Projeto e Análise de Algoritmos

June 2, 2019

Lista 01

3 Resolva as seguintes equações de recorrência segundo o método da árvore de recursão:

- T(n) = T(n - 1) + 2
  - T(n - 1) = T(n - 2) + 2
  - T(n - 2) = T(n - 3) + 2
  - ...
  - T(2) = t(1) + 2
  - T(1) = 2
  - ---
  - T(n) = 2n = Θ (n)
- T(n) = 2T(n - 1) + 1
  - T(n - 1) = 2T(n - 2) + 1   * 2^0
  - T(n - 2) = 2T(n - 3) + 1   * 2^1
  - ...
  - T(2) = 2T(1) + 1   * 2^{n-1}
  - T(1) = 1   * 2^n
  - ---
  - T(n) = $\sum_{i=0}^{n-1}$ 2^i = 1 * (2^n -1)/(2 - 1) = 2^n - 1
  - T(n) = Θ (n)

- T(n) = 2T(n/3) + 1
  - T(n/3) = 2T(n/ 3^2) + 1 * 2^0
  - T(n/ 3^2) = 2T(n/ 3^3) + 1 * 2^1
  - T(n/ 3^3) = 2T(n/ 3^4) + 1 * 2^2
  - ...
  - T(n/ 3^{k-1}) = 2T(n/ 3^k) + 1 * 2^{k-1}
  - T(n/ 3^k) = Θ(1) * 2^k
  - ———————————————————————
  - T(n) = 2^k Θ(1) + $\sum_{i=0}^{k-1}$ 2^i * 1
  - = 2^k * Θ (1) + 1 * (2^k - 1)/(2 - 1)
  - k = $log_2 n$ $2^{log_2 n}$ = n
  - = $2^{log_2 n}$ * Θ(1) + $2^{log_2 n}$ - 1
  - T(n) = Θ(n)
- T(n) = 5T(n/4) + n
  - T(n/4) = 5T(n/ 4^2) + n/4 * 5^0
  - T(n/ 4^2) = 5T(n/ 4^3) + (n/ 4^2) * 5^1
  - T(n/ 4^3) = 5T(n/ 4^4) + T(n/ 4^3) * 5^2
  - ...
  - T(n/ 4^{k-1}) = 5T(n/ 4^k) + T(n/ 4^{k-1}) * 5^{k-1}
  - T(n/ 4^k) = Θ(1) * 5^k
  - ———————————————————————
  - T(n) = T(n/ 5^k)*Θ(1) + $\sum_{i=0}^{k-1}$ * 5 * n/4
  - T(n) = T(n/ 5^k)*Θ(1) + n * $\sum_{i=0}^{k-1}$ * 5 /4
  - k = $log_4 n$ $5^{log_4 n}$ / $4^{log_4 n}$ = $n^{log_4 5}$ / n = 1/4
  - T(n) = T(n/ $5^{log_4 n}$)*Θ(1) + n*1* ($(5/4)^k$ -1)/[(5/4) -1]
  - T(n) = T(n/ $n^{log_4 5}$)*Θ(1) + 4 * [(($n^{log_4 5}$)/ n)-1] / (1/4)
  - T(n) = T(n/ $n^{log_4 5}$)*Θ(1) + 4 * $n^{log_4 5}$ -4n
  - T(n) = Θ($n^{log_4 5}$)

