Prova 2005/1

Questão 1 - Sejam $T(n)$ e $f(n)$ funções dos inteiros nos reais

a) O que significa "$T(n)$ é $O(f(n))$"?

Significa que existem constantes positivas c e n_0 tal que $T(n) \le c \cdot f(n)$ para $\forall\, n \ge N_0$

b) É verdade que $20n^3 + 10n\lg n + 5$ é $O(n^3)$? Justifique.

Vou provar que $20n^3 + 10n\lg n + 5 \le 35n^3$ para $n \ge 1$

Prova:

$$\begin{aligned} 20n^3 + 10n\lg n + 5 &\le 20n^3 + 10n^3 + 5 \\ &\le 20n^3 + 10n^3 + 5n^3 \\ &= 35n^3 \;/\!/ \end{aligned}$$

Portanto $20n^3 + 10n\lg n + 5 = O(n^3)$

c) É verdade que $\frac{1}{2}n^2$ é $O(n)$? Justifique

Não é verdade. Vou provar por absurdo que $\frac{1}{2}n^2$ não é $O(n)$.

Prova

Suponhamos que existam constantes positivas c e n_0 tal que $\frac{1}{2}n^2 \le c \cdot n$ para $\forall\, n \ge N_0$ então temos que

$\frac{1}{2}n^2 \le cn$	$c \ge \frac{1}{2}n$
$\frac{\frac{1}{2}n^2}{n} \le c$	o que é absurdo sendo c uma constante.

d) O que significa "$T(n)$ é $\Omega(f(n))$"?

Significa que existem constantes c e n_0 tal que $T(n) \ge c \cdot f(n)$ para todo $n \ge N_0$.

e) O que significa "$T(n) = n + \Omega(n\lg n)$"?

É um abuso de notação. Significa que $T(n) + n \ge c \cdot n\lg n$

para todo $n \ge n_0$ sendo c e n_0 constantes positivas.

Questão 2 - Considere o seguinte algoritmo que recebe um vetor de números inteiros $A[1..n]$ e um número inteiro positivo k e devolve um vetor $B[1..k]$ com os k maiores elementos de A.

```
Maiores(A, n, k)
1 BUILD-MAX-HEAP(A, n)
2 para i ← 1 até k faça
3     B[i] ← HEAP-MAXIMUM(A)
4     HEAP-EXTRACT-MAX(A, n)
5 devolva B
```

a) Qual o consumo de tempo do algoritmo Maiores? Justifique

Linha	Consumo
1	$O(n)$
2	$O(k)$
3	$O(k) \cdot O(1)$
4	$O(k) \cdot O(\lg n)$
5	$O(1)$

total $= O(n + k \cdot \lg n)$

Mais detalhado:

$$\sum_{i=0}^{k-1} \lg(n-i)$$

b) Para quais valores de k o consumo de tempo do algoritmo MAIORES é $O(n)$? Expresse a sua resposta em função de n e justifique-a.

$$O(n + k\lg n) = O(n)$$

$$k\lg n \le 1 \cdot n$$

$$\boxed{k \le \frac{n}{\lg n}}$$

Questão 3 - Considere o seguinte algoritmo que tem como argumentos dois números inteiros n e r, $n \geq 0$ e $r \geq 0$, e usa como subrotina um algoritmo CAIXA-PRETA

```
Algo(n, r)
  se n = 0
     então CAIXA-PRETA(n, r)
           devolva 1
  K ← r × ALGO(n-1, r)
  para i ← 1 até K faça
       CAIXA-PRETA(n, i)
  devolva K
```

Qual o nº de vezes que o algoritmo CAIXA-PRETA é chamado pelo algoritmo ALGO? Expresse esse número como função de n e r e justifique a sua resposta:

Tamanho da instância = (n, r)

$T(n,r)$ é o nº de vezes que CAIXA-PRETA executa.

A partir do algoritmo Algo $T(n)$ pode ser resolvido a partir da recorrência:

$T(0,r) = 1$

$T(n,r) = T(n-1,r) + r^2$

$$T(n,r) = T(n-1,r) + r^2$$
$$= T(n-2,r) + r^{n-1} + r^2$$
$$= T(n-3,r) + r^{n-2} + r^{n-1} + r^2$$
$$= T(n-i,r) + \sum_{k=0}^{i-1} r^{(n-k)}$$
$$= T(0,r) + \sum_{k=0}^{n-1} r^{(n-k)}$$
$$= 1 + \sum_{k=1}^{n} r^k$$
$$= 1 + \sum_{k=0}^{n} r^k - 1$$
$$= \frac{r^{n+1} - 1}{r - 1} \quad \text{para } r > 1$$

$n - i = 0$

$\boxed{n = i}$

para $n = 1$

$T(n,r) = n + 1$

Para $n = 0$

$T(n,r) = 1$

Questão 4 - Suponha dado um conjunto de livros numerados de 1 a n. Suponha que o livro i tem peso $p[i]$ e que $0 < p[i] < 1$ para cada i. Considere o problema de acondicionar os livros no menor número possível de envelopes de modo que cada envelope tenha no máximo 2 livros e o peso do conteúdo de cada envelope seja no máximo 1. Escreva um algoritmo guloso MIN-ENV(p, n) que recebe um vetor $p[1..n]$ e devolve o número mínimo de envelopes. O consumo de tempo do seu algoritmo deve ser $O(n^2)$. Argumente porque seu algoritmo produz a resposta correta e tem o consumo de tempo pedido.

