PCC104 – Projeto e Análise de Algoritmos

Departamento de Computação - Universidade Federal de Ouro Preto - MG
Professor: **Pedro Silva** (`www.decom.ufop.br/`)
Estagiário docente: ——

Trabalho Prático I (TP I) - 10 pontos, peso 2.

- Data de entrega: 02/05/2024 até 23:59. O que vale é o horário do (a definir), e não do *seu*, ou do *meu* relógio!!!

- Clareza, identação e comentários no código também vão valer pontos. Por isso, escolha cuidadosamente o nome das variáveis e torne o código o mais legível possível.

- O padrão de entrada e saída deve ser respeitado exatamente como determinado no enunciado. Parte da correção é automática, não respeitar as instruções enunciadas pode acarretar em perda de pontos.

- Durante a correção, os programas serão submetidos a vários casos de testes, com características variadas.

- A avaliação considerará o tempo de execução e o percentual de respostas corretas.

- Eventualmente serão realizadas entrevistas sobre o trabalho para complementar a avaliação;

- O trabalho é individual (grupo de UMA pessoa).

- Será aceito trabalhos após a data de entrega, todavia com um decréscimo de 0,05 a cada 10min.

- Os códigos fonte serão submetidos a uma ferramenta de detecção de plágios em software.

- Códigos cuja autoria não seja do aluno, com alto nível de similaridade em relação a outros trabalhos, ou que não puder ser explicado, acarretará na perda da nota.

- Códigos ou funções prontas específicas de algoritmos para solução dos problemas elencados não são aceitos

- Não serão considerados algoritmos parcialmente implementados.

- Procedimento para a entrega:.

1. Submissão: via **(a definir)**.
2. Os nomes dos arquivos e das funções devem ser especificados considerando boas práticas de programação.
3. Funções auxiliares, complementares aquelas definidas, podem ser especificadas e implementadas, se necessário.
4. A solução deve ser devidamente modularizada sempre que cabível.
5. Os arquivos a serem entregues, incluindo aquele que contém *main()*, devem ser compactados (*.zip*), sendo o arquivo resultante submetido via **(a definir)**.
6. Você deve submeter os arquivos fonte e o *.pdf* (relatório) na raiz do arquivo *.zip*.
7. Caracteres como acento, cedilha e afins não devem ser utilizados para especificar nomes de arquivos ou comentários no código.

- Bom trabalho!

Tarefa 1 - Cálculo da multiplicação de duas matrizes

Considere A e B matrizes quadradas de inteiros de ordem $n \times n$ compostas de elementos a_{ij} e b_{ij}, respectivamente, com $i, j = 1, 2, \ldots, n$. No produto $C = A.B$, pode-se definir o elemento c_{ij} de C, para todos $i, j = 1, 2, 3, \ldots, n$, como:

$$c_{ij} = \sum_{k=1}^{n} a_{ik}.b_{kj}.$$

Para calcular a matriz produto C, deve-se calcular n^2 elementos cada um com um custo de $\Theta(n)$ somas e produtos. Portanto, usando-se este procedimento tem-se $\Theta(n^3)$ para o computo da matriz produto C.

utilizando-se do paradigma de **divisão-e-conquista**, é possível realizar o cálculo do produto de duas matrizes quadradas em $O(n^3)$. Apresente um algoritmo com tal complexidade e implemente-o.

Detalhes da implementação

Para atingir o seu objetivo, você deverá construir um Tipo Abstrato de Dados (TAD) (ou Classe) `Matriz` como representação de uma matriz que você quer analisar. As funções a serem implementadas ficam a cargo do aluno, contudo sugere-se funções para criar e destruir a matriz, bem como funções de leitura, impressão e para multiplicação.

Entrada

A entrada é dada por meio do terminal. Para facilitar, a entrada será fornecida por meio de arquivos. [1] A primeira linha especifica as dimensões da matriz que não necessariamente é quadrada, logo serão informados dois valores: o número de linhas (nl) e o número de colunas (nc) respectivamente. Depois serão fornecidas mais $\times nl$, cada linha com nc colunas e depois o mesmo esquema com as informações da segunda matriz.

