Universidade Federal do Ceará Centro de Ciências Departamento de Computação	1^a Avaliação Parcial Construção e Análise de Algoritmos (ck0183/ck0203) - 2021.1 Profa. Ana Karolinna Maia karolmaia@ufc.br Aluno: Matrícula:	Nota

- A prova deve ser escrita a mão (no papel) e fotografada/escaneada para enviar para correção em um único arquivo no formato PDF.
- O upload do arquivo deve ser feito pelo SIGAA.
- Nas questões para as quais a solução é um algoritmo, escreva-o em pseudocódigo. Respostas escritas em linguagens de programação não serão aceitas.

Questão 1 (4,0 pontos)**.** Considere o algoritmo abaixo que recebe um vetor $A[1..n]$ de números inteiros.

Algoritmo 1: X(A, i, j)

```
se A[i] > A[j] então
    Troca A[i] ↔ A[j]
se i + 1 ≥ j então
    Retorna
k ← ⌊(j − i + 1)/3⌋
X(A, i, j − k)
X(A, i + k, j)
X(A, i, j − k)
```

(a) Simule a execução do Algoritmo X no vetor $< 15, 42, 21, 50, 33, 65, 40, 43, 20 >$, com os valores inicias de $i = 1$ e $j = 9$.
Pontuação: 0,5 se mostrou apenas as trocas efetuadas; 1,0 para execuções mais detalhadas.

(b) Descreva sucintamente a funcionalidade do Algoritmo X.
Pontuação: 1,0.
Resposta: Ordenação (Algoritmo Stooge-Sort).

(c) Apresente uma recorrência para o pior caso de tempo de execução do algoritmo.
Pontuação: 0,5.
Resposta: $T(n) = 3T(2n/3) + \Theta(1)$.
Mostre um limite assintótico para a mesma.
Pontuação: 1,0.
Resposta: $T(n) = 3T(2n/3) + 1$
$= 9T(4n/9) + 3 + 1$
$\ldots$
$= 3^{log_{3/2} n} + \cdots + 3^2 + 3 + 1$
$= (3^{log_{3/2} n+1} - 1)/3 - 1$
$= \Theta(3^{log_{3/2} n})$
$= \Theta(3^{log_3 n / log_3 3/2})$
$= \Theta(n^{1/log_3 3/2})$

$= \Theta(n^{2.71})$

Poderia ser resolvido pelo método mestre.

Esse algoritmo é o mais eficiente possível para a realização da tarefa? Explique por que.
Pontuação: 0,5.
Resposta: É mais lento que outros algoritmos de ordenação estudados, mesmo o Insertion-Sort.

Questão 2 (3,0 pontos)**.** Assuma que o Algoritmo Particiona visto em sala, utilizado como subrotina do Algoritmo Mediana, funciona corretamente. Isto é, Particiona recebe um vetor $A[p..r]$ e um elemento $pivot \in A[p..r]$ e retorna o vetor A reorganizado de forma que todos os elementos posicionados antes do $pivot$ são menores do que ele e os elementos em índices superior à posição do $pivot$ são maiores do que ele. Prove que o algoritmo abaixo retorna corretamente a mediana de um vetor de n elementos, para os valores iniciais de $p = 1$ e $r = n$.

Algoritmo 2: Mediana(A, p, r)

```
se p = r = ⌈n/2⌉ então
    Retorna A[p]
pivot ← A[r]
q ← Particiona(A, p, r, pivot)
se q = ⌈n/2⌉ então
    Retorna A[q]
se q > ⌈n/2⌉ então
    Mediana(A, p, q − 1)
se q < ⌈n/2⌉ então
    Mediana(A, q + 1, r)
```

Resposta: Indução no tamanho do vetor. Vamos supor que $r \geq p$ e $n/2$ está entre p e r.
Base: Vetor com 1 elemento. O algoritmo retorna o elemento da posição $\lceil \frac{n}{2} \rceil$ (somente quando $p = r = \lceil \frac{n}{2} \rceil$.)
Hipótese: O algoritmo retorna corretamente o elemento que deveria estar na posição $\lceil \frac{n}{2} \rceil$ (se o vetor estivesse ordenado) quando recebe índices p e r tal que $r - p + 1 < k$.
Passo: $r - p + 1 = k$. O Algoritmo Particiona reorganiza o vetor de forma que todo elemento à esquerda do pivot é menor que ele e todo elemento à sua direita é maior. O pivot fica na posição q. Se $q = \lceil \frac{n}{2} \rceil$, então, pelo Particiona, temos que o elemento que está na posição $\lceil \frac{n}{2} \rceil$ é o que deveria está e o algoritmo retorna tal valor. Se $q > \lceil \frac{n}{2} \rceil$, então na posição $\lceil \frac{n}{2} \rceil$ do vetor deve ficar um elemento menor que o pivot, e tal elemento está entre os atuais elementos de $A[p..q-1]$, pois, pelo Particiona, todos os elementos em outras posições são maiores. Como $A[p..q-1]$ tem pelo menos 1 elemento a menos que $A[p..r]$, aplicamos a hipótese de indução para argumentar que o algoritmo encontra o valor correto. O caso em que $q < \lceil \frac{n}{2} \rceil$ é análogo ao anterior.

Questão 3 (3,0 pontos)**.** Sejam $X[1...n]$ e $Y[1...m]$ dois vetores ordenados. Elabore um algoritmo para encontrar o k-ésimo menor elemento da união de X com Y. Descreva um algoritmo para este problema que execute em tempo $O(\log k)$. Resposta: Fazer chamada ao seguinte algoritmo com valor inicial $n = k$.

```
k-ésimo (X, Y, n)
  Se n = 1
    Retorna min (X[1], Y[1])
  Se X[n/2] < Y[n/2]
    Retorna k-ésimo(X[n/2...n], Y[1..n/2], n/2)
  Senão Retorna k-ésimo(X[1..n/2], Y[n/2..n], n/2)
```