Análise e Projeto de Algoritmos: Identificando a Região de Interesse de uma Imagem

Ana Beatriz Stahl, Gabriela Bley Rodrigues

Universidade do Vale do Rio dos Sinos (UNISINOS) – São Leopoldo, RS - Brazil

ana.beatriz.stahl@gmail.com, bley.gabriela@gmail.com

Resumo. *Este artigo descreve uma implementação de algoritmo para resolver o problema de identificação de uma região de interesse (ROI – Region of Interest) em uma imagem, com foco na identificação do subretângulo com a maior soma em um array bidimensional. Apresenta duas abordagens diferentes e compara seus resultados.*

1. Introdução

O problema de encontrar o subretângulo de soma máxima em uma matriz é um clássico na área da computação, com aplicação em processamento de imagens, análise de dados e muitos outros campos. A soma de um retângulo é definida como a soma de todos os elementos contidos nele, e o objetivo é identificar o subretângulo que maximiza essa soma. A análise eficiente é essencial para que o algoritmo funcione de maneira otimizada, pois para um *array* bidimensional muito grande, calcular e comparar todas as combinações possíveis poderia demandar um tempo excessivo ou até mesmo tornar a execução inviável.

2. Primeira solução: Força bruta

A abordagem de força bruta para encontrar o subretângulo com a maior soma em uma matriz envolve iterar sobre todas as possíveis combinações de subretângulos. O algoritmo começa fixando um ponto inicial (rStart, cStart) e um ponto final (rEnd, cEnd) para definir os limites do subretângulo. Para casa combinação desses pontos, calcula-se a soma dos elementos dentro do subretângulo. A soma máxima encontrada é armazenada e atualizada conforme necessário.

A complexidade dessa abordagem é $O(n^6)$, pois envolve seis loops aninhados para percorres todas as combinações possíveis de subretângulos

3. Segunda solução: Algoritmo de Kadane

A primeira solução – ou abordagem – deve ser otimizada de forma a fixar duas linhas da matriz, transformando o problema da submatriz em um problema de *subarray* unidimensional. Essa técnica se insere no conceito de "Programação Dinâmica", que visa dividir o problema em subproblemas mais simples, conforme apontado por Singhal (2018). Seu funcionamento se dá para cada par de linhas *i1* e *i2*, é necessário calcular a soma de cada coluna entre essas duas linhas, o que resulta em um vetor unidimensional que representa a soma acumulada de cada coluna.

$$temp[j] = \sum_{k=i1}^{i2} M[k][j] \; para \; j = 1 \; até \; n$$

O algoritmo de Kadane para matrizes segue uma sequência de etapas estruturadas para encontrar a submatriz de soma máxima. Primeiramente, o algoritmo começa selecionando um par de colunas da matriz, que delimitará uma faixa de colunas a serem consideradas. Para cada coluna inicial *i*, uma segunda coluna *j* é escolhida, onde $i \leq j$. Em seguida, o algoritmo soma os elementos entre essas colunas para cada linha da matriz, resultando em um vetor temporário unidimensional como mostra a expressão acima. Esse vetor temporário contém a soma dos elementos de cada linha entre as colunas *i* e *j*, o que transforma o problema bidimensional em um problema unidimensional. Com o vetor unidimensional gerado, o algoritmo é então aplicado para encontrar a subsequência contínua de maior soma nesse vetor. O algoritmo percorre o vetor acumulado e calcula a maior soma de uma subsequência contínua. Esse processo é repetido para cada combinação de colunas *i* e *j*, e, a cada iteração, a maior soma é armazenada. Ao final, o algoritmo retorna a maior soma encontrada dentre todas as faixas de colunas processadas.

A complexidade do algoritmo é $O(n^2 \cdot m)$. Para cada par de colunas, somam-se os elementos de cada linha, resultando em um *array* de *m* elementos. Aplicar o algoritmo de Kadane a esse *array*, tem-se o custo $O(m)$. Como há $O(n^2)$ combinações de pares de colunas, a complexidade final é $O(n^2 \cdot m)$, ou, como todos os casos de testes são matrizes quadradas, podemos determinar como $n = m$, portanto, $O(n^3)$

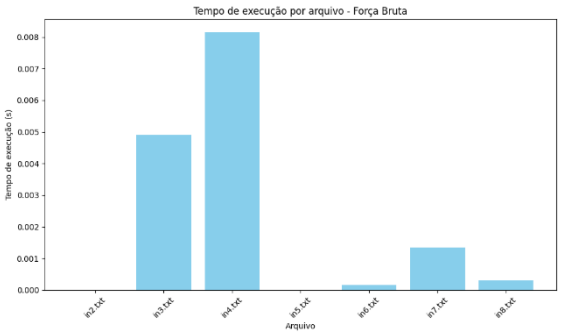
4. Resultados

A abordagem de força bruta, embora simples, possui complexidade $O(n^6)$, tornando-a impraticável para matrizes grandes. Em contraste, a adaptação do algoritmo de Kadane reduz a complexidade para $O(n^3)$, oferecendo uma solução mais eficiente e escalável.

