Universidade Federal de Minas Gerais
Departamento de Computação
Projeto e Análise de Algoritmos – 2024.2
Professor: Marcio Costa Santos
Lista 2

Considere que todas as recorrência descritas possuem caso base (ou casos bases) iguais a 1.

Exercício 1. *Determine e prova uma equivalência assintótica para todas as recorrências abaixo.*

a. $T(n) = T(n-3) + 1$

b. $T(n) = 2T(n-2) + \log n$

c. $T(n) = T(n-1) + n$

d. $T(n) = 2T(n-1) + n^2 + 1$

Exercício 2. *Determine e prove uma equivalência assintótica para todas as recorrências abaixo.Não use o teorema mestre!.*

a. $T(n) = 2T(\frac{n}{2}) + 1$.

b. $T(n) = 4T(\frac{n}{2}) + \log n$.

c. $T(n) = 7T(\frac{n}{3}) + n$.

Exercício 3. *Usando o teorema mestre determine uma equivalência assintótica para:*

i. $T(n) = 2T(\frac{n}{4}) + 1$

ii. $T(n) = 2T(\frac{n}{4}) + n$

iii. $T(n) = 2T(\frac{n}{4}) + \log n$

iv. $T(n) = 4T(\frac{n}{2}) + 1$

v. $T(n) = 4T(\frac{n}{2}) + n$

vi. $T(n) = 4T(\frac{n}{2}) + \log n$

vii. $T(n) = 2T(\frac{n}{2}) + 1$

viii. $T(n) = 2T(\frac{n}{2}) + \log n$

ix. $T(n) = 2T(\frac{n}{2}) + n$

x. $T(n) = 2T(\frac{n}{3}) + 1$

xi. $T(n) = 2T(\frac{n}{3}) + \log n$

xii. $T(n) = 2T(\frac{n}{3}) + n$

Exercício 4. *Determine um limite assintótico para* $T(n) = 2T(\sqrt{n})$. *Dica: Faça uma substituição de variável. Faça* $m = \log n$

Exercício 5. *Determine um limite assintótico para* $T(n) = 2T(\sqrt{n}) + \log n$. *Dica: Faça uma substituição de variável. Faça* $m = \log n$

Exercício 6. *Determine e prove uma equivalência assintótica para* $T(n) = T(\frac{n}{4}) + T(\frac{n}{5}) + T(\frac{n}{6}) + n$. *Dica 1: Divida a prova em limite inferior e limite superior; Dica 2: Aproxime a função para baixo e para cima usando seus próprios termos.*

Complexidade Amortizada

Para as questões a seguir considere uma pilha S que possui duas operações:

`pop(S)`**:** remove (desempilha) o topo da pilha S.

`push(S,x)`**:** empilha o elemento x na pilha S.

Cada uma dessas operações possui custo $O(1)$. Vamos definir uma nova operação para esta estrutura, a operação `multi-pop(S,k)` que remove os k últimos elementos empilhados.

Exercício 7. *Apresente um algoritmo para a operação de* `multi-pop`*. Observe que você pode usar a operação inicial de* `pop` *e a operação* `vazio(S)` *(que verifica se a pilha S é vazia ou não) no seu algoritmo.*

Exercício 8. *Qual a complexidade amortizada da operação de* `multi-pop` *dada uma sequência de operações de* `push`*,* `pop` *e* `multi-pop` *em uma pilha originalmente vazia?*

Exercício 9. *Qual o custo computacional de sequência de n operações de* `push`*,* `pop` *e* `multi-pop` *em uma pilha com inicialmente s_O elementos e que termina com s_n elementos?*