Universidade Federal do Piauí –
Campus Senador Helvídio Nunes de Barros – CSHNB
Curso de Sistemas de Informação
Disciplina: Programação Funcional - Online
Professora: Juliana Oliveira de Carvalho

Análise de Algoritmos feitos na linguagem Haskell

Ericksulino manoel de Araújo Moura

1. Resumo do Projeto

Projeto composto por cinco exercícios, onde devem ser colocados a prova os conhecimentos adquiridos em toda a matéria, os mesmos contêm problemas de listas , que necessitam ser resolvidos de forma eficaz, utilizando funções de alta ordem.

2. Introdução

Projeto realizado com o intuito de completar a segunda nota da disciplina de programação funcional, com cinco exercícios com problemas complexos, envolvendo listas de números, de strings, entre outros. Para serem resolvidas em forma de código fonte, utilizando o paradigma funcional e a linguagem de programação haskell,

Para a realização desse projeto foram necessários vários métodos de manipulação e organização de listas e o mesmo para arquivos também.

Todos os algoritmos foram estruturados por uma função principal onde era feito uma interação com o usuário e funções auxiliares para a realização das tarefas exigidas pelos exercícios, para uma melhor organização dos mesmos.

3. Seções Específicas

a. Informações técnicas

Para o desenvolvimento e testes deste projeto foi utilizado uma máquina equipada com um processador quad-core, dois núcleos e quatro threads, com a velocidade em turbo boost chegando a 3.1Ghz, oito gigas de memoria ram ddr4, sistema operacional com a arquitetura de 64 bits, Linux Ubuntu na versão 20.04Lts. Os códigos foram feitos na ide e editor de texto Visual Studio Code e compilados pelo compilador GHCI versão 8.6.5. Ainda para a resolução dos exercícios foi necessário importar duas bibliotecas, sendo elas a “Data.Char” sendo utilizada apenas no primeiro exercício, e a “Data.List” sendo utilizada em todos os exercícios inclusive no primeiro.

b. Exercício 1

O exercício é pedido para ser feito um programa em Haskell que leia uma lista de strings e então execute essas três ações: uma função para contar o número de caracteres que cada string possui sem repetir; uma função que devolve uma lista contendo os tipos de caracteres que iniciam as

strings da lista, por exemplo: vogal, dígito ou outro tipo de carácter e se o mesmo é ou não maiúsculo, quando possível. uma função que devolva a string que possui o maior número de vogais.

Para contar os caracteres sem repetir, foi somente necessário uma função, na função é usado composição de função, além da utilização das funções do próprio Haskell. A função “sort” recebe uma string e a retorna em ordem alfabética, o retorno de “sort” é utilizado por “group” onde as letras repetidas são agrupadas e retornadas em forma de lista. Assim o “length” conta o tamanho de cada lista retornada por “group”. Por fim, a função “fmap” garante que o método será aplicado a todas as strings da lista.

Para determinar os tipos de caracteres que iniciavam cada string foram necessárias três funções auxiliares sendo elas: uma função que recebe um caractere e compara o mesmo se é igual a uma das cinco vogais existentes se sim é retornado um valor booleano True e caso contário era retornado False; Outra função com o funcionamento igual a anterior, com um caractere como entrada e um valor booleano como saída, mudando apenas que ao invés de vogais é comparado com caracteres especiais; Na última função auxiliar também tem como entrada um caractere, para fazer a verificação esse mesmo caractere foi utilizado como entrada na função para comparar vogais já aqui citada anteriormente se o resultado da mesma for falso então é retornado um True e caso contrario é retornado um False.

A função principal referente às funções citadas no parágrafo anterior, tem um funcionamento bem simples, ela recebe a lista e em cada caso de verificação é pegado a primeira letra da string por meio da função head, com isso, a função está dividida em quatro casos de teste, são eles respectivamente: comparação se é vogal, caractere,consoante e outro caso não entre nos anteriores, em cada um deles é chamada uma função das descritas no parágrafo anterior, com a primeira letra da string em questão, e em cada caso é colocado em uma lista uma string contendo o tipo de dado e em seguida chamando novamente a mesma função de forma recursiva com o resto da lista como parâmetro. Ao final de tudo é retornada a lista completa de strings em que cada uma é o tipo de dado sendo eles em questão: vogal, caractere, consoante e outro.

A última alternativa é composta por duas funções auxiliares e uma principal, sendo a uma das auxiliares a função onde as vogais de cada string da lista é contada de modo recursivo. O seu retorno é usado na função principal junto com um parâmetro de início padrão uma variável denominada de “maior” e ela sempre começa por zero. A principal compara a primeira string com “maior” e acrescenta a uma lista, de modo recursivo. Assim a função faz uma lista das strings com maior número de vogal. A outra função auxiliar é chamada para pegar a última string da lista retornada por pela principal.