Para ilustrar os resultados esperados do algoritmo, implementou-se, juntamente ao código, funções para a plotagem e criação de tabelas para a visualização do tempo tomado na execução.

A máquina utilizada nos testes possui um processador Intel Core i5-10300H com 2.50 GHz, 8 GB de RAM DDR4, sistema operacional Windows 11 Home, SSD de 1 TB e uma placa de vídeo GTX 1650.

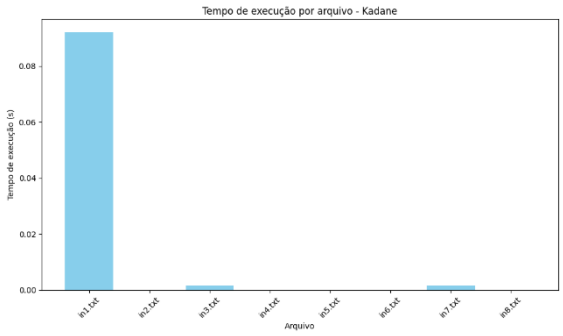
4.1. Gráfico de resultados da força bruta



	Arquivo	Soma máxima	Tempo de execução (s)
0	in2.txt	-1	0.000000
1	in3.txt	2886	0.004910
2	in4.txt	3607	0.008155
3	in5.txt	4	0.000006
4	in6.txt	-2	0.000158
5	in7.txt	18589	0.001342
6	in8.txt	15	0.000303

Figura 1. Tempo de execução do algoritmo para todas as entradas

4.2. Gráfico de resultados do algoritmo de Kadane



	Arquivo	Soma máxima	Tempo de execução (s)
0	in1.txt	9117	0.092170
1	in2.txt	-1	0.000000
2	in3.txt	2886	0.001559
3	in4.txt	3607	0.000000
4	in5.txt	4	0.000000
5	in6.txt	-2	0.000000
6	in7.txt	18589	0.001559
7	in8.txt	15	0.000000

Figura 2. Tempo de execução do algoritmo para todas as entradas

5. Conclusão

O algoritmo de Kadane, reconhecido por sua eficiência na resolução do problema de soma máxima em arrays unidimensionais, pode ser estendido de maneira eficaz para matrizes bidimensionais. Neste trabalho, investigamos essa extensão com o objetivo de encontrar a submatriz contínua com a maior soma, apresentando uma solução eficiente e viável para matrizes de tamanhos moderados a grandes. No contexto do processamento de imagens, onde cada elemento da matriz corresponde à intensidade de um pixel, o algoritmo de Kadane aplicado a matrizes pode ser utilizado para identificar regiões de interesse, como áreas de alta densidade de cor ou padrões específicos. Em vez de recorrer a métodos mais custosos em termos de tempo e recursos computacionais, essa abordagem oferece uma solução otimizada para identificar submatrizes que destacam características importantes da imagem.

6. Referências

KIRUBAHARAN, L. Understanding Kadane's Algorithm: Solving Real-Life Problems. Medium, 2020. Disponível em: <https://medium.com/@laaveniyakirubaharan/understanding-kadanes-algorithm-solving-real-life-problems-a0e99d63a65d>. Acesso em: 28 set. 2024.

SINGHAL, R. Kadane's Algorithm — (Dynamic Programming) — How and Why does it Work? Medium, 2020. Disponível em: <https://medium.com/@rsinghal757/kadanes-algorithm-dynamic-programming-how-and-why-does-it-work-3fd8849ed73d>. Acesso em: 28 set. 2024.

Maximum Subarray Sum using Divide and Conquer algorithm. GeeksforGeeks, 2017. Disponível em: <https://www.geeksforgeeks.org/maximum-subarray-sum-using-divide-and-conquer-algorithm/>. Acesso em: 28 set. 2024.

Largest Sum Contiguous Subarray. GeeksforGeeks, 2014. Disponível em: <https://www.geeksforgeeks.org/largest-sum-contiguous-subarray/>. Acesso em: 28 set. 2024.

Maximum Sum Submatrix, 2022. Disponível em: <https://www.geeksforgeeks.org/maximum-sum-submatrix>. Acesso em: 29 set. 2024.