deti universidade de aveiro
departamento de electrónica,
telecomunicações e informática

1º Trabalho — O TAD image8bit

Algoritmos e Estrutura de Dados

Prof. Joaquim João Estrela Ribeiro Silvestre Madeira

Maria Luís Delgado Linhares	113534
Rui de Faria Machado	113765

Aveiro, Novembro 2023

Índice

Introdução

Este relatório detalha o desenvolvimento e a análise do primeiro trabalho proposto na disciplina "Algoritmos e Estruturas de Dados". O foco central é a criação e avaliação do Tipo Abstrato de Dados (TAD) image8bit, essencial para a manipulação e processamento de imagens em tons de cinza. Para garantir uma implementação eficaz, foi dada especial atenção à gestão de memória e à clareza do código.

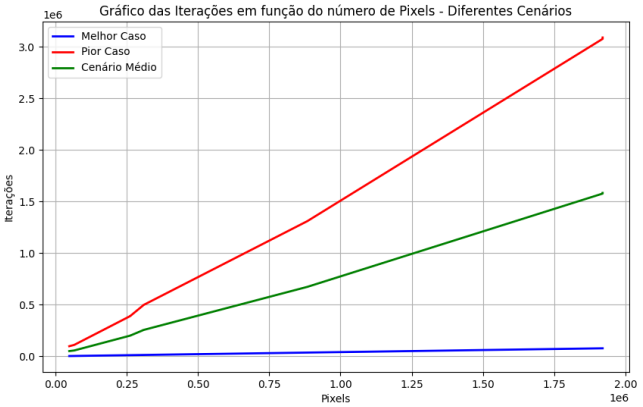
Além da elaboração do TAD, o trabalho inclui a análise da complexidade computacional e formal de duas funções específicas: ImageLocateSubImage(), destinada à identificação e localização de sub imagens noutras imagens; e ImageBlur(), que aplica um efeito de desfoque.

Para aprofundar esta análise, foram desenvolvidos scripts em Python para o tratamento de dados experimentais. Esses scripts permitiram uma avaliação mais precisa e detalhada do desempenho das funções, incluindo a análise de dados em escalas variadas e a criação de gráficos para uma melhor compreensão dos resultados.

ImageLocateSubImage()

Dados Experimentais

Pixel N	Iterações melhor caso	Iterações pior caso	Iterações cenário médio
1920000	76801	3087126	1581963.5
883600	35345	1310036	672690.5
1920000	76801	3077747	1577274
1920000	76801	3090881	1583841
307200	12289	494680	253484.5
307200	12289	492436	252362.5
262144	10405	389528	199966.5
262144	10405	390070	200237.5
48174	1893	109832	55862.5
65536	2602	97179	49890.5



Análise Formal

Etapas:

1. A função em questão possui dois loops seguidos, um a iterar sobre as linhas (altura) e outro sobre as colunas (largura) da imagem img1. Se a imagem img1 tiver dimensões W1 x H1, então o número de iterações será de W1 x H1.

2. Para cada posição (x, y) na img1, verifica-se se é possível posicionar a img2 a partir desse ponto utilizando a função ImageValidRect(). Essa função possui uma complexidade constante de O (Boundary Check), onde verifica se a imagem img2 cabe na img1 a partir de uma determinada posição.

3. Em seguida, cada pixel da img2 é comparado com o pixel correspondente da img1 usando a função MatchSubImage(). Se a img2 tiver dimensões W2 x H2, o número de operações realizadas por essa função será W2 x H2 para cada posição na img1 em que a img2 cabe.

Complexidade Total:

Esta análise considera a complexidade computacional da função em termos de tempo de execução, tendo em conta diferentes cenários possíveis.

Foi usada a terminologia N = W x H (número de pixels), de modo a facilitar a apresentação dos resultados.

Cenário	Descrição	Complexidade Computacional
Melhor Cenário	A img2 é encontrada na primeira posição verificada	O (W1 x H1) = O(N) Complexidade da ImageMatchSubImage()
Pior Cenário	A img2 não é encontrada ou é encontrada na última posição. Todas as posições em img1 são verificadas.	O(W1 x H1 x W2 x H2) = O(N^2)
Cenário Médio	Em média, com uma distribuição uniforme de possíveis locais de correspondência.	O(½ x W1 x H1 x W2 x H2) = O x ((1/2) x N^2)

ImageBlur()

Algoritmo Básico - Implementação Direta (Sem SAT)

A abordagem direta para o desfoque da imagem envolve, para cada pixel, calcular a média dos valores dos pixels numa região ao redor do pixel. Este método é intuitivo e fácil de implementar.

