ISIS 1105 Diseño de algoritmos **Nombre: _____**
Semestre 2014-20 Parcial 3
Noviembre 12, 2014
Prof. Rodrigo Cardoso

N.B: Cuando se soliciten algoritmos se esperan soluciones eficientes en tiempo y en espacio. Por tanto, soluciones menos eficientes que lo esperado pueden ser penalizadas. Si se pide el diseño de un algoritmo se pueden utilizar referencias a algoritmos conocidos, así como a sus costos computacionales.

1 [40/100] *Máxima capacidad de conducción*

Sea $G(V,E,c)$ un grafo dirigido que modela una red de conducción y $c:V\times V\to\mathbf{R}^*$ una función que etiqueta los arcos de modo que $c(x,y)$ es la capacidad de llevar material de x a y. La capacidad de un camino en el grafo es definida como el mínimo de las capacidades de los arcos que lo componen.
Suponga que $\#V=n$ y $\#E=e$.

1a (15/40) Considere el problema de determinar las capacidades máximas de conducción entre cada par de nodos del grafo. Describa un algoritmo que calcule estas capacidades y estime complejidades en tiempo y espacio de su solución.

Se puede usar Floyd Warshall generalizado

[4/15]

con el semianillo $(\mathbf{R}^*, \max, \min, 0, \infty)$.

[4/15]

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

[3/15]

$S(n) = \theta(n^2)$ // si no se permite dañar la entrada
$S(n) = \theta(1)$ // si se permite dañar la entrada

[4/15]

1b (15/40) Suponga que hay dos nodos distinguidos, $f, s \in V$. Considere el problema de determinar la capacidad máxima de conducción de f (*fuente*) a s (*sumidero*). Describa un algoritmo que calcule esta capacidad, argumente por qué debe ser correcto y estime complejidades en tiempo y espacio de su solución.

Nótese que $x \min x = x$.

Entonces, se puede usar Dijkstra generalizado

[4/15]

con el semianillo $(\mathbf{R}^*, \max, \min, 0, \infty)$.

[4/15]

$T(n) = \theta(n \log n + e)$

[3/15]

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

[4/15]

1c **(10/40)** Suponga que `G` no tiene ciclos. ¿Qué mejoras en complejidades puede señalar para los algoritmos descritos en **1a** y en **1b**?

Para 1a : Ninguna ventaja. Floyd-Warshall no distingue nodos visitados.

[5/10]

Para 1b : Se evita marcar nodos.

No hay mejora en el orden de complejidad de `S(n)`, porque `n` corresponde tanto al vector de capacidades óptimas calculadas como al montón en que se llevan los no marcados.

[5/10]

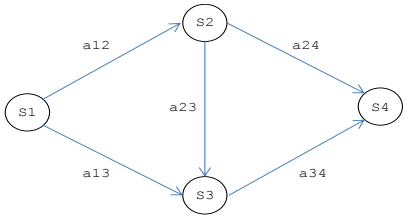
2 [20/100] *Factibilidad y chequeo de obra*

Un proyecto de construcción puede modelarse como un grafo `G(S,A,d)` para el que:

- `S` es un conjunto de *estados*, cada uno de los cuales resume en su descripción el avance de la obra.
- `A` es un conjunto de actividades, cada una de las cuales representa una transición entre dos estados. Para realizar una actividad `a` deben haber terminado todas las que finalizan en el estado en que `a` comienza.

Supóngase que se `#S=s` y `#A=a`.

En la figura, se muestra un ejemplo en el que hay `s=4` estados y `a=5` actividades:



2a **(20/20)** Los administradores del proyecto desean establecer un orden en el tiempo en los estados, para efectuar chequeos de adelanto de la obra. Describa un algoritmo que defina un tal ordenamiento si algo así es posible; si no lo es, su método debe reconocer la situación. Estime complejidades temporal y espacial.

Un algoritmo que solucione el problema debe establecer un orden topológico para los estados del proyecto.

[5/10]

Es posible una implementación con DFS (v. Cormen), que incluye la eventual detección de ciclos de estados, lo que haría imposible el ordenamiento.

