Complexidade – exame

Nome:	**NMEC**: .

<u>**Importante.**</u> **Esta prova consta de 4 páginas. Se num qualquer problema efectuar uma redução (polinomial ou não) descreva claramente a função de transformação associada à redução. Abreviaturas: $\uparrow(x)$– função não definida em nenhum ponto, NPC–"completo em NP", e co-NPC–"completo em co-NP", PR– "definida por recursão primitiva", rec–"recursiva", r.e.–"recursivamente enumerável", dec–"decidível", s-dec–"semi-decidível", co-s-dec–"co-semidecidível". Os problemas que pode utilizar como sendo NPC são: 3SAT, VC, Clique, HC (ciclo hamiltoniano), PART e 3DM. Sabe-se que problemas como 2SAT, 2COL e EC pertencem à classe P.**

1. As seguintes afirmações são verdadeiras ou falsas?

   Nota importante. *Escreva V ou F dentro do quadrado respectivo. Cada quadrado: 100% se resposta certa, 0% se não responde, -100% se resposta errada.*

   [V] $n^3 \in \Omega(n \log n)$.

   [F] Para qualquer função $f : \mathbb{N} \to \mathbb{R}^+$ é $O(f(n)) \subseteq \Theta(f(n))$.

   [V] Uma recorrência cuja equação geral é $f(n+2) - 2f(n+1) + 3f(n) = 2^n + 3^n + 1$ pode ser resolvida pelo método da equação característica.

   [F] $\{i : \{i\}(x)$ pára para 3 ou mais valores de $x\}$ é uma linguagem co-r.e.

   [F] Usando (apenas) o Teorema de Rice pode concluir-se que o seguinte problema não é s-dec: "dado i, $\{i\}(x)$ pára para todo o $x \in \mathbb{N}$?".

   [V] O programa FOR "`inc x0; for x{inc x0}; dec x0`" implementa a função `f(x)=x`.

   [F] Seja qual for `x`, o programa WHILE "`while x{}`" não termina.

   [V] A função de Ackermann é um exemplo de uma função computável total que não pertence à classe PR

   [F] O problema: "dado um grafo não dirigido G, existe uma cobertura de vértices de G com exactamente 5 vértices?" é completo em NP.

   [V] Se P≠NP não existe redução polinomial SAT $\propto$ 2SAT.

   [V] Com base num algoritmo polinomial para o problema Clique, seria possível resolver em tempo polinomial o problema TAUT.

   [F] O problema EME (expressão mínima equivalente) pertence à classe Σ_1^p (supõe-se que a hierarquia polinomial não colapsa, isto é, que para todo o $k \geq 1$ é $\Sigma_k^p \neq \Sigma_{k+1}^p$).

2. Análise de algoritmos

   (a) Considere a função f: $\mathbb{N} \to \mathbb{R}$ seguinte: $f(n) = n^2$ para n par e $f(n) = 3n^2$ para n ímpar. $f(n) \in \Theta(n^2)$ (responda `V` ou `F`): `V` Demonstração:

   Temos: $\forall n \in \mathbb{N} :\ n^2 \leq f(n) \leq 3n^2$. Basta pois tomar $n_0 = 1$, $k_1 = 1$ e $k_2 = 3$ na definição da ordem de grandeza Θ que se recorda a seguir.
   Diz-se que $f(n) \in \Theta(g(n))$ quando

   $$\exists n_0 \in \mathbb{N},\ \exists k_1, k_2 \in \mathbb{R}^+,\ \forall n \geq n_0 :\ k_1 g(n) \leq f(n) \leq k_2 g(n)$$

(b) O tempo de execução de um algoritmo é determinado pela seguinte recorrência

$$\begin{cases} f(0) = 1 \\ f(n+1) = f(n) + 3n^2 + 3n + 1 \end{cases}$$

Resolva a recorrência pelo método "tabelar/suspeitar/demonstrar". **Notas.** Pode ter interesse tabelar n^2 e n^3. Apresente apenas a suspeita e a demonstração.

Suspeita: $f(n) = n^3 + 1$.
Teorema: a solução da recorrência é $f(n) = n^3 + 1$.
Demonstração: por indução matemática em n.
a) Caso $n = 0$: $f(0) = 1$ (da rec.) e $0^3 + 1 = 1$ (da fórmula).
b) Vamos provar que $[f(n) = n^3 + 1] \Rightarrow [f(n+1) = (n+1)^3 + 1]$:

$$\begin{array}{rcll} f(n+1) & = & f(n) + 3n^2 + 3n + 1 & \text{da recorrência} \\ & = & n^3 + 1 + 3n^2 + 3n + 1 & \text{da hip. indutiva} \\ & = & (n^3 + 3n^2 + 3n + 1) + 1 & \\ & = & (n+1)^3 + 1 & \end{array}$$

$\square$

3. Computabilidade para complexidade

(a) Sejam X e Y problemas de decisão. Explique porque é que, a partir do facto X $\leq$ Y se pode concluir que se L_{Y} for s-dec., então L_{X} é s-dec.

Por definição, se L_Y é s-dec, existe uma uma função computável parcial ϕ tal que

$$\phi(a) = \begin{cases} 1 & \text{se } a \in L_Y \\ \uparrow & \text{se } a \notin L_Y \end{cases}$$

Como $X \leq Y$, existe uma função computável total f tal que $x \in L_x$ sse $f(x) \in L_Y$.
Então temos

$$\phi(f(a)) = \begin{cases} 1 & \text{se } a \in L_X \\ \uparrow & \text{se } a \notin L_X \end{cases}$$

Como a função composta $\phi \circ f$ é computável (parcial), L_X é s-dec.

