Merge Sort com Haskell

Lucas Teixeira Gonçalves[1]

[1]Centro de Ciências Computacionais – Universidade Federal do Rio Grande (FURG)
Caixa Postal 474 – 96.201-900 – Rio Grande – RS – Brazil

lucas.teixeira@furg.br

Resumo. *Este trabalho descreve o funcionamento do algoritmo de ordenação Merge Sort na linguagem de programação funcional Haskell.*

1. O Algoritmo

O algoritmo de Merge Sort é um algoritmo de ordenação de divisão e conquista criado por John von Neumann em 1945 [Knuth 1998]. O algoritmo possui uma complexidade no pior caso de $O(nlogn)$, fazendo com que ele seja um dos algoritmos mais eficientes de ordenação.

2. Implementação em Haskell

A implementação do Merge Sort executada no trabalho é composta por duas funções principais, *merge* e *mergeSort*, onde *merge* é uma função que tem como entrada duas listas ordenadas e sua saída é a composição ordenada dessas duas listas. Já a função *mergeSort* realiza a parte de divisão e conquista do algoritmo, separando a lista em várias listas menores até um caso base.

Como Haskell é uma linguagem de programação funcional, a função é extremamente fácil de ser implementada, visto que é fortemente baseada em recursão, um dos pontos fortes da programação funcional.

```
merge :: Ord tipo => [tipo] -> [tipo] -> [tipo]
merge xs [] = xs
merge [] ys = ys
merge (x:xs) (y:ys)
  | x < y     = [x] ++ merge (y:ys) xs
  | otherwise = [y] ++ merge (x:xs) ys


mergeSort :: Ord tipo => [tipo] -> [tipo]
mergeSort [] = []
mergeSort [x] = [x]
mergeSort list = merge (mergeSort (take (length(list) `div` 2) list))
                       (mergeSort (drop (length(list) `div` 2) list))
```

O algoritmo acima funciona da seguinte maneira: *merge* é uma função que recebe duas listas de um tipo e retorna uma lista do mesmo tipo, desde que este tipo seja ordenável. Caso a função receba uma lista qualquer e uma lista vazia, retornará a lista qualquer. Se receber duas listas não-vazias, irá comparar o primeiro elemento das duas listas e retornar o menor destes elementos concatenado em uma lista com a chamada da mesma função com o resto da lista que tinha o primeiro elemento menor e a outra lista inteira.

A função *mergeSort* recebe uma lista de um tipo ordenável e retorna uma lista do mesmo tipo. o *mergeSort* de uma lista vazia é uma lista vazia, o de uma lista de um elemento apenas, é o próprio elemento. O *mergeSort* de uma lista, entretanto, é o resultado da função *merge* do *mergeSort* da primeira metade da lista com o *mergeSort* da segunda metade da lista, fazendo com que essa função seja chamada recursivamente até as listas possuam um ou zero elementos.

3. Execução

O compilador *ghci* foi utilizado para a execução do código. Primeiramente, o compilador pode ser executado do terminal com o comando *ghci*, como pode ser visto na Figura 1.

```
puf3zin@epaminondas:~/haskell$ ghci
GHCi, version 7.6.3: http://www.haskell.org/ghc/  :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
```

Figura 1. Execução do compilador GHCi.

```
Prelude> :l mergeSort.hs
[1 of 1] Compiling Main             ( mergeSort.hs, interpreted )
Ok, modules loaded: Main.
```

Figura 2. Carregamento do arquivo no compilador.

Depois, carregamos o arquivo que contém o módulo da função através do comando *:load* ou *:l*, apenas. Este comando pode ser visto na Figura 2.

Por fim, a função presente no arquivo carregado pode ser executada através do comando com o nome da função seguido por seus parâmetros, como mostrado na Figura 3.

```
*Main> mergeSort [3, 2, 9, 5, 1, 10, 15, 16]
[1,2,3,5,9,10,15,16]
```

Figura 3. Execução da função *mergeSort*.

Logo abaixo da chamada de função, haverá o retorno da função, que neste caso, já que a função *mergeSort* foi chamada na lista $[3, 2, 9, 5, 1, 10, 15, 16]$, teve como retorno a lista ordenada $[1, 2, 3, 5, 9, 10, 15, 16]$.

Referências

Knuth, D. (1998). Section 5.2. 4: Sorting by merging. *The Art of Computer Programming*, 3:158–168.