[5/10]

El sort topológico mediante DFS tiene las siguientes complejidades:

$T(s,a) = \theta(s + a)$

[5/10]

$S(s,a) = \theta(s)$

[5/10]

2b [Bono] Describa un algoritmo que permita establecer si la obra es factible, en el sentido de que toda actividad se pueda realizar. Estime complejidades temporal y espacial de su algoritmo.

Se debe producir un orden topológico para las actividades.

El grafo $G(S,A)$ no es adecuado para usar el algoritmo de 2a, porque las actividades están en los arcos. El grafo requerido se puede construir con un DFS (suponiendo G original representado como lista de adyacencias) en $\theta(s+a)$, espacio $S(s,a)=\theta(a)$.

Se llega a un grafo con a vértices y s arcos, el cual se puede ordenar topológicamente en tiempo $\theta(a+s)$ y espacio $\theta(a)$.

En total:

$T(s,a) = \theta(s + a)$

$S(s,a) = \theta(\max(s,a))$

[+10]

3 [40/100] *El ahorcado*

En el *juego del ahorcado* un jugador busca averiguar una palabra, proponiendo letras del alfabeto que pueden estar en ella.

- Al comenzar, el jugador sabe solo la longitud de la palabra, representada con guiones en el lugar de las letras que se desconocen, y tiene una cuenta de 0 errores cometidos.
- En cada jugada, el jugador propone una letra, de modo que, si ésta aparece en la palabra, se despliega en todas las posiciones en que esté y la cuenta de errores no varía. En otro caso, la cuenta de errores aumenta en 1.
- El jugador gana si adivina la palabra antes de que la cuenta de errores llegue a un máximo preestablecido.

Suponga que:

- Juega con un alfabeto A de n letras, el cual no incluye el carácter '_'.
- $A1 = A \cup \{\_\}$.
- Se trata de adivinar una palabra $p \in A^m$ (i.e, p tiene m letras de A)
- La cuenta de errores no debe llegar a $h>0$.

3a (20/40) Exprese el problema como una búsqueda en grafos, i.e., defina SOLPOS, sat, SOL, BUSQ, s, →.

$SOLPOS = A^m \times 2^A$ // palabras de m letras, conjuntos de letras fallidas durante el proceso

[5/20]

$sat: SOLPOS \rightarrow bool.$
$sat(x,B) \equiv x=p \wedge \#B<h$

[4/20]

$SOL = \{(p,B) \mid \#B<h\}$

[1/20]

$BUSQ = \{(x,B) \mid x \in A1^m, B \subseteq A\}$
$= A1^m \times 2^A$

[4/20]

$s = (\_^m, \emptyset)$: (palabra de m guiones, conjunto vacío de errores)

[2/20]

$(x,B) \rightarrow (x, B \cup \{a\})$, si $a \notin p$ (la letra a no está en la palabra p)
$(x,B) \rightarrow (x+a, B)$, si $a \in B$ ($x+a$ es como x, aumentada con todas las apariciones de a)

[4/20]

3b **(15/40)** Justifique si

(i) hay que marcar nodos;

No es necesario. Incluso, si se repiten letras que se han adivinado, no se cambia de estado (palabra parcialmente adivinada, conjunto de letras fallidas).

[5/15]

(ii) hay que verificar que la agenda se vacíe;

No: si se jugara sin límite de intentos, el juego terminaría por llegar a la solución, a menos que el jugador repitiera letras ya propuestas.

[5/15]

(iii) el algoritmo puede no terminar.

Sí, si el jugador propone letras que están en la palabra y ya han sido propuestas.
No, de lo contrario: termina porque se alcanza h o porque se adivina la palabra.

[5/15]

3c **(5/40)** Calcule $|BUSQ|$ y $|\rightarrow|$, como estimación de la complejidad temporal de su algoritmo.

$$\#BUSQ = \#(A1^m \times 2^A)$$
$$= (n+1)^m * 2^n$$

[5/5]

3d [Bono] Proponga una heurística para tratar de ganar el juego.

Averiguar la frecuencia de las letras de A en las palabras.

Heurística: no repetir letras y proponer primero las letras más frecuentes.

No es, necesariamente, una heurística admisible.

[+5]