# **Computabilidade e Complexidade**

1. **Disserte sobre a Tese de Church-Turing. Deixe claro (i) o que ela afirma, e (ii) qual o seu significado para a Ciência da Computação. Explique também (iii) porque ela não pode ser provada formalmente e (iv) por que, mesmo sem prova formal da sua verdade, ela é aceita como verdadeira sendo um dos fundamentos da Teoria da Computabilidade.**
2. **Um dos mais importantes e interessantes teoremas da Teoria da Computabilidade é o Teorema de Rice. Ele afirma que toda propriedade não trivial de funções computadas por programas é não decidível. Uma propriedade de funções parciais é dita trivial se ela é verdadeira para todas as funções ou para nenhuma. Note que o Teorema se refere a funções computadas por programas e não à representação/forma como a função é computada. Propriedades tais como “o número de transições de um programa em uma Máquina de Turing é maior que 100” ou “o número de linhas de código de um programa em C é igual a 5000”, por exemplo, são propriedades da representação/forma, não da função computada.**
   1. **Explique por que o teorema se refere apenas a propriedades não triviais, ou seja, por que ele também não “cobre” propriedades triviais.**
   2. **Dê pelo menos três exemplos de propriedades não triviais de funções computadas por programas.**
   3. **Disserte sobre a significância do Teorema de Rice para a prática da Ciência da Computação.**
3. **Dê uma prova da indecidibilidade do Problema da Parada usando o Teorema de Rice.**
4. **Explique como uma redução pode ser usada para provar que um problema é indecidível.**
5. **Um conjunto S é contável se é vazio ou se existe uma sequência s0, s1, s2,... com todos (e somente todos) os elementos de S. Está sequência é chamada de uma enumeração de S.**

**Exemplos de conjuntos contáveis:**

* **O conjunto dos números naturais é contável. Uma sequência óbvia mencionando todos os elementos é 0, 1, 2, ...**
* **O conjunto Z de todos os números inteiros é contável. E uma sequência para seus elementos é 0, 1, -1, 2, -2, ...**
* **O conjunto N x N é contável. Uma sequência é (0, 0), (0, 1), (1, 0) ...**

**Esta terminologia também se aplica a conjuntos de funções, parciais ou totais. Sejam A e B conjuntos e seja S um conjunto não vazio cujos elementos são funções parciais de A para B. Então S é contável se existe uma sequência f0, f1, f2,... tal que g ∈ S se e somente se g = fi para algum i, ou seja, g ocupa alguma posição i na sequência.**

**Prove as seguintes afirmações:**

* 1. **O conjunto de todas as funções totais de N para N é incontável.**
  2. **O conjunto de todas as funções parciais de N para N é incontável.**
  3. **O conjunto de todas as funções computáveis parciais de N para N é contável.**
  4. **O conjunto de todas as funções computáveis totais de N para N é contável.**
  5. **Existe função não computável total de N para N.**
  6. **Existe função não computável parcial de N para N.**

**Observações:**

1. **A prova de (a) acima pode ser feita por contradição usando o método da diagonalização.**
2. **A prova de (b) segue imediatamente do resultado (a) acima e de teoria dos conjuntos.**
3. **A prova de (c) faz uso da Tese de Church-Turing.**
4. **A prova de (d) é imediata a partir da afirmação (c) e de conhecimentos básicos de funções e de teoria dos conjuntos.**
5. **A prova de (d) e (e) também é imediata a partir dos resultados anteriores e de teoria dos conjuntos.**

**Ou seja, o esforço de prova está na verdade nos itens (a) e (c) acima. A prova das demais afirmações pode ser feita assumindo a verdade das afirmações anteriores usando resultados básicos da Teoria dos Conjuntos de acordo com as sugestões acima.**

1. **Prove que (i) o conjunto Q dos números racionais é contável, e (ii) o conjunto R dos números reais é incontável.**
2. **Explique porque boa parte da teoria da Computabilidade e da teoria da Complexidade é feita com base em problemas de decisão. De exemplos de problemas que são naturalmente problemas de decisão, e problemas cuja versão de interesse na prática não são problemas de decisão. Para estes, apresente a sua versão como problema de decisão.**
3. **Explique porque boa parte da teoria da Computabilidade e da Complexidade é abordada com representações de problemas como linguagens cujos elementos são *strings* de 0’s e 1’s.**
4. **Disserte sobre os seguintes tópicos abaixo. Deixe bem claro o significado a importância de cada um para a Ciência da Computação:**
   1. **Teorema de Cook-Levin.**
   2. **Questão "P = NP?"**
   3. **Dê uma definição de redução de tempo polinomial e explique porque reduções feitas na prova de NP completude de um problema devem ser polinomiais.**
   4. **Dê uma definição da classe de complexidade NP.**
5. **Usando os conceitos (a) e (b) acima dê uma definição de problema NP-Hard.**
6. **Usando os conceitos acima, dê uma definição de problema NP-Completo.**
7. **Seja Double-SAT o conjunto de fórmulas booleanas φ em formato CNF (forma normal conjuntiva) tais que φ é satisfeita por pelo menos duas atribuições de valores verdades às suas variáveis. Prove que o problema Double-SAT é NP-Completo. Nesta prova deixe bem claro que Double-SAT satisfaz todos itens da sua definição de problema NP-Completo dada no item (d) acima. Na sua resposta você pode também usar outros resultados conhecidos sobre a complexidade do problema de satisfatibilidade booleana.**

