Instituto Politécnico de Beja
Escola Superior de Tecnologia e Gestão
Licenciatura em Engenharia Informática

Estruturas de Dados e Algoritmos

enunciado do trabalho de avaliação

José Jasnau Caeiro

2020

1 Introdução

O trabalho encontra-se dividido em 4 partes que correspondem às divisões da matéria leccionada na disciplina:

1. algoritmos de ordenação e análise da complexidade computacional dos algoritmos;
2. estruturas de dados simples: conjuntos dinâmicos, *hash-tables* e árvores de pesquisa binária;
3. grafos e algoritmos de: pesquisa simples; de geração de árvores de envergadura mínima e de pesquisa de caminhos mais curtos;
4. introdução à concorrência e paralelismo na algoritmia.

Ao longo do semestre e com prazos pré-determinados os alunos irão realizar parcelas do trabalho que correspondem às diversas partes do programa da disciplina.

O trabalho deve ser realizado numa pasta designada por `EDA-Trabalho-2020`. O código deve residir numa sub-pasta designada por `src` e a documentação que fôr produzida deve residir numa sub-pasta designada por `docs`. Os ficheiros em formato `PDF` devem residir numa pasta designada por `pdfs`.

Os trabalhos que não obedeçam a esta estrutura de pastas terão avaliação nula.

Apenas serão considerados ficheiros que contenham a seguinte identificação:

- nome completo do aluno;
- número mecanográfico do aluno;
- e-mail do aluno.

É imprescindível que estes elementos identificativos estejam em cada ficheiro submetido para avaliação.

2 Objectivos/Requisitos

As sub-secções seguintes enunciam os objectivos e requisitos para cada parcela do trabalho de avaliação e as suas datas para realização.

As datas de cada parcela são indicativas mas se não forem cumpridas os resultados da sua avaliação serão diminuídos em 10% na primeira semana de atraso e em 20% por cada semana adicional.

2.1 Complexidade Computacional e Algoritmos de Ordenação

Nesta parcela do trabalho enumeram-se as seguintes tarefas:

1. instalar o seu próprio sistema de desenvolvimento de código para a linguagem de programação `Kotlin` acordo com as preferências de cada aluno;
2. instalar a biblioteca `OpenCV` e colocar esta a funcionar com a linguagem de programação `Kotlin` numa qualquer variante do sistema operativo; `Ubuntu`[1] de acordo com guia próprio de instalação que se encontra disponível na página `Moodle` da disciplina;
3. programar o algoritmo Bucket-Sort[2];
4. realizar o gráfico em formato `PDF` representativo da curva teórica da taxa de crescimento do tempo de execução do algoritmo Bucket-Sort com a aplicação `Gnuplot`;
5. projectar o protocolo experimental que permita realizar o gráfico representativo da taxa de crescimento do tempo de execução do algoritmo programado no trabalho com um erro relativo das medidas inferior a 1%;
6. realizar o gráfico em formato `PDF` representativo da curva experimental da taxa de crescimento do tempo de execução do algoritmo Bucket-Sort com a aplicação `Gnuplot`;
7. realizar um programa usando o algoritmo Bucket-Sort que aplicado a uma imagem monocromática, de dimensão correspondente a `Full HD`, e lida com a biblioteca `OpenCV`, devolva um histograma, sob a forma de ficheiro, com 100 *buckets*[3];
8. realizar o gráfico que representa o histograma em formato `PDF` com a aplicação `Gnuplot`;
9. usar adequadamente o repositório `Git`, nomeadamente editando o ficheiro `.gitignore` de modo a não colocar no repositório material desnecessário;

[1] Preferencialmente a versão LTS (*Long Term Support*).
[2] Pode encontrar-se uma introdução ao algoritmo em `https://en.wikipedia.org/wiki/Bucket_sort`.
[3] Em português **recipientes**.

10. documentar todos os procedimentos, e código, no ficheiro `parte1.md` escrito no formato `Markdown`[4].

O algoritmo deve ser programado usando como sub-algoritmo de ordenação[5] o algoritmo `Insertion-Sort`. O programa deve ser aplicado a vetores de números inteiros.

