Examen de Programación 3
13 de diciembre de 2021

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

Ejercicio 1 (30 puntos)

Considere el algoritmo de Gale-Shapley para formar un emparejamiento estable entre dos conjuntos, $M = \{m_1, m_2, \ldots, m_n\}$ y $W = \{w_1, w_2, \ldots, w_n\}$. En la versión presentada en la figura 1, las preferencias de los elementos de M están determinadas por listas de preferencias que constituyen parte de la entrada del algoritmo (igual que en el libro de referencia), pero las preferencias de los elementos de W se resuelven en tiempo de ejecución mediante invocaciones a una función de booleana, $pref$. Para $w \in W$, $m \in M$ y $m' \in M$, el resultado de $pref(w, m, m')$ es **true** si w prefiere a m antes que a m' y **false** en caso contrario.

```
1  Algorithm Gale-Shapley
2      Inicialmente p está libre para todo p ∈ M ∪ W
3      while existe m ∈ M libre que no se ha propuesto a todo w ∈ W do
4          Sea w el elemento de W de mayor preferencia para m al cual m no se ha propuesto
5          if w está libre then
6              Emparejar m con w
7          else
8              Sea m' la actual pareja de w
9              if pref(w, m, m') then
10                 Separar a w de m' y emparejar m con w
11             else
12                 w rechaza a m
```

Figura 1: Algoritmo para formar un emparejamiento estable.

(a) Muestre que el algoritmo termina. Repita cualquier argumento que utilice de los estudiados en el curso.

(b) Supongamos que para toda instancia del problema se cumple, para todo $w \in W$ y $m' \in M$, que $pref(w, m, m')$ requiere tiempo $O(n)$ para $m = m_1$ y requiere tiempo $O(1)$ para $m \in M \setminus \{m_1\}$. Muestre que este algoritmo admite una implementación cuyo tiempo de ejecución es $O(n^2)$. Puede citar sin demostrar proposiciones y resultados presentados en el libro del curso; en particular con respecto al tiempo de ejecución de pasos que son iguales al algoritmo estudiado en el curso.

Solución:

(a) Ver (1.3) en el libro de referencia.

(b) Por la definición del paso 4, ningún $m \in M$ repite una propuesta a un mismo $w \in W$. Por lo tanto, m_1 realiza como máximo n propuestas, una para cada $w \in M$. En consecuencia, no hay más de n invocaciones a $pref$ con $m = m_1$ como argumento. El tiempo total incurrido en estas llamadas, T_1, es entonces $O(n^2)$. Por otra parte, la cantidad de invocaciones $pref(w, m, m')$ con $m \neq m_1$ no supera n^2 y cada una requiere tiempo $O(1)$, por lo cual el tiempo total insumido en estas llamadas, T_2, es también $O(n^2)$. Concluimos entonces que el tiempo total consumido en ejecuciones del paso 9, T_{pref}, es $O(n^2)$, ya que es la suma de T_1 y T_2, que ambos son $O(n^2)$. Con una implementación como la vista en el curso, el tiempo total requerido para el resto de los pasos, T_{resto}, es $O(n^2)$. El tiempo total requerido por este algoritmo es la suma de dos funciones que son $O(n^2)$, $T_{pref} + T_{resto}$, lo cual es también $O(n^2)$.

Ejercicio 2 (40 puntos)

Un *camino hamiltoniano* en un grafo dirigido es un camino que pasa por todos los vértices del grafo sin repetir ninguno. Sea $G = (V, E)$ un *DAG* (grafo dirigido y acíclico).

(a) Supongamos que G contiene un camino hamiltoneano $C = v_1, v_2, \ldots, v_n$. Muestre que C es un orden topológico para G.

(b) En las mismas condiciones que en la parte anterior, muestre que C es el **único** orden topológico para G.
Sugerencia: Sea $C' = w_1, w_2, \ldots, w_n$ un orden topológico para G. Notar que si $C \neq C'$, entonces C tiene al menos una inversión con respecto a C', esto es, un par de índices i, j, con $j > i$, tales que v_j aparece antes que v_i en C'.

