Universidade Federal de Itajubá

Relatório de Análise de Algoritmos

Trabalho submetido ao Prof. João Paulo Leite.

Aluno: Felipe dos Santos
Matrícula: 2019002970

Itajubá
24 de Junho de 2021

INTRODUÇÃO

Algoritmos são um conjunto de passos para resolver um determinado problema. Uma receita de bolo é o exemplo mais comum que, de fato, representa o conceito de um algoritmo. Os passos de um algoritmo são compostos por operações, essas operações podem ser simples ou complexas como, por exemplo, uma equação de segundo grau comum e uma equação exponencial. Ao desenvolver um algoritmo, existem diversas maneiras de se obter a solução, porém algumas soluções são mais eficientes que as outras. Isso é relativo a diversas variáveis como, por exemplo, um algoritmo pode ser ruim para determinada arquitetura e organização e bom para outra. No caso deste trabalho, o foco dos testes foram os processadores de uso geral e as linguagens de desenvolvimento de alto nível, x86 e C + +. Os algoritmos estudados foram algoritmos para busca em uma matriz ordenada, isto é, uma matriz em que os termos das colunas e linhas são sempre maiores que os termos das colunas e linhas passadas. Dois algoritmos foram testados, sendo um que realiza a busca pelo modo da força bruta(testes para todos os valores) e outro que realiza a busca de maneira inteligente e faz uso da ordenação aplicada na matriz.

OBJETIVO

O objetivo do relatório foi coletar a quantidade de comparações realizadas para identificar um termo um uma matriz com relação aos algoritmos apresentados e realizar a análise dos dados para identificar a eficiência dos algoritmos.

CÓDIGO-FONTE

```cpp
//search is O(n^2)
bool brute_search(int m[][SIZE], int key)
{
    int comparacao = 0;
    for(int i = 0; i < SIZE; i++)
    {
        for (int j = 0; j < SIZE; j++)
        {
            comparacao++;
            if (m[i][j] == key)
            {
                cout << "Encontrei o numero " << key << " apos " << comparacao << " comparacoes" << endl;
                return true;
            }
        }

    }
    cout << "Nao foi possivel enconrtar o numero " << key << " apos " << comparacao << " comparacoes" << endl;
    return false;
}
```

Figura 01: Função Busca Bruta.

```cpp
//search is O(n)
bool smart_search(int m[][SIZE], int key)
{
    int comparacao = 0;
    int row, col;
    row = SIZE-1;
    col = 0;
    while(row >= 0 && col < SIZE)
    {
        comparacao++;
        if (m[row][col] == key)
        {
            cout << "Encontrei o numero " << key << " apos " << comparacao << " comparacoes" << endl;
            return true;
        }
        else if (m[row][col] > key)
        {
            row = row - 1;
        }
        else if (m[row][col] < key)
        {
            col = col + 1;
        }
    }
    cout << "Nao foi possivel enconrtar o numero " << key << " apos " << comparacao << " comparacoes" << endl;
    return false;
}
```

Figura 02: Função Busca Inteligente.

TESTES E RESULTADOS

brute	Dimensão da matriz (inteiro)	smart	Dimensão da matriz (inteiro)
100	10	10	10
400	20	20	20
900	30	30	30
2500	50	50	50
10000	100	100	100

Figura 03: Tabela com os valores obtidos nos testes.

