Relatório 2 da Implementação de Projeto e Análise de Algoritmos

Camila Lopes[1], **Victor Colen**[1]

[1]Instituto de CIências Exatas e Informática
Pontifícia Universidade Catórila de Minas Gerais (PUC-MG)
30535-901 – Coração Eucarístico – Belo Horizonte – MG - Brasil

{lopes.camila, victor.costa}@sga.pucminas.br

Resumo. *Este artigo apresenta três algoritmos para busca de elementos em um vetor de inteiros, sendo eles: busca linear, busca binária e busca linear com o uso de sentinela. Ademais, é realizada uma comparação entre os métodos com base no número de operações necessárias para localizar um elemento específico no vetor.*

1. Introdução

A busca eficiente de elementos em estruturas de dados é um problema fundamental na ciência da computação e no desenvolvimento de algoritmos, com aplicações em diversas áreas. Dessa forma, neste artigo projetamos três algoritmos para busca em vetores de inteiros: busca linear, busca binária e busca linear com sentinela.

A busca linear é o método mais simples, percorrendo sequencialmente o vetor até encontrar o elemento desejado. A busca binária, por sua vez, aproveita-se de vetores ordenados para reduzir drasticamente o espaço de busca a cada iteração. Já a busca linear com sentinela é uma variação otimizada da busca linear que elimina a necessidade de verificar o fim do vetor a cada iteração.

Posto isto, neste estudo, comparamos esses três algoritmos em termos do número de operações necessárias para encontrar um elemento específico, em quatro cenários diferentes: um cenário com o elemento no começo do vetor, no meio do vetor, no final do vetor e um cenário onde o elemento não está presentes no vetor. Dessa forma, nossa análise considera as distribuições de elementos, proporcionando uma visão abrangente da eficiência relativa de cada método.

2. Implementação

2.1. Busca Linear

A implementação da busca linear é direta e simples, o algoritmo percorre sequencialmente o vetor, comparando cada elemento com o valor buscado. Dessa forma, retorna verdadeiro se encontrado ou um indicador de falha caso contrário. Esta abordagem tem complexidade de tempo **O(n)** no pior caso, onde n é o número de elementos no vetor.

Algorithm 1 Algoritmo de Busca Linear

Entrada: Array $array$, tamanho do array n, elemento x
operações $\leftarrow 0$
for $i = 0$ até $n - 1$ **do**
    operações $\leftarrow$ operações $+1$
    if $array[i] = x$ **then**
        operações $\leftarrow$ operações $+1$
        retorna Verdadeiro, operações
    end if
    operações $\leftarrow$ operações $+1$
end for
operações $\leftarrow$ operações $+1$
retorna Falso, operações

2.2. Busca Binária

A implementação da busca binária requer um vetor previamente ordenado. Dessa forma, o algoritmo começa comparando o elemento buscado com o elemento central do vetor. Se forem iguais, a busca é concluída, caso contrário, é descartado metade do vetor baseado na comparação e repete o processo na metade restante. Assim, este processo continua, reduzindo pela metade o espaço de busca a cada iteração, até encontrar o elemento ou determinar que ele não existe no vetor.

A complexidade de tempo deste algoritmo é **O(log n)**, tornando-o eficiente para grandes conjuntos de dados ordenados.

Algorithm 2 Algoritmo de Busca Binária

Entrada: Array $array$, índices $left$ e $right$, elemento x
operações $\leftarrow 0$
while $left \leq right$ **do**
    operações $\leftarrow$ operações $+1$
    $mid \leftarrow left + (right - left)/2$
    operações $\leftarrow$ operações $+4$
    if $array[mid] = x$ **then**
        operações $\leftarrow$ operações $+1$
        retorna Verdadeiro, operações
    else if $array[mid] < x$ **then**
        $left \leftarrow mid + 1$
        operações $\leftarrow$ operações $+2$
    else
        $right \leftarrow mid - 1$
        operações $\leftarrow$ operações $+2$
    end if
end while
operações $\leftarrow$ operações $+1$
retorna Falso, operações

2.3. Busca Linear com Sentinela

A implementação da busca linear com sentinela é considerado uma otimização da busca linear tradicional. Neste algoritmo, antes de iniciar a busca, o valor procurado é temporariamente adicionado ao final do vetor como uma 'sentinela', dessa forma, é eliminado a necessidade de verificar o fim do vetor a cada iteração, simplificando o loop de busca. Assim, após percorrer o vetor, é verificado se o elemento encontrado está na posição da sentinela (indicando que o elemento não estava originalmente no vetor) ou em uma posição anterior (indicando sucesso na busca).

Embora mantenha a complexidade **O(n)** no pior caso, esta técnica pode oferecer uma melhoria de desempenho na prática, especialmente para vetores grandes.

Algorithm 3 Algoritmo de Busca Linear com Sentinela

Entrada: Array $array$, índice do último elemento $lastIndex$, elemento x
operações $\leftarrow 0$
$array[lastIndex] \leftarrow x$
operações $\leftarrow$ operações $+2$
$i \leftarrow 0$
operações $\leftarrow$ operações $+1$
while $array[i] \neq x$ **do**
    $i \leftarrow i + 1$
    operações $\leftarrow$ operações $+2$
end while
operações $\leftarrow$ operações $+1$
retorna (i $\neq lastIndex$), operações

3. Resultados e Discussão

Nesta seção, são apresentados os gráficos que comparam o número de operações necessárias para encontrar um elemento em um vetor de inteiros usando os três algoritmos implementados.