A função principal “main” é usada para exibir na tela do usuário todos os elementos.

c. Exercício 2

O objetivo desse exercício é fazer um programa em Haskell que leia duas listas de inteiros ordenadas, A e B, e então executa duas funções sendo elas: Uma função que devolve uma lista contendo a união ordenada entre elementos que tem em A e não tem em B e os elementos que têm em B e não tem em A; Uma função que devolve uma lista contendo a soma entre os quadrados dos elementos das duas listas que forem maiores do que a soma entre o cubo dos dois primeiros elementos da lista.

Na primeira alternativa é criada duas funções que basicamente recebem listas de inteiros retornam A – B e B – A. Utilizando uma função da biblioteca Data.List "\\" que realiza a diferença entre duas listas, e a diferença de uma função para a outra é só a ordem das listas, na primeira é A – B e na segunda é B – A.

A segunda alternativa é composta por quatro funções auxiliares e uma principal, sendo elas : "prim","maiores","elev" e "soma". Na função "prim" recebe uma lista, com o "take 2" pega os dois primeiros elementos da mesma, com o "map(^3)" esses dois elementos são elevados ao cubo e com o "sum" é feito a soma de tudo. Na função "maiores" recebe uma lista, com o "filter>prim" filtra na lista os maiores de que o retorno da função "prim" e por final retorna essa lista resultante. A função "soma" basicamente pega duas listas e vai somando os elementos das individualmente, a soma do primeiro de uma com o primeiro da outra e assim sucessivamente. Por fim a função "letraB" recebe as duas listas, chama a função "soma" que faz a soma da primeira lista que é o resultado de a função elev com a primeira lista como parâmetro e a mesma coisa com a segunda.

Ainda tem uma função para fazer uma interação com o usuário, onde tem duas listas como parâmetros e em forma de mensagens de texto é perguntado se o mesmo quer usar essas listas na primeira ou na segunda alternativa, tendo escolhido, a função escolhida é chamada com as duas listas e o resultado é printado na tela, é possivel fazer isso ate que seja digitado 0 pelo usuário, nesse caso é mostrada uma mensagem que o programa foi finalizado e o mesmo é encerrado.

d. Exercício 3

Nesse exercício é pedido um código fonte que leia uma lista de times de futebol contendo nome do clube, estado e país a qual pertence e ano de fundação do clube. Faça uma função que ordene a lista pelo campo nome do clube usando o Quicksort, e depois possibilite ao usuário ver toda a lista e permita também o usuário buscar informações de um clube pelo nome do clube.Para a resolução desse exercício que foram necessárias quatro funções auxiliares, e a função principal chamada somente de main.

Antes de tudo tipos para, os dados do time exigidos pela questão, e ainda um tipo denominado de "Time" que é uma lista que contém os tipos antes mencionados.

Foi criada uma função para o cadastro dos dados do time onde a mesma colocava esses dados em uma lista do tipo “Time” e a retornava, cada dado antes de ser colocado na lista tinha uma mensagem informando ao usuário qual era o dado que ele deveria digitar, de uma forma bem simples e intuitiva.

Como foi pedido na questão foi criado uma função chamada de “quicksort” cujo como o nome mesmo já diz, é a já conhecida função de ordenação quicksort, com o seu simples funcionamento de receber uma lista depois retorná la ordenada.

Uma função de busca também foi criada, a mesma recebe uma lista de listas e uma string que é o nome do time, esse nome é comparado com os nomes dos times presentes nas listas e quando os dois são iguais a lista contendo as informações do mesmo é retornada.

Uma simples função denominada apenas por “buscar” foi criada no simples intuito de receber uma lista e printa-la na tela do usuário.

Função principal ou simplesmente “main”, é a responsável por tornar o código mais atrativo para o usuário, trazendo mais interações com o mesmo, ela é bem simples, aparece uma mensagem de texto com as opções e pedindo para o usuário digitar a opção em que ele deseja até que seja digitado um “0” assim encerrando o código com uma mensagem de texto informando que o mesmo foi encerrado, cada opção chama uma das funções já mencionadas, e também são exibidas mensagens de texto ao usuário.

E ainda tem uma função chamada de “iniciar” que tem com a sua única função chamar a função “main”.

e. Exercício 4

Aqui é pedido um algoritmo que é basicamente um sistema escolar, possibilitando o usuário fazer o cadastro de notas, alunos, disciplinas e cursos, ainda possibilitando a visualização dos mesmos e a pesquisa por meio de códigos e fazer isso quantas vezes o usuário desejar. Como no exercício anterior, aqui também foram necessárias várias funções auxiliares muitas delas com o mesmo funcionamento das do algoritmo anterior mudando somente poucas coisas para se adaptar ao que é pedido nessa questão e a função principal, nomeada somente por “main”.

