Universidade Federal de Alagoas
Ciência da Computação
Computação Paralela

Danillo Rodrigues Abreu
Luiz Fernando da Silva
12/01/2021

**Projeto 2 – Quicksort (Fork-join)**

**SUMÁRIO**



## 1 Objetivo

Implementar uma versão do algoritmo Quicksort utilizando Fork-join.

## 2 Quicksort

O algoritmo Quicksort foi proposto por Hoare em 1960 e publicado em 1962, considerado o algoritmo de ordenação interna mais rápido que se conhece, sendo provavelmente o mais utilizado. A ideia básica definida por esta abordagem é dividir o problema de ordenar um conjunto de $n$ itens em dois problemas menores, onde estes problemas são ordenados independentemente e os resultados obtidos são combinados para produzir a solução final (ZIVIANI, 2007).

O Quicksort rearranja os elementos de um determinado vetor $v[esq..dir]$, a partir da escolha aleatória de um pivô $x$. Posteriormente, o referido vetor é particionado em duas partes, onde a parte esquerda irá conter os elementos menores ou iguais a $x$ e a parte direita os elementos maiores ou iguais a $x$. De modo geral, o algoritmo funciona de acordo com os passos a seguir (ZIVIANI, 2007; JAIN et al., 2020):

1. Escolha aleatoriamente um pivô $x$.
2. Percorra o vetor a partir da esquerda até que $v[i] \geq x$.
3. Percorra o vetor a partir da direita até que $v[j] \leq x$.
4. Troque $v[i]$ com $v[j]$.
5. Continue este processo até que os elementos $i$ e $j$ se cruzem.

Por fim, o vetor $v[esq..dir]$ estará particionado de tal forma que os elementos em $v[esq], v[esq+1], \ldots, v[j]$ são menores ou iguais a $x$ e os elementos $v[esq]$, $v[esq+1], \ldots, v[j]$ são maiores ou iguais a $x$ (ZIVIANI, 2007; JAIN et al., 2020).

## 3 Fork-join

O Fork-join é considerado um padrão de projeto paralelo, projetado para acelerar a execução de tarefas que podem ser dividas em outras subtarefas menores, executando-as em paralelo e combinando seus resultados para alcançar um só ao final. Sendo assim, as subtarefas são independentes e aplicando o princípio de divisão e conquista, esta abordagem divide a tarefa em sub-rotinas menores até que um limite seja atingido, daí a origem do nome Fork-join, em tradução livre “bifurcar-juntar” (SOBRAL, 2018).

## 4 Sobre a implementação

Diante das diversas implementações existentes do Quicksort, para o presente trabalho e como mencionando anteriormente, a versão escolhida do algoritmo com o intuito de ser codificada foi a proposta por Hoare, uma vez que a mesma é considera a mais utilizada e a primeira a ser publicada.

Desta maneira, o algoritmo foi implementado com a linguagem de programação Java na versão 11.0.9 utilizando a classe `ForkJoinPool` implementada pela mesma. Onde, para executar as tarefas em paralelo, é utilizado um pool de threads que por padrão é igual ao número de núcleos disponíveis para a Java Virtual Machine (JVM), sendo esta a abordagem escolhida. De modo geral, cada thread tem sua própria fila de destino duplo para armazenar as tarefas que serão executadas.

Sobre a implementação, foram definidas duas classes para o projeto, a classe `QuicksortRecursiveAction` e a classe `Main`. Na primeira, encontra-se o código responsável por fazer a ordenação de um determinado vetor usando o algoritmo Quicksort, onde neste projeto é uma especificação da classe abstrata `RecursiveAction` que representa tarefas que não produzem um valor de retorno. A classe `RecursiveAction` possui o método abstrato `compute()` no qual é o responsável por executar em paralelo a tarefa de ordenação por meio dos método `partition()` que por sua vez evoca o método `swap()` todos definidos em `QuicksortRecursiveAction`.

Referente a classe `Main`, nela é definido de forma randômica um `vector` com 1000 números naturais, além de ser instanciado um objeto denominado `quicksort`, referente a classe do Quicksort que recebe como parâmetro este `vector`, e um objeto `pool` da classe `ForkJoinPool`, onde por meio do método `invoque()` é o responsável “bifurcar” e “juntar” as tarefas requeridas pela instância `quicksort`. Ao término da execução da `Main`, o programa apresenta a saída com o vetor de números randômicos desordenados, posteriormente o vetor com os elementos ordenados e o tempo gasto para a execução da tarefa.

Portanto, tendo em vista a natureza do Quicksort, com o método do Fork-join é possível tornar o processo de ordenação mais eficiente, pois, com ele é aumentado o rendimento do algoritmo, uma vez que existem muitas tarefas a serem executadas.

## 5 Referências

JAIN, P. et al. Quicksort. *GeeksforGeeks*, 2020. Disponível em: https://www.geeksforgeeks.org/quick-sort. Acesso em: 08 jan. 2021.

SOBRAL, J. B. M. Fork-join framework. *Bosco HomePage*, 2018. Disponível em: https://bit.ly/3i6LyEf. Acesso em: 08 jan. 2021.

ZIVIANI, N. *Projeto de algoritmos*: com implementações em Java e C++. São Paulo: Editora Thomson Learning, 2007.