Análise Empírica dos Algorítimos de Pesquisa Binária e Pesquisa Sequencial

Kalyl Heings[1]

[1]Departamento de Ciências da Computação - Universidade do Estado de Santa Catarina (UDESC)
R. Paulo Malschitzki, 200 - Zona Industrial Norte, Joinville - SC, 89219-710

kalyl.henigs@edu.udesc.br

1. Introdução

No presente trabalho, será apresentado uma análise comparativa do desempenho de dois algoritmos de pesquisa em vetores com relação ao tamanho de suas entradas, sendo um deles uma busca sequencial simples e o outro uma busca binária

2. Algoritmos

Abaixo, estão, respectivamente, os algoritmos de busca sequencial e de busca binária, ambos implementados na linguagem de programação C, seguidos de uma breve explicação sobre seu funcionamento.

2.1. Busca Sequencial

```c
int busca_sequencial(int v[], int n, int elem)
{
    for (size_t i = 0; i < n; i++)
    {
        if (v[i] == elem)
        {
            return i;
        }
    }
    return -1;
}
```

O algoritmo de busca sequencial consiste em iterar pela lista, visitando as posições da lista uma a uma, até encontrar o elemento, retornando seu índice, ou senão -1 caso ele não tenha sido encontrado.

2.2. Busca Binária

```
int busca_binaria(int v[], int esq, int dir, int elem)
{
    while (esq <= dir)
    {
        int pivo = esq + (dir - esq) / 2;
        if (v[pivo] == elem)
        {
            return pivo;
        }
        if (v[pivo] < elem)
        {
            esq = pivo + 1;
        }
        else
        {
            dir = pivo - 1;
        }
    }

    return -1;
}
```

O algoritmo de busca binária funciona definindo um pivô central no elemento médio da lista, e então o compara com o elemento sendo buscado. Se for o elemento, ele retorna o índice, se o elemento for menor, um novo pivô ao lado esquerdo da lista é estabelecido, senão, um novo pivô é estabelecido do lado direito da lista, e o processo é repetido de maneira iterativa enquanto o índice mais a esquerda (inicializado em 0) for menor ou igual ao índice mais a direita (inicializado em n-1, onde n é o tamanho da entrada)

3. Análise Comparativa de Desempenho

Nesta sessão, será explicada a metodologia utilizada para comparar o desempenho dos dois algoritmos de busca, além dos resultados obtidos e uma breve discussão dos mesmos.

3.1. Metodologia

Para a comparação do desempenho dos dois algoritmos, os seguintes passos foram tomados:

- Primeiramente, um programa auxiliar escrito em um notebook do Jupyter foi criado
- Este programa tem por finalidade gerar entradas de tamanhos variados, indo de de $1 * 10^0$ até $1 * 10^6$, em incrementos de $1 * 10^3$.
- Para cada uma das entradas geradas, os dois algoritmos são alimentados com a entrada, e seus tempos de execução são avaliados utilizando a biblioteca time.h conforme o algoritmo apresentado abaixo
- Os tempos de execução para os dois algoritmos são então anotados, e a iteração segue para todas as 1000 entradas de tamanhos variados geradas

- Então, os dados coletados são organizados, e um gráfico utilizando os dados é plotado.

  Segue o excerto principal utilizado para a coleta dos tempos de execução:

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main()
{
    int size;
    scanf("%d", &size);
    int *v = malloc(size * (sizeof(int)));
    for (size_t i = 0; i < size; i++)
    {
        scanf("%d", &v[i]);
    }
    int elem = -1;
    // timestamp inicio da contagem
    clock_t t = clock();

    // CHAMADA DO ALGORITMO DE BUSCA

    // timestamp final da contagem
    t = clock() - t;
    double tempo = ((double)t) / CLOCKS_PER_SEC;
    printf("%f", tempo);
    return 0;
}
```

3.2. Resultados e Discussões

Abaixo está um gráfico que mostra o tempo de execução dos algoritmos para cada um dos 1000 pontos coletados, seguindo a metodologia descrita acima.

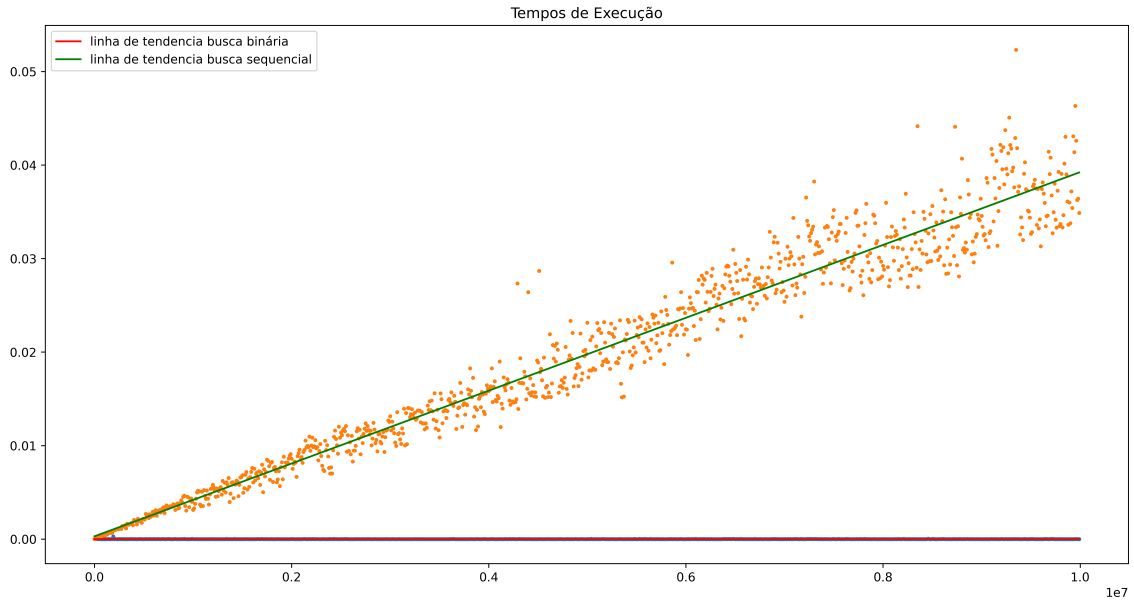

Figure 1. Tempos de execução dos dois algoritmos de pesquisa para entradas de $1 * 10^0$ até $1 * 10^6$ em passos de $1 * 10^3$

Conforme demonstrado no gráfico, o tempo de execução do algoritmo de busca binária mantém-se estável em $\mathcal{O}(log(n))$, enquanto o tempo do algoritmo de busca sequencial fica mais lento conforme o tamanho da entrada aumenta, com complexidade $\mathcal{O}(n)$.

As curvas verde e vermelha foram derivadas dos pontos coletados, e aproximam-se do valor assintótico teórico.