- T(n) = 7T(n/7) + n
  - T(n/7) = 7T(n/ 7^2) + n/7 * 7^0
  - T(n/ 7^2) = 7T(n/ 7^3) + (n/ 7^2) * 7^1
  - T(n/ 7^3) = 7T(n/ 4^4) + (n/ 7^3) * 7^2
  - ...
  - T(n/ 7^{k-1}) = 5T(n/ 7^k) + (n/ 7^{k-1}) * 7^{k-1}
  - T(n/ 7^k) = Θ(1) * 7^k
  - ———————————————————————————
  - T(n) = T(n/ 7^k)* Θ (1) + $\sum_{i=0}^{k-1}$ * 7 * n/7
  - k = $log_7 n$
  - T(n) = T(n/ $7^{log_7 n}$) * Θ(1) + n * $\sum_{i=0}^{k-1}$ * 7/7
  - T(n) = n* Θ (1) + n $log_7 n$
  - T(n) = Θ (n $log_7 n$)
- T(n) = 9T(n/3) + n^2
  - T(n/3) = 9T(n/ 3^2) + n^2 * 9^0
  - T(n/ 3^2) = 9T(n/ 3^3) + ($(n/3^2)^2$) * 9^1
  - T(n/ 3^3) = 9T(n/ 3^4) + ($(n/3^3)^2$) * 9^2
  - ...
  - T(n/ 3^{k-1}) = 9T(n/ 3^k) + ($(n/3^{k-1})^2$) * 9^{k-1}
  - T(n/ 3^k) = Θ(1) * 9^k
  - ———————————————————————————
  - T(n) = T(n/ 9^k)* Θ(1) + $\sum_{i=0}^{k-1}$ * 9 * $n^2/9$
  - k = $log_9 n$
  - T(n) = T(n/ $9^{log_9 n}$) * Θ(1) + n^2 * $\sum_{i=0}^{k-1}$ * 9/9
  - T(n) = n* Θ(1) + n^2 $log_9 n$
  - T(n) = Θ(n^2 log n)

- T(n) = 8T(n/2) + n^3
  - T(n/2) = 8T(n/ 2^2) + n^3 * 8^0
  - T(n/ 2^2) = 8T(n/ 3^3) + ($(n/2^2)^3$) * 8^1
  - T(n/ 2^3) = 8T(n/ 3^4) + ($(n/2^3)^3$) * 8^1
  - ...
  - T(n/ 2^{k-1}) = 8T(n/ 2^k) + ($(n/2^{k-1})^3$) * 8^{k-1}
  - T(n/ 2^k) = Θ(1) * 8^k
  - ———————————————————————
  - T(n) = T(n/ 8^k)*Θ(1)+ $\sum_{i=0}^{k-1}$ * 8 * $n^3/8$
  - k = $log_8 n$
  - T(n) = T(n/ $8^{log_8 n}$)*Θ(1) + n^3 * $\sum_{i=0}^{k-1}$ * 8/8
  - T(n) = n*Θ(1) + n^3 $log_8 n$
  - T(n) = Θ(n^3 log n)
- T(n) = T(0.99n) + 7
  - T((99/100) * n) = T(($(99/100)^2$) * n) + 7
  - T(($(99/100)^2$) = T(($(99/100)^3$) * n)+ 7
  - T(($(99/100)^3$) = T(($(99/100)^4$) * n) + 7
  - ...
  - T(($(99/100)^{k-1}$) = T(($(99/100)^3$)* n) +7
  - T(($(99/100)^k$) = Θ(1)
  - ———————————————————————
  - T(n) = Θ(1) + k*7
  - T(n) = Θ(log n)
- T(n) = T($\sqrt{n}$) + 7
  - T(n) = T($n^{1/2}$) +1
  - T($n^{1/2}$) = T($n^{1/2^2}$) +1
  - ...
  - T($n^{1/2^{k-1}}$) = T($n^{1/2^k}$) +1
  - T($n^{1/2^k}$) = Θ(1)
  - ———————————————————————
  - T(n) = k + 1
  - k = log $log_2 n$
  - T(n) = log $log_2 n$ + 1
  - T(n) = Θ(loglog n)

4 Suponha que você está tentando escolher entre esses três algoritmos abaixo. Qual o tempo de cada um em notação assintótica e qual você escolheria?

