Proyecto 1

Análisis de Algoritmos

Primer Semestre 2024, Prof. Cecilia Hernández

Fecha Inicio: Jueves 4 de Abril 2024.

Fecha Entrega: Lunes 22 de Abril 2024 (23:59 hrs).
Integrantes: Vicente Cuello - Maximiliano Lopez - Iván Zapata

1. **[5 puntos]** Determine justificadamente si las siguientes afirmaciones son verdaderas o falsas. En caso de resultar verdadera, muestre los valores de las constantes c, c_1, c_2 y n_0 según corresponda.

   a) $n^2/\log(n^{20})$ es $o(n^2)$
   Debe cumplirse que:
   $\lim_{n\to\infty}(\frac{n^2/\log n^{20}}{n^2}) = 0$
   $\lim_{n\to\infty}(1/\log n^{20})$
   $\lim_{n\to\infty}(1/20\log n)$
   $\frac{1}{20}\lim_{n\to\infty}(1/\log n)$
   Como el logaritmo es una función continua y creciente:
   $\frac{1}{20}\lim_{n\to\infty}(1/\log n) = 0 \to n^2/\log(n^{20})$ **sí** es $o(n^2)$

   b) $\ln(n) + \sqrt{n}$ es $\Theta(\sqrt{n})$
   Debe cumplirse que $\exists c_1, c_2 \in \mathbb{R}^+$:
   $0 \leq c_1\sqrt{n} \leq \ln(n) + \sqrt{n} \leq c_2\sqrt{n}$
   $0 \leq c_1\sqrt{n} - \sqrt{n} \leq \ln(n) \leq c_2\sqrt{n} - \sqrt{n}$
   $0 \leq \sqrt{n}(c_1 - 1) \leq \ln(n) \leq \sqrt{n}(c_2 - 1)$
   Si tomamos $n_0 = 1$:
   $0 \leq c_1 - 1 \leq 0 \leq c_2 - 1$
   $0 \leq c_1 \leq 1 \leq c_2$
   Tomando $c_1 = 1, c_2 = 2$:
   $0 \leq 1 \leq 1 \leq 2$
   Como la función es creciente, entonces existen c_1 y c_2 que cumplen los requisitos para $1 \leq n$ y la función **sí** es $\Theta(\sqrt{n})$

   c) $2\sum_{i=1}^{n} i$ es $\Theta(n^2)$
   Debe cumplirse que $\exists c_1, c_2 \in \mathbb{R}^+$:
   $0 \leq c_1 n^2 \leq 2\sum_{i=1}^{n} i \leq c_2 n^2$
   $0 \leq c_1 n^2 \leq n(n+1) \leq c_2 n^2$
   $0 \leq c_1 n \leq n + 1 \leq c_2 n$
   $0 \leq c_1 n - n \leq 1 \leq c_2 n - n$
   $0 \leq n(c_1 - 1) \leq 1 \leq n(c_2 - 1)$
   $0 \leq c_1 - 1 \leq 1/n \leq c_2 - 1$

Tomando n = 1:
$0 \leq c_1 - 1 \leq 1 \leq c_2 - 1$
Tomando $c_1 = 1$ y $c_2 = 2$:
$0 \leq 0 \leq 1 \leq 1$
Como la función es creciente, entonces existen c_1 y c_2 que cumplen los requisitos para $1 \leq n$ y la función **sí** es $\Theta(n^2)$

d) $2^n + 4^n$ es $O(2^n)$
Debe cumplirse que: $\exists c \in \mathbb{R}^+$:
$0 \leq 2^n + 4^n \leq c2^n$
$0 \leq 2^n + 2^{2n} \leq c2^n$
$0 \leq 2^n(1 + 2^n) \leq c2^n$
$0 \leq 1 + 2^n \leq c$
Sea $f(n) = 1 + 2^n \to f'(n) = 2^n \ln 2$ una funcion siempre creciente y por tanto, no acotable por ninguna constante.
Como $\nexists c \in \mathbb{R}^+$ que cumpla el criterio, la función **no** es $O(2^n)$

e) 2^n es $O(n^2)$
Analizando el límite:
$\lim_{n\to\infty}(2^n/n^2)$
Aplicando la regla de L'Hopital:
$\lim_{n\to\infty}(2^n \ln 2/2n)$
Aplicando la regla de L'Hopital por segunda vez:
$\lim_{n\to\infty}(2^n \ln^2 2/2)$
$\ln^2 2/2 * \lim_{n\to\infty}(2^n) = \infty$
Como el resultado del límite es igual a ∞, entonces se puede decir que 2^n es $\omega(n^2)$. Al ser una cota absolutamente superior de n^2, entonces la función 2^n **no** es $O(n^2)$.