**Obs.: uma fórmula está no formato CNF se ela é uma conjunção de disjunções de literais. A seguinte fórmula por exemplo, está em formato CNF:**

**(x1 ∨ x2) ∧ (x3 ∨ ¬x1 ∨ ¬x4)∧(¬x2 ∨ x5)**

1. **Considere os seguintes problemas de decisão:**

**IndSet: Dado um grafo G = (V, A) com n nodos e um inteiro positivo k ≤ n. O grafo G possui um subconjunto W de V com pelo menos k nodos tais que nenhum par de nodos de W é conectado por uma aresta em A?**

**3-COL: Dado um grafo G = (V, A), é possível atribuir a cada nodo de G uma cor dentre as cores vermelho, azul e verde, de tal forma que nenhum par de nodos ligados por aresta em A seja da mesma cor?**

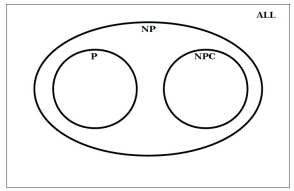
**EulerCycle: Dado um grafo G = (V, A), ele contém ciclo no qual todas as arestas de A são visitadas exatamente uma vez?**

**Sabendo que:**

* **IndSet é NP-Completo (pertence a NP e é NP-Hard).**
* **Indset é redutível em tempo polinomial a 3-COL (3-COL é NP-Hard).**
* **EulerCycle pode ser resolvido em tempo O(|A|), ou seja, polinomial (é P).**

**diga qual o valor verdade (se verdadeiro, falso ou desconhecido) das seguintes afirmações. Justifique suas respostas.**

1. **EulerCycle é redutível em tempo polinomial a IndSet.**
2. **EulerCycle é redutível em tempo polinomial a 3-COL.**
3. **3-COL é redutível em tempo polinomial a EulerCycle.**
4. **Teorema: A classe de funções recursivas parciais é idêntica à classe de funções Turing-computáveis.**
   1. **Defina essas classes de funções e faça um esquema da prova deste teorema, explicando o que deve ser provado em cada passo.**
   2. **Explique por que este teorema é importante.**
5. **Disserte sobre redução de problemas, tanto no contexto de indecibilidade de problemas quanto no contexto da definição de classes de complexidade.**
6. **Disserte sobre os seguintes tópicos abaixo. Deixe bem claro o significado e a importância de cada um para a Ciência da Computação:**
   1. **Tese de Church-Turing.**
   2. **Problema da Parada.**
   3. **Teorema de Rice.**
   4. **Teorema de Cook-Levin.**
   5. **Questão “P = NP”?**
7. **Explique intuitivamente o que diz o Teorema de Cook-Levin e por que ele é relevante para a Ciência da Computação? (Não precisa mostrar definições formais nesta questão.)**
8. **Suponha que um *lower-bound­* exponencial para um problema NPC tenha sido provado (logo P != NP). Nesse caso, o Diagrama de Venn ilustrando as classes de problemas P, NP, NPC, e problemas fora de NP seria o seguinte:**

****

* 1. **Seja A um problema tal que SAT ≤p A. Escreva a letra A nas partes do diagrama onde o problema A pode estar e justifique.**
  2. **Seja B um problema tal que B ≤p SAT. Escreva com a letra B as partes do diagrama onde o problema B pode estar e justifique.**

1. **Suponha que existe um problema X que pertence a NP e X não pertence a P. Prove que baseado nessa afirmação, nenhum problema NPC pode ser resolvido em PTIME.**
2. **Considere os seguintes problemas de decisão:**

**HamCycle: Dado um grafo G = (V, E), ele contém ciclo cujos vértices são visitados exatamente uma vez?**

**HamPath: Dado um grafo G = (V, E e dois vértices distintos s e t, ambos em V, existe caminho simples em G que começa em s, termina em t e cujos vértices são visitados exatamente uma vez?**

