# Trabalho Prático 2
# Biblioteca Digital de Arendelle

Misael Saraiva de Rezende

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte – MG – Brasil

## 1. Introdução

O reino de Arendelle está se expandindo e se atualizando. Então solicitaram que seja feita a análise de vários algoritmos de Quicksort a fim de testar o desempenho das variações do mesmo e ajudar o reino a decidir qual o algoritmo mais adequado para o uso na implantação da biblioteca digital. Para isso foram examinados várias versões do Quicksort com o objetivo de ajudar a determinar a melhor solução a ser utilizada.

Foram testados sete versões do algoritmo. A primeira foi a versão *Clássica*, que escolhe o pivô no meio da partição a ser ordenada. A segunda foi a *Mediana de Três* que escolhe o pivô a partir da mediana entre o valor mais à esquerda, o mais à direita e o central da partição. O terceiro algoritmo testado foi o *Primeiro Elemento* que sempre escolhe como pivô o primeiro elemento da partição a ser ordenada. O quarto, quinto e sexto algoritmos são bem parecidos. Eles se baseiam no *Clássico* porém fazem uso do algoritmo de *Inserção* e também do conceito do Quicksort Mediana de Três. Esses três são executados normalmente igual o *Mediana de Três* até que a partição tenha menos de 1, 5 ou 10% de elementos. A partir desse momento é usado o algoritmo de *Inserção* para resolver essa partição menor. O último algoritmo testado foi o *Não Recursivo*, que basicamente usa o mesmo conceito do clássico porém é implementado sem o uso da recursão.

## 2. Implementação

O programa foi desenvolvido na linguagem de programação C, e compilado pelo compilador gcc.

### I. Estrutura de Dados

A implementação do programa teve como base a estrutura de dados de uma estrutura (struct) do tipo Item que armazena uma variável inteira de nome Chave. Nessa estrutura foram armazenados os elementos do vetor a serem testados. O código também foi modularizado para agrupar funções com objetivos relacionados.

## II. Modularização

Para modularizar a implementação, foram usados dois módulos principais, *Quicksort* e *MinhaBiblioteca* e o *main.c* (programa principal).

O módulo Quicksort contem a implementação dos algoritmos relacionados às variações dos quicksorts. Como os *Quicksort Clássico* e o *Mediana de Três* se diferem apenas na escolha do elemento pivô, foram juntados em uma implementação somente com variação na função *Partição*, na qual nela seria feito a escolha do pivô. Já o *Quicksort Primeiro Elemento* foi implementado um pouco diferente do mencionado anteriormente sem usar o procedimento *Ordena*. Por fim o *Quicksort Não Recursivo* implementado também não precisa do método *Ordena*, porém foi elaborado usando a estrutura de dados *pilha*, no qual, seus métodos, foram implementados no módulo *MinhaBiblioteca*.

Por sua vez, o módulo *MinhaBiblioteca* contém além da implementação da estrutura *pilha* e suas funções básicas, já mencionada, a implementação de outras funções. A função *escolheMediana* foi utilizada para escolher a mediana do tempo da execução do algoritmo de Quicksort. Para isso foi utilizado o método de *Inserção* para ordenar, de forma crescente, o vetor e assim obter o valor. O módulo também contém a função *gerarVetor* que como objetivo principal é criar os vetores de acordo com a ordenação requerida pelo usuário e, armazenar também em um vetor, se for passado o parâmetro -p, os elementos de cada vetor gerado, antes de serem ordenados. Por último a função *imprimeResultados* retorna para o usuário a média de comparações e movimentações do algoritmo de quicksort escolhido assim como a mediana do tempo de execução.

O módulo principal *main.c* contém a implementação que realiza a comunicação direto com o usuário do programa. Nele não há nenhuma função implementada. Apenas as variáveis que armazenam os dados a serem usados. Observa-se aqui que foi feito uma implementação especial caso o parâmetro de impressão não fosse passado. Essa elaboração possibilitou que não fosse necessária uma versão especial (quase igual) da função *gerarVetor* com apenas uma modificação nos parâmetros da função. Assim a variável *Auxiliar* foi alocada como uma matriz de tamanho 20x1 para satisfazer essa inevitabilidade. Com isso foi possível usar a mesma função sem precisar repetir código muito parecido.