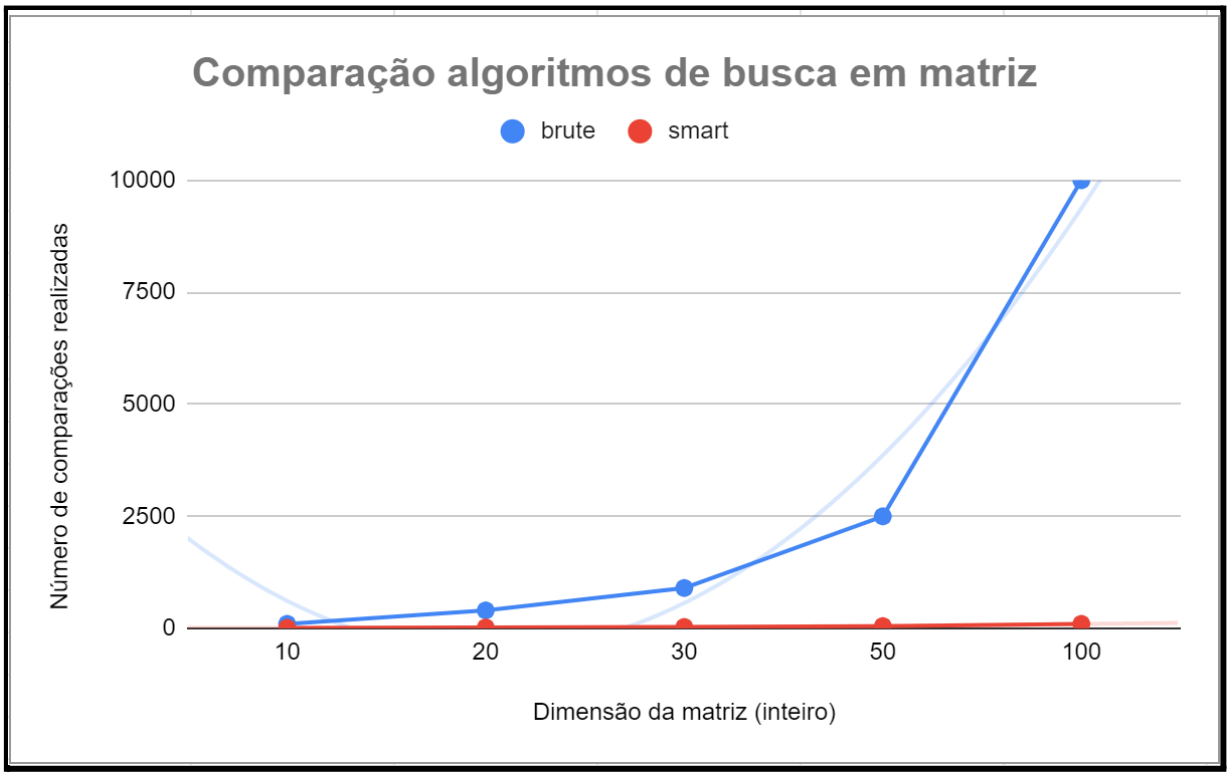

Figura 04: Gráfico dos valores obtidos nos testes.

DISCUSSÃO

A realização dos algoritmos foi realizada na linguagem C + + e a biblioteca utilizada foi apenas iostream. A arquitetura foi a x86. Sob a perspectiva da análise do pior caso de execução dos algoritmos, o algoritmo de busca bruta apresentou complexidade O(n^2), enquanto o algoritmo busca inteligente apresentou complexidade O(n). Tal afirmação pode ser comprovada por meio da visualização das linhas de tendência dos gráficos na Figura 04. Para a busca bruta, a linha de tendência demonstra a forma de uma curva semelhante à aproximação polinomial de segundo grau. Para a busca inteligente o problema apresenta linha de tendência semelhante à aproximação linear.

A matriz utilizada nos testes é uma matriz quadrada, sendo a dimensão desta matriz modificada em cada etapa do teste. Os testes foram realizados utilizando valores que não estão presentes na matriz em ordem de identificar o pior caso do algoritmo. Uma observação que pode ser realizada ao analisar a Figura 03 é a igualdade entre os valores da dimensão da matriz e o número de comparações realizadas pelo algoritmo busca inteligente. O algoritmo realiza, no máximo, o valor da dimensão da matriz em comparação. Isto se deve devido a ordenação da matriz e como o algoritmo realiza a busca. A busca inteligente elimina linhas ou colunas inteiras, enquanto a busca bruta elimina apenas um termo da matriz por comparação.

Os códigos fornecidos foram levemente alterados para ficar em conformidade com o padrão do C + +, assim como a biblioteca cstdio foi excluída.

CONCLUSÃO

Ainda que para determinados casos ambos os algoritmos resolvem o problema, a eficiência é um ponto que deve ser levado em consideração em algoritmos. A eficiência de um algoritmo está relacionada à quantidade de processamento que essa solução necessita para resolver o problema. Quanto maior a necessidade de processamento, maior o uso de energia elétrica. Resumindo, quanto maior a complexidade do algoritmo, maior o tempo para encontrar uma solução em um mesmo ambiente. O algoritmo que realiza a busca inteligente apresentou resultados superiores ao algoritmo busca bruta para a busca em matrizes ordenadas, porém para matrizes não ordenadas ele não resolve o problema. É interessante levar em consideração a complexidade de um algoritmo de ordenação de matrizes para analisar matrizes não ordenadas.