Diseño y Análisis de Algoritmos
Examen 1

Santiago Sinisterra Sierra

13 de enero de 2021

1. Problema 1

Para que $f(n)$ sea $\Theta(n^3)$ es necesario encontrar dos constantes c_1 y c_2 así como un valor n_0 que haga que la Ec.(1) se cumpla.

$$c_1 n^3 \leq f(n) \leq c_2 n^3 \tag{1}$$

En primera instancia se desarrolla $f(n)$, el cual se muestra en la Ec.(2)

$$\begin{aligned} f(n) &= 32n(\frac{n-1}{2}+3) + 8n^2 + 10n + 4 \\ f(n) &= 32n(\frac{n^2-n+6}{2}) + 8n^2 + 10n + 4 \\ f(n) &= 16n(n^2-n+6) + 8n^2 + 10n + 4 \\ f(n) &= 16n^3 - 16n^2 + 96n + 8n^2 + 10n + 4 \\ f(n) &= 16n^3 - 8n^2 + 106n + 4 \end{aligned} \tag{2}$$

En la Ec.(3) se calcula c_1, resolviendo la desigualdad de la primera línea.

$$\begin{aligned} c_1 n^3 &\leq 16n^3 - 8n^2 + 106n + 4 \\ c_1 n^3 - 16n^3 &\leq -8n^2 + 106n + 4 \\ (c_1 - 16)n^3 &\leq -8n^2 + 106n + 4 \\ \frac{(c_1-16)n^3}{n^3} &\leq \frac{-8n^2+106n+4}{n^3} \\ (c_1 - 16) &\leq \frac{-8}{n} + \frac{106}{n^2} + \frac{4}{n^3} \\ (c_1 - 16) &\leq \frac{-8}{n} + \frac{106}{n^2} + \frac{4}{n^3} \\ c_1 &\leq \frac{-8}{n} + \frac{106}{n^2} + \frac{4}{n^3} + 16 \end{aligned} \tag{3}$$

En la Ec.(4) se calcula c_2, resolviendo la desigualdad de la primera línea.

$$
\begin{aligned}
c_2 n^3 &\geq 16n^3 - 8n^2 + 106n + 4 \\
c_2 n^3 - 16n^3 &\geq -8n^2 + 106n + 4 \\
(c_2 - 16)n^3 &\geq -8n^2 + 106n + 4 \\
\frac{(c_2 - 16)n^3}{n^3} &\geq \frac{-8n^2 + 106n + 4}{n^3} \\
(c_2 - 16) &\geq \frac{-8}{n} + \frac{106}{n^2} + \frac{4}{n^3} \\
(c_2 - 16) &\geq \frac{-8}{n} + \frac{106}{n^2} + \frac{4}{n^3} \\
c_2 &\geq \frac{-8}{n} + \frac{106}{n^2} + \frac{4}{n^3} + 16
\end{aligned} \tag{4}
$$

Evaluando la Ec.(3)con $n = n_0$ con un n_0 seleccionado de 20 para obtener c_1

$$
\begin{aligned}
c_1 &\leq \frac{-8}{n} + \frac{106}{n^2} + \frac{4}{n^3} + 16 \\
c_1 &\leq \frac{-8}{20} + \frac{106}{20^2} + \frac{4}{20^3} + 16 \\
c_1 &\leq \frac{31731}{2000}
\end{aligned} \tag{5}
$$

Evaluando la Ec.(6) con $n = n_0$ con un n_0 seleccionado de 20 para obtener c_2

$$
\begin{aligned}
c_2 &\geq \frac{-8}{n} + \frac{106}{n^2} + \frac{4}{n^3} + 16 \\
c_2 &\geq \frac{-8}{20} + \frac{106}{20^2} + \frac{4}{20^3} + 16 \\
c_2 &\geq \frac{31731}{2000}
\end{aligned} \tag{6}
$$

Con un c_1 y c_2 calculado, se sustituye la propiedad inicial en la Ec.(1), resultando en la Ec.(7)

$$
\begin{aligned}
c_1 n^3 \leq & f(n) \leq c_2 n^3 \\
\frac{31731}{2000}(20)^3 \leq & f(20) \leq \frac{31731}{2000} n^3 \\
126924 \leq & 126924 \leq 126924
\end{aligned} \tag{7}
$$

Al realizar la evaluación de la Ec.(7) y verificarse que se cumple la propiedad, se completa la demostración.

2. Problema 2

Sólo hay un orden topológico: a, c, b, d.

Siempre se inicia por a porque es el único nodo sin aristas que entran. b no puede ser el segundo elemento porque no hay forma de eliminar la arista (c, b) sin antes agregar al orden a c; por lo que c es el segundo elemento y c el tercero. d tiene que ser el último elemento porque no sale ninguna arista del mismo.

3. Problema 3

3.1. (a)

$O(f(n))$ es n^3, ya que primero hay que recorrer i de 1 a n. Para cada elemento en i, hay que recorrer j hasta n. Finalmente, para B_{ij} hay que sumar en un rango, el cual es una operación a la que le toma n pasos. Multiplicando, resulta en $n \cdot n \cdot n = n^3$. Ya que es posible encontrar una c (1) y n_0 (1) tal que $cg(n) \geq f(n)$, $f(n)$ forma parte de O.