Como observado nos gráficos, a busca binária mantém uma eficiência constante em diferentes cenários, exceto no caso onde o elemento não está presente, enquanto as buscas lineares apresentam variações significativas dependendo da posição do elemento no vetor. Este comportamento reforça a teoria de que a busca binária é ideal para grandes conjuntos de dados ordenados, enquanto a busca linear pode ser mais adequada em casos específicos ou quando o vetor não é ordenado.

4. Conclusão

Este estudo comparou três algoritmos de busca - busca linear, busca binária e busca linear com sentinela. Dessa forma, concluímos que a eficiência dos algoritmos de busca varia significativamente dependendo do cenário e do tamanho do vetor. Assim, em nossa análise com um vetor de 100.000 elementos, o algoritmo de busca linear demonstrou ser mais eficiente que a busca linear com sentinela, contrariando expectativas iniciais e ressaltando a importância de testes práticos além da análise teórica.

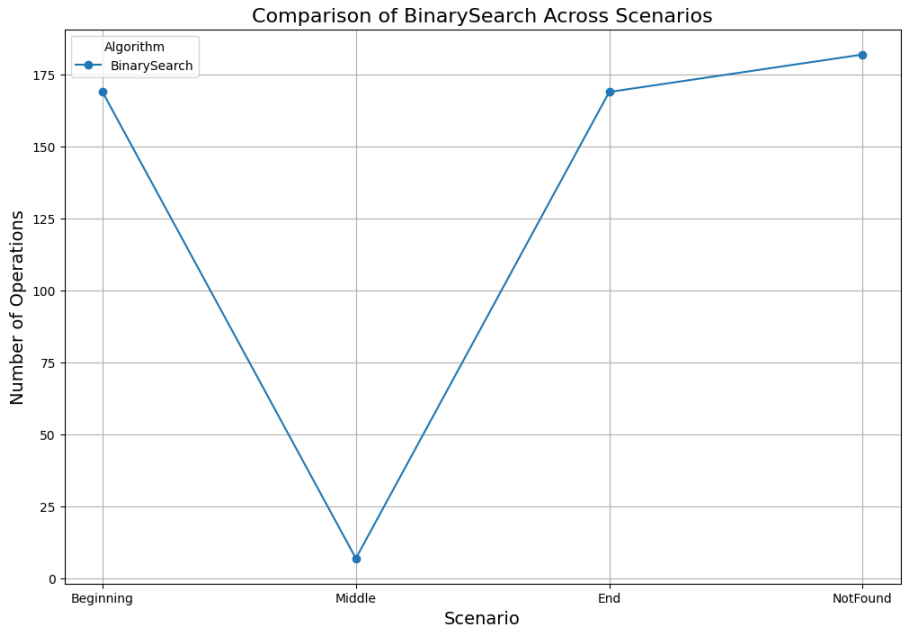

Figure 1. Comparação de operações da busca binária em diferentes cenários.

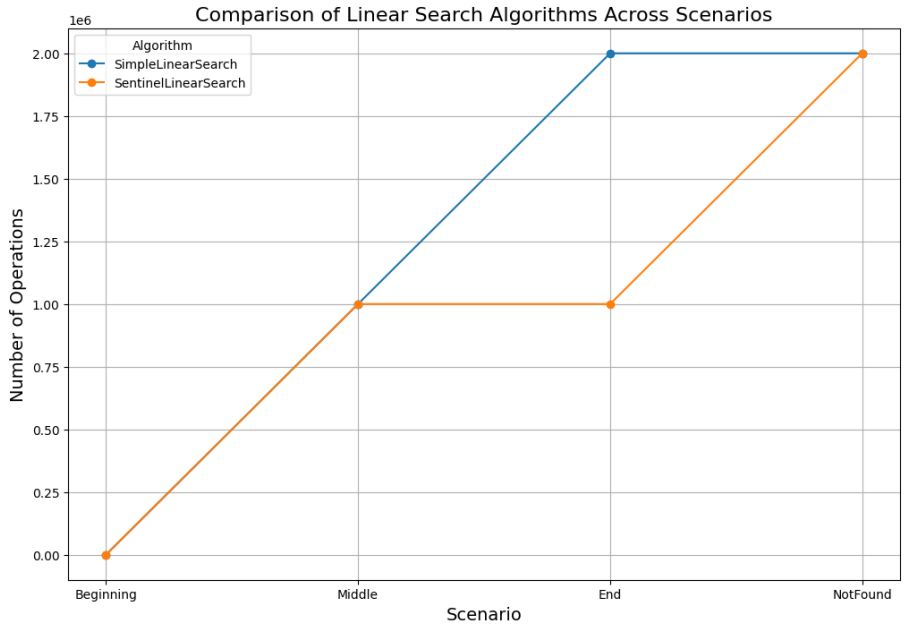

Figure 2. Comparação de operações das buscas lineares (simples e com sentinela) em diferentes cenários.

A busca binária, como esperado, mostrou-se mais eficiente em cenários com elementos no meio do vetor ou ausentes, especialmente em grandes conjuntos de dados. No entanto, sua necessidade de um vetor pré-ordenado deve ser considerada no cálculo do custo total da operação.

A performance superior da busca linear sobre a busca linear com sentinela em um vetor grande sugere que otimizações teóricas nem sempre se traduzem em ganhos práticos, possivelmente devido a fatores como cache de memória ou características es-

pecíficas do hardware utilizado.

Em conclusão, este estudo reafirma a importância da escolha adequada do algoritmo de busca baseada no contexto específico da aplicação, considerando fatores como o tamanho do conjunto de dados, a frequência das buscas e a possibilidade de manter os dados ordenados. Futuros estudos poderiam explorar o desempenho em cenários mais específicos ou adicionar mais algoritmos para comparação.