## III. Entrada e saída de dados

Espera-se que o usuário entre com os parâmetros a serem executados pelo programa pelo terminal de comando do sistema - veja as instruções de execução no próximo tópico. São passados por parâmetro o tipo de quicksort a ser testado, qual o tipo de ordenação inicial e o tamanho da entrada além de, opcionalmente, um parâmetro **-p**.

Quanto ao tamanho dos dados de entrada, existem três tipos: desordenados (aleatórios), ordenados de forma crescente e ordenados de forma decrescente. Em relação ao tamanho dos dados, há variações de cinquenta mil (50.000) até quinhentos mil (500.000) pergaminhos e livros a serem ordenados.
Para a saída, haverá o retorno da média das comparações e movimentações e também a mediana do tempo de execução, além da opção de retornar os vetores gerados para o terminal ou arquivo de acordo com a opção do utilizador - veja o próximo tópico para entender como os dados de saída são formatados.

# 3. Instruções de compilação e execução

Esse programa foi editado e compilado em máquinas usando o sistema operacional Linux (variante Ubuntu versão 16.02 LTS). A compilação sempre foi feita utilizando o terminal disponível por padrão no sistema.
Para compilar esse programa, foi usado o arquivo de configuração Makefile que facilita a compilação de todos os códigos nos arquivos modularizados.
Para compilar basta digitar ***make*** na linha de comando do terminal. Para limpar os arquivos gerados pela compilação pode-se digitar o comando ***make clean***.

De acordo com o que foi sugerido pelo enunciado, o modelo para executar é o:

*./nomedoprograma <variacao> <tipo> <tamanho> [-p].*

Exemplo 1:
*./tp2 QPE Ale 50000 -p*

que vai executar o algoritmo Quicksort Primeiro Elemento em um vetor de cinquenta mil elementos e imprimir no terminal os 20 vetores aleatórios (com cinquenta mil elementos cada) que foram gerados.

Exemplo 2:
*./tp2 QNR OrdD 100000 -p > qnr-ordd-100000.txt*

que vai executar o algoritmo Quicksort Não Recursivo com vetores gerados já ordenados de forma decrescente e imprimir esses vetores no arquivo de saída *qnr-ordd-100000.txt*.

# 4. Análise experimental

O computador que rodou o código implementado de todos os quicksort foram do laboratório do CRC, com sistema operacional Linux Ubuntu 16.02, processador Intel® Core™ i7-6700 CPU @ 3.40GHz × 8, 64-bit e Memória RAM de 15.6 GiB.

Para realizar a análise dos algoritmos foi observado os três tipos de fatores de comparação dos algoritmos. O número de comparações, movimentações e o tempo de execução total dos algoritmos. Cada teste foi realizado 20 vezes e desse resultado foi extraído a média de comparações, média de movimentações e a mediana do tempo de execução. Foi verificado os três tipos básicos de ordenação inicial, se aleatório, ordenado crescente ou decrescente. E cada teste foi feito para dez tamanhos diferentes que variava de cinquenta mil a quinhentos mil elementos.

O resultado dos testes foi anotado em uma tabela e posteriormente foi plotado um gráfico de dispersão para ajudar a entender melhor os resultados. Foi decidido usar a escala logarítmica para o eixo vertical do gráfico pois o mesmo tinha uma enorme variação e era possível ver que a maioria dos dados se aglutinavam bem próximos e havia poucos pontos dispersos. Do contrário, se fosse usado a escala normal, não seria possível discernir melhor o comportamento dos dados com resultados parecidos.

