

Trabalho 2 - Analise

🕒 Created	@May 25, 2021 1:16 PM
🏷️ Tags	

Questão 1

Quanto as listas de I1 a I7, o primeiro algoritmo parece demorar um pouco mais que os outros dois. As variações 1 e 2 parecem ter um tempo bastante aproximado, não consegui diferenciar o tempo de execução entre eles.

Eles definitivamente entregam mais rápido que o original, algo que pode ser visto mais claramente em I1, no qual as variações entregam instantaneamente enquanto o original demora um pouco.

A quantidade de trocas permanece a mesma.

Questão 2

O original é definitivamente mais lento. Cheguei a deixar por uns 5 minutos cada lista entre I2 e I7 para tentar ordenar e não tive resposta, então abortei a execução. As variações não tiveram problemas com essas execuções.

O feito com foldr1 me pareceu um pouco mais rápida que a variação 1.

Quanto ao número de trocas, houveram algumas mudanças. A variação 1 teve 3999 trocas em I5 e a variação 2 apenas 2000. Também há discrepâncias nas listas x1 a x7

```

ghci> selection x1
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],0)
ghci> selection1 x1
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],0)
ghci> selectionFold x1
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],0)
ghci>
ghci> selection x2
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],19)
ghci> selection1 x2
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],19)
ghci> selectionFold x2
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],19)
ghci>
ghci> selection x3
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],10)
ghci> selection1 x3
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],19)
ghci> selectionFold x3
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],10)
ghci>
ghci> selection x4
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],18)
ghci> selection1 x4
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],18)
ghci> selectionFold x4
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],18)
ghci>
ghci> selection x5
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],10)
ghci> selection1 x5
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],19)
ghci> selectionFold x5
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],10)
ghci>
ghci> selection x6
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],13)
ghci> selection1 x6
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],18)
ghci> selectionFold x6
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],13)
ghci>
ghci> selection x7
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],19)
ghci> selection1 x7
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],19)
ghci> selectionFold x7
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],19)
ghci>

```

Questão 3

A velocidade entre o original e com o foldr parece quase idêntica, mas a variação parece ligeiramente mais rápida. O número de trocas permanece o mesmo nos dois algoritmos.

Questão 4

A variação 2 é extremamente mais lenta que as outras 2. Tentei executar em I2, mas abortei a execução quando começou a travar meu computador. Não tentei executar essa variação a

partir de l3, apenas de x1 a x7.

A variação 1 parece ser a mais rápida.

A variação 1 é a que tem menos comparações, metade do que o código original. A variação 2 tem menos comparações que o original quando a lista está ordenada, e bem mais quando não está.

```
quick l1 --3998000
quick1 l1 --1999000
quick2 l1 --2000000

quick l2 --3998000
quick1 l2 --1999000

quick l3 --3998002
quick1 l3 --1999001

quick l4 --4002000
quick1 l4 --2001000

quick l5 --8004000
quick1 l5 --4002000

quick l6 --8004000
quick1 l6 --4002000

quick l7 --8004000
quick1 l7 --4002000
```

```

ghci> quick x1
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],380)
ghci> quick1 x1
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],190)
ghci> quick2 x1
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],200)
ghci>
ghci> quick x2
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],380)
ghci> quick1 x2
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],190)
ghci> quick2 x2
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],815)
ghci>
ghci> quick x3
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],200)
ghci> quick1 x3
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],100)
ghci> quick2 x3
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],495)
ghci> quick x4
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],200)
ghci> quick1 x4
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],100)
ghci> quick2 x4
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],545)
ghci>
ghci> quick x5
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],160)
ghci> quick1 x5
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],80)
ghci> quick2 x5
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],484)
ghci>
ghci> quick x6
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],168)
ghci> quick1 x6
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],84)
ghci> quick2 x6
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],466)
ghci>
ghci> quick x7
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],162)
ghci> quick1 x7
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],81)
ghci> quick2 x7
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],544)

```


Questão 5

- Melhor versão do selectionSort → Variação 2
 - Algo estranho com o que foi pedido é que se pede o número de trocas na implementação do selection sort, e não o de comparações. Numa implementação inocente do código, esse número seria igual, já que se contaria como troca quando o valor já estivesse no devido lugar, mas eu não implementei assim.

- Essa ideia de que não precisaria contar troca quando o valor já estivesse no lugar veio de uma postagem no Stack Overflow que eu li enquanto eu pesquisava quantas trocas eu deveria esperar no pior e no melhor caso.
- Primeira resposta

What are the number of swaps required in selection sort for each case?

Each iteration of selection sort consists of scanning across the array, finding the minimum element that hasn't already been placed yet, then swapping it to the appropriate position. In a naive implementation of selection sort, this means that

 <https://stackoverflow.com/questions/26688765/what-are-the-number-of-swaps-required-in-selection-sort-for-each-case>



- Melhor versão do quickSort → Variação 1

O merge sort parece extremamente mais rápido nas listas de l1 a l7 e tem consideravelmente menos comparações que o quicksort. Ele também é mais rápido que o selection sort, que também parece mais rápido que o quicksort.

```
merge l1 -- 10864
selectionFold l1 -- 0
quick1 l1 -- 1999000

merge l2 -- 11088
selectionFold l2 -- 1999
quick1 l2 -- 1999000

merge l3 -- 10880
selectionFold l3 -- 1
quick1 l3 -- 1999001

merge l4 -- 11102
selectionFold l4 -- 1999
quick1 l4 -- 2001000

merge l5 -- 25966
selectionFold l5 -- 2000
quick1 l5 -- 4002000

merge l6 -- 25958
selectionFold l6 -- 3999
quick1 l6 -- 4002000

merge l7 -- 26190
selectionFold l7 -- 3999
quick1 l7 -- 4002000

merge x1 -- 40
selectionFold x1 -- 0
quick1 x1 -- 190

merge x2 -- 48
selectionFold x2 -- 19
quick1 x2 -- 190

merge x3 -- 40
selectionFold x3 -- 10
quick1 x3 -- 100

merge x4 -- 48
selectionFold x4 -- 18
quick1 x4 -- 100
```

```
merge x5 -- 49
selectionFold x5 -- 10
quick1 x5 -- 80

merge x6 -- 60
selectionFold x6 -- 13
quick1 x6 -- 84

merge x7 -- 65
selectionFold x7 -- 19
quick1 x7 -- 81
```

Questão 6

```
PS C:\git\FACOM\prog-funcional\trab02> ghci .\trab02.hs
GHCi, version 9.0.1: https://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Main             ( trab02.hs, interpreted )
Ok, one module loaded.
ghci> avalia (Div (Mult (Add (Val 3) (Val 12)) (Sub (Val 15) (Val 5))) (Mult (Val 1) (Val 3)))
50.0
ghci>
ghci> avalia (Sub (Val 0) (Mult (Add (Sub (Add (Val 6) (Val 8)) (Val 5)) (Val 1)) (Add (Val 2) (Div (Val 6) (Val 2)))))
-50.0
ghci> █
```