Estructura de datos y teoría de algoritmos
Tarea 1

Fabián Romero

November 27, 2013

Pregunta 1 Demuestra que todo torneo tiene un rey y un súbdito. Es decir, un vértice desde el cual se puede llegar a todos los demás vértices pasando a lo más por dos arcos, y uno al cual todos pueden llegar pasando a lo más por dos arcos

Inducción: Demostremos por inducción que cada torneo tiene un rey
El caso base es la gráfica de 1 vértice, que por vacuidad del conjunto de aristas se tiene que el único vértice es rey.
Asumamos que es cierto que cualquier torneo de n vértices tiene un rey.
Sea G un torneo de $n+1$ vértices y tomemos un vértice cualquiera v, entendamos G_{-v} la grafíca G eliminando v y todas las aristas incidentes a v, G_{-v} es un torneo, pues G es un torneo y entre cualquiera dos vertices hay una arista. Por hipótesis de inducción siendo G_{-v} un torneo de n vértices, tiene un rey, sea $r_{G_{-v}}$ dicho rey, por definición, sabemos que $\forall x \in vertices(G_{-v})$ son dominados por $r_{G_{-v}}$ directamente, o son dominados por $r_{G_{-v}}$ en 2 arcos, llamemos δ_r al conjunto de vertices que son directamente dominados y γ_r al conjunto de vertices que son dominados por dos arcos. Observacion i: si un vértice está en γ_r significa que hay un vértice en δ_r que lo domina directamente.

Analicemos a G. Si $r_{G_{-v}} \to v$ dado que $r_{G_{-v}}$ es rey en G_{-v}, entonces $r_{G_{-v}}$ es rey en G y por lo tanto G tiene un rey.
En caso de que $v \to r_{G_{-v}}$ Observemos que si un pasa que $x \to v$ para algun vértice $x \in \gamma_r$ entonces $r_{G_{-v}}$ es un rey en G, pues $r_{G_{-v}}$ es rey en G_{-v} y domina en 2 arcos a v en G. Así entonces solo queda el caso

de que $v \to r_{G_{-v}}$ y $\forall x \in \gamma_r v \to x$, pero en ese caso v domina a todo vertice en δ_r y por la observación i domina en (cuando más) dos arcos a todo vértice en γ_r y por lo tanto v es rey en G ■.

Reducción al absurdo: Probaremos un enunciado todavía mas estricto, el enunciado a probar es:
Cada torneo tiene al menos un rey, y el vértice v con mayor número de elementos dominados es uno de ellos.
Supongamos que no es cierto, es decir: Hay un torneo G donde el vértice v con mayor número de elementos dominados no es rey.
Sea δ_v el conjunto de todos los vértices dominados directamente por v en G.
Como v no es rey, podemos afirmar que hay en G un vértice x que no es dominado por v ni directamente ni en 2 arcos, es decir $x \to v$ y $x \to v_i \forall v_i \in \delta_v$. Observemos el número de vértices dominados por x, es al menos el número de elementos en δ_v más 1 pues $x \to v$, es decir mayor al de v lo cual contradice nuestra hipotesis ■.

Existencia del súbdito: Para demostrar que cada torneo es un súbdito, basta notar que un rey en un torneo G es un súbdito en el torneo $\widehat{G}$ que se obtiene cambiando la orientación de cada arista en G, así, por su significado simétrico, como cada torneo tiene un rey, cada torneo tiene un súbdito.

Pregunta 2 Presenta un algoritmo para encontrar un rey y analiza su complejidad y correctez.

Algoritmo El algoritmo es el sigiente: Recorrase el torneo buscando el elemento que tiene mayor número de elementos dominados.

```
maximo = none
rey = none
for {v | v in vertices(G)}:
    if |v| > maximo:
      maximo = |v|
      rey = v
```

Este algoritmo es correcto en encontrar el vértice de mayor grado, pues busca exhaustivamente.
Asumiendo que la representación de G es la gráfica de adyacencia, el algoritmo es del orden de $O(n+m)$, pues cada vértice se recorre una vez y ya en el y solo se toma el tamaño de la lista de adyacencia, lo cual es en el caso de una lista ligada $O(m)$. Por la prueba por demostración al absurdo, sabemos que el vértice con mayor número de elementos dominados es un rey, lo cual hace al algoritmo correcto en buscar al rey.

