

Universidade Federal de Minas Gerais
Programa de Pós-Graduação em Ciência da Computação
Exame de Qualificação 1º Estágio
2º Semestre de 2018

Em 17/08/2018, **09:00 horas**.

Prova individual sem consulta com duração de **3 horas**.

Observações:

1. A prova deve ser resolvida no próprio caderno de questões.
2. As questões desta prova estão nas páginas seguintes, as quais estão numeradas de 1 a 6.
3. Faz parte da prova a interpretação das questões. Caso você ache que falta algum detalhe nos enunciados ou nos esclarecimentos, você deverá fazer as suposições que achar necessárias e escrever essas suposições juntamente com as respostas.
4. **Todas** as respostas devem ser justificadas.
5. Somente serão corrigidas respostas legíveis.
6. Não se esqueça de escrever seu **nome completo em todas as páginas**.

Desejamos a você uma boa prova!

A COPEQ

Atenção: Esta prova contém um total de 6 (seis) questões, das quais você deve fazer 4 (quatro). Marque abaixo as questões que devem ser consideradas para avaliação:

1 2 3 4 5 6 (selecione até quatro)

Nome completo: _____

Assinatura: _____

Nome completo:

Questão 1

Temos um caixa eletrônico com infinitas notas de valores p_1, p_2, \dots, p_k reais. Dado um inteiro positivo n pretende-se determinar um conjunto de notas da menor cardinalidade possível cujo valor seja n reais ou determinar que tal conjunto não existe.

- (a) Modele o problema como um problema em grafos.
- (b) Mostre o grafo que resulta de ter notas de 3, 5 e 8 reais para saques de até 15 reais.
- (c) Qual algoritmo usaria para resolver o problema? Qual é a complexidade de dito algoritmo em função de k e n .

Solução:

1. *Estados representando a quantia de dinheiro. Cada estado possível é representado por um vértice no grafo. Cada arco representa a possível transição de um estado a outro entregando uma nota. O problema consiste em determinar o caminho mínimo do vértice representando o estado "0" até o vértice representando o estado "n".*
2. *vértices de 0 a 15 com arcos saindo desde cada vértice i ao vértice $i+3$, $i+5$ e $i+8$.*
3. *Busca em largura a partir de 0.*



Nome completo:

Questão 2

Dado um grafo não orientado e não ponderado:

- (a) Desenvolva um algoritmo polinomial para computar um emparelhamento maximal do grafo. Um emparelhamento maximal é um emparelhamento no qual nenhuma aresta pode ser acrescentada sem que o conjunto de arestas deixe de ser um emparelhamento.
- (b) Usando o algoritmo desenvolvido em (a) desenvolva um algoritmo 2-aproximativo para o problema de cobertura por vértices. Mostre que o conjunto de vértices obtido cobre todas as arestas e que contém não mais do que o dobro do número de vértices de uma cobertura mínima.

Solução:

1. *Percorra as arestas do grafo selecionando aquelas que não compartilhem nenhum vértice incidente com outra aresta selecionada.*
2. *Para cada aresta selecionada selecione os dois vértices nas quais a aresta incide. O tamanho da cobertura será de duas vezes o tamanho do emparelhamento maximal. Sabe-se que o tamanho de qualquer emparelhamento é um limite inferior para o tamanho de qualquer cobertura por vértices (pelo menos um dos vértices incidentes a cada aresta tem que estar na cobertura). Portanto o tamanho da cobertura obtida não é maior que duas vezes o tamanho da cobertura mínima.*

Nome completo:

Questão 3

Escreva um algoritmo que encontra os k maiores elementos de um vetor v com n elementos, onde $k \ll v$. A complexidade assintótica no pior caso de seu algoritmo deve ser $O(n + k \log n)$. Uma implementação cuja complexidade é maior que isso, e.g. $O(n \log n)$, vale zero pontos. Você pode usar qualquer estratégia que quiser, mas segue uma sugestão usando *heaps*.

