Universidade Estadual de Santa Cruz – UESC

COMPARAÇÕES DOS ALGORITMOS DE ORDENAÇÃO QUICK E MERGE EM HASKELL, C E JAVA.

Aluna: Letícia da Silva Bomfim.

Matricula: 201310461.

Disciplina Conceitos de Linguagem de Programação.

Curso Ciência da Computação

Semestre 2015.2

Professor Cesar Bravo.

Ilhéus – BA
2015

RELATÓRIO

O presente relatório mostra o desempenho e comparação dos algoritmos Quicksort e MergeSort em Linguagem Haskell, Linguagem C e Java. Abaixo contem as linhas de comando para a compilação e execução do código, e as saída para as questões requeridas.

1. **Linha de comando de compilação do projeto:**

   Linguagem C: gcc Leticia-Proj2B.c -o prog

   Linguagem Haskell: ghci merge.hs –o prog e ghci quick.hs –o prog

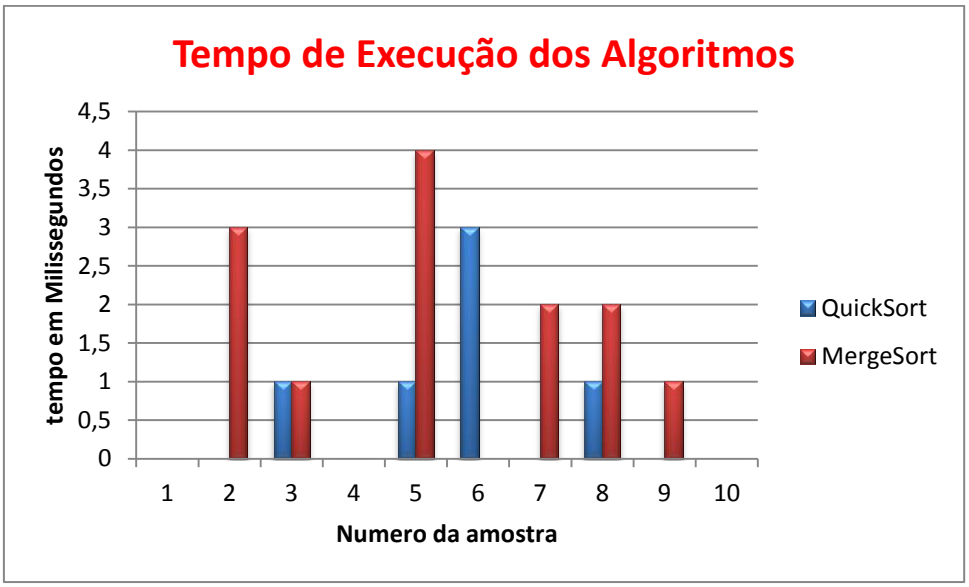
   Linguagem Java: Através da IDE.

2. **Linha de comando de execução do projeto:**

   Linguagem C: ./prog

3. **Comparação da execução de tempo entre Merge e Quick.**

   Na Linguagem de Programação Java:




```
Tempo de Processamento de QuickSort: 0ms
Tempo de Processamento de MergeSort: 0ms
Tempo de Processamento de QuickSort: 0ms
Tempo de Processamento de MergeSort: 3ms
Tempo de Processamento de QuickSort: 1ms
Tempo de Processamento de MergeSort: 1ms
Tempo de Processamento de QuickSort: 0ms
Tempo de Processamento de MergeSort: 0ms
Tempo de Processamento de QuickSort: 1ms
Tempo de Processamento de MergeSort: 4ms
```

```
Tempo de Processamento de QuickSort: 3ms
Tempo de Processamento de MergeSort: 0ms
Tempo de Processamento de QuickSort: 0ms
Tempo de Processamento de MergeSort: 2ms
Tempo de Processamento de QuickSort: 1ms
Tempo de Processamento de MergeSort: 2ms
Tempo de Processamento de QuickSort: 0ms
Tempo de Processamento de MergeSort: 1ms
Tempo de Processamento de QuickSort: 0ms
Tempo de Processamento de MergeSort: 0ms
```

Exemplo de Execução:

```
22 26 31 4 44 53 3 58 9 7 79 59 83 46 38 99 42 86 1 24 51 19 31 77 60 21 67 19 94 52 73 19 3 52 8 24 72 51 4 1 20 87 52 75 15 56 8 72 1 49
Tempo de Processamento de QuickSort: 0ms
1 1 1 3 3 4 4 7 8 8 9 15 19 19 19 20 21 22 24 24 26 31 31 38 42 44 46 49 51 51 52 52 52 53 56 58 59 60 67 72 72 73 75 77 79 83 86 87 94 99
Tempo de Processamento de MergeSort: 1ms
1 1 1 3 3 4 4 7 8 8 9 15 19 19 19 20 21 22 24 24 26 31 31 38 42 44 46 49 51 51 52 52 52 53 56 58 59 60 67 72 72 73 75 77 79 83 86 87 94 99
```

