**Relatório Trabalho I**

**Algoritmos Recursivos de Busca e de Ordenação**

*Algoritmos e Estrutura de Dados II*

*Departamento de Informática*

*Universidade Federal do Paraná - UFPR*

*Eduardo Gabriel Kenzo Tanaka*

*Rafael Gonçalves dos Santos*

*Curitiba - PR*

1. **Introdução**

Este presente relatório tem como objetivo discorrer sobre os testes realizados com os algoritmos recursivos de busca e de ordenação do Trabalho 1 - Busca Sequencial (Ingênua), Busca Binária, Insertion Sort, Selection Sort, Merge Sort e Quick Sort.

1. **Testes e Análises**
2. **Busca Sequencial**

No melhor caso, todos os vetores farão 1 comparação no tempo de 0,000001s. Já no pior, serão n comparações.

| **comparações** | **tamanho** | **tempo** |
| --- | --- | --- |
| 1.000 | 1.000 | 0.000019s |
| 10.000 | 10.000 | 0.000209s |
| 50.000 | 50.000 | 0.001056s |
| 100.000 | 100.000 | 0.002015s |

*Tabela 1: relação comparações x tamanho x tempo para instâncias na busca sequencial.*

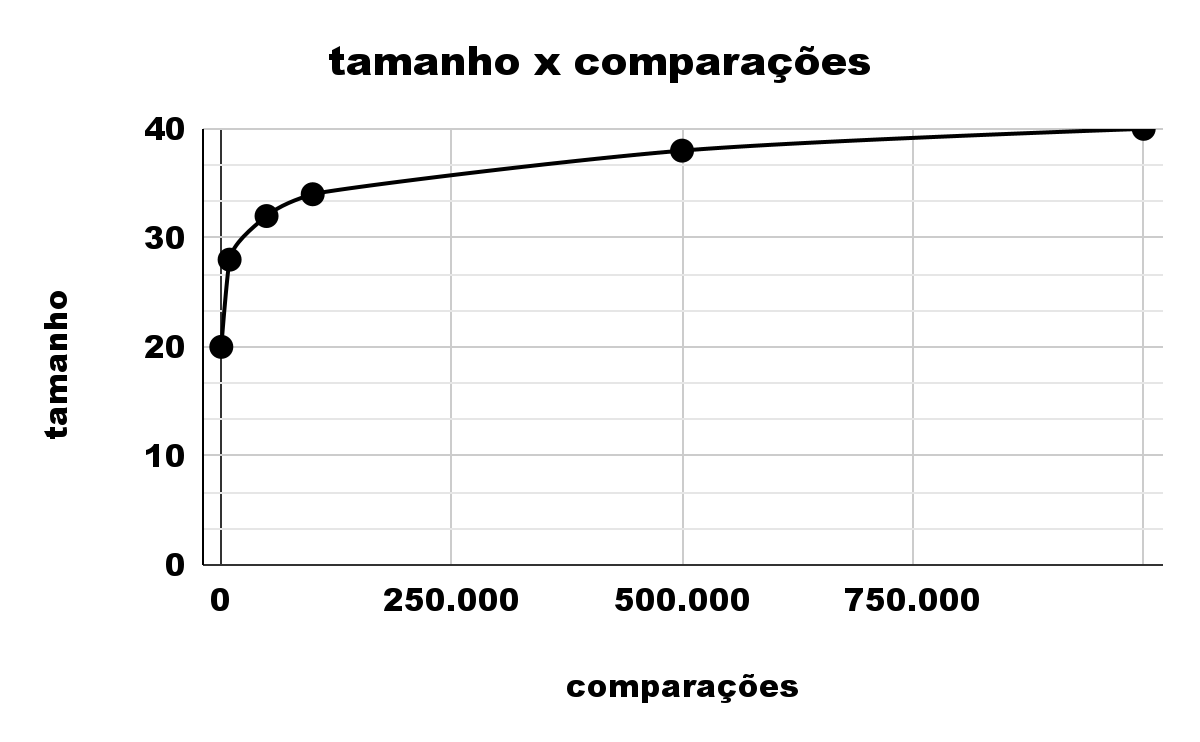
Pelos dados da tabela é notável a ineficiência da busca sequencial que possui crescimento linear.

1. **Busca Binária**

Para o melhor caso, o número de comparações é sempre 1 e o tempo 0,000001s. No pior caso o algoritmo realiza comparações.

| **comparações** | **tamanho** | **tempo** |
| --- | --- | --- |
| 20 | 1.000 | 0,000005s |
| 28 | 10.000 | 0,000001s |
| 32 | 50.000 | 0,000002s |
| 34 | 100.000 | 0,000006s |

*Tabela 2: relação comparações x tamanho x tempo para instâncias na busca binária.*



*Gráfico 1: relação tamanho x número de comparações na busca binária*

No gráfico 1 é possível perceber o crescimento logarítmico da relação tamanho *versus* comparações, o que mostra a eficiência do algoritmo de busca binária, principalmente quando utilizado em vetores maiores.