- Algoritmo A : Resolve o problema dividindo a entrada em cinco subproblemas com a metade do tamanho. Resolve cada subproblema recursivamente e depois combina-os em tempo linear.
  T(n) = 5T(n/2) + n
  - T(n/2) = 5T(n/ 2^2) + n/2   * 5^0
  - T(n/ 2^2) = 5T(n/ 2^3) + (n/ 2^2)   * 5^1
  - T(n/ 2^3) = 5T(n/ 2^4) + T(n/ 2^3)   * 5^2
  - ...
  - T(n/ 2^{k-1}) = 5T(n/ 2^k) + T(n/ 2^{k-1})   * 5^{k-1}
  - T(n/ 2^k) = Θ(1)   * 5^k
  - ---
  - T(n) = T(n/ 5^k)*Θ(1) + $\sum_{i=0}^{k-1}$ * 5 * n/2
  - T(n) = T(n/ 5^k)*Θ(1) + n * $\sum_{i=0}^{k-1}$ * 5 /2
  - k = $log_2 n$ $5^{log_2 n}$ / $2^{log_2 n}$ = $n^{log_2 5}$ / n = 1/2
  - T(n) = T(n/ $5^{log_2 n}$)*Θ(1) + n*1* ($(5/2)^k$ -1)/[(5/2) -1]
  - T(n) = T(n/ $n^{log_2 5}$)*Θ(1) + 2 * [(($n^{log_2 5}$)/ n)-1] / (1/2)
  - T(n) = T(n/ $n^{log_2 5}$)*Θ(1) + 2 * $n^{log_2 5}$ - 2n
  - T(n) = Θ($n^{log_2 5}$)
- Algoritmo B : Resolve o problema dividindo a entrada em dois subproblemas de tamanho n-1 (onde n é o tamanho da entrada) resolve cada subproblema recursivamente e depois conbina-os em tempo constante.
  T(n) = 2T(n - 1) + 1
  - T(n - 1) = 2T(n - 2) + 1   * 2^0
  - T(n - 2) = 2T(n - 3) + 1   * 2^1
  - ...
  - T(2) = 2T(1) + 1   * 2^{n-1}
  - T(1) = 1   * 2^n

- ---
- T(n) = 2^n *Θ(1) $\sum_{i=0}^{n-1}$ 2^i
- T(n) = 2^n *Θ(1) + 1 * (2^n -1)/(2 - 1)
- T(n) = 2^n *Θ(1) + 2^n - 1
- T(n) = Θ (2^n)

- Algoritmo C : Resolve o problema dividindo a entrada em nove subproblemas com um terço do tamanho, resolve cada subproblema recursivamente e depois combina-os em tempo quadrático.
  T(n) = 9T(n/3) + n^2
  - T(n/3) = 9T(n/ 3^2) + n^2 * 9^0
  - T(n/ 3^2) = 9T(n/ 3^3) + ($(n/3^2)^2$)   * 9^1
  - T(n/ 3^3) = 9T(n/ 3^4) + ($(n/3^3)^2$)   * 9^2
  - ...
  - T(n/ 3^{k-1}) = 9T(n/ 3^k) + ($(n/3^{k-1})^2$)   * 9^{k-1}
  - T(n/ 3^k) = Θ(1)   * 9^k
  - ---
  - T(n) = T(n/ 9^k)* Θ(1) + $\sum_{i=0}^{k-1}$ * 9 * $n^2/9$
  - k = $log_9 n$
  - T(n) = T(n/ $9^{log_9 n}$) * Θ(1) + n^2 * $\sum_{i=0}^{k-1}$ * 9/9
  - T(n) = n* Θ(1) + n^2 $log_9 n$
  - T(n) = Θ(n^2 log n)

  Resposta:

  O algoritmo C é o melhor. Pois os algoritmos A e B possuem um crescimento exponencial, Θ($n^{log_2 5}$) e Θ (2^n) respectivamente. Enquanto o algoritmo C tem crescimento logaritmico Θ(n^2 log n).