(b) Mostre que o seguinte problema (M3) não é s-dec
INSTÂNCIA: Um índice i de uma MT.
PERGUNTA: $\{i\}(x)\downarrow$ sse x é múltiplo de 3?
Sugestão. Sabe-se que $\overline{\text{PAP}}$ não é s-dec; use uma redução $\overline{\text{PAP}} \leq$ M3. Descreva a redução e complete a prova.

Temos que definir uma função computável f total tal que

$$\begin{array}{ccl} \overline{\text{PAP}} & \leq & \text{M3} \\ i & \rightarrow & k = f(i) \\ \{i\}(i)\uparrow & \text{sse} & \{k\}(x)\downarrow \text{ sse } x \text{ múltiplo de 3} \end{array}$$

$k = f(i)$ é o índice de uma máquina de Turing com o seguinte comportamento:

Se x é múltiplo de 3, pára
senão:
  Apaga a fita
  Escreve i na fita
  Simula $\{i\}(i)$

$\Rightarrow$ Se $i \in L_{\text{PAP}}$, isto é, se $\{i\}(i)\uparrow$, a máquina de Turing de índice k pára quando x é múltiplo de 3 e não pára quando x não é múltiplo de 3.
$\Leftarrow$ Se $k \in L_{\text{M3}}$, isto é, se Se a computação $\{k\}(x)$ termina sse x é múltiplo de 3, isso implica $\{i\}(i)\uparrow$ (pois, se $\{i\}(i)\downarrow$, a computação termina para todo o x).
Se M3 fosse s-dec, $\overline{\text{PAP}}$ também seria s-dec (ver a alínea 3(a)), o que sabemos ser falso.
Logo M3 não é s-dec.

4. Teoria dos problemas completos em NP

(a) Sejam A e B problemas da classe P, sendo B não trivial (isto é, tal que existe pelo menos uma instância com resposta $\textsc{Sim}$ e pelo menos uma instância com resposta $\textsc{Não}$). Mostre que se verifica sempre $A \propto B$.

A ideia é incluir na função f associada à redução $A \propto B$ a solução do problema A (o que se pode fazer, pois existe um algoritmo polinomial para A), e, conforme o rsultado, retornar um elemento apropriado de B.
Mais concretamente, seja b uma instância de B com resposta SIM e seja b' uma instância de B com resposta NÃO (tais instâncias existem porque B não é trivial). Define-se a redução $A \propto B$ através da função f:

$$f(x) = \begin{cases} b & \text{se a resposta a } x \text{ em } A \text{ é SIM} \\ b' & \text{se a resposta a } x \text{ em } A \text{ é NÃO} \end{cases}$$

(o conjunto imagem $f(L_A)$ reduz-se a$\{b, b'\}$) A função f pode ser implementada em tempo polinomial (dado que $A \in$ P) e $x \in L_A$ sse $f(x) \in L_B$.

(b) Uma fórmula lógica está na forma DNF (forma disjuntiva normal) quando consiste numa disjunção de claúsulas, sendo cada claúsula uma conjunção de literais; cada literal é uma variável lógica ou uma variável lógica negada. Um exemplo: $(a \wedge \overline{b}) \vee (b \wedge \overline{c} \wedge \overline{d}) \vee (\overline{a})$.
Considere o problema de decisão "FORMA-DNF":
INSTÂNCIA: Uma fórmula lógica E na forma DNF. PERGUNTA: Existe alguma valorização lógica das variáveis de E que torna E falsa?
(para o exemplo dado a resposta é sim; considere-se por exemplo a valorização $\nu(a) = \nu(b) = \nu(c) = \nu(d) =$ `Verd`.) Mostre que FORMA-DNF é um problema completo na classe NP. Para esse fim, que redução polinomial vai utilizar? `3SAT` $\propto$ `FORMA-DNF`
Descrição da redução e justificação de que FORMA-DNF é completo em NP:

(1) Redução pretendida:

3SAT	$\propto$	FORMA-DNF
(U, C)	$\rightarrow$	E
C satisfazível	sse	E falsificável

onde "E falsificável" significa que existe uma valorização lógica que torna E falsa. Consideremos o conjunto da claúsulas C de 3SAT na forma conjuntiva normal; a sua negação pode ser transformada (pelas leis de Morgan) numa expressão na forna DNF; esta transformação pode facilmente ser efectuada em tempo polinomial. Por exemplo, se $C = (\overline{u} \vee v \vee w) \wedge (u \vee \overline{w} \vee \overline{z})$, temos

$$\overline{C} = \overline{(\overline{u} \vee v \vee w) \wedge (u \vee \overline{w} \vee \overline{z})} = (u \wedge \overline{v} \wedge \overline{w}) \vee (\overline{u} \wedge w \wedge z)$$

É óbvio que existe uma valorização lógica que torna C verdadeira sse existe uma valorização lógica que torna $\overline{C}$ falsa (a valorização pode ser a mesma). Concluimos pois que a existência da função

$$f(U, C) = \overline{C} \text{ (com } \overline{C} \text{ escrito na forma DNF)}$$

(computável em tempo polinomial) permite concluir que 3SAT $\propto$ FORMA-DNF.

(2) O problema FORMA-DNF pertence à classe NP, pois, dada uma palavra y, pode verificar-se em tempo polinomial se

– y é uma valorização lógica das variáveis de E e

– com essa valorização, E assume o valor FALSO.

Nota. Os problemas "uma expressão dada E na forma CNF é falsificável?" e "uma expressão dada E na forma DNF é satisfazível?" pertencem à classe P.