2. **[7 puntos]** Ordene de menor a mayor orden asintótico las siguientes funciones. Justifique.
   - $\frac{3}{2}n$
   - $n^2 \log^2(n)$
   - $n!$
   - $\log \log n^3$
   - 2^n
   - $\log^2(n)$
   - $(10^7)^{120!6}$

   Recordando el orden de complejidad asintótica:
   $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^3) < O(2^n) < O(n!)$

   - La menor de todas es $(10^7)^{120!6}$ debido a que es $O(1)$ (Esto puede demostrarse tomando cualquier constante $c \geq (10^7)^{120!6}$)

   - Despues vendría la función $\log \log n^3$, ya que es $O(\log n)$ como sigue:
   Debe cumplirse que: $\exists c \in \mathbb{R}^+$:
   $0 \leq \log \log n^3 \leq c \log n$
   $0 \leq \log 3 \log n \leq c \log n$
   $0 \leq \log 3 + \log n \leq c \log n$

$-\log n \leq \log 3 \leq c \log n - \log n$
$-\log n \leq \log 3 \leq \log n(c-1)$
Si tomamos $n \geq 3$ y $c = 2$:
$-\log 3 \leq \log 3 \leq \log 3(2-1)$
Entonces como existe c positiva para $n \geq 3$, entonces $\log \log n^3$ es $O(\log n)$
- Luego vendría $\frac{3}{2}n$ ya que posee orden lineal $O(n)$ (Esto puede demostrarse tomando cualquier constante $c \geq \frac{3}{2}$)

- Luego vendría la función $\log^2(n)$ que es $O(n \log n)$ como sigue:
$\lim_{n\to\infty}(\frac{\log^2 n}{n \log n})$
$\lim_{n\to\infty}(\frac{\log n}{n})$
Utilizando la regla de L'Hopital:
$\lim_{n\to\infty}(1/\ln 2n) = \frac{1}{\ln 2}\lim_{n\to\infty}(1/n) = 0$
Entonces la función es $o(n \log n)$, y por tanto, $O(n \log n)$

- Luego vendría $n^2 \log^2(n)$ que es $O(n^3)$ como sigue:
$\lim_{n\to\infty}(\frac{n^2 \log^2 n}{n^3}) = \lim_{n\to\infty}(\frac{\log^2 n}{n})$
Utilizando la regla de L'Hopital:
$\lim_{n\to\infty}(\frac{2\log n/\ln 2n}{1}) = \lim_{n\to\infty}(\frac{2\log n}{\ln 2n}) = \frac{2}{\ln 2} lim_{n\to\infty}(\frac{\log n}{n})$
Utilizando la regla de L'Hopital otra vez:
$\frac{2}{\ln 2} lim_{n\to\infty}(\frac{1/\ln 2n}{1}) = \frac{2}{\ln 2} lim_{n\to\infty}(\frac{1}{ln 2n}) = 0$
Entonces la función es $o(n^3)$, y por tanto, $O(n^3)$

- Luego vendría la función 2^n que es $O(2^n)$ (Bastan tomar cualquier constante c positiva tal que $c \geq 1$)

- Por último vendría la función $n!$ que es $O(n!)$ (Bastan tomar cualquier constante c positiva tal que $c \geq 1$)

Por lo tanto, ordenadas de menor a mayor complejidad asintótica resulta:
$(10^7)^{120!6} \to \log \log n^3 \to \frac{3}{2}n \to \log^2(n) \to n^2 \log^2(n) \to 2^n \to n!$

3. **[4 puntos]** Para cada una de las siguientes funciones, encuentre un $c > 0$ y un $n_0 \in \mathbb{N}$ que muestre que $f(n) \in O(g(n))$. Explique por qué tales valores funcionan bien.

   a) $f(n) = \sqrt{n^5} + 4n^{10} + n\log(5)$, $g(n) = 10n^{10}$
   Tomando $n = 1$ y $c = 1$ se cumple la condición como sigue:
   $\exists c \in \mathbb{R}^+$:
   $0 \leq \sqrt{n^5} + 4n^{10} + n\log(5) \leq c10n^{10}$
   $0 \leq \frac{\sqrt{n^5} + 4n^{10} + n\log(5)}{n^{10}} \leq 10c$
   $0 \leq n^{-15/2} + 4 + n^{-9}\log(5) \leq 10c$
   Tomando $n \geq 1$
   $0 \leq 1 + 4 + \log(5) \leq 10c$
   $0 \leq \frac{5+\log(5)}{10} \leq c \to c \geq 0{,}732192$

