# Trabalho Prático 03 - Algoritmos I

Data de Entrega: 02/12/2025

## 1 Introdução

Com o Natal se aproximando, o Papai Noel iniciou os preparativos finais para garantir que cada criança ao redor do mundo receba seu presente a tempo. Entre renas, trenós e listas de desejos, uma das tarefas mais importantes é montar a equipe de duendes responsáveis por organizar, embalar e auxiliar nas entregas.

O problema é que os duendes são criaturas talentosas, mas também são conhecidos por seus temperamentos fortes, e frequentemente entram em conflito uns com os outros. Alguns pares de duendes simplesmente não conseguem trabalhar lado a lado sem transformar o Polo Norte em um verdadeiro campo de guerra com bolas de neve e pisca-pisca voando para todos os lados. Colocar dois duendes brigões na mesma equipe pode atrasar toda a operação natalina.

Como há muitos duendes e várias rivalidades antigas entre eles, o bom velhinho percebeu que não conseguia resolver essa questão sozinho. Por isso, ele pediu a sua ajuda para selecionar o maior grupo possível de duendes que consiga trabalhar em harmonia, sem que nenhum par dentro da equipe seja formado por duendes que brigam entre si.

Agora cabe a você ajudar o Papai Noel a montar a equipe perfeita, garantindo que todos colaborem em paz e que os presentes cheguem a tempo para iluminar o Natal de pessoas no mundo inteiro. Ho, ho, ho!

## 2 Descrição do Problema

Cada duende é representado por um índice de 1 a $N$, e algumas duplas de duendes brigam entre si. Seu objetivo é ajudar o Papai Noel a montar a maior equipe possível sem conflitos, determinando quantos duendes farão parte da equipe e quais são eles.

A saída consiste em uma linha contendo o número total de duendes selecionados, seguida por uma segunda linha com os índices dos duendes escolhidos, apresentados em ordem crescente e de menor ordem lexicográfica.

## 3 Solução

### 3.1 Modelagem

Este trabalho prático aborda a parte de programação dinâmica da ementa desta disciplina. Por esse motivo, sua solução deve utilizar um algoritmo de programação dinâmica e ser ótimo. Suas escolhas de implementação devem ser descritas na documentação do seu trabalho.

### 3.2 Entrada e Saída

Nesta sessão está especificado o que será enviado como entrada e o que é esperado como saída da sua solução. Para cada problema, o programa deve imprimir a resposta no formato especificado na seção anterior. Abaixo segue o formato generalizado:

**Entrada:**

```
N M
A1 B1
A2 B2
A3 B3
...
XM YM
```

**Saída:**

```
X
C1 C2 C3 C4 ... CX
```

Temos como **entrada**:

- 1ª linha: dois números inteiros positivos $N$, $1 \leq N \leq 40$, e $M$, $0 \leq M \leq N * (N-1)/2$, onde $N$ indica o número de duendes em uma região do Polo Norte e $M$ indica quantos pares de duendes brigam entre si.
- Próximas $M$ linhas: cada linha contém dois números inteiros positivos $A_i$ e $B_i$, $0 \leq A_i, B_i \leq N-1$, onde $A_i$ e $B_i$ representam os índices das duendes que brigam entre si.

Como **saída**, temos:

1. **X**
   - A primeira linha da saída contém um inteiro positivo $X$, indicanco o número de duendes escolhidos para a equipe do Papai Noel.
2. **C1 C2 C3 C4 ... CX**
   - A segunda linha contém $X$ inteiros positivos $C_i$, $0 \leq C_i \leq N-1$, separados por espaço. Cada $C_i$ indica o índice de um duende selecionado para a equipe. Sua resposta deve estar em ordem crescente e ser lexicograficamente mínima.

     Um vetor A é lexicograficamente menor que um vetor B se no primeiro índice onde os dois se diferenciam, o vetor A tem um elemento menor que o de B. Ou seja, se dois vetores, por exemplo, "1 2 3" e "1 2 4", resultarem em equipes de duendes válidas, deve-se imprimir "1 2 3".

