Universidad Nacional Mayor de San Marcos

Facultad de Ingeniería Informática y Sistemas

Separata de Ejercicios

Complejidad Algorítmica

Autores :
Prof. Herminio Paucar
Prof. Luis Guerra
Prof. Edson Huillca

Version 0.1
20 de marzo de 2018

Recomendaciones

Esta separate está dirigida a todos los estudiantes de la FISI-UNMSM que estén llevando la materia de Análisis y Diseño de Algoritmos. Está primera separata contiene los ejercicios de la primera semana de clases donde se dictaron los topicos de: Repaso de Matemáticas y Complejidad de Algoritmos. Como fuente de información recomiendo revisar los siguientes libros:

1. Introduction to Algorithms. Cormen, Leiserson and Rivest and Cliff Stein.
2. Algorithms Design. Eva Tardos and Jon Kleinberg.
3. Algorithms. Dasgupta, Papadimitriou, and Vazirani.

Índice general

Los algoritmos y su complejidad **v**

0.1. Introducción a la Inducción Matemática v

0.2. Demostración de Complejidad v

0.3. Calcular la Complejidad de los Algoritmos viii

0.4. División y Conquista I - Recursividad xi

Los algoritmos y su complejidad

0.1. Introducción a la Inducción Matemática

1. É verdade que $\| \log n \| \geq \log(n-1)$ para todo inteiro $n \geq 2$? Justifique sua resposta.
2. É verdade que $\| \log n \| \leq \log(n+1)$ para todo inteiro $n \geq 1$? Justifique sua resposta.
3. Quanto vale a soma
$$\sum_{i=1}^{n} \log i = \log 1 + \log 2 + ... + \log n$$
onde $n \leq 1$ é um inteiro.
4. Prove por indução em n que $2^0 + 2^1 + 2^2 + ... + 2^n = 2^{n+1} - 1$
5. Ache uma fórmula para a soma $1{,}2 + 2{,}3 + 3{,}4 + .. + n(n+1)$ e prove sua afirmação.
6. Ache uma fórmula para a soma $1^2 + 2^2 + 3^2 + .. + n^2$ e prove sua afirmação.
7. Ache uma fórmula para a soma $\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + ... + \frac{1}{2n-1} + \frac{1}{2n}$ e prove sua afirmação.
8. Prove que as regiões formadas por n círculos no plano podem ser coloridas com duas cores de modo que regiões vizinhas tenham cores distintas.
9. **O Princípio da Casa do Pombo** (em sua forma mais simples) afirma o seguinte: se $n+1$ bolas são distribuídas de modo arbitrário em n caixas, então pelo menos uma caixa contém mais que uma bola. Prove este princípio por indução.

0.2. Demostración de Complejidad

1. Considerar las siguientes Funciones [P1, Poggi, 18/03/2015, PUC-Rio]

   a) $10.n^{\pi}$

   b) $\log n$

c) $\log(n^2)$

d) $0{,}005.n^{0{,}00001}$

e) $1000.(\log n)^2$

f) $30.n^3.\log n$

g) $50.n.\log^2 n$

h) $(\log n)^{\log n}$

i) $\frac{n}{\log n}$

j) $70.n$

k) $\log\log n$

l) $(1{,}5)^{\log^2 n}$

m) $90.n^2.\log n + n^3.\log n$

- Coloque las funciones de arriba en orden de crecimiento Asintótico, i.e. Valor cuando $n \to \infty$.
- Agrupar aquellas funciones por su respectivo Orden O, Ω, Θ.

2. ¿Qué significa que una función $g(n)$ es $O(f(n))$.

3. ¿Qué significa que una función $g(n)$ es $\Theta(f(n))$.

4. ¿Qué significa que una función $g(n)$ es $\Omega(f(n))$.

5. Supongamos un algoritmo A y un algoritmo B con funciones de complejidad de tiempo $a(n) = n^2 - n + 549$ y $b(n) = 49n + 49$, respectivamente. Determine cuáles son los valores de n pertenecientes al conjunto de los números naturales para los cuales A tiene menor tiempo para ejecutar que B.

6. Esprese una función $10n^3 - 5n^2 - 10n + 3$ en terminos de notación Θ.

7. ¿Es verdad que $2n^3 + 5 = \Theta(n^3)$? Exlique.

8. Dos algortimos A y B poseen complejidad n^5 y 2^n respectivamente. ¿Usted utilizará el algoritmo B en lugar de A, en que caso? Explicar.

9. Cuál es el orden de complejidad en el peor caso de:

   a) $2n + 10$

   b) $(1/2)n(n + 1)$

   c) $n + \sqrt{n}$

   d) $n/1000$

e) $(1/2)n^2$

f) $(1/2)n^2 - 3n$

10. ¿Cuáles son las magnitudes físicas que influencian la eficiencia de tiempo de un algoritmo en la práctica?