Por lo tanto $c = 1$ cumple el requisito, y $f(n) = O(10n^{10})$

b) $f(n) = \sqrt{\log(n)n} + n^2$, $g(n) = (n\log(n))^2$
Se cumple para $n = 2$ y $c = 2$ como sigue:
$\exists c \in \mathbb{R}^+$:
$0 \leq \sqrt{\log(n)n} + n^2 \leq c(n\log(n))^2$
$0 \leq \dfrac{1}{\sqrt{(\log(n)n)^3}} + \dfrac{1}{\log^2 n} \leq c$
Tomando $n \geq 2$
$0 \leq \dfrac{1}{\sqrt{8}} + 1 \leq c \to c \geq 1{,}35355339$
Por lo tanto $c = 2$ cumple el requisito, y $f(n) = O((n\log(n))^2)$

4. **[6 puntos]** Resuelva las siguientes recurrencias.

a) $T(n) = 12T(n/10) + n^{1/c}$, $c > 1$
Utilizando el teorema maestro vamos a intentar verificar si $n^{1/c}$ es $O(n^{\log_{10} 12-\epsilon})$
Debemos comprobar que $\exists d \in \mathbb{R}^+$:
$0 \leq n^{1/c} \leq dn^{\log_{10} 12-\epsilon}$
$0 \leq n^{1/c} \leq dn^{\log_{10} 12-\epsilon}$
Si tomamos $\epsilon = 2$ y $d = 1$:
$0 \leq n^{1/c} \leq n \to \dfrac{1}{c} \leq 1 \to 1 \leq c$
Como $c > 1$, la desigualdad se cumple y por lo tanto:
$T(n) \in \theta(n^{\log_{10} 12})$

b) $T(n) = 3T(\sqrt[3]{n}) + c\log n$, $c > 0$
Utilizando el cammbio de variable $n = 2^m$, se tiene que:
$S(m) = T(2^m) = 3T(2^{m/3}) + mc \to S(m) = 3S(m/3) + mc$ Utilizando el caso dos del teorema maestro para S, vamos a verificar que:
$\exists d_1, d_2 \in \mathbb{R}^+$:
$0 \leq d_1 m^{\log_3 3} \leq mc \leq d_2 m^{\log_3 3}$
$0 \leq d_1 m \leq mc \leq d_2 m$
$0 \leq d_1 \leq c \leq d_2$
Tomando $d_1 = c = d_3$ se cumple la desigualdad, luego $mc = \theta(m)$ y por tanto, $S(m) = \Theta(m\log m)$
Luego volviendo al caso anterior:
$2^m = n \to m = \log n$. Así:
$T(2^m)$ es $\theta(m\log m) \to T(n)$ es $\Theta(\log n \log(\log n))$

5. **[6 puntos]** Construya los árboles recursivos para las siguientes recurrencias, estime la solución de la recurrencia y luego demuestre por substitución.

a) $T(n) = 2T(n/4) + c\sqrt{n}$, con $T(1) = 1$
Dibujando el arbol, se obtiene (Por niveles):
Nivel 0:——————-$c\sqrt{n}$——————-
Nivel 1:——$c\sqrt{n/4}$———$c\sqrt{n/4}$——
Nivel 2:–$c\sqrt{n/4^2}$–$c\sqrt{n/4^2}$—$c\sqrt{n/4^2}$–$c\sqrt{n/4^2}$–
Nivel k:–$c\sqrt{n/4^k} - c\sqrt{n/4^k} - c\sqrt{n/4^k} - c\sqrt{n/4^k} - c\sqrt{n/4^k}$...
Vemos que en cada nivel la suma de los costos puede obtenerse como:
En el nivel k: $c\sqrt{\dfrac{n}{4^k}}2^k = c\sqrt{\dfrac{n}{(2^k)^2}}2^k = c\dfrac{\sqrt{n}}{2^k}2^k = c\sqrt{n}$
Para calcular la altura, necesitamos llegar al caso base de la recurrencia como sigue
$c\sqrt{\dfrac{n}{4^k}} = 1 \to c^2 n = 4^k \to \log_4 c^2 n = k \to 2\log_4 c + \log_4 n = k$