**EulerCycle: Dado um grafo G = (V, E), ele contém ciclo cujas arestas são visitadas exatamente uma vez?**

* **HamCycle ∈ NPC**
* **Hamcycle ≤ p HamPath**
* **EulerCycle ∈ P**

**Mostre que HamCycle é redutível em tempo polinomial para HamPath.**

asd

1. **Dada uma prova da não existência de TM para decidir o Problema da Parada, explique o papel da tese de Church-Turing na prova de que o Problema da Parada é indecidível.**
2. **Escreva sobre a tese de Church-Turing. Em sua resposta, você deve:**
   1. **Escrever o enunciado da tese.**
   2. **Explicar porque é uma tese e não um teorema (ou seja, explicar por que ela não pode ser provada).**
   3. **Escrever sobre as evidências que suportam a sua verdade (mencionar duas dessas evidências).**
   4. **Explicar a sua importância para a Ciência da Computação, em particular explicar porque ela é importante em provas de resultados negativos, como a indecibilidade do Problema da Parada.**
3. **Faça um breve ensaio discorrendo sobre os seguintes conceitos relativos à Teoria da Complexidade.**
   1. **Complexidade de tempo e um algoritmo e complexidade de tempo de um problema algorítmico.**
   2. **Complexidade de tempo polinomial e complexidade de tempo exponencial.**
   3. **Problemas tratáveis e intratáveis (com exemplos de problemas tratáveis e intratáveis, e exemplos de problemas com “status” em relação a tratabilidade desconhecido).**
4. **A fim de entender melhor problemas algorítmicos com respeito ao uso de recursos, pesquisadores começaram a agrupá-lo de acordo com as suas semelhanças em relação a requisitos de tempo (e espaço). Este estudo deu origem as classes de complexidade. Faça um breve ensaio sobre as classes**

**P, NP, NPC, EXP**

**Descreva quais são as propriedades determinantes para um problema estar dentro de cada uma dessas classes. Complemente o ensaio com diagramas de Venn que mostram a relação entre essas classes. Discutir sobre a questão P = NP.**

1. **Disserte sobre os conjuntos de problemas NP, NP-Hard e NPC. Desenhe diagramas de Venn mostrando a relação entre estes três conjuntos para o caso de P = NP e P != NP.**
2. **Disserte sobre os seguintes aspectos da Teoria da Computabilidade:**
   1. **Qual é o seu principal foco de estudo.**
   2. **Exemplos dos principais resultados (teoremas, teses, etc).**
   3. **Relevância para a Computação desses resultados principais.**
3. **Disserte sobre os seguintes aspectos da Teoria da Complexidade:**
   1. **Qual é o seu principal foco de escuto.**
   2. **Exemplos dos principais resultados (teoremas, teses, etc).**
   3. **Relevância para a Computação desses resultados principais.**
   4. **Problemas em aberto.**
4. **Imagine que se quer provar que uma propriedade P é válida para a classe de funções computáveis. Faça esboços de como poderiam ser as provas desta propriedade usando TM e usando Funções Recursivas Parciais, justificando a validade de cada prova. Defina critérios e faça uma análise comparativa das duas provas.**
5. **Disserte sobre problemas NPC. Além das definições, explicações e exemplos que você julgar relevantes, sua resposta deve necessariamente explicar 2 formas diferentes de se provar que um dado problema é NPC e também uma justificativa da importância desta classe de problemas. Note que a sua escolha do que é relevante sobre este assunto será considerada na nota desta questão.**
6. **Um professor gostaria de automatizar a correção dos exercícios de programação Python dos alunos, construindo um programa que implemente a função PROG-CORRETO?, definida assim.**

**Entrada: Dois programas escritos em Python, um representando a solução (sol) e outro o programa submetido pelo aluno (sub).**

**Saída: Verdadeiro, se o programa sub implementa a mesma função que o programa sol, ou seja, se para todas as entradas para as quais sol produz um valor de saída, sub produz o mesmo valor, e para todas as entradas para as quais sol não produz saída, sub também não produz saída. A saída deve ser falso caso contrário.**

**Responda às seguintes questões:**

* 1. **Explique o que é uma “função computável”.**
  2. **A função PROG-CORRETO é computável? Prove.**
  3. **Transforme a definição de PROG-CORRETO em PROG-CORRETOmod da seguinte forma:**
     1. **Se sua resposta na questão b) foi SIM, PROG-CORRETOmod deve ser uma função mais geral que PROG-CORRETO que não seja computável.**
     2. **Se sua resposta na questão b) foi NÃO, PROG-CORRETOmod deve ser uma função mais restrita que PROG-CORRETO que seja computável.**