Pregunta 3 Demuestra que todo torneo tiene un camino dirigido Hamiltoniano.

Por inducción: El caso base es la gráfica de 1 vértice, el camino es el mismo vértice.
Supongamos que es cierto que todo torneo de tamaño n tiene un camino Hamiltoniano, y demostremos que un torneo de tamaño $n+1$ tambien lo tiene.

Análogamente a la pregunta 1 tomemos un vértice v y la gŕafica G_{-v} inducida al eliminarlo de G, sabemos que en G_{-v} hay un camino Hamiltoniano por hipotesis de inducción, sea $H = (v_1, v_2, ..., v_n)$ tal camino. Si $v \to v_1$, $(v, v_1, v_2, ..., v_n)$ es un camino Hamiltoniano en G.
en caso contrario $v_1 \to v$, sea v_i el primer elemento en H tal que $v \to v_i$, ahi tenemos que $(v_1, v_2, ., v_{i-1}, v, v_i, .., v_n)$ es un camino Hamiltoniano en G. Y si no hay un primero elemento, es decir cada vértice en H domina a v $(v_1, v_2, ..., v_n, v)$ es un camino Hamiltoniano en G■.

Pregunta 4 Prueba que en un torneo aleatorio de n vértices, la probabilidad de que todo vértice sea un rey y un subdito tiende a 1 conforme n tiende a infinito.

Respuesta sea $p(n)$ la probabilidad de que cada vértice sea rey (observese que por simetría es la misma que cada vertice sea súbdito), y denotemos $\overline{p}(n)$ su complemento 1 - $p(n)$ la probabilidad de que no cada vértice sea rey.

Si no todo vértice es rey, entonces existen vértices x_1 y x_2 tal que $x_2 \to x_1$ y $\forall v \in G x_1 \to v \Rightarrow d \to x_2$ sea ψ la probabilidad de que un par de vertices cualquiera (x_1, x_2) cumpla con las caracteristicas descritas.

Puesto que hay $n \times (n-1)$ pares en G tenemos que

$$\overline{p}(n) \leq n \times (n-1) \times \psi$$

dado que $\lim_{n\to\infty} n \times (n-1) = \infty$ y que $1 \geq \psi \geq 0$ se desprende que $\lim_{n\to\infty} \overline{p}(n) = 0$ y por lo tanto $\lim_{n\to\infty} p(n) = 1$■

Pregunta 5 Sean T1 y T2 dos torneos sobre el mismo conjunto de vértices. Demuestra la existencia de un vértice desde el cual se puede llegar a todos los demás en un arco de T1 , o un arco de T2 , o un arco de T1 seguido de uno en T2.

Llamemos rey doble, al vértice que cumple la condicion anterior

Inducción Para el caso de los torneos de Tamaño 1, el único vértice es el rey, tanto en $T1$ como en $T2$ por vacuidad en el número de aristas.

denotemos $x \Rightarrow y, x, y \in V$ si

$$(x \to y \in T1) \vee (x \to y \in T2) \vee (\exists z \in V | x \to z \in T1 \wedge z \to y \in T2)$$

Supongamos que es cierto que para todo torneo de tamaño n existe un rey doble por demostrar, existe un rey noble en todo par de torneos sobre el mismo conjunto de n vértices V.
sea j un vértice cualquiera en V, en las gráfica inducida V_{-j} por los 2 Torneos $T1$ y $T2$ al retirar j por hipotesis de inducción hay un rey doble r en V_{-j}.
Caso 1: $r \to j$ esta en $T1$ o en $T2$ por lo tanto r es un rey