Así, una estimación de la complejidad de la recurrencia puede ser:
$T(n)$ es $O(\sqrt{n}\log n)$
Demostración por sustitución:
Debebemos demostrar que:
$\exists d \in \mathbb{R}^+$:
$2d\sqrt{n/4}\log n/4 + c\sqrt{n} \leq d\sqrt{n}\log n$
$d\sqrt{n}\log n/4 + c\sqrt{n} \leq d\sqrt{n}\log n$
$d\log n/4 + c \leq d\log n$
$c \leq d(\log n - \log n/4)$
$c \leq d(\log n - \log n + \log 4)$
$c \leq d(\log 2^2)$
$c \leq 2d$
Tomando $c = 2d$ se demuestra que $T(n)$ es $O(\sqrt{n}\log n)$

b) $T(n) = T(n-1) + n$ Dibujando el arbol, se obtiene (Por niveles):
Nivel 0: n
Nivel 1: n - 1
Nivel 2: n - 2
Nivel 3: n - 3
Nivel k: n - k
Vemos que la amplituda del arbol esta dada por:
$\sum_{i=0}^{n} n - i$

6. **[5 puntos]** Use substitución para demostrar las siguientes recurrencias. Note que debe aplicar las definiciones asintóticas.

a) $T(n) = T(n-1) + log(n)$ su solución es $T(n) = \Theta(n\log(n))$
Vemos que la función puede escribirse como $T(n) = T(n-1) + O(\log n)$
Aqui, debemos encontrar cotas superiores e inferiores tales que:
$T(n) \leq T(n-1) + clog(n)$ para alguna $c > 0$
Es decir:
$d(n-1)\log(n-1) + c\log n \leq dn\log(n)$
$d(n-1)\log(n) + c\log n \leq dn\log(n)$
$d(n-1) + c \leq dn$
$dn - d + c \leq dn$
$-d + c \leq 0$
$c \leq d$
Tomando $c = d$ se cumple la condición para cota inferior y superior y por tanto $T(n)$ es $\Theta(n\log n)$

b) $T(n) = T(n/2) + T(n/4) + T(n/16) + cn$ su solución es $T(n) = \Theta(n)$
Utilizando sustitución, se debe comprobar que:
$T(n)$ es $\Theta(n) \rightarrow dn/2 + dn/4 + dn/16 + cn \leq dn$
$8dn + 4dn + dn + 16cn \leq 16dn$
$8d + 4d + d + 16c \leq 16d$
$13d + 16c \leq 16d$
$16c \leq 3d$
$c \leq 3d/16$
Tomando $c = 3d/16$, se tiene que $T(n)$ es $\Theta(n)$

7. **[6 puntos]** Proporcione un análisis asintótico de peor caso en notación $O\,()$ para el tiempo de ejecución de los siguientes fragmentos de programa.

```
(a)

for( int i = 1; i <= n; i *= 2 ) {
  for( int j = 1; j < n; j += 2 ) {
    f(); // O(log n)
    for( int k = 1; k < 3; k *= 2 ) {
      g() // O(n)
    }
  }
}
```

```
(b)

for( int i = 1; i <= n; i *= 2 ) {
  for( int j = 1; j < n; j *= 2 ) {
    for( int k = 1; k < n; k += 1 ) {
       f(); // O(n)
   for( int l = 1; l < 100; k += 1 ) {
     g() // O(n)
   }
    }
  }
  h() // O(n)
}
```

```
(c)
void f(int n){
  if( n > 1){
    f(n/3);
    f(n/3);
    f(n/3);
    for(int i=0; i<n; i++){
      ProcesaA();// O(n)
    }
  } else {
      ProcesaB();// O(1)
  }
}
```

a) La complejidad del algoritmo esta dada por las siguientes operaciones:
$\sum_{i=1}^{\log n} \sum_{j=1}^{n/2} (cn + \sum_{k=1}^{3} dn)$
$\sum_{i=1}^{\log n} \sum_{j=1}^{n/2} (cn)$
$\sum_{i=1}^{\log n} cn^2$
$cn^2 \log n$
Por lo cual el algoritmo es $O(n^2 \log n)$

b) La complejidad del algoritmo esta dada por las siguientes operaciones:
$\sum_{i=1}^{\log n} \sum_{j=1}^{\log n} \sum_{k=1}^{n} (cn + \sum_{l=1}^{100} dn) + O(n)$
$\sum_{i=1}^{\log n} \sum_{j=1}^{\log n} \sum_{k=1}^{n} cn + O(n)$
$\sum_{i=1}^{\log n} \sum_{j=1}^{\log n} cn(n) + O(n)$
$\sum_{i=1}^{\log n} cn^2 \log n + O(n)$
$cn^2 \log^2 n + O(n) = cn^2 \log^2 n + dn$
Por lo cual el algoritmo es $O(n^2 \log^2 n)$

c) La complejidad del algoritmo recursivo esta dada por la siguiente ecuación:
$T(n) = 3T(n/3) + O(n^2)$
Utilizando sustitución para $T(n) \leq dn^2$:
$3d(n/3)^2 + cn^2 \leq dn^2$
$dn^2/3 + cn^2 \leq dn^2$
$d/3 + c \leq d$
$d + 3c \leq 3d$
$3c \leq 2d$