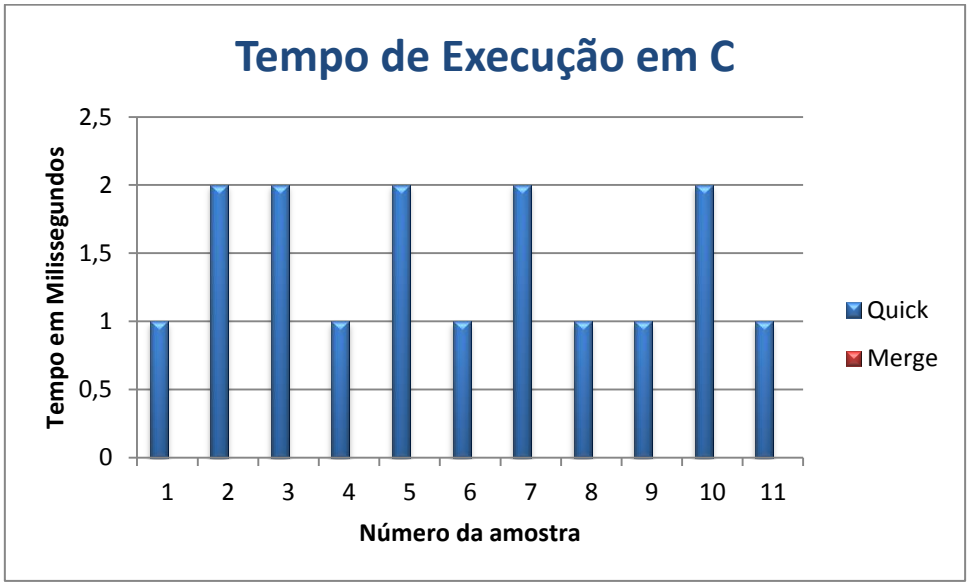
Na Linguagem de Programação Haskell:

```
*Main> quicksort [3,4,5,6,7,8,9,6,7,8,9,7,6,53,1,2,3,4]
[1,2,3,3,4,4,5,6,6,6,7,7,7,8,8,9,9,53]
*Main> mergesort [3,4,5,6,3,9,6,5,42,3,5,7,8,9,5,6,7,3,4,1,2]
[1,2,3,3,3,3,4,4,5,5,5,5,6,6,6,7,7,8,9,9,42]
```

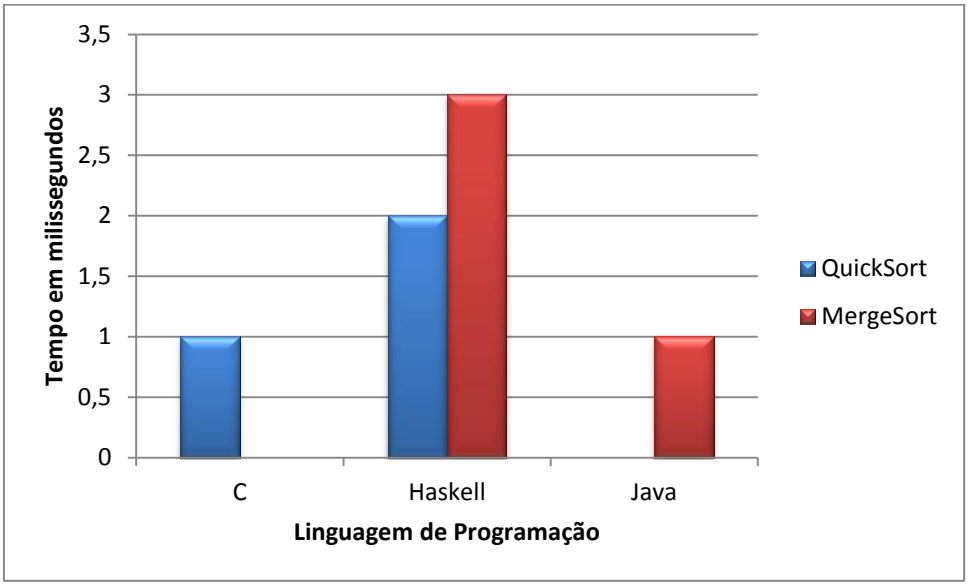
La Linguagem de Programação C:

```
Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 1.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 2.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 2.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 1.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 2.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 1.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 2.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 1.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 1.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 2.000000
```

```
Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 1.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 1.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 2.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 1.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 2.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 2.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 1.000000

Tempo de execucao mergesort = 0
Tempo de execucao quicksort = 1.000000
```



Comparação Entre C, Java e Haskell.



4. **Considerações:**

   No MergeSort o tempo independe da qualidade da entrada, já no QuickSort o tempo de execução varia de acordo com a qualidade da entrada.

Algoritmo	Tempo	Espaço	Pior Caso
QuickSort	O(nlogn)	O(nlogn)	$O(n^2)$
MergeSort	O(nlogn)	O(nlogn)	O(nlogn)

5. **Link para download do projeto 2 B:**

   https://www.dropbox.com/sh/dgregcy04owoqey/AAAJyS_Malc400PdA002AcVda?dl=0