Caso 2: $j \to t$ tanto en $T1$ como en $T2$, considere $\delta_{T1}(r)$ que es el conjunto de los vértices dominados directamente en T1 por r, si alguno de ellos domina a j en $T2$ r seria rey doble, por lo que j domina en $T2$ a todo vértice en $\delta_{T1}(r)$ pero como $j \to t$ en $T1$ y $j \Rightarrow \delta_{T1}(r)$ luego $j \Rightarrow \delta_{T2}(r)$ puesto que $j \to t$ en $T1$ y también $j \Rightarrow \delta_{T2}(\delta_{T1}(r))$ puesto que $j \to \delta_{T1}(r)$ en $T1$, pero por r ser rey doble $r \cup \delta_{T1}(r) \cup \delta_{T2}(r) \cup \delta_{T2}(\delta_{T1}(r)) = V_{-j}$ j es rey doble en V■

Pregunta 6 El algoritmo de BFS visto en clase utiliza una cola de vértices, y cada vértice tiene asignado un color (gris,blanco,negro), y un estimado de distancia d. El estimado d(v) siempre es mayor o igual a la distancia

del origen a v, y al final del algoritmo, d(v) es igual a la distancia del origen a v. Presenta y demuestra las invariantes que satisfacen los vértices en la cola, con respecto a d y a su color.

Por claridad cambiemos (gris,blanco,negro) por (encontrado,visitados y no visitado)

```
1 def BFS(G, v):
2     create a queue Encontrados
3     create a set Visitados
4     create queue Ordenados
5     enqueue v onto Encontrados
6     while Encontrados is not empty:
7         t <- Encontrados.dequeue()
8         for all vertex u in G.adjacentEdges(t) do
9             if u is not in Ordenados:
10                if u is not in Encontrados:
11                    enqueue u into Encontrados
12            add u into Ordenados
```

invariantes

- En cada iteración de (6) el conjunto Ordenados aumenta en uno su tamaño.
  Lo cual es cierto pues 12 es ejecutado incondicionalmente una vez cada que se pasa por 6.
- Cada elemento entra una unica vez y sale una unica vez de Encontrados.
  Esto se debe a que 11 es condicion de 9 y de 10, es decir, que se añade a Encontrados, si no esta en Encontrados o en Ordenados, es decir que se visita por primera vez, de ahi que solo entra una vez a Encontrados

Pregunta 7 Adapta el algoritmo del emparejamiento estable para instancias de n hombres y m mujeres en donde n = m. El objetivo es encontrar un emparejamiento estable de cardinalidad máxima, con la definición original de emparejamiento estable (es decir, que no hay alguna pareja inestable). Demuestra que tu adaptación siempre termina y que da un emparejamiento estable de cardinalidad máxima posible.

¿Cuál es el tiempo de ejecución del algoritmo en términos de n y m? ¿Se puede caracterizar el tamaño del emparejamiento estable de máxima

cardinalidad en términos de n y m? ¿Tu adaptacién sigue optimizando individualmente a cada hombre (cada hombre termina con su mejor pareja válida)?

Respuesta Adaptación: supongamos que $n > m$, lo que hacemos es crear $n - m$ mujeres ficticias de tal forma que cada hombre siempre prefiera a cualquier mujer real sobre cualquier mujer ficticia, en caso de $m > n$ creamos $m - n$ hobres ficticios, de tal forma que cada mujer prefiera a cualquier hombre real sobre cualquier hombre ficticio. en el caso $n = m$ es el visto en clase y alimentamos con estos datos el algoritmo $G - S$.

Tiempo de ejecución Como lo "reducimos" al caso visto en clase cuya complejidad es de $\theta(n^2)$ añadiendo elementos ficticios para hacerlo cuadricular, la complejidad es de $\theta(max(n, m)^2)$

Optimilidad Sí, esto se debe a que si un hombre obtiene su mejor pareja válida (real o ficticia), en el caso de que sea ficticia, como el prefiere a cualquier mujer real sobre cualquier ficticia por construcción quiere decir que no habia pareja válida real para el y por lo tanto la mujer que recibe (i.e. ninguna) es la mejor pareja válida, en caso de que le toque pareja real, la conclusión se sigue de $G - S$ directamente.

