ser el orden del tiempo de ejecución del algoritmo.

(d) Para hallar el camino de tiempo mínimo desde s a t, se parte del router destino t y utilizando la matriz OPT calculada en b), para cada router se determina cual router lo precede en el camino óptimo y se lo agrega al camino, hasta llegar a s.

```
1  Algorithm CaminoDeTiempoMinimo (G = (V,E),t,OPT)
2      Camino = CreoListaVacia()
3      InsertFront(t,Camino)
4      v= t
5      for i=n-1 downto 1 do
6          if OPT[i,v] ≠ OPT[i-1,v] then
7              u = argmin_{u/(u,v)∈E}{OPT[i − 1, u] + TP(u) + TT(u, v)}
8              InsertFront(u,Camino)
9              v = u
10         end
11     end
12     return Camino
13 end
```

Ejercicio 3 (30 puntos)

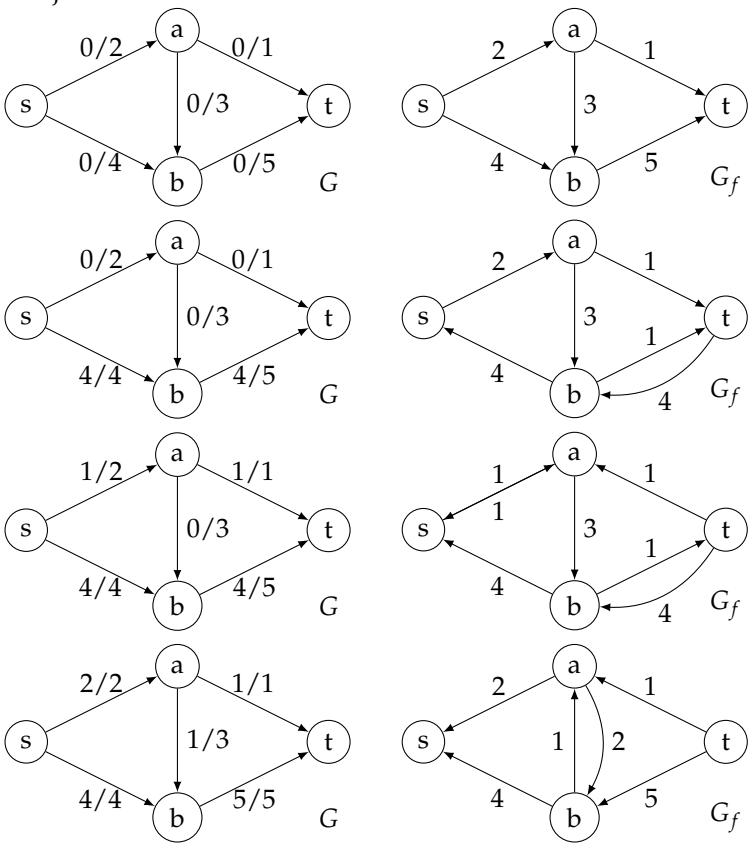
(a) Ejecute, paso a paso, el algoritmo de Ford-Fulkerson sobre la red de flujo de la figura. Recuerde dibujar la red G y el grafo residual G_f.



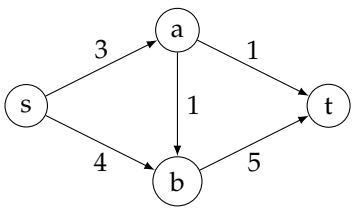
(b) La siguiente afirmación ¿es verdadera o falsa? En caso afirmativo verdadera, da una breve explicación. Si es falsa, dé un contraejemplo.
Sea G una red de flujo arbitraria, con una fuente s, un sumidero t y una capacidad entera positiva c_e en cada arista e; y sea (A, B) un corte $s-t$ mínimo con respecto a estas capacidades $\{c_e : e \in E\}$. Supongamos ahora que añadimos 1 a cada capacidad entonces (A, B) sigue siendo un corte $s-t$ mínimo con respecto a estas nuevas capacidades $\{1 + c_e : e \in E\}$.

Solución:

(a) La ejecución de FF es:



(b) La afirmación es falsa. Por ejemplo en la siguiente red de flujo,



El corte $(\{s, a\}, \{b, t\})$ tiene una capacidad de 6 y es mínimo. Sin embargo, al sumar 1 a cada arista, el mismo corte tiene capacidad 9 y sin embargo, un corte mínimo del esta nueva red, $(\{s, a, b\}, \{t\})$, es 8.