Trabalho Prático II
Algoritmos I

Data de Entrega: **19 de Dezembro de 2024**

1. Introdução

A Metalmax é uma grande siderúrgica especializada na produção de aço. Para alimentar seus processos industriais, como fornos, laminadores e sistemas auxiliares, a empresa utiliza uma rede elétrica interna complexa. Essa rede é composta por geradores, consumidores e conexões entre eles. Recentemente, a Metalmax enfrentou dificuldades no fornecimento de energia, o que tem impactado sua produção e elevado os custos operacionais.
Na rede elétrica da Metalmax, os geradores produzem energia sem restrições de capacidade, enquanto os equipamentos consumidores possuem demandas específicas que precisam ser atendidas. Além disso, as conexões têm capacidade máxima limitada para transmitir energia. Os engenheiros da Metalmax precisam avaliar o funcionamento da rede para identificar gargalos e possíveis falhas no atendimento energético. Esse diagnóstico é essencial para garantir que a produção de aço não seja prejudicada.
O seu trabalho é elaborar um algoritmo para analisar a rede elétrica da Metalmax, relatando um diagnóstico preciso do seu funcionamento identificando situações críticas e pontos de melhoria.

2. Diagnóstico da Rede Elétrica

A rede elétrica possui as seguintes propriedades fundamentais:

- **Geradores**: Pontos de fornecimento de energia, a partir dos quais a energia sai e passa pela rede elétrica.

- **Consumidores**: Pontos de consumo de energia, a partir dos quais a energia que é transmitida pela rede que chega neste ponto é consumida dada uma certa demanda específica.
- **Conexões**: Os caminhos na rede pelos quais a energia flui, indo sempre em uma única direção partindo dos geradores para os consumidores ou de um consumidor para outro consumidor. Cada conexão possui uma capacidade máxima.

Dado essas características, seu diagnóstico deve abordar as seguintes questões:

2.1 Energia Total

Apesar dos geradores sempre conseguirem fornecer energia sem nenhuma restrição, as conexões possuem seus limites. Sua primeira análise portanto, seria para identificar qual o limite que uma determinada rede consegue operar, ou seja:

Qual a capacidade máxima que essa rede comporta?

Ou seja, considerando todos os geradores e todas as conexões, qual o máximo que essa rede consegue transmitir? A empresa tem como requerimento mínimo que todos os consumidores sejam atendidos.

2.2 Energia Não-atendida

Pode ser o caso de que, mesmo com os geradores fornecendo sempre a capacidade máxima para a rede, a energia que chega nos consumidores não está sendo suficiente para atender a demanda. Ou seja, os consumidores não operam em suas melhores condições por falta de energia. Assim, temos o segundo diagnóstico requerido:

Quanta energia falta para os consumidores operarem efetivamente?

Considere qual a demanda total dos consumidores e a situação atual da rede, ou seja, o quanto de energia de fato está chegando nos mesmos.

2.3 Energia Perdida

A partir de cada gerador, sai uma determinada quantidade de energia que vai para a rede. Entretanto, devido as limitações das conexões, parte da energia pode acabar se perdendo ao longo do caminho. Assim, podemos dizer que a rede acaba desperdiçando ao longo do trajeto até os consumidores. Com isso, temos a parte final do diagnóstico:

Quanta energia é desperdiçada ao longo da rede?

Em outras palavras, quanto de energia sai dos geradores que não chega nos consumidores? Seu diagnóstico deve apontar o total de energia perdida ao longo da rede.

2.4 Conexão Crítica

É importante saber qual é a conexão na rede que consome mais energia, e que portanto demandaria uma atenção maior tanto em quesito de importância como em estabilidade da rede elétrica como um todo. Assim, sua última análise seria:

Qual as conexões que demandam mais energia na rede?

Considere dentre todos os geradores e todos os consumidores, qual seriam as conexões da rede que estão operando em sua capacidade máxima e poderiam ser potenciais pontos de falha.

3. Solução

3.1 Modelagem

A solução do problema deverá utilizar grafos como estrutura de dados principal, utilizando algoritmos que operam nessa modelagem.

3.2 Entrada & Saída

A entrada para seu programa será no formato descrito nos exemplos a seguir. Considere como uma descrição da rede elétrica, apontando quem são os geradores, os consumidores e as conexões que levam energia dos primeiros para os últimos.