A realização dos gráficos deve ser realizada com a aplicação `Gnuplot`[6].

A data de entrega é: **27 de abril**.

2.2 Estruturas de Dados Simples

Nesta parcela do trabalho enumeram-se as seguintes tarefas:

1. programar a estrutura de dados designada por **árvore de pesquisa binária balanceada *red-black*** leccionada na disciplina e que se descreve completamente no livro adotado (Cormen *et al.*, 2009) [1];

2. a gestão de memória desta estrutura de dados deve ser programada pelo aluno de acordo com os conteúdos leccionados na disciplina;

3. deve ser realizada uma aplicação que pegue num texto e que coloque na árvore de pesquisa binária cada uma das palavras. Deve usar-se uma função de comparação de `string` existente na biblioteca padrão da linguagem de programação `Kotlin`;

4. documentar todos os procedimentos, e código, no ficheiro `parte2.md` escrito no formato `Markdown`.

A data de entrega é: **11 de maio**.

2.3 Grafos

Nesta parcela do trabalho enumeram-se as seguintes tarefas:

1. programar a estrutura de dados designada por **grafo** por meio de listas de adjacência, na forma leccionada na disciplina e que se descreve completamente no livro adotado (Cormen *et al.*, 2009) [1];

2. o aluno deve usar a representação de listas disponível para a linguagem de programação `Kotlin`;

[4] A descrição do que é este formato está disponível em `https://en.wikipedia.org/wiki/Markdown`. Um tutorial encontra-se disponível em `https://www.tutorialspoint.com/grav/grav_markdown_syntax.htm`. Uma folha em formato `PDF` com o resumo de `Markdown` está disponível em `https://guides.github.com/pdfs/markdown-cheatsheet-online.pdf`.

[5] O algoritmo Next-Sort na representação que se encontra em `https://en.wikipedia.org/wiki/Bucket_sort`.

[6] Esta aplicação está disponível em `http://www.gnuplot.info/`.

3. deve ser realizada uma aplicação que recebe uma tabela de `N` distâncias, sob a forma de números pertencentes ao corpo dos reais positivos, e que usa o algoritmo de Bellman-Ford para estimar os caminhos mais curtos entre cada nó. O número N deve ser escolhido de modo a que no computador do aluno o tempo de execução do algoritmo seja de aproximadamente 600 segundos;

4. documentar todos os procedimentos, e código, no ficheiro `parte3.md` escrito no formato `Markdown`.

A data de entrega é: **1 de junho**.

2.4 Concorrência e Paralelismo

A linguagem de programação `Kotlin` suporta a programação com *threads* multi-núcleo o que permite acelerar em muitos casos os algoritmos que operam sobre imagens. Este suporte é proporcionado pela JVM (*Java Virtual Machine*). É de mencionar neste caso o *package* `kotlin.concurrent` que segue de perto a implementação em Java (Kotlin, 2019) [2].

Nesta parcela do trabalho enumeram-se as seguintes tarefas:

1. realizar a leitura duma imagem a cores de dimensão correspondente ao formato `4K UHD`[7], (3840 × 2160 pxl), com a biblioteca `OpenCV`;

2. dividir a imagem em blocos de dimensão 192 × 108 pxl e para cada bloco calcular a média para cada componente de côr: vermelho, verde e azul. Calcular, também, o desvio padrão, em cada bloco;

3. medir o tempo de execução do procedimento anterior com erro relativo inferior a 1%;

4. colocar a informação das médias e dos desvio-padrão numa lista e depois escrever o conteúdo da lista num ficheiro de texto;

5. programar os procedimentos de cálculo da média e do desvio-padrão tomando em conta a programação *multicore*;

6. medir o tempo de execução do procedimento para *multicore* com erro relativo inferior a 1%;

7. documentar todos os procedimentos, descrever completamente a máquina virtual que foi usada e o *hardware* subjacente. Documentar o código, no ficheiro `parte4.md` escrito no formato `Markdown`.

A data de entrega é: **22 de junho**.

[7] Wikipedia, `https://en.wikipedia.org/wiki/4K_resolution`.