11. Explique qué tipos de problemas o algoritmos suelen tener complejidad del orden de $nlogn$ y cómo los identificamos.

12. ¿Qué problemas suelen tener complejidad del orden de $\log n$?

13. ¿Cuáles son los problemas que suelen ser exponenciales?

14. Escribir las siguientes funciones en notación O.

    a) $3n^3 + 20n^2 \log n$

    b) $3n^n + (5)2^n$

    c) $(n-1)^n + n^{(n-1)}$

    d) $4n + 2n \log n$

    e) 34

15. Verdadero o Falso, Justificar [P10, Poggi, 18/03/2015, PUC-Rio]

    a) $(\log n)^{100} = O(n^\varepsilon), \forall \varepsilon > 0$.

    b) $2^{n+1} = O(2^n)$.

    c) Se $g(n,m) = m \log_d n$ onde $d = [m/n + 2]$ ($[x]$ *é* o menor inteiro maior que x), $m = O(n^2)$, então $g(n,m) = O(m^{(1+\varepsilon)})$ $\forall \varepsilon > 0$.

16. Usando la definición de O, pruebe formalmente los siguientes enuncioados [P3, Wootters, 06/10/2017, Stanford]

    a) $2\sqrt{n} + 6 = O(\sqrt{n})$

    b) $n^2 = \Omega(n)$

    c) $\log_2(n) = \Theta(ln(n))$

    d) 4^n no es $O(2^n)$.

    [Se espera para cada parte, una prueba rigurosa (pero breve), usando la definición de O, Ω y Θ]

17. Busca dos funciones $f(n)$ y $g(n)$ que satisfagan las siguientes relaciones. Si no existen tales $f(n)$ y $g(n)$, explica brevemente porqué. [P2, HW2, Leitert, Summer 2017, Kent]

a) $f(n) \in O(g(n))$ y $f(n) \notin \Theta(g(n))$

b) $f(n) \in \Theta(g(n))$ y $f(n) \in O(g(n))$

c) $f(n) \in \Theta(g(n))$ y $f(n) \notin O(g(n))$

d) $f(n) \in \Omega(g(n))$ y $f(n) \notin O(g(n))$

18. Sejam $f(n)$ e $g(n)$ funções assintoticamente não-negativas. Usando a definição básica da notação Θ, mostre que a função $h(n) = \text{máx}\, f(n), g(n)$ pertence a $\Theta(f(n)+g(n))$.

19. Mostre que para quaisquer constantes a, b onde $b > 0$ temos que $(n+a)^b \in \Theta(n^b)$.

20. É verdade que $2^{n+1} \in O(2^n)$? E $2^{2n} \in O(2^n)$?

21. Explique por que a afirmação "o tempo de execução do algoritmo A é pelo menos $O(n^2)$" não faz sentido.

22. Mostre que $n! \in o(n^n)$, $n! \in w(2^n)$ e $\log n! \in \Theta(n \log n)$. Não utilize a aproximação de Stirling.

23. Prove ou apresente um contra-exemplo para cada uma das afirmações abaixo.

    a) se $f(n) \in O(g(n))$ então $g(n) \in O(f(n))$

    b) $f(n) + g(n) \in \Theta(\text{mín}(f(n), g(n)))$

    c) se $f(n) \in O(g(n))$ então $2^{f(n)} \in O(2^{g(n)})$

    d) se $f(n) \in O(g(n))$ então $g(n) \in (f(n))$

    e) se $h(n) \in o(f(n))$ então $f(n) + h(n) \in \Theta(f(n))$

0.3. Calcular la Complejidad de los Algoritmos

1. Obtenha como função de n a melhor análise de complexidade possıvel para os dois pseudo-codigos apresentados abaixo [Laber p1, 26/06/2007, PUC-Rio]

    Pseudocodigo-1
    $Soma \leftarrow 0$
    for $i \in \{1, \ldots, n\}$ **do**
        for $j \in \{1, \ldots, n^3\}$ **do**
            $Soma \leftarrow Soma + 2$
            $Aux \leftarrow Soma$
            while $Aux \geq 1$ **do**
                $Aux \leftarrow Aux/2$
            end while

```
        end for
    end for

    Pseudocodigo-2
    Cont ← 0
    for i ∈ {1, ..., n} do
        Aux ← i
        Aux ← Soma
        while Aux ≥ 0 & Cont < n^{3/2} do
            Cont++
            Aux--
        end while
    end for
```

2. Obtenha como função a melhor análise de complexidade possível para os dois pseudo-códigos apresentados abaixo. [Laber p1, 24/04/2007, PUC-Rio]

```
    t ← 0
    Cont ← 1
    for i ∈ {1, ..., n} do
        Cont ← 2 * Cont
    end for
    while Cont ≥ 1 do
        Cont ← Cont/2
        for j ∈ {1, ..., n} do
            t++
        end for
    end while
```