## 3.3 Exemplos

### 3.3.1 Exemplo 01

**Entrada:**

```
8 10
0 1
2 3
4 5
6 7
0 2
2 4
4 6
1 3
3 5
5 7
```

**Saída:**

```
4
0 3 4 7
```

#### 3.3.2 Exemplo 02

**Entrada:**

```
7 9
0 1
1 2
2 0
2 3
3 4
4 2
4 5
5 6
6 4
```

**Saída:**

```
3
0 3 5
```

### 3.4 Dica

Note que o tamanho máximo do conjunto de duendes é 40. O problema ficaria mais fácil se houvesse até 20 duendes? Como podemos usar isso para achar a solução para 40 duendes?

## 4 Implementação

### 4.1 Linguagem

O trabalho prático deverá ser implementado em `C`, `C++` ou `Python` e utilizar apenas as bibliotecas padrão das respectivas linguagens. Não será permitido o uso de bibliotecas exclusivas de um sistema operacional ou compilador.

#### 4.1.1 C

No caso de `C`, você deve usar apenas bibliotecas do padrão ANSI C, das versões `C99`, `C11` ou `C17`.

#### 4.1.2 C++

No caso de `C++`, utilize bibliotecas do padrão `ISO/IEC C++`, das versões `C++11`, `C++14`, `C++17` ou `C++20`.
Poderão ser utilizadas bibliotecas que implementam algumas funcionalidades mais básicas, como:

- Algumas estruturas de dados simples: `<vector>`, `<set>`, `<list>`, etc.
- Entrada e saída: `<iostream>`, `<fstream>`, etc.
- Manipulação de strings: `<string>`, `<sstream>`, etc.
- Algumas utilidades simples: `<utility>`, `<compare>`, etc.

Não poderão ser utilizadas bibliotecas que realizam funcionalidades de mais alto nível, como:

- Algoritmos mais complexos: `<algorithm>`, `<functional>`, `<ranges>`, etc.
- Gerenciamento automático de memória: `<memory>`, `<memory_resource>`, etc.

Em caso de dúvidas, pergunte aos monitores.

#### 4.1.3 Python

No caso de Python, serão aceitas versões feitas em Python 3.9+, com acesso a todas as bibliotecas padrão da linguagem. Pacotes adicionais não serão permitidos. Em caso de erro do código devido a importação de bibliotecas proibidas, ele NÃO SERÁ AVALIADO.

## 4.2 Ambiente

O aluno pode implementar em qualquer ambiente de programação que desejar, mas deve garantir que o programa seja compilável nas máquinas do DCC com sistema operacional **Linux**, que são acessíveis aos alunos e disponibilizadas pelo Centro de Recursos Computacionais (para mais informações, acesse o site: https://www.crc.dcc.ufmg.br/).

## 4.3 Código

Utilize boas práticas de programação que você aprendeu em outras disciplinas, para que seu código seja legível e possa ser interpretado corretamente pelo leitor. A qualidade do seu código será avaliada. Algumas dicas úteis:

- Seja consistente nas suas escolhas de indentação, formatação, nomes de variáveis, funções, estruturas, classes e outros.
- Escolha nomes descritivos e evite nomear variáveis como `aux`, `tmp` e similares.
- Comente o seu código de forma breve e objetiva para descrever funções, procedimentos e estruturas de dados, mas evite comentários muito longos e de várias linhas.
- Para c,c++, separe seu código em diferentes arquivos, como `.c` e `.h` para C ou `.cpp` e `.hpp` para C++, de forma que facilite navegar pelo seu código e compreender o fluxo de execução.
- Para o Python, não é necessária a utilização de arquivos de headers.
- Evite funções muito grandes ou muito pequenas, que fazem várias coisas diferentes ao mesmo tempo, ou que tenham ou retornem muitos parâmetros diferentes.
- Seu código deve ser legível e devidamente comentado.

## 4.4 Compilação

### 4.4.1 C,C++

Ao compilar o programa, você deverá utilizar no mínimo as seguintes flags:

```
-Wall -Wextra -Wpedantic -Wformat-security -Wconversion -Werror
```

Se seu programa apresentar erros de compilação, seu código não será corrigido.

### 4.4.2 Python

Para Python, por ser uma linguagem interpretada, o código não precisa ser compilado. Ainda assim, ele não deve apresentar erros em sua execução

## 4.5 Parâmetros

O aluno deve ler o arquivo de entrada do programa pela entrada padrão através de linha de comando, como por exemplo:

```
./tp3.out < testCase01.txt
```

E gerar o resultado na saída padrão, não por arquivo.