3 Avaliação

3.1 Material Produzido pelo Aluno

Apenas o material que seja integralmente original e realizado pelo aluno contará para a sua avaliação.

Exclui-se da avaliação qualquer código ou material não original quer se encontre na *Internet* ou sob a forma de qualquer outra documentação. Exclui-se da avaliação qualquer código ou texto que tenha sido realizado por outra pessoa.

No caso de existir dúvida sobre a originalidade do trabalho então o aluno será sujeito a procedimento de avaliação alternativo que deve incluir a produção de novo código em tarefa escolhida pelo docente.

No caso de utilização de código ou texto não original as fontes devem ser sempre adequadamente citadas/referenciadas.

No caso de não serem citadas/referenciadas as fontes originais usadas no trabalho, toda a parte do trabalho em que estas sejam utilizadas será anulada.

3.2 Percentagens da Avaliação

A realização de cada parte tem o mesmo peso na classificação final.

A avaliação de cada parte do trabalho está dividida em:

10% Funcionamento e facilidade de utilização da aplicação. A classificação é proporcional à facilidade e correção de procedimentos documentados com que o docente porá o sistema a funcionar na sua própria máquina virtual.

40% Código desenvolvido e as suas consequências em termos de tempos de execução (desempenho computacional). A qualidade do código é avaliada em muitas vertentes: o uso apropriado das bibliotecas; a forma como se adotam as boas práticas da programação; a arquitetura e a estrutura do código; o uso correto de nomes de variáveis, funções e classes; a documentação do código; os paradigmas de programação, *etc.*;

20% A documentação quer em formato `Markdown` quer a que se encontra presente no código sob a forma de comentários;

30% As metodologias de desenvolvimento, nomeadamente o uso do sistema de controlo de versões e a originalidade das soluções adotadas, serão avaliadas aqui. A classificação será proporcional à utilização correta do sistema de controlo de versões ao longo do tempo e à originalidade demonstrada no desenvolvimento.

A data de entrega final dos elementos de avaliação está na página da coordenação do curso.

3.3 Critérios Gerais de Avaliação

Os critérios gerais de avaliação são:

0-5 valores quando a componente avaliada é de **má** qualidade, não demonstrando trabalho original do autor;

6-9 valores quando o aluno desenvolveu trabalho mas com o funcionamento errado e sem corresponder aos requisitos pedidos. O trabalho corresponde de forma **medíocre** às exigências para a realização positiva da disciplina;

10-13 valores quando o aluno cumpre os requisitos mas com falhas diversas na documentação, falta de qualidade geral no código, demonstrando de forma geral que consegue realizar o trabalho em graus diversos de qualidade meramente **suficiente**;

14-17 valores quando o aluno apresenta boa qualidade geral no seu trabalho, com soluções desenvolvidas por ele, documentação conveniente e código que demonstra a sua proficiência na disciplina. O trabalho é, portanto, **bom** em vários graus;

18-20 valores o aluno apresenta todos os requisitos de qualidade do trabalho e demonstra um conhecimento aprofundado que lhe permite destacar-se pela originalidade das soluções apresentadas com pesquisa da documentação científica disponível e propondo alternativas às realizações comuns dos algoritmos. O trabalho salienta-se por ser **muito bom** em todas as vertentes.

Referências

[1] Thomas Cormen et al. *Introduction to algorithms*. Cambridge, Mass: MIT Press, 2009. ISBN: 9780262533058.

[2] Kotlin. *kotlin.concurrent*. Web. `https://kotlinlang.org/api/latest/jvm/stdlib/kotlin.concurrent/index.html`. Abr. de 2019.

Observações

Nenhuma parte do trabalho entregue fora do prazo será avaliada. É apenas admitida a utilização do sistema de controlo de versões para a submissão do trabalho.

Os alunos que tenham realizado regularmente ao longo do semestre os guias da disciplina podem beneficiar de valorização suficiente do trabalho para avaliação positiva. Isto quando se coloque uma situação em que classificação inicialmente atribuída se encontre entre os 8 e os 10 valores.