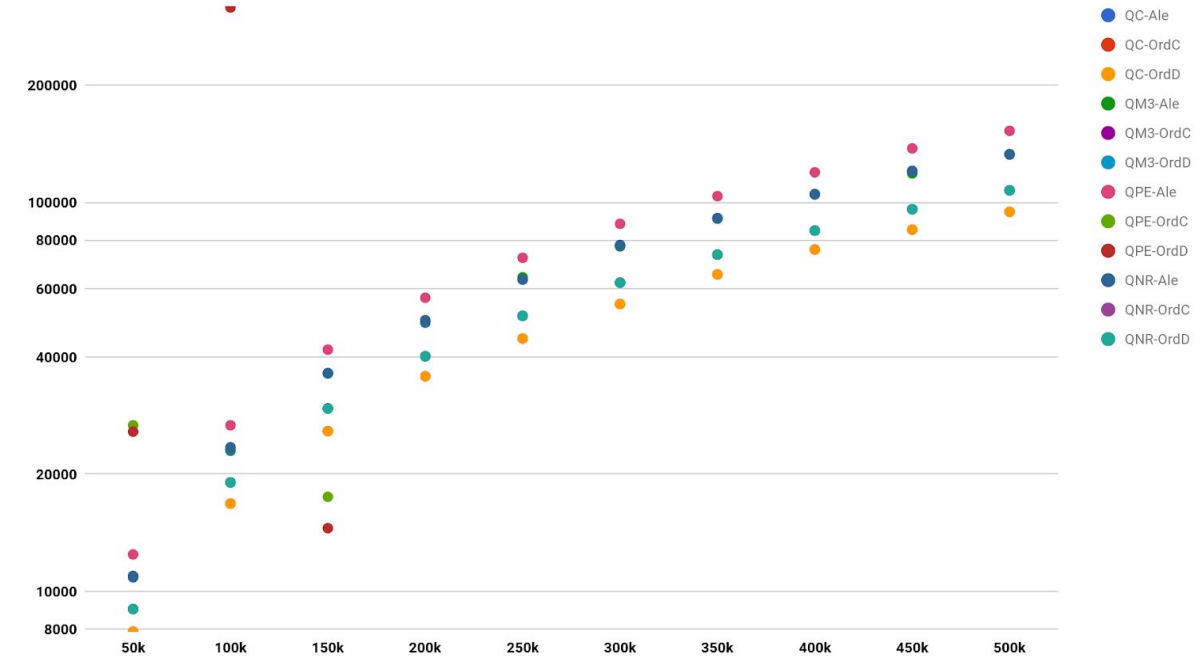
Do resultado final constata-se que o *Quicksort Clássico* obtém na maioria das vezes o melhor desempenho em relação aos demais. Também nota-se que o *Quicksort Mediana de Três* é o segundo colocado na maior parte das vezes com pouca diferença para o Clássico. Em seguida vem o *Quicksort Não Recursivo*.

Por outro lado o Quicksort Primeiro Elemento normalmente tem o pior desempenho quando os seus elementos já estão ordenados. Porém se os dados são aleatórios, o Primeiro Elemento costuma ser um pouco mais rápido que o *Mediana de Três* e o *Não Recursivo*. Em contrapartida o seu número de movimentações e comparações dos elementos (entrada aleatória) é o maior de todos.

A seguir estão as três tabelas geradas para realizar o estudo de comparação entre os algoritmos.