- (a) Descreva o pseudocódigo de uma função $Max\text{-}Heapfy(A, i)$ que recebe um vetor A e um índice i . Essa função assume que $LEFT(i)$ e $RIGHT(i)$ são *heaps* de máximo, mas que $A[i]$ pode ser menor que seus filhos. Esta função deve transformar $A[i]$ em uma *heap* em $O(\log n)$.
- (b) Assuma que a questão do item (a) está implementada corretamente. Descreva o pseudocódigo de uma função $MAX\text{-}HEAPIFY(A)$ que converte o vetor $A[1..n]$, em uma *heap* de máximo. Discuta informalmente porque a complexidade no pior caso dessa função é $O(n)$.
- (c) Assuma que a questão do item (b) está implementada corretamente. Descreva o pseudocódigo de uma função $HEAP\text{-}EXTRACT\text{-}MAX(A)$ que recebe uma *heap* de máximo A e retira o maior elemento desta *heap* em $O(\log n)$. Utilize esta função para resolver a questão do enunciado.

Solução:

- (a) Gabarito na seção 6.2 da terceira edição de Cormen et al.
- (b) Gabarito na seção 6.3 da terceira edição de Cormen et al.
- (c) Gabarito na seção 6.5 da terceira edição de Cormen et al.



Nome completo:

Questão 4

- (a) Descreva o pseudocódigo de uma função **recursiva** que transforma uma árvore binária de busca balanceada A em uma lista duplamente encadeada ordenada em $\Theta(n)$, onde n é o número de nós em A . Uma implementação cuja complexidade é maior que $\Theta(n)$ vale zero pontos.
- (b) Enuncie o caso do Teorema Mestre necessário para determinar a complexidade computacional da função proposta. O Teorema Mestre também é conhecido como “método mestre para resolver recorrências”.
- (c) Descreva a equação de recorrência de sua função e mostre que a complexidade dela é $\Theta(n)$.

Solução:

- (a) $Listfy(A) \{ \text{if } \{|A|=0\} \text{ then } \{\text{return } A\} \text{ else } \{\text{return } \text{append}(Listfy(Left(A)), Root(A), Listfy(Right(A)))\} \}$
- (b) Teorema 4.1 da seção 4.5 da terceira edição de Cormen et al.
- (c) $T(n) = 2T(n/2) + \Theta(1) = \Theta(n)$, pelo caso 1 do T.M.

Nome completo:

Questão 5

Uma clique em um grafo $G = (V, E)$ é um conjunto de vértices $S \subseteq V$ tal que cada dois vértices em S são adjacentes. Dizemos que uma clique é par se seu número de elementos é par. Considere o seguinte problema: dado um grafo G e um inteiro k , determinar se existe uma clique par de tamanho pelo menos k .

- (a) Mostre que este problema está em NP.
- (b) Mostre que este problema é NP-difícil.
- (c) Este problema é NP-completo?

Solução:

1. *Um certificado natural para este problema é fornecer a lista de vértices que compõem a clique. Um algoritmo verificador precisaria conferir se a lista de vértices de fato tem um tamanho par maior ou igual a k e se todos os vértices da lista são adjacentes entre si, o que pode ser feito facilmente em tempo quadrático.*
2. *Basta fazer uma redução de clique. Se na instância de clique k já for par, podemos copiar a instância. Caso k seja ímpar, basta incrementar o valor de k e adicionar um vértice universal.*
3. *Basta afirmar que um problema é NP-completo se é NP-difícil e está em NP.*

Nome completo:

Questão 6

Considere o procedimento `calcula(n)` abaixo, que recebe um inteiro $n > 0$ como parâmetro.

```
calcula (n)
{
    soma = 0;
    para i = 1 até n
    {
        para j = i até n
        {
            x = f(j);
            k = 1;
            enquanto k * k <= i
            {
                soma = soma + g(x);
                k = k + 1;
            }
        }
    }
}
```

- (a) Determine o número de chamadas feitas à função $f(j)$, em função de n .
- (b) Assumindo que as funções $f(\cdot)$ e $g(\cdot)$ podem ser computadas em tempo $O(1)$, mostre que o algoritmo possui complexidade de tempo $O(n^3)$.
- (c) Este algoritmo possui complexidade de tempo $\Theta(n^3)$? Justifique.

Solução:

1. Basta escrever o somatório de 1 a n . Não é necessário resolver o somatório, fornecendo uma fórmula fechada.
2. Basta observar que cada um dos três laços são executados uma quantidade de vezes menor ou igual a n .
3. Não. O laço mais interno executa um número de passos proporcional a \sqrt{i} . Desta forma, a complexidade está limitada superiormente por $n^{2.5}$, que é assintoticamente inferior a n^3 . Formalmente, basta mostrar que não existe constante tal c que $n^{2.5} \geq cn^3$ e portanto $n^{2.5}$ não é $\Omega(n^3)$.