****

**UNIVERSIDADE FEDERAL DO CEARÁ - CAMPUS SOBRAL**

**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E COMPUTAÇÃO**

**DISCIPLINA: ESTUDOS ESPECIAIS**

**ALUNO: IAGO MAGALHÃES DE MESQUITA**

**ALGORITMOS PARA ENCONTRAR O MAIOR VALOR**

Sobral, 2023

**ALGORITMOS PARA ENCONTRAR O MAIOR VALOR**

Relatório apresentado como requisito para aprovação na disciplina de Estudos Especiais do programa de pós-graduação em engenharia elétrica e computação da Universidade Federal do Ceará.

Sobral, 2023

**SUMÁRIO**

**1 INTRODUÇÃO........................................................................................................... 3**

**2 OBJETIVOS................................................................................................................ 4**

**3 METODOLOGIA....................................................................................................... 4**

**4 RESULTADOS........................................................................................................... 5**

**5 CONCLUSÃO............................................................................................................ 15**

**6 REFERÊNCIAS......................................................................................................... 16**

1. **INTRODUÇÃO**

Algoritmos para encontrar o maior valor funcionam de forma simples, dado uma entrada se busca pelo maior presente nessa instância. Porém, existem diferentes formas de se encontrar esse valor, sendo assim, pode-se construir algoritmos com melhores desempenhos, reduzindo o tempo de processamento e uso de memória.

Tais algoritmos podem ser divididos em diferentes complexidades, tais como os 0(n). Um algoritmo é dito que usa tempo linear, ou tempo O(n), se sua complexidade de tempo é O(n). Informalmente, isto significa que para entradas grandes o suficiente o tempo de execução delas aumenta linearmente com o tamanho da entrada. Por exemplo, um procedimento que adiciona todos os elementos em uma lista requere tempo proporcional ao tamanho da lista. Esta descrição é levemente imprecisa, visto que o tempo de execução pode desviar significantemente de uma proporção precisa, especialmente para valores pequenos de n [1].

Em ciência da computação, a complexidade de tempo de um algoritmo quantifica a porção de tempo tomada por um algoritmo para rodar em função do tamanho da entrada do problema. A complexidade de tempo de um algoritmo é comumente expressada usando a notação big O, que suprime constantes multiplicativas e outros termos de menor ordem. Quando expressada dessa forma, a complexidade de tempo é dito ser descrita assintoticamente, i.e., como o tamanho da entrada vai para o infinito. Por exemplo, se o tempo requisitado por um algoritmo em todas as entradas de tamanho n é no máximo 5n3 + 3n, a assíntota da complexidade de tempo é O(n3) [1].

Neste trabalho iremos analisar algoritmos para encontrar o maior valor em uma instância de entrada. Observaremos o tempo necessário para realizar essa busca e o consumo de memória em dois algoritmos.

1. **OBJETIVOS**
   1. Objetivo Geral

* Realizar experimentos com algoritmos para encontrar o maior valor, visando analisar tempo de execução e uso de memória para diferentes instâncias.
  1. Objetivos Específicos
* Realizar leitura de instância não ordenadas;
* Realizar implementação dos algoritmos para encontrar o maior valor com a linguagem Python;
* Obter tempo de processamento de cada algoritmo;
* Obter uso memória de cada algoritmo;
* Plotar gráficos de resultados de tempo e consumo de memória.

1. **METODOLOGIA**

Neste trabalho, foram analisados dois algoritmos para encontrar o maior valor, sendo eles:

* maxVal1;
* maxVal2.

Os testes realizados foram utilizados instâncias com valores numéricos não ordenados, em relação a máquina na qual os dados foram processados, este trabalho se utilizou da seguinte máquina:

* Notebook Lenovo, AMD Ryzem 5, 8Gb de RAM.

As seguintes instâncias foram utilizadas para realizar as análises, sendo elas não ordenadas:

* 100
* 200
* 1000
* 2000
* 5000
* 10000
* 50000
* 100000
* 500000
* 1000000
* 5000000
* 10000000
* 100000000

Devido a questões computacionais e de tempo, as 4 últimas instâncias não foram utilizadas para análise dos algoritmos de buscas. Sendo eles:

* 1000000
* 5000000
* 10000000
* 100000000

Ao final da execução de todos os algoritmos, foram analisadas o tempo e o consumo de memória, plotado gráficos para realização de comparação entre eles e todos os scrips desenvolvidos foram postados no GitHub.

