Segundo Trabalho de Programação Funcional – 2020/1

Data de entrega: 25/05/2021 (até 23:55)

Valor: 22 pontos

Obs1: Esse trabalho pode ser desenvolvido em **Dupla** ou **Individual,** mas a arguição na apresentação e a respectiva nota será individual.

Obs2: A nota será dada pela nota de desenvolvimento (de 0 a 20) multiplicada pela nota de arguição (de 0 a 1), onde o aluno deverá demonstrar completo entendimento do trabalho desenvolvido.

PARTE A – Algoritmos de ordenação

Para os exercícios de ordenação, considere as 14 listas a seguir como exemplos para teste dos diversos algoritmos:

```
l1=[1..2000]

l2=[2000,1999..1]

l3=l1++[0]

l4=[0]++l2

l5=l1++[0]++l2

l6=l2++[0]++l1

l7=l2++[0]++l2

x1=[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

x2=[20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1]

x3=[11,12,13,14,15,16,17,18,19,20,1,2,3,4,5,6,7,8,9,10]

x4=[10,9,8,7,6,5,4,3,2,1,20,19,18,17,16,15,14,13,12,11]

x5=[11,12,13,14,15,5,4,3,2,1,16,17,18,19,20,10,9,8,7,6]

x6=[1,12,3,14,5,15,4,13,2,11,6,17,8,19,20,10,9,18,7,16]

x7 = [20,8,2,11,13,3,7,18,14,4,16,10,15,1,9,17,19,12,5,6]
```

Exercício 1) Em relação ao algoritmo de ordenação **Bolha,** faça as implementações em Haskell solicitadas abaixo**. Em todas as variações abaixo, incluindo o código original, você deve incluir um contador de trocas realizadas durante a ordenação.**

- Original Bolha: implementação do código visto em aula.

- Variação 1: parada do algoritmo é antecipada quando uma iteração de trocas é finalizada sem que nenhuma troca efetiva seja realizada na iteração completa.

- Variação 2: faz parada antecipada e, a cada iteração de trocas, a avaliação é realizada desconsiderando-se o último elemento cuja posição foi fixada. Ou seja, diminui o tamanho da lista a ser ordenada a cada iteração.

- Realizar execuções comparativas nas listas dadas como exemplo entre os algoritmos das variações e do algoritmo original para avaliar: 1) o número de **trocas** em cada execução; 2) o tempo de execução (apenas verifiquem se existe uma mudança aparente de tempo de processamento). Apresentar esses resultados em um arquivo PDF, fazendo uma análise dos resultados encontrados.

Exercício 2) Em relação ao algoritmo de ordenação **Selecao,** faça as implementações em Haskell solicitadas abaixo. **Em todas as variações abaixo, incluindo o código original, você deve incluir um contador de trocas realizadas durante a ordenação.**

- Original Seleção: implementação do código visto em aula.

- Variação1: Refaça o código original para que a busca pelo menor elemento (função mínimo) e a eliminação desse menor elemento da lista a ser ordenada (função remove) ocorra numa mesma função (remove_menor), sem a necessidade de se percorrer a lista duas vezes a cada iteração.

- Variação 2: Refaça a implementação do algoritmo Seleção usando funções genéricas (foldr ou foldr1) .

- Realizar execuções comparativas nas listas dadas como exemplo entre os algoritmos das variações e do algoritmo original para avaliar: 1) o número de **trocas** em cada execução; 2) o tempo de execução (apenas verifiquem se existe uma mudança aparente de tempo de processamento). Apresentar esses resultados em um arquivo PDF, fazendo uma análise dos resultados encontrados.

Exercício 3) Em relação ao algoritmo de ordenação **Inserção,** faça as implementações em Haskell solicitadas abaixo. **Em todas as variações abaixo, incluindo o código original, você deve incluir um contador de comparações realizadas durante a ordenação.**

- Original Inserção: implementação do código visto em aula.

- Variação 1: Refaça a implementação do algoritmo Inserção usando funções genéricas (foldr ou foldr1).

- Realizar execuções comparativas nas listas dadas como exemplo entre os algoritmos das variações e do algoritmo original para avaliar: 1) o número de **trocas** em cada execução; 2) o tempo de execução (apenas verifiquem se existe uma mudança aparente de tempo de

processamento). Apresentar esses resultados em um arquivo PDF, fazendo uma análise dos resultados encontrados.

Exercício 4) Em relação ao algoritmo de ordenação **quicksort** visto em sala de aula e laboratório. **Em todas as variações abaixo, incluindo o código original, você deve incluir um contador de comparações realizadas durante a ordenação.**

- Original Quicksort: implementação do código visto em aula.

- Variação 1: modifique o algoritmo original para que ao invés dos elementos maiores e menores serem encontrados com buscas independentes, que seja elaborada e utilizada a função **divide** que percorre a lista uma única vez, retornando os elementos menores em uma lista e os elementos menores em outra.

EX: > divide 'j' "pindamonhangaba" Resposta: ("idahagaba","pnmonn")

- Variação 2: modifique a variação 1 para que o elemento pivô seja obtido a partir da análise dos 3 primeiros elementos da lista, sendo que o pivô será o elemento mediano entre eles. Exemplo: na lista [3, 9, 4, 7, 8, 1, 2], os elementos 3, 9 e 4 seriam analisados e o pivô escolhido seria 4. Caso a lista a ser analisada tenha menos que 3 elementos, o pivô é sempre o primeiro.

- Realizar execuções comparativas nas listas dadas como exemplo entre os algoritmos da Variação 1, 2 e do algoritmo original para avaliar: 1) o número de comparações em cada execução; 2) o tempo de execução (apenas verifiquem se existe uma mudança aparente de tempo de processamento). Apresentar esses resultados em um arquivo PDF, fazendo uma análise dos resultados encontrados.

Exercício 5)

- Pesquise e implemente em Haskell o algoritmo mergesort, incluindo um contador de comparações elementares realizadas durante a ordenação.

- Realizar execuções comparativas nas listas dadas como exemplo entre os algoritmos selecionados como as melhores versões do Seleção e Quicksort, além do algoritmo Mergesort, para avaliar: 1) o número de comparações em cada execução; 2) o tempo de execução (apenas verifiquem se existe uma mudança aparente de tempo de processamento). Apresentar esses resultados em um arquivo PDF, fazendo uma análise dos resultados encontrados.

Obs: as análises solicitadas nos exercícios 1, 2, 3, 4 e 5, devem ser entregues em um único arquivo PDF, que deve ser enviado junto com os códigos dos exercícios.

PARTE B – Tipos Algébricos e Ávores