# 5 Documentação

O aluno deverá fornecer uma documentação do trabalho contendo as seguintes informações:

1. **Introdução:** Uma breve explicação, em suas palavras, sobre qual é o problema computacional a ser resolvido.
2. **Modelagem:** Como você modelou o problema, traduzindo a situação fictícia em uma estrutura de dados, e quais algoritmos foram utilizados.
3. **Solução:** Como os algoritmos que você implementou resolvem o problema proposto e qual a ideia geral de cada um deles. A explicação deve ser de alto nível e/ou utilizar pseudocódigo, sem trechos diretos do código fonte.
4. **Análise de Complexidade:** Para cada um dos três problemas deve haver uma analíse da complexidade assíntotica demonstrada e explicada de tempo e memória da solução escolhida.
5. **Considerações Finais:** Descreva sua experiência em realizar este trabalho prático, quais partes foram mais fáceis, quais foram mais difíceis e por quê.
6. **Referências:** Liste aqui as referências que você utilizou, considerando aquilo que foi relevante para a resolução deste trabalho prático.

A documentação deverá ser **sucinta e direta**, explicando com clareza o processo, contendo não mais do que 5 páginas.

# 6 Entrega

A entrega deverá ser feita através do Moodle, sendo um arquivo `.zip` ou `.tar.gz` no formato `MATRICULA_NOME`, contendo os seguintes itens:

- A documentação em um arquivo `.pdf`;
- Um arquivo `Makefile` que crie um executável com o nome `tp3.out` (desnecessário para resoluções em Python);
- Todos os arquivos de código fonte.
- No caso de Python, seu arquivo de código principal deve se chamar `tp3.py`

**Atenção:** caso seu trabalho não siga as instruções dadas nessa seção, **você será penalizado.**

# 7 Correção

Seu trabalho prático será corrigido de forma automática, e portanto, você deverá garantir que, ao rodar o comando `make` na pasta contendo os arquivos extraídos, seja gerado um binário executável com o nome `tp3.out` na raiz, para que seu código seja avaliado corretamente.

Serão avaliados casos de teste básicos, bem como casos mais complexos e específicos, que testarão não somente a corretude, mas também a performance da sua solução. Para que seu código seja avaliado, você deverá garantir que seu programa dê a resposta correta e ótima, conforme pedido na descrição dos problemas. Esses casos serão disponibilizados no Moodle para que você possa testar seu programa.

Você deve garantir que seu programa não apresente erros de execução ou vazamentos de memória. Caso seu programa apresente erros de execução em algum caso de teste, sua nota será zerada para o caso específico. Vazamentos de memória serão penalizados.

Em caso de suspeita de plágio, seu trabalho será zerado e os professores serão informados. É importante fazer o trabalho por conta própria e não simplesmente copiar a solução inteira de outra pessoa. Caso você tenha suas próprias ideias inspiradas em outras, deixe isso claro na seção de referências.

A entrega do código-fonte e da documentação é **obrigatória**. Na ausência de algum desses, seu trabalho não será corrigido, e portanto, será zerado.

# 8 Avaliação

A nota final (NF) do trabalho prático será composta por três fatores: corretude (CR) e clareza (CL) do código, e documentação (DC). A corretude dirá respeito ao código do aluno passar nos casos testes de correção, a clareza dirá respeito à facilidade de compreensão do código escrito, tal como nomes de variáveis e comentários, e a documentação dirá respeito à qualidade do texto e atenção ao formato proposto na seção 5 acima.

## 8.1 Nota Final do Trabalho Prático (NF)

A nota final do trabalho prático será obtida pela equação:

$$NF = 0{,}4 \times CR + 0{,}2 \times CL + 0{,}4 \times DC$$

## 8.2 Atraso

Devido à proximidade do fim do semestre, esse trabalho **não aceitará submissões atrasadas.** Se programe para entregar no prazo estipulado.

# 9 Considerações Finais

Leia atentamente a especificação e comece o trabalho prático o quanto antes. A interpretação do problema faz parte da modelagem. Em caso de dúvidas, procure os monitores, eles estão a disposição para solucionar dúvidas e ajuda-los.

Busque primeiro usar o fórum de dúvidas no Moodle, pois a dúvida de um geralmente é a dúvida de muitos.