1. **RESULTADOS**

Todos os scripts desenvolvidos podem ser visualizados no GitHub no seguinte endereço: <https://github.com/IagoMagalhaes23/BBP1008---ESTUDOS-ESPECIAIS/tree/main/Trabalho_esquenta_02>

A seguir será analisado cada um dos algoritmos de forma individual com o tempo de processamento e memória utilizada, além disso, as instâncias de 1, 5, 10 e 100 milhões não foram testadas devido a problemas com memória e tempo de execução.

ALGORITMO 1 – maxVal1

A busca linear é o algoritmo de busca mais simples para vetores e qualquer outro tipo de estrutura de dados linear. A ideia básica do algoritmo é comparar o elemento procurado com cada elemento do vetor até encontrá-lo partindo, em geral, da primeira posição do vetor [3].

Tabela 01. Busca linear v1 para instâncias ordenadas.

|  |  |  |
| --- | --- | --- |
| **Instância** | **Tempo** | **Memória** |
| 100 | 0.0007576942443847656s | 215937024 |
| 200 | 0.0005022287368774414s | 201719808 |
| 1000 | 0.0005040884017944336s | 215937024 |
| 2000 | 0.000639653205871582s | 201719808 |
| 5000 | 0.0009804248809814453s | 201719808 |
| 10000 | 0.0016920328140258788s | 216203264 |
| 50000 | 0.006335234642028809s | 201719808 |
| 100000 | 0.012808132171630859s | 201719808 |
| 500000 | 0.035229849815368655s | 218005504 |
| 1000000 | X | X |
| 5000000 | X | X |
| 10000000 | X | X |
| 100000000 | X | X |

Fonte: Autor.

ALGORITMO 2 – maxVal2

A busca linear é o algoritmo de busca mais simples para vetores e qualquer outro tipo de estrutura de dados linear. A ideia básica do algoritmo é comparar o elemento procurado com cada elemento do vetor até encontrá-lo partindo, em geral, da primeira posição do vetor [3].

Tabela 01. Busca linear v1 para instâncias ordenadas.

|  |  |  |
| --- | --- | --- |
| **Instância** | **Tempo** | **Memória** |
| 100 | 0.0007576942443847656s | 215937024 |
| 200 | 0.0005022287368774414s | 201719808 |
| 1000 | 0.0005040884017944336s | 215937024 |
| 2000 | 0.000639653205871582s | 201719808 |
| 5000 | 0.0009804248809814453s | 201719808 |
| 10000 | 0.0016920328140258788s | 216203264 |
| 50000 | 0.006335234642028809s | 201719808 |
| 100000 | 0.012808132171630859s | 201719808 |
| 500000 | 0.035229849815368655s | 218005504 |
| 1000000 | X | X |
| 5000000 | X | X |
| 10000000 | X | X |
| 100000000 | X | X |

Fonte: Autor.

1. **CONCLUSÃO**

O trabalho foi realizado com êxito, sendo possível implementar todos os algoritmos utilizando a linguagem Python, realizando a leitura de todas as instâncias no formato ‘txt’. As métricas solicitadas, tempo e memória, foram obtidas e analisadas neste trabalho.

Com este trabalho foi possível ver que dado um problema, existem diferentes maneiras de soluciona-los e que utilizando uma melhor organização de código e recursos de hardware é possível melhorar os desempenhos.

1. **REFERÊNCIAS**

[1] ALGORITMOS de Busca - Algoritmos e Programação de Computadores. [S. l.], 1 jan. 2023. Disponível em: https://ic.unicamp.br/~mc102/aulas/aula11.pdf. Acesso em: 5 set. 2023.

[2] COMPLEXIDADE de tempo. [S. l.], 8 jan. 2023. Disponível em: https://pt.wikipedia.org/wiki/Complexidade\_de\_tempo#Tabela\_de\_complexidade\_de\_tempo\_comum. Acesso em: 5 set. 2023.

[3] Busca Linear. [S. l.], 14 set. 2017. Disponível em: <https://www.blogcyberini.com/2017/09/busca-linear.html>. Acesso em: 4 set. 2023.

[4] Busca Binária. Disponível em: https://pt.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search#:~:text=A%20busca%20bin%C3%A1ria%20%C3%A9%20um,localiza%C3%A7%C3%B5es%20poss%C3%ADveis%20a%20apenas%20uma.. Acesso em: 4 set. 2023.

[5] Pesquisa ternária vs Pesquisa binária. Disponível em: https://www.techiedelight.com/pt/ternary-search-vs-binary-search/. Acesso em: 4 set. 2023.