A disparidade no crescimento do número de comparações dos algoritmos pode ser analisada na tabela 3, na qual se observa uma diferença de 3.000x menos comparações em um vetor de 100.000 elementos.

| **tamanho** | **sequencial** | **binária** |
| --- | --- | --- |
| 1.000 | 1.000 | 20 |
| 10.000 | 10.000 | 28 |
| 50.000 | 50.000 | 32 |
| 100.000 | 100.000 | 34 |

*Tabela 3: número de comparações da busca sequencial e da busca binária*

1. **Insertion Sort**

No melhor caso o Insertion Sort fará: n - 1 comparações com elementos do vetor.

| **comparações** | **tamanho** | **tempo** |
| --- | --- | --- |
| 999 | 1.000 | 0.000032s |
| 9.999 | 10.000 | 0.000262s |
| 49.999 | 50.000 | 0.001375s |

*Tabela 4: relação comparações x tamanho x tempo para instâncias no Insertion Sort, melhor caso*

Já no pior caso comparará elementos.

| **comparações** | **tamanho** | **tempo** |
| --- | --- | --- |
| 499.500 | 1.000 | 0.009349s. |
| 49.995.000 | 10.000 | 0.599787s. |
| 199.990.000 | 20.000 | 2,401983s |
| 1.249.975.000 | 50.000 | 15.085648s |

*Tabela 5: relação comparações x tamanho x tempo para instâncias no Insertion Sort, pior caso*

1. **Selection Sort**

Para o Selection Sort as comparações são dadas por , uma vez que percorre, obrigatoriamente, todos os elementos do vetor.

| **comparações** | **tamanho** | **tempo** |
| --- | --- | --- |
| 499.500 | 1.000 | 0,026788s |
| 49.995.000 | 10.000 | 0,457488s |
| 199.990.000 | 20.000 | 1,676303s |
| 1.249.975.000 | 50.000 | 10,289607s |

*Tabela 6: relação comparações x tamanho x tempo para instâncias no Selection Sort.*

1. **Merge Sort**

Em todos os casos o Merge Sort fará comparações.

| **comparações** | **tamanho** | **tempo** |
| --- | --- | --- |
| 9.976 | 1.000 | 0,004081s |
| 133.616 | 10.000 | 0,042315s |
| 784.464 | 50.000 | 0,212875s |
| 1.668.928 | 100.000 | 0,417158s |
| 19.951.424 | 1.000.000 | 4,117102s |

*Tabela 7: relação comparações x tamanho x tempo para instâncias no Merge Sort, vetores aleatórios.*

1. **Quick Sort**

A quantidade de comparações no melhor caso será e, no pior caso,

| **comparações** | **tamanho** | **tempo** |
| --- | --- | --- |
| 11.586 | 1.000 | 0.000084s |
| 171.122 | 10.000 | 0.001143s |
| 977.082 | 50.000 | 0.006365s |
| 2.022.799 | 100.000 | 0.013671s |
| 27.289.991 | 1.000.000 | 0.171323s |

*Tabela 8: relação comparações x tamanho x tempo para instâncias no Quick Sort, vetores aleatório.*

Como mostrado na Tabela 8, o algoritmo do Quick Sort sempre realizou mais comparações se comparado com o Merge Sort, porém o Quick Sort, em todos os casos, realizou uma ordenação em tempo menor.

| **tamanho** | **merge sort** | **quick sort** |
| --- | --- | --- |
| 1.000 | 0,004081s | 0.000084s |
| 10.000 | 0,042315s | 0.001143s |
| 50.000 | 0,212875s | 0.006365s |
| 100.000 | 0,417158s | 0.013671s |
| 1.000.000 | 4,117102s | 0.171323s |

*Tabela 9: tempo de execução do Merge Sort e do Quick Sort.*

Isso se deve ao fato da principal função do Quick Sort, a Partição, ser mais leve do que a principal função do Merge Sort, o Intercala, cujo algoritmo possui uma cópia de vetor, o que acaba tirando a eficiência do mesmo.

**III. Conclusão**

Levando em consideração os testes, análises e comparações realizadas, conclui-se que os diferentes algoritmos de busca e de ordenação possuem prós e contras, seja em relação ao tempo físico ou às comparações, as quais devem ser levadas em conta quando implementadas em um projeto.