# Universidade Federal de Minas Gerais
# Programa de Pós-Graduação em Ciência da Computação
# Exame de Qualificação 1º Estágio
# 1º Semestre de 2021

Em 28/05/2021, **14:00 horas**.

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

**Observações**:

1. **Atenção**: Esta prova contém um total de 6 (seis) questões, das quais você deve fazer 4 (quatro).
2. Você deve indicar na primeira página da sua solução quais as questões foram selecionadas. Todas as respostas devem indicar claramente a que questões elas se referem.
3. As questões desta prova estão nas páginas seguintes, as quais estão numeradas de 1 a 7.
4. 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.
5. **Todas** as respostas devem ser justificadas.
6. Somente serão corrigidas respostas legíveis.
7. Não se esqueça de escrever seu **nome e número de matrícula em todas as páginas**.
8. A forma de comunicação oficial durante a prova é através da plataforma Microsoft Teams.
9. Ao final da prova, sua solução escaneada deve ser enviada através da plataforma em até 30 minutos após o término da prova, isto é, até **17:30**. Este prazo não será prorrogado e o sistema recusará soluções automaticamente após esse horário.

Desejamos a você uma boa prova!

A COPEQ

---

# Questão 1

Preencha a tabela abaixo com V (verdadeiro) ou F (falso) assumindo que $c$ e $k$ são constantes positivas maiores do que 1. Para cada linha da tabela forneça uma justificativa baseada na definição de $O, \Omega$ e $\Theta$.

| | $f(n) = O(g(n))$ | $f(n) = \Omega(g(n))$ | $f = \Theta(g(n))$ |
|---|---|---|---|
| $f(n) = n^{2c}, g(n) = 2c^n$ | | | |
| $f(n) = n^2 \log_3 n, g(n) = n \log_2 n$ | | | |
| $f(n) = c^n, g(n) = k^n$ | | | |
| $f(n) = \log_3 n + \log_2 (n^{\log_2 c}), g(n) = \log_2(k^{\log_4(n)})$ | | | |

*Solução:*

| | $f(n) = O(g(n))$ | $f(n) = \Omega(g(n))$ | $f = \Theta(g(n))$ |
|---|---|---|---|
| $f(n) = n^{2c}, g(n) = 2c^n$ | $V$ | $F$ | $F$ |
| $f(n) = n^2 \log_3 n, g(n) = n \log_2 n$ | $F$ | $V$ | $F$ |
| $f(n) = c^n, g(n) = k^n$ *(assumindo $c > k$)* | $F$ | $V$ | $F$ |
| $f(n) = \log_3 n + \log_2 (n^{\log_2 c}), g(n) = \log_2(k^{\log_4(n)})$ | $V$ | $V$ | $V$ |

# Questão 2

Considere um vetor de $n \geq 2$ posições, cada uma preenchida com um dígito no intervalo $0 - 9$. Queremos decompor este vetor em subvetores, ou seja, em trechos consecutivos. Além disso, todo subvetor deve ser par, isto é, o número formado pelos dígitos do subvetor deve ser par.

Por exemplo, caso o vetor de entrada seja 805438452, este pode ser particionado como "8054, 38, 452", com 3 subvetores, ou como "8, 0, 54384, 52", com 4 subvetores, dentre outras possibilidades. Escreva um algoritmo que determine o número de maneiras diferentes em que um em que o vetor pode ser decomposto. Por exemplo, caso o vetor de entrada seja 1234, o vetor pode ser decomposto de apenas duas maneiras: "12, 34" e "1234". Determine a complexidade de seu algoritmo. Seu algoritmo deve ter uma complexidade tão baixa quanto possível. Você pode assumir que o último dígito do vetor é par.

*Solução:*

*Basta notar que a solução é sempre da forma* $2^{k-1}$*, onde* $k$ *é o número de dígitos pares do vetor. Isso segue do fato de que todo número par termina com um dígito par e, para cada dígito par (com exceção do último) podemos decidir se ali termina ou não um dos subvetores.*