$c \leq 2d/3$

Tomando $c = \dfrac{2}{3}d$, se prueba que $T(n)$ es $O(n^2)$

8. **[6 puntos]**Determine qué realiza el siguiente algoritmo, demuestre que es correcto y determine su complejidad asintótica.

```
int F(int* A, int n, float x){
  p=0;
  for i=0 to n{
    p = A[n-i] + x*p;
  }
  return p;
}
```

Indicación: El arreglo A de tamaño n representa los coeficientes de un polinomio $p(x) = \sum_{i=0}^{n} a_i x^i$. -
El algoritmo lo que realiza es el cálculo del valor x entregado en un polinomio con coeficientes de A
Un ejemplo de como funciona:
Sea $p(x) = 3x^2 + 2x + 1$ un polinomio de grado . En este caso $A = [1, 2, 3]$ y el polinomio evaluado en 5 se calcula en el ciclo de la siguiente forma:
$p = 3$
$p = 3(5) + 2$
$p = 3(5)^2 + 2(5) + 1$
Si queremos hacer un análisis de correctitud, tenemos la siguiente invariante del loop:
$p = A[n-i] + x\sum_{j=0}^{i-1} A[n-j]x^{n-j}$
Fase de inicialización ($n = 0$):
Se tiene que $p = A[0]$, lo cual es consistente ya que un polinomio de grado 0 es constante
Fase de mantención $n = k$:
Se tiene que $A = [A[0], A[1], A[2], A[3]..., A[k+1]]$
$p = A[k-i]+x\sum_{j=0}^{i-1} A[k-j]^{k-j} = A[k-i]+x(A[k]x^k+A[k-1]x^{k-1}+A[k-2]x^{k-2}...+A[k-i+1]x^{k-i+1})$
$p = A[k-i] + A[k]x^{k+1} + A[k-1]x^k + A[k-2]x^{k-1}... + A[k-i+1]x^{k-i+2}$
Vemos que el polinomio resultante satisface el orden de los coeficientes
Fase de terminación:
El loop de la suma se mantiene completamente inalterado, por lo cual el resultado final sera simplemente el polinomio entregado (funciona para n = k+1) por lo cual el algoritmo es correcto.

Análisis asintótico:
Vemos que el ciclo de operaciones esta dado por:
$1+\sum_{i=0}^{n} O(1)+1 = 1+O(1)n+1 = cn+2$ con c una constante positiva. Por tanto el algoritmo posee una complejidad de $O(n)$

9. **[7 puntos]** Considere un conjunto de puntos tridimensionales con coordenadas reales con signo. Reporte la distancia mínima entre dos puntos. Para ello utilizaremos la distancia euclidiana entre dos puntos. Dados dos puntos $p_1(x_1, y_1, z_1)$ y $p_2(x_2, y_2, z_2)$, el cálculo de la distancia euclidiana es la siguiente:

$$d_{p_1,p_2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

   a) Escriba un pseudo código para un algoritmo secuencial que resuelva el problema.
   b) Diseñe un algoritmo basado en dividir para conquistar que resuelva el problema en escriba su pseudo código.
   c) Demuestre correctitud de sus algoritmo y realice los análisis de tiempo de ejecución.

10. **[8 puntos]** Considerando el problema anterior, modifique los algoritmos propuestos para que reporte todos los pares de puntos cuya distancia sea menor o igual a una distancia d.

a) Modifique los dos algoritmos propuestos para el problema anterior para que resuelvan esta nueva versión. Escriba los pseudocódigos para los nuevos problemas.

b) Explique los cambios realizados a la hora de adaptar los algoritmos.

c) Demuestre correctitud de sus algoritmo y realice los análisis de tiempo de ejecución.

d) Implemente sus algoritmos usando C/C++ definiendo una función para cada uno.

e) Realice análisis experimental para lo cual se pide que construya un gráfico que muestre como varían los tiempos de ejecución en nanosegundos variando el tamaño de la entrada (n).