Pregunta 8 Sean P y Q dos programas que ordenan números. P lo hace con mergesort, cuya complejidad es $\theta(nlogn)$ y Q con BubbleSort, cuya complejidad es $\theta(n^2)$. Juanito Hacker dice que corrio a P y Q en la misma computadora, dándoles como entrada a ambos una lista de $1,000,000$ de elementos, y que consistentemente P hizo 1000 veces más operaciones más que Q. ¿Es esto posible?

Respuesta Esto es perfectamente posible y en el casos de BubbleSort y de MergeSort fácilmente explicable, pues resulta que BubbleSort es $\theta(n^2)$ en el peor de los casos y $\theta(n)$ en el caso ideal, en tanto que MergeSort tiene una complejidad $\theta(nlogn)$ en el peor caso y tambien en el caso esperado por lo que existen instancias donde BubbleSort es lineal y Mergesort $\theta(nlogn)$ (por ejemplo cuando la entrada esta ordenada o solo tiene un número de transposiciones de elementos consecutivos sobre la lista ordenada), y como $n = 1,000,000 = 1,000^2$ $nlogn$ es 1000 veces n, lo cual coincide perfectamente con los datos de Juanito Hacker, que aqui entre nos, parece ser un hacker de peluche si no entiende la ejecución

y complejidad de los programas y lo sorprende en lo más mínimo este resultado.

Pregunta 9 Sean P y Q dos programas concretos de complejidades $\theta(n^2)$ y $\theta(n)$ respectivamente que resuelven el mismo problema. Ante una instancia I de tamaño n, P hace 100 veces más operaciones que Q. Si ejecutamos P y Q con otra instancia I de tamaño n, ¿que podemos asegurar respecto al número de operaciones de ambos programas?

Respuesta Nada absolutamente, $\theta(n^2)$ y $\theta(n)$ son construcciones no acotadas por un tamaño específico, es decir un programa puede tomar cualquier tiempo arbitrario para una instancia y segir siendo $\theta(n^2)$ o $\theta(n)$, estas notaciones solo explican el comportamiento *asintótico*, una instancia especifica, o de hecho un tamaño específico de la entrada no tiene ninguna garantía de ningún tipo.

Pregunta 8 Supón que tienes programas concretos que se ejecutan hoy en una computadora a cierta velocidad. Supón que dentro de dos años tendrás una computadora el doble de rápida. ¿Qué tan grandes serán los tamaños de los problemas que podrás resolver en dos años, usando una hora de procesamiento, respecto a los tamaños máximos de los problemas que puedes resolver hoy, usando una hora de procesamiento, si tus complejidades son algunas de las siguientes:

$\theta(log(n))$, $\theta(log^2 n)$, $\theta(n)$, $\theta(nlogn)$, $\theta(n^2)$, $\theta(n^2 logn)$, $\theta(n^3)$, $\theta(1.1^n)$, $\theta(2^n)$ y $\theta(3^n)$ Por supuesto ignora detalles de arquitectura (tamaño de memoria, jerarquía de memoria, etc.) y concéntrate unicamente en las características de escalabilidad teóricas que te da cada complejidad diferente.

Respuesta Supongamos para la simplicidad que si aqui decimos que un programa es de $\theta(n^2)$ sea exactamente $C * n^2 \forall |I| = n$ con C una constante arbitraria pero fija, y más aún, supongamos por simplicidad que esa constante C es tal que por haga que para cada problema, pudiera resolver una entrada de tamaño 1000 el dia de hoy. así teniendo mañana una computadora 2 veces más rápida nos daría:

Orden	Tamaño hoy	Tamaño mañana
$\theta(log(n))$	1,000	1,000,000
$\theta(log^2(n))$	1,000	17,484
$\theta(n)$	1,000	2,000
$\theta(nlog(n))$	1,000	1,838
$\theta(n^2)$	1,000	1,412
$\theta(n^2log(n))$	1,000	1,382
$\theta(n^3)$	1,000	1,260
$\theta(1.1^n)$	1,000	1,007
$\theta(2^n)$	1,000	1,001
$\theta(3^n)$	1,000	1,000.6