Saída

A saída corresponde a matriz resultante da multiplicação. Também pode ocorrer que as dimensões das matrizes não possibilitem a multiplicação. Quando isso ocorrer, apresente a mensagem "EPIC FAIL!".

Exemplo de um caso de teste

Exemplos de saídas esperadas dada diferentes entradas:

Entrada	Saída
3 2 1 1 1 1 1 1 2 3 1 1 1 1 1 1	2 2 2 2 2 2 2 2 2

Entrada	Saída
1 2 1 1 1 2 1 1	EPIC FAIL!

[1] Para usar o arquivo como entrada no terminal, utilize `./executavel < nome_do_arquivo_de_teste`.

Tarefa 2 - Algoritmo de Dijkstra

O algoritmo de Dijkstra é um algoritmo de caminho mínimo usado em grafos. Ele é usado para encontrar o caminho mais curto entre dois vértices em um grafo ponderado, onde os pesos são associados às arestas.

Para este trabalho, você deve implementar o algoritmo de Dijkstra e utilizar as instâncias da competição USA-road-d.NYc.gr (`http://www.decom.ufop.br/haroldo/1acompcm/index.html`) para realizar os testes.

Tanto a entrada, como as saídas são fornecidas no próprio site da competição.

Detalhes da implementação

Para atingir o seu objetivo, você deverá construir Tipos Abstratos de Dados (TAD) (ou Classes) `Grafo`, `Aresta` e `Vértice` para representar um grafo e suas arestas e vértices. As funções a serem implementadas ficam a cargo do aluno, contudo sugere-se funções para criar e destruir o grafo, bem como funções de leitura, impressão e para achar a saída.

Imposições e comentários gerais

Neste trabalho, as seguintes regras devem ser seguidas:

- Você deve usar das boas práticas para o desenvolvimento do código, como identação, modularização, etc. **Não é autorizado o uso de qualquer biblioteca pronta para o desenvolvimento do projeto.**
- A separação das operações em funções e procedimentos está a cargo do aluno, porém, **não deve haver acúmulo** de operações dentro de uma mesma função/procedimento.
- O limite de tempo para solução de cada caso de teste é de apenas **um segundo**.

O que deve ser entregue

- Código fonte do programa em qualquer linguagem (**bem identado e comentado**).
- Documentação do trabalho (relatório[2]). A documentação deve conter:
  1. **Introdução:** descrição sucinta do problema a ser resolvido e visão geral sobre o funcionamento do programa.
  2. **Implementação:** descrição sobre a implementação do programa. **Não faça** "*print screens*" de telas. Ao contrário, procure resumir ao máximo a documentação, fazendo referência ao que julgar mais relevante. É importante, no entanto, que seja descrito o funcionamento das principais funções e procedimentos utilizados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado. Muito importante: os códigos utilizados na implementação devem ser inseridos na documentação.
  3. **Estudo de Complexidade:** estudo da complexidade do tempo de execução dos procedimentos implementados e do programa como um todo (notação O), considerando entradas de tamanho n.
  4. **Testes:** descrição dos testes realizados e listagem da saída (não edite os resultados).
  5. **Análise**: deve ser feita uma análise dos resultados obtidos com este trabalho. Por exemplo, avaliar o tempo gasto de acordo com o tamanho do problema.
  6. **Conclusão**: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
  7. **Bibliografia:** bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso.
  8. **Formato:** PDF ou HTML.

[2]Exemplo de relatório: `https://www.overleaf.com/latex/templates/modelo-relatorio/vprmcsdgmcgd`.

Como deve ser feita a entrega

Verifique se seu programa compila e executa na linha de comando antes de efetuar a entrega. Quando o resultado for correto, entregue via *(a definir)* até a 02/05/2024 até 23:59 um arquivo **.ZIP** com o nome e sobrenome do aluno. Esse arquivo deve conter: (i) os arquivos fonte utilizados na implementação, (ii) instruções de como compilar/executar o programa no terminal, e (iii) o relatório em **PDF**.

A SAÍDA DA SUA IMPLEMENTAÇÃO DEVE SEGUIR EXATAMENTE A SAÍDA PROPOSTA.

O SEU CÓDIGO SERÁ TESTADO EM AMBIENTE LINUX