3.2.1 Exemplo

Considere a entrada do exemplo a seguir:

```
6 7
1 0
2 20
3 30
4 40
5 50
6 50
1 2 50
1 3 60
```

```
1 4 70
1 5 80
1 6 30
2 4 10
5 6 15
```

A primeira linha contém dois números inteiros V e E, onde V é o número de pontos na rede e E o número de conexões.
Nas próximas V linhas, temos 2 números V_i e T, onde V_i é o identificador do ponto na rede e T indicando qual o tipo. Se for o caso onde $T = 0$, então é um gerador. Mas se $T > 0$, então é um consumidor e o número indica sua demanda.
Em seguida, temos E linhas contendo 3 números V_i, V_j e C, onde V_i e V_j são dois identificadores de pontos na rede representando que entre eles existe uma conexão indo de V_i até V_j que possui capacidade máxima $C > 0$.
Para essa entrada temos a saída:

```
185
5
105
2
1 6 30
5 6 15
```

Temos 3 linhas, cada uma contendo um número referente aos diagnósticos descritos anteriormente.
A primeira, temos $E_{Total} > 0$ que é a energia total que a rede comporta.
A segunda, temos $E_{Missing} \geq 0$ que é a energia não-atendida.
A terceira, temos $E_{Loss} \geq 0$ indicando a energia perdida ao longo da rede.
Na quarta linha, temos um número $P_{Critical}$ indicando quantas conexões críticas temos na rede. Nas próximas $P_{Critical}$ linhas, temos 3 números V_i, V_j e E_{ij} indicando uma conexão que vai de V_i para V_j e está operando em sua capacidade máxima E_{ij}.

4. Implementação

4.1 Linguagem

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

4.1.1 C

No caso de `C`, você deve usar bibliotecas apenas 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 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.

4.2 Ambiente

O aluno pode implementar em qualquer ambiente de programação que desejar, mas deve garantir que o programa seja compilável nas máquinas do DCC, que são acessíveis aos alunos 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 na suas escolhas de indentação, formatação, nomes de variáveis, funções, estruturas, classes e outros, etc.

- 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.
- 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.
- Evite funções muito grandes ou muito pequenas, que fazem várias coisas diferentes ao mesmo tempo, ou que tenham ou retornem muitos parâmetros diferentes.

4.4 Compilação

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.

4.5 Parâmetros

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

```
./tp2 < testCase01.txt
```

e gerar o resultado na 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 qual 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 mais alto nível e utilizar pseudocódigo, sem trechos diretos do código fonte.

4. **Análise de Complexidade**: Uma análise assintótica de tempo e memória da sua solução, devidamente demonstrada e justificada.

5. **Considerações Finais**: Descreva sua experiência em realizar este trabalho prático, quais partes foram mais fáceis, quais foram mais difíceis e porquê.

6. **Referências**: Coloque 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 o seguinte:

- A documentação em um arquivo `.pdf`.
- Um arquivo `Makefile` que crie um executável com o nome `tp2`.
- Todos os arquivos de código fonte.

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 `tp2` para que seu código seja avaliado corretamente.
Serão avaliados casos de teste básicos como também casos mais complexos e específicos que testarão não somente a corretude mas também performance da sua solução. Para que seu código seja avaliado, você deverá garantir que seu programa dê a resposta correta pelo menos nos casos de teste mais básicos. Os mesmos 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 informados. É importante fazer o trabalho por conta própria e não simplesmente copie a solução inteira de outra pessoa. Caso você tenha suas próprias ideias inspiradas em outras, deixe 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 zerado.

8. Avaliação

A nota final (NF) do trabalho prático será composta por dois fatores: o Fator Parcial de Implementação (FPI) e o Fator Parcial de Documentação (FPD).

8.1 Fator Parcial de Implementação (FPI)

A implementação será avaliada com base nos seguintes aspectos:

Aspecto	**Sigla**	**Valores Possíveis**
Compilação no ambiente de correção	co	0 ou 1
Respostas corretas nos casos de teste	ec	0 a 100%
Tempo de execução abaixo do limite	te	0 ou 1
Qualidade do código	qc	0 a 100%

Tabela 1: Aspectos de avaliação da implementação.

A fórmula para cálculo do FPI é:

$$FPI = co \times (ec - 0,15 \times (1 - qc) - 0,15 \times (1 - te))$$

Caso o valor calculado do FPI seja menor que zero, ele será considerado igual a zero.

8.2 Fator Parcial da Documentação (FPD)

A documentação será avaliada com base nos seguintes aspectos:

Aspecto	**Sigla**	**Valores Possíveis**
Apresentação (formato, clareza, objetividade)	ap	0 a 100%
Modelagem computacional	mc	0 a 100%
Descrição da solução	ds	0 a 100%
Análise de complexidade	ac	0 a 100%

Tabela 2: Aspectos de avaliação da documentação.

A fórmula para cálculo do FPD é:

$$FPD = 0,4 \times mc + 0,4 \times ds + 0,2 \times ac - 0,25 \times (1 - ap)$$

Caso o valor calculado do FPD seja menor que zero, ele será considerado igual a zero.

8.3 Nota Final do Trabalho Prático (NF)

A nota final do trabalho prático será obtida pela equação:

$$NF = 10 \times (0,6 \times FPI + 0,4 \times FPD)$$

8.4 Atraso

Para trabalhos entregues com atraso, haverá uma penalização conforme a fórmula:

$$\Delta p = \frac{2^{d-1}}{64}$$

onde d representa a quantidade de dias de atraso. Assim, sua nota final será atualizada da seguinte forma:

$$NF := NF \times (1 - \Delta p)$$

Note que a penalização é exponencial e, **após 7 dias de atraso, a penalização irá zerar a sua nota**.

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