ISIS 2103 Diseño de algoritmos
Semestre 2011-10 Parcial 3
Mayo 12, 2011
Prof. Rodrigo Cardoso

Nombre: ______________________________
Sección: ________

1 **[40 puntos]** La Superintendencia Bancaria tiene un registro de préstamos que una entidad hace a otra en el país. Para simplificar, hay n entidades en el país, identificadas con los números en $1..n$. El registro de préstamos permite saber si la entidad i debe dinero a la entidad j. La Superintendencia está interesada en detectar si hay *autopréstamos* en el sistema, i.e., si existe una secuencia de entidades $j_1, j_2, \ldots, j_k$, tal que j_i le presta a j_{i+1}, para $1 \leq i < k$, y j_k le presta a j_1.

1a **[20/40]** Describa un algoritmo para decidir si hay autopréstamos en el sistema.
(N.B.: Se califican mejor los algoritmos más eficientes)

Considérese el grafo dirigido $G(V, \rightarrow)$, donde
$V=1..n$: (identificadores de) empresas
$.\rightarrow.$: relación de préstamos en V, i.e., $i \rightarrow j$ vale ssi i le presta a j.

Hay autopréstamos si la relación $\rightarrow$ no es acíclica, i.e. si G tiene ciclos. Al representar G con una matriz de conectividad A, el problema se reduce a verificar si, para algún $i \in V$, la pareja $(i,i) \in A^+$. Si esto vale, hay un ciclo en el grafo y la relación no es acíclica.

Variante 1:

Para $i \in V$: Algoritmo de Dijkstra en $(M_n(\mathbf{B}), \vee, \wedge, 0, 1)$

[15/20]

con parada forzada si "i es alcanzable$^+$ desde i"

[5/20]

Variante 2:

Algoritmo de Warshall

[10/20]

+
"Chequeo de $\neg(i \ A^+ \ i), i \in V$"

[5/20]

Variante 3 (ineficiente)

Calcular A^+ mediante multiplicaciones sucesivas en el semianillo $(M_n(\mathbf{B}), \vee, \wedge, 0, 1)$.

$A^+ = (\vee i | \ 1 \leq i < n \ : \ A^i)$

Más exactamente, sea $B_k = (\vee i | \ 1 \leq i < k \ : \ A^i)$, para $1 \leq k \leq n$. Entonces

$B_1 = A$.

Además, para $1 < k < n$:

$$
\begin{aligned}
B_{k+1} &= (\vee i | \ 1 \leq i < k+1 \ : \ A^i) \\
&= A(\vee i | \ 0 \leq i < k \ : \ A^i) \\
&= A(I \vee (\vee i | \ 1 \leq i < k \ : \ A^i)) \\
&= A(I \vee B_k)
\end{aligned}
$$

Finalmente: $B_n = A^+$.

[7/20]

1b **[20/40]** Estime las complejidades temporal y espacial de su algoritmo.

Variante 1:

Complejidad espacial:

Un vector d[1..n] adicional.

$S(n) = \theta(n)$

[5/20]

Complejidad temporal:

$$T(n) = n\ \theta(n^2)$$
$$= \theta(n^3).$$

[15/20]

Variante 2:

Complejidad espacial:

Una matriz $n \times n$ adicional.

$S(n) = \theta(n^2)$

[5/20]

Complejidad temporal:

$$T(n) = \theta(n^3) + \theta(n)$$
$$= \theta(n^3).$$

[15/20]

Variante 3:

Complejidad espacial:

Una matriz $n \times n$ adicional.

$S(n) = \theta(n^2)$

[5/20]

Complejidad temporal:

$$T(n) = (n-1)\theta(n^3)$$
$$= \theta(n^4).$$

[15/20]

2 **[40 puntos]** Sea $G(V,E,c)$ un grafo etiquetado que representa una red de oleoductos. Los nodos representan estaciones de la red y los arcos representan conexiones entre estaciones. Por razones de diseño, cada estación está conectada de manera directa, a lo sumo, con 3 estaciones.

