Universidade de Aveiro
Departamento de Eletrónica, Telecomunicações e Informática

Algoritmos e Estruturas de Dados

Relatório do Trabalho 1 – Análise de Algoritmos sobre Imagens RGB

Autores:
Nome do Aluno 1 – Nº Mecanográfico
Nome do Aluno 2 – Nº Mecanográfico

Ano Letivo: 2024/2025
Docentes: AED Team

Universidade de Aveiro – DETI
Algoritmos e Estruturas de Dados

Introdução

Este relatório apresenta uma análise aprofundada dos algoritmos desenvolvidos no âmbito do módulo **imageRGB** para o Trabalho 1 da unidade curricular **Algoritmos e Estruturas de Dados**. O projeto envolve operações sobre imagens representadas por LUT (look-up tables) e uma matriz de labels, incluindo funções de comparação, cópia, rotação, segmentação e region growing. O foco principal deste relatório é analisar detalhadamente o comportamento da função **ImageIsEqual**, complementada por testes experimentais, bem como comparar três abordagens diferentes para a operação de flood-filling numa imagem digital.

1. Estrutura Interna do TAD imageRGB

O TAD utilizado representa imagens RGB através de uma matriz de rótulos que apontam para uma tabela LUT onde se encontram as cores reais. Esta estratégia reduz o uso de memória, evita repetição de valores RGB e permite um acesso rápido e eficiente. A estrutura inclui: **width** – número de colunas **height** – número de linhas **image** – matriz de valores inteiros (labels) **LUT** – vetores com cores RGB **num_colors** – número de cores em uso

2. Análise Formal da Função ImageIsEqual

A função **ImageIsEqual** compara duas imagens pixel a pixel, verificando a igualdade dos seus valores RGB reais. É importante notar que labels podem diferir entre duas imagens; por isso, é a LUT que determina a cor verdadeira. O funcionamento baseia-se num duplo ciclo aninhado que atravessa a imagem linha a linha, coluna a coluna, efetuando comparações de valores RGB.

2.1 Melhor Caso – Ω(1)

O melhor caso ocorre quando as imagens diferem logo no primeiro pixel. Apenas uma comparação é necessária, resultando numa complexidade constante.

2.2 Pior Caso – O(width × height)

Se as imagens forem rigorosamente iguais, todos os pixeis são comparados. Isto resulta numa complexidade linear relativamente ao número total de pixeis da imagem.

2.3 Caso Médio – Θ(width × height)

Assumindo diferenças aleatórias entre pixeis, a divergência ocorrerá, em média, aproximadamente a meio do processo, mantendo uma tendência linear.

3. Avaliação Experimental

A tabela seguinte apresenta o número de comparações observadas para vários tamanhos de imagem, em três cenários distintos: Pior caso – imagens totalmente iguais Melhor caso – diferença no primeiro pixel Caso médio – diferença aleatória (~33% da imagem)

Tamanho	Pixels	Iguais	Diferença cedo	Caso médio
50×50	2 500	2 500	1	830
100×100	10 000	10 000	1	3 300
150×150	22 500	22 500	1	7 500
200×200	40 000	40 000	1	13 200

4. Comparação Teórica vs Experimental

Os resultados obtidos coincidem exatamente com a análise teórica: O melhor caso apresenta sempre 1 comparação. O pior caso aumenta proporcionalmente ao número total de pixeis. O caso médio segue um comportamento claramente linear. Isto comprova a eficiência e previsibilidade do algoritmo.

5. Comparação das Estratégias de Region Growing

Foram implementadas três versões da operação de flood-filling: **Recursiva** – simples e elegante, mas arriscada para imagens grandes devido ao risco de stack overflow. **Stack (DFS)** – elimina o risco da abordagem recursiva, mantendo a ordem de profundidade. **Queue (BFS)** – ideal para preenchimento radial uniforme, mais adequado para grandes regiões. A abordagem com Queue apresenta normalmente melhor estabilidade, enquanto a versão com Stack é a mais eficiente em profundidade.

Conclusão

O trabalho permitiu explorar profundamente o funcionamento de algoritmos aplicados à manipulação de imagens. As análises formais e experimentais confirmam a natureza linear da função **ImageIsEqual**, enquanto a comparação entre estratégias de flood-filling demonstra como diferentes estruturas de dados influenciam diretamente o desempenho e segurança da execução. O módulo **imageRGB** revela-se eficiente, modular e bem projetado.