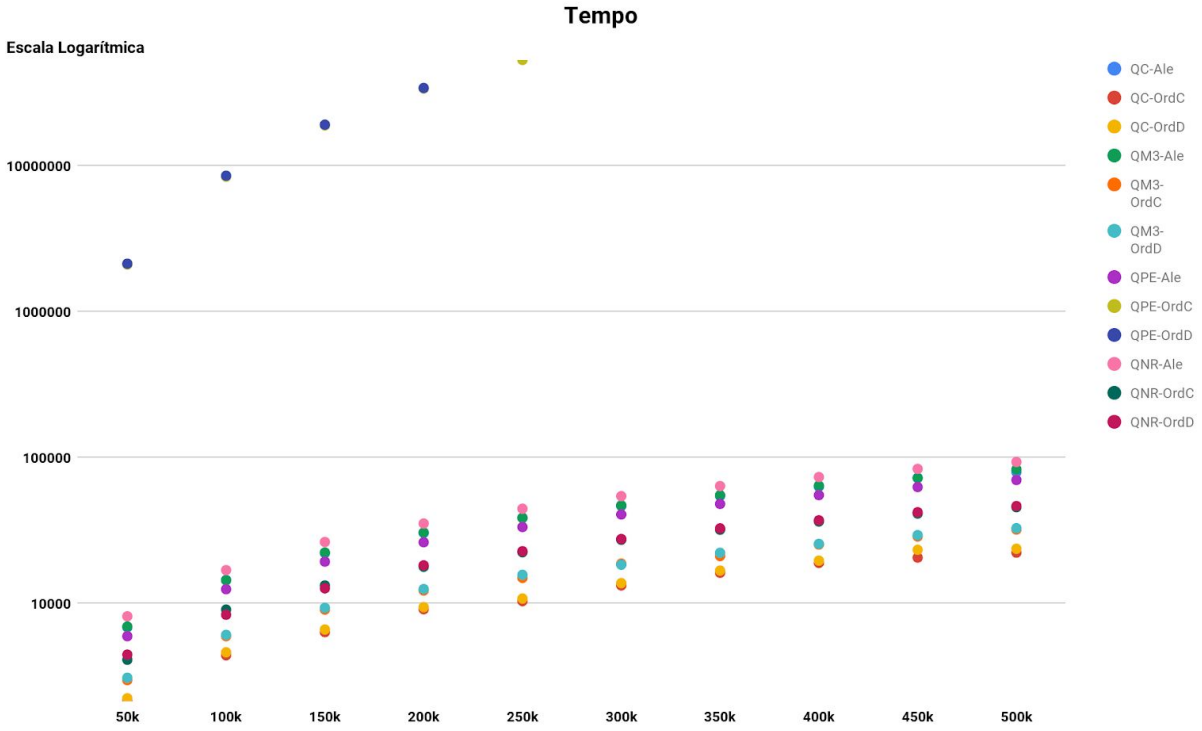
## Comparações

Escala Logarítmica




## Movimentações

Escala Logarítmica




# 5. Conclusão

Foi realizado um estudo comparativo de várias implementações do algoritmo Quicksort. Constata-se que o *Quicksort Clássico* é a melhor escolha que o reino de Arendelle pode tomar. Ele é o algoritmo mais rápido para as três variações de entradas, e na grande maioria das vezes é o que realiza menos trocas e comparações dos dados. Por fim, um algoritmo que o reino não deve usar é o *Primeiro Elemento*, que é o mais lento quando os dados já estão ordenados, e mesmo que os dados fossem aleatórios o seu desempenho ainda perde um pouco para os outros algoritmos. Dessa forma recomenda-se o *Quicksort Clássico* para o uso na biblioteca digital do reino.

Quanto aos contratempos, tive dificuldades para implementar a variante do *Quicksort Inserção*, e como o prazo estava no fim, não o inclui nas minhas análises. Também tive dificuldades na hora de transformar os dados da tabela em uma visualização gráfica de forma que possibilitasse uma comparação geral entre os algoritmos e resultasse no menor número de gráficos possível. Tive a ideia de usar um gráfico de linha temporal ou o de barras porém como havia muitos dados próximos, escolhi o gráfico de dispersão por apresentar uma melhor visualização e entendimento dos resultados. E também decidi usar escala logarítmica para que o observador veja a diferença no grupo de dados com resultados mais próximos.

# 6. Bibliografia


- slides da professora Raquel Prates:
    - ED_Aula8 - Bolha Seleção Inserção - (Apresentação dos métodos de ordenação básicos)
    - ED_Aula9-Quicksort - (Apresentação do algoritmo Quicksort)
    - ED_Aula12_ComparacaoMetodosOrd - (Comparação dos métodos de ordenação)

- https://en.wikipedia.org/wiki/Quicksort

- https://stackoverflow.com/a/51284766

- https://www.mkrevuelta.com/en/2016/02/28/7-ways-to-fail-horribly-while-implementing-quicksort/