Assim como no anterior foram criados tipos para os dados e dessa vez quatro tipos de listas, que continham os tipos dos dados anteriormente tipados.

Uma função para receber uma lista e printar na tela do usuário com o mesmo funcionamento da criada no exercício anterior desta vez denominada de “visualizar”.

Exatamente cinco funções de busca, iguais às do exercício anterior, sendo apenas adaptadas para esse, todas recebem uma lista, algumas recebem só um parâmetro para a busca e outras recebem dois, todas também retornam listas de acordo com o que foi pesquisado.

Quatro funções para fazerem cadastro dos dados, armazenando-os em listas dos tipos anteriormente criados, com o funcionamento igual mudando apenas algumas coisas de acordo com o dado, sendo as funções

respectivamente nomeadas por “cadastro_curso” , “cadastro_disc” , “cadastro_alunos” e “cadastro_notas”.

Depois disso existe uma função chamada de “menu” que é basicamente o como o próprio nome diz, um menu contendo todas as opções numeradas e com mensagens com breve descrição sobre as mesmas no final é pedido para o usuário digitar o número correspondente a opção escolhida, o mesmo é armazenado e retornado pela função.

Por fim, a função “main”,, primeiramente a função menu é chamada e um variável recebe o retorno, isso serve para ter uma interação maior com o usuário, o mesmo é instruído a digitar um número ou zero para finalizar o programa , o número digitado é usado para executar uma ação que corresponda com o mesmo, para cada opção escolhida é mostrada uma mensagem antes e depois de ser chamada uma das funções citadas anteriormente. Assim como no exercício anterior, aqui também existe uma função “iniciar” que chama a função “main” dando início ao loop que só vai finalizar quando for digitado 0 pelo usuário.

f. Exercício 5

Nesse exercício é pedido uma solução para a questão anterior utilizando arquivos. Todo o funcionamento e as funções exceto as que envolvem a manipulação de arquivos são iguais às do exercício anterior, e também foram adicionados mais duas opções no menu do usuário, sendo elas, escrever os dados em um arquivo e visualizar um arquivo respectivamente. Tendo isso em vista aqui vão ser abordadas somente as funções que são exclusivas desse exercício, após a outras já foram muito bem abordadas no tópico anterior.

Função “escrever”, a mesma recebe como parâmetros quatro listas de listas que são onde devem estar os dados escolares e uma string que deve ser o nome e a extensão do arquivo. Primeiro o arquivo é criado, ficando no modo de escrita com o “WriteMode” e nomeado pela variável que é recebida como parâmetro o mesmo é atribuído várias mensagens de texto com o intuito de dar um melhor efeito visual, em seguida as listas de listas são adicionadas no arquivo com algumas mensagens de texto informando o tipo de dado de cada uma antes delas, quando todas elas são adicionadas no arquivo é exibido uma mensagem para o usuário indicando que o arquivo foi escrito com sucesso e depois o “hFlush” faz com que todos os itens armazenados em buffer para saída em handle hdl sejam enviados imediatamente para o sistema operacional, o mesmo é fechado em seguida com o “hClose”.

Na Função “ler”, como o nome diz é onde é possível visualizar qualquer arquivo de texto que esteja na mesma pasta que esse código, ela tem como o único parâmetro uma string, a mesma é usada com o “readFile” para abrir um arquivo e depois o mesmo é impresso na tela do usuário.

Como já foi explicado anteriormente algumas funções são iguais às do exercício anterior, função “main” é quase igual, mudando apenas duas opções do menu de usuário que já foram nomeadas aqui neste tópico, na opção de escrita de arquivo, antes de tudo é perguntado ao usuário se ele quer mesmo criar um arquivo com esses dados já cadastrados, se sim é

pedido para o mesmo dar um nome para esse arquivo, depois a função “escrever” é chamada com os dados e o nome do arquivo escolhido pelo usuário, caso o mesmo escolha que não é exibido uma mensagem e então o mesmo é redirecionado para o menu principal, como nas outras opções também. Na opção de leitura de arquivo é simplesmente pedido para o usuário digitar o nome do arquivo que ele deseja visualizar, e a função “ler” é chamada com esse nome como parâmetro.

4. Resultados da Execução do Programa

Dados os códigos feitos para esse projeto, esses são os resultados de alguns testes feitos nos mesmos, será mostrado a entrada e a saída em uma tabela abaixo da descrição de cada código, os números das entradas foram escolhidos de forma aleatória, por mim quando fiz os testes.

