Trabalho Prático 2: Biblioteca Digital de Arendelle

Victor Hugo Silva Moura

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

victorhugosmoura@gmail.com

Abstract. *The following text describes the "Arendelle's Digital Library"'s problem and its solution using different implementations of QuickSort algorithm. For each implementation, a series of analyses are made in order to compare their respective performances.*

Resumo. *O texto a seguir descreve o problema "Biblioteca Digital de Arendelle" e sua resolução usando diferentes implemenações do algoritmo QuickSort. Para cada implementação é feita uma série de análises a fim de comparar suas respectivas performances.*

1. Introdução

O problema *Biblioteca Digital de Arendelle* consiste em ordenar o acervo de livros e pergaminhos que o reino de Arendelle possui em suas bibliotecas físicas. Por simplificação, cada item do acervo é considerado como um número inteiro e, deste modo, o real problema é ordenar um array de inteiros. Para realizar essa ordenação existem 7 algoritmos candidatos que devem ser comparados a fim de descobrir qual é a eficiência deles para a resolução desse problema. Os algoritmos candidatos são variações do algoritmo QuickSort, sendo eles:

1. **Quicksort clássico** - Seleção de pivô usando o elemento central.
2. **Quicksort mediana de três** - Seleção do pivô usando a "mediana de três" elementos, em que o pivô é escolhido usando a mediana entre a chave mais à esquerda, a chave mais à direita e a chave central (como no algoritmo clássico).
3. **Quicksort primeiro elemento** - Seleção do pivô como sendo o primeiro elemento do subconjunto.
4. **Quicksort inserção 1%** - O processo de partição é interrompido quando o subvetor tiver menos de k = 1% chaves. A partição então deve ser ordenada usando uma implementação especial do algoritmo de ordenação por inserção, preparada para ordenar um subvetor. Seleção de pivô usando a "mediana de três" elementos, descrita acima.
5. **Quicksort inserção 5%** - Mesmo que o anterior, com k = 5%.
6. **Quicksort inserção 10%** - Mesmo que o anterior, com k = 10%.
7. **Quicksort não recursivo** - Implementação que não usa recursividade. Utiliza pilha para simular as chamadas de função recursivas e identificar os intervalos a serem ordenados a cada momento. A seleção do pivô deve ser feita assim como no Quicksort clássico.

Para cada algoritmo são feitos testes com diferentes tipos de arrays de entrada (ordenado crescente, ordenado decrescente e aleatório) e diferentes tamanhos de array com intuito de obter aproximadamente o tempo de execução, número de comparações entre chaves e número de movimentações de registro médios (mediana no caso do

tempo de execução). Posteriormente deve ser feita uma análise dos três parâmetros citados anteriormente.

2. Implementação

3. Instruções de Compilação e Execução

4. Análise Experimental

5. Conclusão

6. Referências