O algoritmo MIN-ENV recebe um vetor $p[1..n]$ onde $0 < p[i] < 1$ para $i = 1..n$ e um inteiro n e devolve o nº mínimo de envelopes com elementos de p.

```
MIN_ENV(p, n)
1  ordena p por peso crescente
2  i ← 1
3  j ← n  env ← 0
4  enquanto i ≤ j faça
5        se p[i] + p[j] ≤ 1
6           então env ← env + 1
7                 i ← i + 1
8                 j ← j - 1
9           senão env ← env + 1
10                j ← j - 1
11 devolva env
```

Consumo:

Linha	consumo
1	$n \lg n$
2-3	$O(1)$
4-10	$O(n)$
11	$O(1)$

O consumo total é $O(n \lg n)$ considerando que a ordenação é feita por um algoritmo Mergesort ou heapsort. O agrupamento nos envelopes leva no máximo $O(n)$.

Corretude:

Escolha gulosa

Teorema: Qualquer solução ótima possui aproveitamento máximo de cada envelope.

Prova:

Suponhamos que temos uma solução ótima com um dos envelopes com $x+y \leq 1$ sendo $x = min\{1..n\}$ e $y = max\{1..n\}$, agora vamos supor que tenhamos uma outra solução ótima com $k+a$ sendo

$x \leq k$ e $y = a$ - então se $k > x$ quer dizer que existe a possibilidade do o envelope que conterá x ficar subutilizado

$x = k$ e $y \leq a$ - então se $a < y$ quer dizer que existe a possibilidade de $a+k$ utilizar um envelope e ser necessário mais 2 envelopes com k e y já que y é maior e combinaria com x mas ele foi utilizado por outro que tb combinaria com um outro elemento maior que k

Subestrutura ótima

Teorema: Seja A uma solução ótima para $1..n$ com tamanho k de envelopes então a escolha gulosa do mínimo e do máximo elemento produz solução ótima também para a instância $1.2..n-1$ se $p_1 + p_n \leq 1$ ou

$2.1..n-1$ se $p_1 + p_n > 1$

Prova: 1- Vamos supor $A_{2..n-1}$ não é solução ótima, então existe uma outra solução A^* que possui envelopes $< k-1$ então adicionando 1 envelope correspondente a $1+n$,

teríamos uma solução ótima com menos envelopes que k o que é uma contradição.

2- Suponhamos que $A_{1..n-1}$ com env $= k-1$ não é uma solução ótima e então existe uma outra solução que possui menos envelopes que $k-1$ dessa forma se adicionarmos o último envelope teríamos uma solução para a instância $1..n$ com menos que k envelopes. Uma contradição.

Questão 5 - Escreva um algoritmo EMBARALHAMENTO(Z, X, m, Y, n) que recebe os vetores $Z[1..m+n]$, $X[1..m]$ e $Y[1..n]$ e devolve 1 se Z é um embaralhamento de X e Y e devolve 0 em caso contrário. O consumo de tempo do seu algoritmo deve ser $O(m+n)$. Argumente porque seu algoritmo produz a resposta correta e tem o consumo de tempo pedido.

```
Embaralhamento (Z, X, m, Y, n)
  k ← 1   i ← 1   j ← 1
  enquanto k ≤ m+n
      se Z[k] = X[i]
          então k ← k+1
                i ← i+1
          senão B[j] ← Z[k]
                k ← k+1
                j ← j+1
  se i = m+1 e j = n+1
      então Para j ← 1 até n
                se Y[j] ≠ B[j]
                    então devolva 0
            devolva 1
      senão devolva 0
```

/ /

Consumo
$T(n) = O(m+n)$
Corretude
Se o embaralhamento permite qq posição entre X e Y desde que permaneçam suas posições. A solução é "extrair" os dois vetores de Z e confirmar suas posições.

Questão 6 - Responda cada um dos itens abaixo e dê uma justificativa curta para a sua resposta
a) O que significa dizer que um problema π pode ser polinomialmente reduzido a um problema π'?
Significa que qualquer instância de π pode ser reformulada como uma instância de π', cuja solução fornece resposta para a instância de π
(Significa que o problema π' não é mais fácil que o problema π)

b) Defina as classes P e NP e problema NP-completo.
Classe P - Problemas que podem ser resolvidos em tempo polinomial, ou seja, em tempo $O(n^k)$ para k constante e n como tamanho da entrada.

Classe NP - Problemas que são verificáveis em tempo polinomial. Se tivéssemos um certificado para a solução esse certificado poderia ser verificado em tempo polinomial, $O(n^k)$ k constante, n (tamanho da entrada).

Classe NP-completo: Problemas que estão em NP e que todos os problemas em NP podem ser reduzidos polinomialmente a ele.