(c) Muestre que G contiene un camino hamiltoniano si y solo si G tiene un único orden topológico.

(d) Escriba un algoritmo que dado un DAG $G = (V, E)$ determine si G tiene un camino hamiltoniano.

Solución:

(a) Por definición, un orden topoógico es un ordenamiento de los vértices de G $v_1, v_2, \ldots, v_n$, tal que para cada arista de G (v_i, v_j) se cumple que $i < j$. Supongamos por absurdo que (v_j, v_i) es una arista de G, con $j > i$. Entonces el subcamino de C, $v_i, \ldots, v_j$, junto con la arista (v_j, v_i) forman un ciclo, contradiciendo el hecho de que G es un DAG.

(b) Sea $C' = w_1, w_2, \ldots, w_n$ un orden topológico para G y supongamos por absurdo que $C \neq C'$. Entonces C tiene al menos una inversión con respecto a C', esto es, un par de índices i, j, con $j > i$, tales que v_j aparece antes que v_i en C'. Siguiendo las aristas recorridas por el camino C a partir de v_i, $(v_i, v_{i+1}), (v_{i+1}, v_{i+2}), \ldots, (v_{j-1}, v_j)$, a más tardar al llegar a (v_{j-1}, v_j) alguna de estas aristas debe ir hacia atrás con respecto al orden C'. Esto contradice el hecho de que C' es un orden topológico y la contradicción proviene de asumir que $C \neq C'$.

Otra demostración. Sea C' una secuencia genérica distinta de C en que aparecen todos los vértices de G exactamente una vez. Como las secuencias son diferentes hay al menos una inversión.

Sea i el menor índice que ocurre en una inversión, o sea, C' tiene como prefijo $v_1, \ldots, v_{i-1}, v_j$, con $j > i$. De $j - 1 > i - 1$ se sigue que v_{j-1} ocurre en C' después de v_j. Como C es un camino, (v_{j-1}, v_j) es una arista por lo que C' no es un ordenamiento topológico para G.

Por lo tanto, como C' es genérica, C es el único ordenamiento topológico para G.

(c) El directo queda demostrado por las partes anteriores. Para el recíproco, sea $O = v_1, v_2, \ldots, v_n$ el único orden topológico de G. Mostramos que O es un camino hamiltoniano en G. Notar que O contiene todos los vértices de G sin repetir ninguno, por lo cual solo resta probar que es un camino, es decir, que $(v_i, v_{i+1}) \in E$ para todo i, $1 \leq i < n$.

Supongamos por absurdo que existe i, $1 \leq i < n$, tal que $(v_i, v_{i+1}) \notin E$. Sea O' el ordenamiento de vértices que se obtiene a partir de O intercambiando los lugares de los vértices v_i, v_{i+1}. El orden relativo entre todo par de vértices se preserva en O' con respecto a O, con la única excepción de los vértices v_{i+1}, v_i, que aparecen en ese orden en O'. Como por hipótesis de absurdo $(v_i, v_{i+1}) \notin E$, ninguna arista va hacia atrás en O' porque no lo hacen en O. Por lo tanto existen al menos dos órdenes topológicos distintos, lo cual se contradice con que O es único.

(d) El algoritmo de la figura 2 resuelve el problema.

```
1  Algorithm ExisteCaminoHamiltoniano(G=(V,E))
2      if V tiene un único vértice then
3          retornar que existe camino hamiltoniano
4      else
5          Encontrar todos los vértices sin aristas entrantes en G.
6          if existe más de un vértice sin aristas entrantes en G then
7              retornar que no existe camino hamiltoniano.
8          else
9              Sea v el vértice sin aristas entrantes
10             Sea G' el grafo resultante luego de eliminar v de G y sus aristas adyacentes en E
11             retornar ExisteCaminoHamiltoniano(G')
12 end
```

Figura 2: Algoritmo para determinar si existe un camino hamiltonianon en G.