3. Calcule a complexidade, no pior caso, do fragmento de codigo abaixo:

```
    Pseudocodigo-1
    int i, j, k;
    for (i = 0; i < N; i++) do
        for (j = 0; j < N; j++) do
            R[i][j] = 0;
            for (k = 0; k < N; k++) do
                R[i][j]+ = A[i][k]*B[k][j];
            end for
        end for
    end for
```

4. Calcule a complexidade, no pior caso, do fragmento de codigo abaixo:

Pseudocodigo-2
int i, j, k, s;
for $(i = 0; i < N - 1; i++)$ **do**
  for $(j = i + 1; j < N; j++)$ **do**
    for $(k = 1; k < j - 1; k++)$ **do**
      $s = 1$;
    end for
  end for
end for

5. Calcule a complexidade, no pior caso, do fragmento de codigo abaixo:

   Pseudocodigo-3
   int i, j, s;
   for $(i = 0; i < N - 1; i++)$ **do**
     for $(j = 1; j < 2{*}N; j++)$ **do**
       $s = s + 1$;
     end for
   end for

6. Obter uma equação matemática relativa a uma análise do melhor e melhor caso do fragmento de código abaixo:

   Pseudocodigo-1
   for $(i = 0; i < N; i = i + 2)$ **do**
     for $(j = N - i; j >= 0; j--)$ **do**
       if $(V[i] < V[j])$ **then**
         printf(i)
       end if
     end for
   end for

7. Escreva um algoritmo eficiente que procure por um dado número em vetor ordenado. Quais são sua funçã de custo e ordens de complexidade O e Ω?

8. Supongamos que disponemos de la siguiente definición de tipo:

   CONST n = ...;
   TYPE vector = ARRAY [1..n] OF INTEGER;

   Consideramos entonces los procedimientos y funciones siguientes:

   PROCEDURE Algoritmo (VAR a:vector)
   for $(i = 0; i < N; i = i + 2)$ **do**
       for $(j = N - i; j >= 0; j - -)$ **do**
           if $(V[i] < V[j])$ **then**
               printf("%d " , i) ;
           end if
       end for
   end for

0.4. División y Conquista I - Recursividad

Método de Sustitución.

1. Demuestre que la solución de la recurrencia $T(n) = T([\![n/2]\!])+1$ pertenece a $O(\log n)$.
2. Demuestre que la solución de $T(n) = 2T([\![n/2]\!]) + n$ es $\Theta(n \log n)$.
3. Demuestre que la solución de $T(n) = T([\![n/2]\!]) + T([\![n/2]\!]) + n$ es $\Omega(n \log n)$.
4. Resuelva la recurrencia $T(n) = T([\![n/3]\!]) + T([\![2n/3]\!]) + 1$.

Método de Iteración y Árbol de Recurrencia.

1. Determine un buen limete superior asintótico para la recurrencia $T(n) = 3T([\![n/2]\!])+ n$ usando el metodo de iteración.
2. Determine un buen limete superior asintótico para la recurrencia $T(n) = T(n/3) + T(2n/3) + n$ es $\Theta(n \log n)$ utilizando el método del árbol de recurrencia. No se preocupe por redondeos.
3. Dibuje el árbol de recurrencia para $T(n) = 4T([\![n/2]\!]) + n$ y obtenga la clase Θ a la que pertenece la solución.
4. Use el método de iteración para resolver la recurrencia $T(n) = T(n - a) + T(a) + a$ donde $a \geq 1$ es un entero positivo.
5. Use el metodo de arbol de recurrencia para resolver la recurrencia $T(n) = T(\alpha n) + T((1 - \alpha)n + n$ donde $0 < \alpha < 1$ es una constante.

Teorema Master.

1. Use el Teorema Master para resolver las recurrencias de abajo.

   a) $T(n) = 4T(n/2) + n$

   b) $T(n) = 4T(n/2) + n^2$

   c) $T(n) = 4T(n/2) + n^3$

2. El tiempo de ejecución de un algoritmo A es descrito por la recurrencia $T(n) = 7T(n/2) + n^2$, otro algoritmo A' tiene complejidad de tiempo descrito por $T'(n) = aT'(n/4) + n^2$. ¿Cuál es el mayor entero a tal que A' es asintoticamente mas rápido que A?

3. ¿El Teorema Master puede ser aplicado a la recurrencia $T(n) = 4T(n/2) + n^2 \log n$? Justifique su respuesta. Obtenga un buen limite superior asintótico para la recurrencia, sin usar el Teorema Master directamente.

4. Determine una expresión cerrada para cada una de las siguitenes recuerrencias. [P11, Poggi, 18/03/2015, PUC-Rio]

   a) $T(1) = 1$
   $T(2) = 6$
   $T(n) = T(n-2) + 3n + 4$, $\forall n \geq 3$.

   b) $T(1) = 1$
   $T(2) = 6$
   $T(n) = 2.T(n-2) + 3$, $\forall n \geq 3$.

   c) $\sum_{i=1}^{n-1}(T(i) + T(n+i)) + 1$, $\forall n \geq 2$.