/ /

c) Se um problema π pode ser polinomialmente reduzido a um problema π' e π' está em P então π está em P?
Sim. Pois P é fechado polinomialmente.

d) Se um problema π pode ser polinomialmente reduzido a um problema π' e π' é NP-completo então π é NP-completo?
Não, pois é garantido ser NP ou NP-completo.

e) Há problemas em NP que não são NP-completos?
Sim desde que $P \neq NP$.

f) $P \cap NP \neq \emptyset$?
Sim. $P \subseteq NP$

g) Existem problemas NP-completos em P?
Considerando que $P \neq NP$, não existem problemas NP-completos em P.

Questão 7 - [redacted] pg 105
Questão 8 - a) Escreva um algoritmo MEDIANAS-1(A, n, K) que recebe um vetor de números inteiros A[1..n] e um número inteiro positivo k e devolve um vetor B[1..n-K+1] das medianas dos K-segmentos de A. Seu algoritmo deve consumir tempo $O(nk)$. Explique sucintamente porque seu algoritmo está correto e tem o consumo de tempo pedido.

```
MEDIANAS-1(A, n, K)
1   q ← ⌊K+1/2⌋
2   Para i ← 1 até n-K+1
3       A' ← A[i..i+K-1]
4       x ← SELECT-BFPRT(A', 1, K, q)
5       B[i] ← A[x]
6   devolva B
```

O invariante:
na linha 2 B[1..i-1] é um vetor das medianas dos K-segmentos de A[1..i+K-2]

1+3-2 3-2 1=2
1..1-2 -1+3 2
1..2 3/2

PanAmericana

/ /

Consumo

A linha 2 consome $O(n-k+1)$ com k é constante e podemos considerar $O(n)$. O algoritmo utiliza o algoritmo select BFPRT que recebe um vetor de k elementos e o q-ésimo menor elemento. Tem consumo $O(k)$.

A linha 3+b consome $O(k)$. Portanto o algoritmo possui consumo $O(n \cdot k)$

b) Escreva um algoritmo Mediana2(A,n,k) que recebe um vetor de n inteiros $A[1..n]$ e k e devolve um vetor $B[1..n-k+1]$ das medianas dos k-segmentos de A. Seu algoritmo deve ter tempo $O(n \lg k)$. Explique pq ta correto e tem o consumo. Dica: usar árvore balanceada.

Mediana2(A,n,k)

1	TREE-BUILD$(T, A, 1, k-1)$	$O(k \lg k)$
2	Para $i \leftarrow 1$ até $n-k+1$	$O(n)$
	TREE-INSERT$(T, A[i+k-1])$	$O(\lg k) \cdot O(n)$
	$B[i] \leftarrow T[0]$	$O(1) \cdot O(n)$
	TREE-DELETE$(T, A[i])$	$O(\lg k) \cdot O(n)$
	devolva B	$O(1)$

Consumo

$O(n \lg k)$

Corretude

Inicia-se a árvore com $k-1$ elementos. Para cada nó se estabelece a árvore $T[i..i+k-1]$ e pega-se sua raiz que representa a mediana e após isso deleta-se o elemento $A[i]$ permanecendo com $T[i..i-k-2]$ sempre na linha 2, para que ela unir à o próximo elemento e deletado o 1º.

$V[i] - V[i] - 1$

/ /

~~Bx~~ PROVA 2004/1

Questão 1 - pg 138

Questão 2 - Meu manual diz que um problema "está em NP" concluo que não existe algoritmo polinomial para o problema. Estou certo ou errado? Justifique.

Errado. Pois $P \subseteq NP$ dessa forma existe um subconjunto em NP onde os problemas são resolvidos em tempo polinomial.

Questão 3 -

FATOR(n)
1 Para $d \leftarrow 2$ até $n-1$
2 se resto$(n/d) = 0$
3 devolva d
4 devolva 0

Este algoritmo não é polinomial. Está certo?

Tamanho da instância: $(\lfloor \log(n+1) \rfloor)$ → nº de dígitos e $n = 2^{\lfloor \lg(n+1) \rfloor}$

Linha consumo
1-3 $O(2^{\lfloor \log(n+1) \rfloor})$
4 $O(1)$

$T(\lfloor \log n+1 \rfloor) = 2^{\lfloor \log(n+1) \rfloor}$

Está certo

Seja o tamanho da entrada m (nº de dígitos), então o consumo de tempo será $T(n) = O(n)$ mas $m = \lg n \Rightarrow 2^m = 2^{\lg n}$
$\Rightarrow \boxed{2^m = n}$

Questão 4 - $|V[i] - V[i+1]| \leq 1$ - BUSCA(V, n, x) - $O(\lg n)$

Ex: 1 2 3 4 3 4 5 4 5 6 K-ordenado

BUSCA(V, p, r, x)
se $p = r+1$
se $A[r] = x$
devolva r
senão devolva -1

senão $q \leftarrow \lfloor p+r/2 \rfloor$
se $A[q] < x$
então devolva BUSCA$(V, q+1, r, x)$
senão devolva BUSCA$(V, p, q-1, x)$