Una solución alternativa se presenta en el algoritmo de la figura **??**.

```
1  Algorithm ExisteCaminoHamiltoniano2(G=(V,E))
2      Sea L una lista vacía
3      Sea S un conjunto vacío
4      while S ≠ V do
5          Encontrar un vértice v sin aristas entrantes en G
6          Insertar v al final de L
7      foreach par de vértices consecutivos, v_i, v_{i+1}, en L do
8          if (v_i, v_{i+1}) ∉ E then
9              retornar que no existe camino hamiltoniano.
10     Retornar que existe camino hamiltoniano.
11 end
```

Figura 3: Algoritmo alternativo para determinar si existe un camino hamiltonianon en G.

Ejercicio 3 (30 puntos)

Para proporcionar un servicio médico adecuado a sus pacientes, los administradores de los hospitales deben buscar constantemente formas de mantener los niveles de personal en números adecuados. Un hospital tiene tres departamentos: la sala de emergencias (D_1), la sala de cuidados intermedios (D_2), y el CTI (D_3). Cada uno puede recibir **a diario** un máximo de R_1, R_2, y R_3 enfermeros. El hospital tiene tres turnos de trabajo, T_1,T_2, y T_3, cada uno con una cantidad de personal de enfermería disponible E_1, E_2, y E_3, respectivamente. El hospital desea identificar el número de enfermeras y enfermeros que debe asignar a cada departamento y cada turno sabiendo que el máximo número de enfermeras y enfermeros que puede asignarse a cada departamento en un turno específico es M_{ij}, donde i es el turno y j el departamento. Tenga en cuenta que cada enfermero o enfermera puede trabajar en sólo un departamento por turno

(a) Defina *Grafo Residual* de una red de flujo.

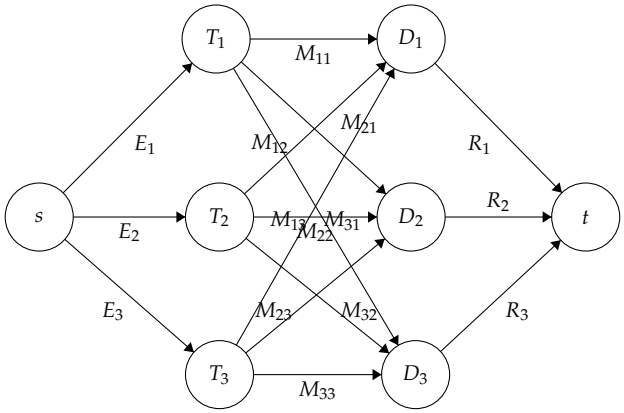
(b) Proponga un algoritmo basado en Redes de Flujo que asigne la mayor cantidad posible de personal en los tres turnos de un día cumpliendo con las restricciones mencionadas, devolviendo la cantidad de personal de cada turno asignado a cada departamento. Puede citar y utilizar algoritmos vistos en el curso sin reescribirlos.

Solución:

(a) El grafo residual G_F se define de la siguiente manera.

- El conjunto de nodos de G_f es el mismo que el de G.
- Para cada arista $e = (u, v)$ de G en la que $f(e) < c_e$, hay $c_e - f(e)$ unidades de capacidad "sobrantes" en las que podríamos intentar empujar el flujo. Entonces incluimos la arista $e = (u, v)$ en G_f, con una capacidad de $c_e - f(e)$. Llamamos a estas aristas aristas *forward*.
- Para cada arista $e = (u, v)$ de G en la que $f(e) > 0$, hay $f(e)$ unidades de flujo que podemos "deshacer"si queremos, empujando el flujo hacia atrás. Entonces incluimos la arista $e = (v, u)$ en G_f, con una capacidad de $f(e)$. Llamaremos a estas aristas aristas *backwards*.

(b) Se crea una red de flujo G como la que puede verse en la siguiente figura:



y se calculan los $f(e)$, para todos las aristas e de G, correspondientes a su flujo máximo con, por ejemplo, el algoritmo de Ford-Fulkerson. La cantidad máxima de personal del turno T_i asignada al departamento D_j es $f((T_i, D_j))$.