Análise Formal

Etapas:

1. Criação de uma imagem temporária: Cria uma imagem temporária (tempImg) do mesmo tamanho que a imagem original. Complexidade: O(1) para a alocação da estrutura, mas a alocação de memória para os pixels é O(W x H)
2. Loop externo para percorrer cada pixel: Dois loops seguidos percorrem cada pixel da imagem. Complexidade: O(W x H).
3. Cálculo do desfoque para cada pixel: Para cada pixel, é calculado o desfoque considerando um retângulo de dimensões (2 x dx + 1) x (2 x dy + 1) em volta do pixel.

Complexidade:

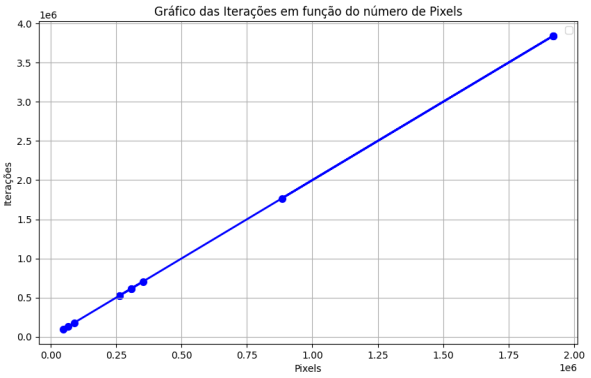
A complexidade total é o produto da complexidade do loop externo e da complexidade do cálculo do desfoque para cada pixel. Assim, a complexidade total é O(W x H x dx x dy).

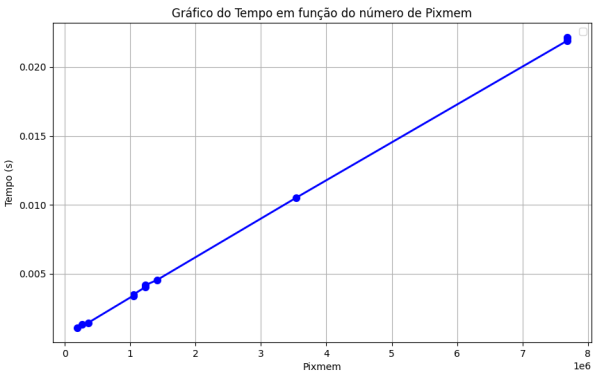
Algoritmo Melhorado - Implementação Utilizando SAT

A SAT é uma técnica que permite calcular rapidamente a soma dos valores dos pixels em qualquer região retangular da imagem. Uma vez calculada a SAT para a imagem, a soma dos valores dos pixels em qualquer região pode ser encontrada em tempo constante.

Dados Experimentais

Pixel N	Time	PIXMEM	Iterações
1920000	0.022116	7680000	3840000
1920000	0.021905	7680000	3840000
1920000	0.022157	7680000	3840000
883600	0.010503	3534400	1767200
351900	0.004536	1407600	703800
307200	0.004165	1228800	614400
307200	0.004024	1228800	614400
262144	0.003486	1048576	524288
262144	0.003408	1048576	524288
65536	0.001444	360000	180000
48174	0.001315	262144	131072
9000	0.001074	192696	96348





Análise Formal

Etapas:

1. Computação da SAT: A SAT é calculada para cada pixel da imagem. Para cada pixel, o valor da SAT depende dos valores da SAT dos pixels adjacentes. A complexidade desta etapa é de O(W x H).

2. Aplicação do desfoque (for loops na ImageBlur): Para cada pixel na imagem, a função calcula a soma dos valores dos pixels no retângulo de desfoque usando ComputeSumUsingSAT e, em seguida, calcula a média. A função ComputeSumUsingSAT tem uma complexidade constante O(1). Portanto, a complexidade para aplicar o desfoque a todos os pixels é O(W x H), pois cada pixel é processado uma vez.

