MC458 - Projeto e Análise de Algoritmos I
Exame - 10/12/2018

Instruções:

1. Comece pelas questões que você tem mais certeza de saber fazer.
2. Não perca tempo com detalhes menores. Foque no que é relevante para resolver a questão e passe logo para a próxima.
3. Se sobrar tempo, você pode acrescentar detalhes às questões já resolvidas.

Questão 1 (valor 2,5) Analise o algoritmo a seguir e dê a ordem de crescimento do número de operações básicas para o pior caso e para o melhor caso em função de n, o tamanho do vetor A:

```
procedure InsertionSort(A)
    n ← length[A]
    for j ← 2 to n do
        key ← A[j]
        i ← j − 1
        while i ≥ 1 and A[i] > key do
            A[i + 1] ← A[i]
            i ← i − 1
        A[i + 1] ← key
```

Questão 2 (Valor 2,5) Suponha que sejam dados n números inteiros no intervalo $[0..n^4 - 1]$. Descreva um método para ordenar estes números em $O(n)$.

Você pode usar qualquer algoritmo visto em classe sem precisar produzir o código deste algoritmo, mas especifique-o bem e defina claramente seus parâmetros, se houver.

Questão 3 (Valor 2,5) Um grupo de amigos, identificados pelos números de 1 a n, reúne-se toda quinta-feira à noite para seu tradicional jogo de pôquer. Para registrar os resultados das rodadas, eles mantêm uma matriz M de dimensão $n \times n$, que começa zerada. A cada jogo, se o jogador i perdeu o valor v para o jogador j, a entrada $M[i,j]$ é acrescida de v. As entradas da matriz M são sempre positivas ou nulas, e a diagonal é sempre nula. No final da noite, a matriz é consultada para saldar as dívidas. Por exemplo, considere a matriz final abaixo:

	1	2	3	4
1	0	1	8	1
2	0	0	5	1
3	2	3	0	1
4	1	4	6	0

Nela podemos ver que o jogador 1 deve 8 para o jogador 3, mas também o jogador 3 deve 2 para o jogador 1. Como resultado, o saldo líquido que 1 deve pagar a 3 é 6.

Um *ganhador da noite* é um jogador i tal que $M[i,j] < M[j,i]$ para todo $j \neq i$. Escreva um algoritmo que, dada a matriz M, retorne um ganhador da noite, ou 0 se não houver tal jogador. Analise a complexidade computacional de seu algoritmo. Tente fazer um algoritmo mais eficiente possível assintoticamente.

Questão 4 (Valor 2,5) Exiba um código binário ótimo para a seguinte lista de caracteres e suas frequências:

`a`: 1; `b`: 1; `c`: 3; `d`: 3; `e`: 4.

Mostre que seu código é otimo. Por exemplo, se você o obteve através do algoritmo de Huffman, isto é um argumento suficiente, pois o mencionado algoritmo garante códigos ótimos.

Boa sorte!

Soluções

Questão 1 (valor 2,5)

Na tabela abaixo, o melhor caso reflete a contagem de operações básicas em cada linha quando o vetor está em ordem crescente, e o pior caso reflete a situação com o vetor em ordem decrescente.

Linha	Melhor caso	Pior caso
1	$O(1)$	$O(1)$
2	$O(1)$	$O(1)$
3	$O(n)$	$O(n)$
4	$O(n)$	$O(n)$
5	$O(n)$	$O(n)$
6	$O(n)$	$O(n^2)$
7	$O(n)$	$O(n^2)$
8	$O(n)$	$O(n^2)$
9	$O(n)$	$O(n)$
Total	$O(n)$	$O(n^2)$

Questão 2 (valor 2,5)

Usar RadixSort com base n. Teremos n números com 4 dígitos cada um. O algoritmo RadixSort fará 4 iterações de CountingSort, sendo que cada iteração de CountingSort gasta $O(n+k)$, onde k é o tamanho do intervalo dos dígitos. No nosso caso, $k = O(n)$.

A complexidade total será então $O(4(n+k)) = O(4(2n)) = O(8n) = O(n)$.

Questão 3 (valor 2,5)

Este problema pode ser resolvido com a mesma estratégia usada no problema da celebridade. Pode-se eliminar um ganhador da noite com cada pergunta. Começamos com um candidato G a ganhador igual ao primeiro jogador, e a cada iteração o candidato é testado contra um novo jogador. No final, o candidato restante é testado para ver se realmente é ganhador da noite.

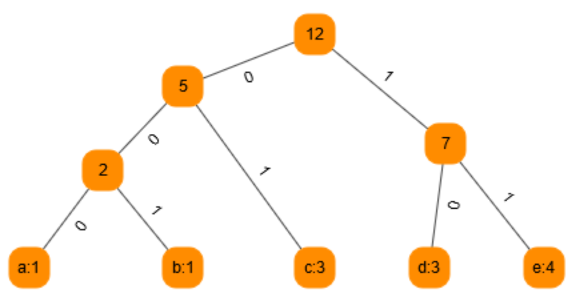
function GanhadorDaNoite(M, n)
  $G \leftarrow 1$
  for $i \leftarrow 2$ **to** n **do**
    if $M[G, i] > M[i, G]$ **then**
      $G \leftarrow i$
  for $i \leftarrow 1$ **to** n **do**
    if $i \neq G$ **and** $M[G, i] \geq M[i, G]$ **then**
      $G \leftarrow 0$
      break
  return G

Questão 4 (valor 2,5)

Uma possível lista de passos para o algoritmo de Huffman:

junta `a` e `b`
junta `ab` e `c`
junta `d` e `e`
junta `abc` e `de`

Árvore final resultante:



Código ótimo:

Caractere	Código
a	000
b	001
c	01
d	10
e	11