**Justifique por que PROG-CORRETOmod é computável / não computável.**

1. **Disserte sobre as principais classes de complexidade e explique a importância de se demonstrar a classe de complexidade de um problema.**
2. **Diga se as afirmações a seguinte são verdadeiras ou falsas e justifique suas respostas.**
   1. **Se um problema pertence à classe P, sempre existe um algoritmo com complexidade O(2n) que o resolve.**
   2. **Se P = NP então a lógica de primeira ordem seria decidível.**
   3. **Se L é NP-Hard, L é polinomialmente redutível a L’ e L’ pertence a P então P != NP.**
3. **RSA é um algoritmo de criptografia baseado em um par de chaves, uma pública e uma prova. Uma mensagem cifrada usando uma chave pública só pode ser decodificada usando a respectiva chave privada. Neste algoritmo a contrução das chaves se baseia na geração de dois números primos muito grandes e da sua multiplicação, de tal forma que é uma tarefa computacionalmente custosa dada uma chave pública, encontrar os números primos originais, com os quais a mensagem pode ser decodificada. Hoje em dia, grande parte das transações realizadas na internet usam criptografia RSA.**

**Assuma que a chave pública do RSA consiste apenas na multiplicação de dois números primos e a chave privada correspondente é formada pelos dois números primos (na realidade, as chaves são mais complexas que isso no RSA). Neste caso, dada uma chave pública n, o código poderia ser quebrado se fossem encontrados os dois números primos p e q que multiplicados geram a chave pública (ou seja, a fatoração de n em números primos é n = p x q). Considere a seguinte frase de um fórum de criptografia:**

**“Quebrar RSA é NPC, portando não existe maneira melhor que usar força bruta para isso.”**

**Avalie esta frase, levando em consideração as seguintes questões:**

* 1. **O que significa precisamente “quebrar o RSA é NPC”? Ou seja, qual é o problema de decisão envolvido e o que significa dizer que ele é NPC?**
  2. **O problema pertence à classe NP? Justifique.**
  3. **Assumindo que a premissa está correta, ou seja, que “quebrar o RSA é realmente NPC”, a conclusão da frase é uma consequência lógica? Justifique.**
  4. **Fale sobre o impacto da questão P = NP? Para criptografia baseada no RSA.**

1. **O que é modelo de computação? Para que serve um modelo de computação? Dê 2 exemplos de modelo de computação, explicando cada um deles e deixando evidente como “computação” é definida em cada modelo. Qual o significado de dizer “Os modelos de computação A e B são equivalentes”? Os modelos que você citou são equivalentes? Justifique (não precisa fazer uma prova formal, somente explicar o que deve ser feito para provar a equivalência / não equivalência entre esses modelos).**
2. **Um desenvolvedor de *software* precisa construir um programa para solucionar um problema X. Procurando em uma biblioteca, ele encontra um programa eficiente (com ordem de complexidade O(n2)) que soluciona um problema PE que parece ser muito similar ao problema X. Por outro lado, ele encontra um texto que fala sobre um problema PNC, dizendo que ele não é computável, e o desenvolvedor acha que PNC também parece similar a X.**
   1. **Explique o que é uma redução e como definimos reduções entre problemas. Quando dizemos que uma redução é polinomial?**
   2. **Diga se as afirmações a seguir são verdadeiras ou falsas e justifique:**
      1. **O problema PE não é NPC.**
      2. **Se conseguirmos reduzir X para PNC, X também não é computável.**
      3. **Se conseguirmos reduzir X para PE, X pertence à classe P.**
3. **Seja Parada o problema da parada e X um problema que não se sabe se é computável ou não. Assuma ainda que existe uma função δ: Parada → X, mapeando instâncias do problema Parada para instâncias do problema X.**

**Diga se as afirmações a seguir são verdadeiras ou falsas e justifique:**

* 1. **Com base nas informações acima, pode-se afirmar que o problema X não é computável.**
  2. **Se a função δ não for computável, o problema X não é computável.**
  3. **Se pudermos provar que P = NP, então existe uma TM determinística que resolve o problema Parada.**
  4. **Se a Tese de Church não for verdadeira, o problema Parada seria computável**
  5. **Se dois modelos de Computação são equivalentes, eles resolvem a mesma classe de problemas.**

1. **Responda as questões a seguir:**
   1. **Defina as principais classes de complexidade (min 4).**
   2. **Dê um exemplo de problema NPC, definindo claramente o problema e dando exemplos de instâncias do problema com respostas sim e não.**
   3. **O problema apresentado no item anterior pertence à classe NP? Justifique (prove).**
   4. **Defina e explique o que é uma redução polinomial e como ela pode ser usada para provar que um problema X é NPC.**