ACH-2002 - Introdução à Análise de Algoritmos - Prova 1, turma 04 – 25.10.2024 – Prof. Fábio Nakano

Nome: **nusp**

Orientações

- Duração: 1h30min. Entregar a avaliação ao professor.
- Mostre que você sabe, dê respostas detalhadas
- Escrever a lápis ou tinta, como preferir
- Escrever seu nome e número USP em todas as folhas. Quando houver local indicado, usá-lo.
- Entregar esta folha junto com as folhas de resposta.
- Indicar claramente a que questão e item refere-se a resolução
- Apresentar a resolução na ordem que preferir. (o enunciado deve ser lido sequencialmenete ;-)
- Na entrega, colocar as folhas de resposta e esta uma dentro da outra de forma que formem um único bloco.
- Caso o tempo tenha se esgotado e o professor precise ir ao aluno recolher a avaliação será atribuída nota ZERO. Avaliações que não forem entregues receberão nota ZERO.
- É proibida qualquer consulta, por exemplo (não limitado a) colegas, livros, anotações feitas antes da avaliação e anotações de colegas.
- Mostrar o encadeamento lógico das idéias e conceitos é essencial nas respostas.

---

1. Considere a execução do quicksort ,implementado como no livro-texto, no **melhor** caso. Qual a função de complexidade que corresponde à altura da árvore de execução (árvore de chamadas) do algoritmo? Justifique. **2,5pt**

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
```

2. Demonstre que $f(n)$ {é/não é} $O(g(n))$ (use a aproximação de Stirling, nota: independente do contexto em que foi apresentada, é uma função como muitas outras) **2,5pt**

   $f(n) = log(n!)$, $g(n) = n * log(n)$

3. Segundo o livro-texto, a complexidade de tempo de pior caso do Max-Heapify(A,i) é $T(n) \le T(2n/3) + \Theta(1)$. Explique por quê **1,5pt**.

```
MAX-HEAPIFY(A, i)
l = LEFT(i)
r = RIGHT(i)
if l ≤ A.heap-size and A[l] > A[i]
    largest = l
else largest = i
if r ≤ A.heap-size and A[r] > A[largest]
    largest = r
if largest ≠ i
    exchange A[i] with A[largest]
    MAX-HEAPIFY(A, largest)
```

   Aplique o Teorema Mestre (mostre que você testou os casos, diga qual a função de complexidade obtida pela aplicação do Teorema Mestre). **1,0pt**

4. Considere o Merge-sort codificado abaixo:

```

void mergeThreeSort(int *A, int ini, int fim) {

  if (ini >= fim) {
    return;
  } else {
    int terco1 = ini + (fim - ini) / 3;
    int terco2 = ini + (2 * (fim - ini) / 3);

    mergeThreeSort(A, ini, terco1);
    mergeThreeSort(A, terco1 + 1, terco2);
    mergeThreeSort(A, terco2 + 1, fim);
    merge(A, ini, terco1, terco2);
    merge(A, ini, terco2, fim);
  }
}
```

   a função merge é a mesma dada no livro-texto. Qual a função de complexidade de tempo em forma de recorrência? **1,5pt** Aplique o Teorema

Mestre (mostre que você testou os casos, diga qual a função de complexidade obtida pela aplicação do Teorema Mestre) **1,0pt**

Teorema Mestre:
Dada a recorrência $T(n) = aT(\frac{n}{b}) + f(n)$
caso 2: Se $f(n) \in \Theta(n^{log_b(a)}) \Rightarrow T(n) \in \Theta(n^{log_b(a)} * lg(n))$
caso 1: Se $f(n) \in O(n^{log_b(a)-\epsilon}) \Rightarrow T(n) \in \Theta(n^{log_b(a)})$
caso 3: Se $f(n) \in \Omega(n^{log_b(a)+\epsilon})$ ...
e $a * f(\frac{n}{b}) \leq c * f(n); 0 < c < 1 \Rightarrow T(n) \in \Theta(f(n))$
Aproximação de Stirling:

$$n! = \sqrt{2\pi n}(\frac{n}{e})^n(1 + O(\frac{1}{n}))$$

Alguns operadores assintóticos:
$O(g(n)) = \{f(n) : \exists\, c, n_0$, positivas, tais que $0 \leq f(n) \leq c \cdot g(n)\, \forall n \geq n_0\}$
$o(g(n)) = \{f(n) : \exists\, c, n_0$, positivas, tais que $0 < f(n) < c \cdot g(n)\, \forall n \geq n_0\}$
$\Theta(g(n)) = \{f(n) : \exists\, c_1, c_2, n_0$, positivas, tais que $0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\, \forall n \geq n_0\}$
Escrito usando Overleaf.