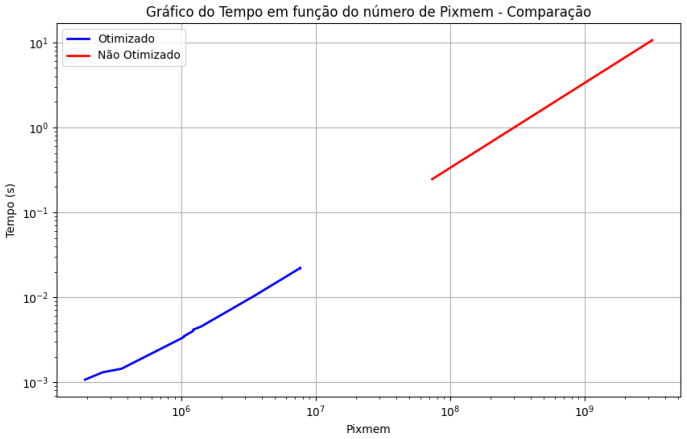
3. Substituição dos pixels da imagem original: Os pixels da imagem original são substituídos pelos pixels da imagem temporária. Esta etapa também tem uma complexidade O(W x H).

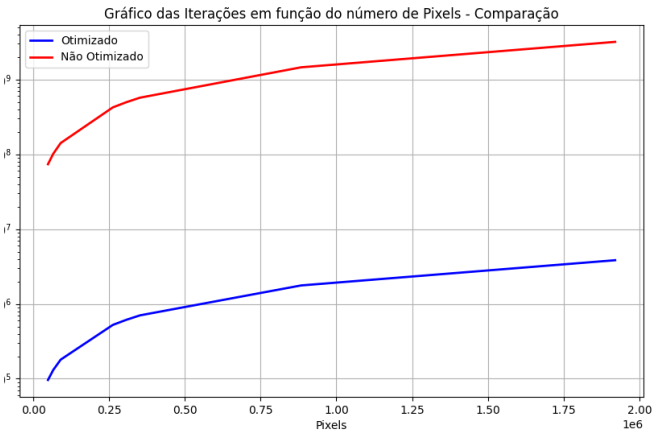
Complexidade:

A complexidade total da função ImageBlur é dominada pelas operações que têm complexidade O(N x M). Portanto, a complexidade total da função ImageBlur é O(N x M).

Análise Comparativa das diferentes abordagens

Critério	**Abordagem Direta**	**Abordagem com SAT**
Complexidade Computacional	Aumenta com o tamanho do kernel de desfoque (O(W x H x dx x dy))	Constante (O(W x H)), ou (O)N. Independente do tamanho do kernel
Eficiência	Menos eficiente para imagens grandes ou grandes kernels de desfoque	Mais eficiente, especialmente para imagens grandes e kernels grandes
Aplicabilidade	Adequada para imagens menores e aplicações menos exigentes	Superior para processamento em tempo real e imagens grandes
Simplicidade vs. Otimização	Mais simples e direta para implementar	Mais complexa na implementação, mas oferece melhor desempenho





Para analisar a eficiência do Pixmem (número de acessos à memória de cada pixel), adotamos uma escala logarítmica devido à grande discrepância nos dados. Esta abordagem destacou a melhoria no desempenho do algoritmo otimizado comparado ao não otimizado. Por exemplo, a maior imagem testada processada pelo algoritmo não otimizado levou 10.64 segundos, contra apenas 0.022 segundos pelo otimizado.

Além disso, um gráfico adicional mostrou a relação entre o número de pixels e iterações. Aqui, o algoritmo otimizado demonstrou superioridade, completando o processo com 3840000 iterações, enquanto o não otimizado necessitou de 3183320400 iterações.

Estes resultados evidenciam a eficiência aprimorada do algoritmo otimizado tanto em tempo quanto em número de iterações.

Conclusão

Em suma, este trabalho foi fundamental para aprofundar o nosso conhecimento sobre estruturas de dados, processamento de imagens e análise de complexidade. Através da implementação das funções, dos testes práticos e das análises formais, alcançamos sucesso na resolução do desafio proposto. Esta jornada foi enriquecedora, tanto em termos de conhecimento adquirido ao superar diversos obstáculos, quanto na experiência de trabalho em equipa.

Aveiro, Novembro 2023