3.2. (b)

Para que $f(n)$ sea $\Omega(n^3)$ se debe encontrar una c (1) y n_0 (1) tal que $cg(n) \leq f(n)$, $f(n)$ forma parte de Ω.

Como previamente se comprobó que $f(n)$ es tanto $O(n^3)$ como $\Omega(n^3)$; $f(n)$ pertenece a $\Theta(n^3)$ ya que la condición es que debe pertenecer tanto a O como a Ω.

3.3. (c)

En lugar de sumar de A_i hasta A_j, se reemplaza la línea 3 del algoritmo 1 por $B_{i,j} < -Bi, j - 1 + A_j$. Se elimina la n extra porque la suma y el acceso tienen costo constante. El pseudocódigo queda en la Fig.(3.3)

```
for i = 1, 2, ..., n do
  for j = i + 1, i + 2, ..., n do
      Bij <- B[i, j-1] + A[j]
  end
end
```

Figura 1: Fig(3.3.)

Este algoritmo pertenece a $O(n^2)$, al calcular el límite se divide $\frac{n^2}{n^2} = 1$, donde su límite es 0.

```
ciclo_detectado <- false
explorados <- [(n, false) for n in nodos(G)]

def DFS(G, u):
  marcar la arista u como explorada
  foreach Arista(u,v) do:
    if not marcado(v):
      DFS(G, v)
    else
      ciclo_detectado <- true
    end
  end
return ciclo_detectado == false
```

Figura 2: Algoritmo para el problema 4

4. Problema 4

Para determinar si un grafo no contiene ciclos se puede recorrer el grafo con el algoritmo de búsqueda en profundidad (DFS). Se puede plantear como una versión modificada de DFS, cuando v es un nodo ya marcado entonces significa que ya fue visitado previamente, generando un ciclo. Si se encuentra un ciclo, se activa una bandera en el procedimiento que marca que se encontró un ciclo. Si el algoritmo no tiene ciclos, regresa verdadero si *ciclo_detectado* es falso. En la Fig.(2) está el pseudocódigo del algoritmo

Tiene complejidad $O(m + n)$ porque la iteración del foreach lo hace para m aristas, y la inicialización de la lista de explorados lo hace para n nodos.

5. Problema 5

5.1. (a)

El árbol de expansión mínima está dado en la Fig.(3)

1. Se selecciona de la cola de prioridad la arista de menor peso (a,c) y se comprueba que no forme ciclos en el árbol
2. La siguiente de menor peso es (b,c). Como no forma ciclos en el árbol, se agrega al árbol.
3. La siguiente de menos peso es (c, f). Como no forma ciclos, se agrega al árbol

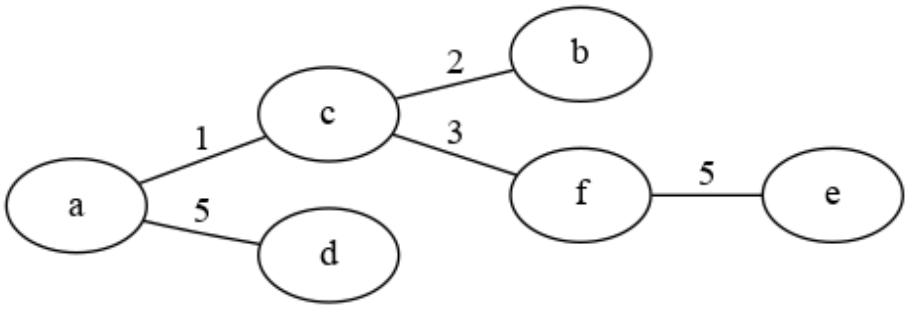

Figura 3: MST propuesto

4. La siguiente de menor peso es (a, b). Pero al agregarla se forma un ciclo (a, b, c), por lo cual se descarta.
5. La siguiente de menor peso es (a, d). Como no forma ciclos en el árbol, se agrega al árbol
6. La siguiente de menor peso es (f, e). Como no forma ciclos en el árbol, se agrega al árbol.
7. La siguiente de menor peso es (d, f). Si se agregara, formaría un ciclo entre ACDF, por lo cual se descarta.
8. Finalmente está (a, e). Si se agregara, forma un ciclo entre ACFE, por lo que se descarta

Una vez evaluados todas las aristas, el algoritmo termina.

5.2. (b)

(d,f) no puede formar parte de ningún MST porque el algoritmo nunca la consideraría al ser un algoritmo voraz. Los vértices que conecta siempre tienen una opción mejor para unirse al árbol que (d,f), que se evalúan antes porque las aristas siempre se mantienen ordenadas.

Para uno de los vértices que conecta, d, siempre se evalúa antes (a, d) ya que tiene menor costo que (d, f). Del mismo modo, para que f se conecte con otros nodos, (c,f) y (f,e) siempre se evalúan antes de (d,f), los cuales tienen costo menor que (d,f).