El conjunto de estaciones es $V=1..n$, y se supone que $n>1$. Para $i,j\in V$, $c(i,j)$ es la capacidad de transporte de i a j por la conexión (i,j), con valores en el semianillo $(R^*,max,min,0,\infty)$; $cm(i,j)$ es la capacidad máxima de transporte de i hasta j, usando conexiones de la red.

Se quiere resolver el siguiente problema de optimización:

PP : Encontrar un nodo *principal* de la red, i.e., un nodo $x_0\in V$ tal que la capacidad máxima de transporte promedio desde el nodo x_0 a los demás sea máxima.

2a **[30/40]** Explique y justifique un algoritmo para resolver PP de manera óptima.

Es claro que, para `i,j∈V`:
`cm(i,j) = c*(i,j)`

Para un nodo `x`, sean

```
scm(x)    = (+y:V| y≠x : cm(x,y))      // suma de capacidades máximas de x a los demás
cmp(x)    = scm(x)/(n-1)               // capacidad máxima promedio de x a los demás
```

El predicado $ppal(x_0)$ es verdadero si y solo si x_0 es un nodo principal de la red.

Para resolver PP, se observa que

$ppal(x_0)$

=

$cmp(x_0) = (\max x \mid x \in V: cmp(x))$

=

$scm(x_0)/(n-1) = (\max x \mid x \in V: scm(x)/(n-1))$

=

$scm(x_0) = (\max x \mid x \in V: scm(x)))$

Es decir, x_0 es un principal si la suma de capacidades máximas se maximiza para x_0. Para calcular la suma de las capacidades máximas se requiere calcular todas las capacidades máximas de x_0 a todos los demás nodos, i.e., la matriz `C*`.

[10/30]

Se requiere calcular la matriz `C*`. Esto se puede realizar con `n` llamadas al algoritmo de Dijkstra.

[10/30]

Hay que sumar las filas de la matriz de distancias `C*` calculada con las `n` llamadas al algoritmo de Dijkstra.

[5/30]

Finalmente, se elige como x_0 el nodo cuya fila tenga una suma máxima.

[5/30]

El algoritmo podría escribirse así:

```
Dijkstra(G,Cx[.],1)
scm:= (+y| y≠1: Cx[y]);
x0:= 1;
for x:=2 to n →
      [1.1] Dijkstra(G,Cx[.],x)
            {Q: Cx[.] = c*[x,.]}
      [1.2] scmx:= (+y| y≠x : Cx[y]);
            if scmx>scm
                  then  x0:= x;
                        scm:= scmx
            fi
rof
{Q: ppal(x0)}
```

2b [10/40] Estime las complejidades temporal y espacial de su propuesta.

$S(G) = \theta(n)$ // el vector de distancias para Dijkstra

[5/10]

$T(G) = \theta(n^2 \log n)$

Cada llamada a Dijkstra es $\theta(n \log n + e)$. Ahora, el número de arcos e es $\theta(3n)$, o bien $\theta(n)$. Con una implementación adecuada (heaps, listas de adyacencia), se logra un desempeño $\theta(n \log n)$.

[5/10]

3 [20 puntos] Teniendo dos vasijas de `a` y `b` litros de capacidad, se quiere medir `c` litros de agua en alguna de las dos. Se puede trasvasar líquido entre las dos vasijas, pero no medir cantidades intermedias.

3a [10/20] Exprese el problema como una búsqueda en grafos.

```
SOLPOS = 0..a × 0..b
```

[3/10]

```
sat(x,y) ≡ x=c ∨ y=c
```

[2/10]

```
BUSQ = SOLPOS
```

[2/10]

```
(x,y) → (w,z) ≡  "(w,z) es resultado de trasvasar en (x,y)"
                  (Ver Notas de clase Cap 3, en Sicua+)
```

[2/10]

```
s = (0,0)
```

[1/10]

3b [10/20] Justifique si (i) hay que marcar nodos (ii) hay que verificar que la agenda se vacíe. (iii) el algoritmo puede no terminar.

(i) Hay que marcar. El grafo tiene ciclos.

[4/10]

(ii) Sí: puede no haber solución por agenda vacía.

[3/10]

(iii) No: el número de nodos en `BUSQ` es finito. Debe terminar.

[3/10]