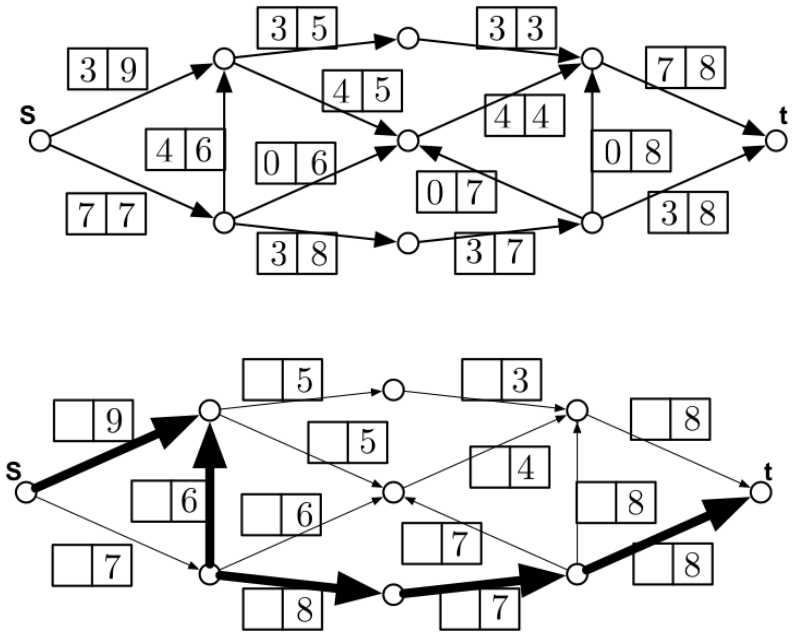
*Contar o número* $k$ *de dígitos pares do vetor pode ser feito em tempo linear, portanto a solução é linear.*

# Questão 3

O método de Ford-Fulkerson é utilizado para encontrar um fluxo máximo de um vértice $s$ para um vértice $t$ em um grafo dirigido, com capacidades nos arcos. Ele consiste basicamente em encontrar um caminho de $s$ para $t$ e realizar algum procedimento que aumenta, se possível, o fluxo total que flui de $s$ para $t$.

No grafo abaixo, o par de números em cada arco representa | `fluxo atual` | `capacidade` | de uma instância do método. O algoritmo de busca utilizado encontrou então o caminho em negrito no segundo grafo.

(a) Escreva em **todos os espaços em branco** os fluxos obtidos após uma iteração do método de Ford-Fulkerson que usa o caminho em negrito, baseado no estado abaixo.

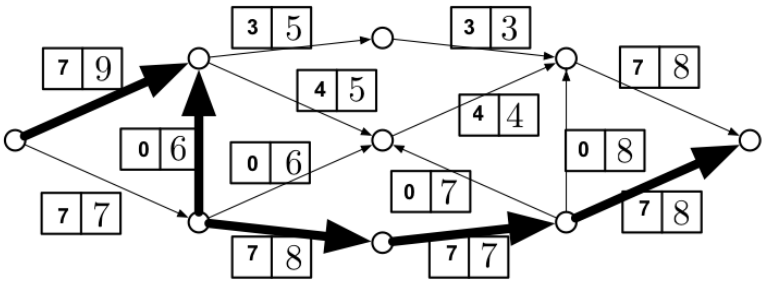



(b) Qual o valor máximo de um $st$-fluxo neste grafo?

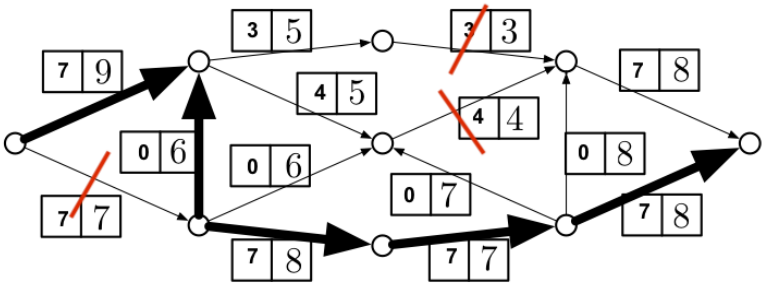
Um conjunto de arcos cuja remoção elimina todos os caminhos dirigidos de $s$ para $t$ é chamado de um corte. Após o término do método, é possível identificar um corte cuja soma das capacidades dos arcos no corte é igual ao fluxo máximo encontrado.

(c) Marque no primeiro grafo os arcos de um corte com mesmo valor do fluxo máximo.

*Solução:*




*Fluxo máximo tem valor 14 (o encontrado já é máximo). O corte está abaixo*



# Questão 4

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 $\Omega(n^3)$.*

# Questão 5

Você possui um grafo conectado, ponderado e não-direcionado $G(V, E, w)$ sem ciclos de peso negativo. O diâmetro do grafo é definido como sendo o caminho mínimo de maior peso no grafo, i.e. para cada par de vértices $u, v$ existe algum caminho de peso $\delta(u, v)$, e o diâmetro é definido como sendo $max_{(u,v)}\{\delta(u, v)\}$. Faça um algoritmo de tempo polinomial (em relação a $|E|$ e/ou $|V|$ ) para achar o diâmetro de $G$. Qual é seu tempo de execução?

*Solução:*

*Execute Bellman-Ford $|V|$ vezes, uma para cada nó. Isso encontrará todos os $|V|^2$ caminhos mínimos. Então obtenha o valor máximo dos caminhos mínimos. O tempo de execução de executar Bellman-Ford $|V|$ vezes é $O(|V|^2|E|)$.*

# Questão 6

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.*