Exercício número um é um algoritmo que lê uma lista de strings e então devolve uma lista contendo número de caracteres que cada string possui sem repetir, na letra “a”, uma lista contendo os tipos de caracteres que iniciam as strings da lista, na letra “b” e a string que possui o maior número de vogais, na letra “c”.

Entrada	Saída
["maca","uva","pera","banana","abacaxi","aeiou","!salada"]	**a)** **[3,3,4,3,5,5,5]** **b)** **["consoante","vogal","consoante","consoante","vogal","vogal","caractere"]** **c) "aeiou"**
["maca","!va","Pera","rock"]	**a)** **[3,3,4,4]** **b)** **["consoante","caractere","consoante","consoante"]** **c) "maca"**

Exercício dois a entrada contém duas listas de inteiros ordenados e a saída uma lista contendo a união ordenada entre (A – B) e (B – A) na letra “a”, e uma lista contendo a soma entre os quadrados dos elementos das duas listas que forem maiores do que a soma entre o cubo dos dois primeiros elementos da lista, na letra “b”.

Entrada	Saída

[1,4,5,6,10]
e
[1,2,3,5,10]
	`a) [4,6] [2,3]`
	
	`b)[100]`

[1,4,5,6,11,50,1690] **e** **[1,2,3,5,8,10,37,653]**	`a) [4,6,11,50,1690] e [2,3,8,10,37,653]` `b)[2856200,1369,426409]`

No exercício três é pedido um algoritmo que leia uma lista de times de futebol contendo nome do clube, estado e país a qual pertence e ano de fundação do clube e de acordo com a escolha exiba uma lista.

Entrada	Saída
Opção c (cadastro) **Flamengo,RJ,Brasil,1895** **Palmeiras,SP,Brasil,1914** **Corinthians,SP,Brasil,** **1910** **River,PI,Brasil,1946**	**Opção o (lista ordenada)** **[(“Corinthians”,”SP”,”Brasil”,** **1910),(“Flamengo”,”RJ”,”Brasil”,** **1895),(“Palmeiras”,”SP”,”Brasil”,** **1914),(“River”,”PI”,”Brasil”,1946)]**
*** Com os mesmos dados cadastrados anteriormente.**	**[(“Flamengo”,”RJ”,”Brasil”,**

Opção b (Busca) **Flamengo**	**1895)]**
*** Com os mesmos dados cadastrados anteriormente.** **Opção l (Lista dos times)**	**("Corinthians","SP","Brasil",1910)** **("Flamengo","RJ","Brasil", 1895)** **("Flamengo","RJ","Brasil", 1895)** **("Palmeiras","SP","Brasil" ,1914)** **("River","Pi","Brasil",1946)**

Exercício quatro é pedido um programa em Haskell, que cadastre dados de uma universidade e de acordo com a opção realize uma ação que resulte na exibição de uma lista.

Entrada	Saída
Opção 2 (cadastro de curso) **Código do curso – 702** **Nome – Física** **Q. de períodos – 8**	**Opção 7(Ver cursos)** **Cursos ofertados:** **(702,"fisica",8)**
Opção 3 (cadastro de disciplinas) **Código da disciplina – 7021** **Código do curso – 702** **Nome – Cálculo**	**Opção 8 (Disciplinas por curso)** **Código do curso - 702** **(7021,702,"Cálculo",1)** **(7022,702,"Mecânica",1)**

Período – 1 **Código da disciplina – 7022** **Código do curso – 702** **Nome – Mecânica** **Período – 1**	
Opção 1 (cadastro de alunos) **Código de matrícula - 1237** **Nome do aluno – Alberto** **Código de curso – 702** **Período - 1**	**Opção 5 (Alunos por curso)** **Código do curso - 702** **(1237,"Alberto",702,1)**

Exercício cinco é igual ao anterior mudando apenas que agora as informações podem ser armazenadas em um arquivo..

Entrada	Saída
Opção 2 (cadastro de curso) **Código do curso – 702** **Nome – Física** **Q. de períodos – 8**	**Opção 11 (armazenar em um arquivo) arquivo.txt** **Opção 12 (visualizar um arquivo) arquivo.txt** **----------Dados escolares----------** **Cursos:** **(702,"fisica",8)** **Disciplinas:** **Alunos:**

	Notas:
***com os dados anteriores já cadastrados** **Opção 3 (cadastro de disciplinas)** **Código da disciplina – 7021** **Código do curso – 702** **Nome – Cálculo**	**Opção 11 (armazenar em um arquivo) arquivo.txt** **Opção 12 (visualizar um arquivo) arquivo.txt** **----------Dados escolares----------** **Cursos:** **(702,"fisica",8)** **Disciplinas:** **(7021,702,"Cálculo",1)** **Alunos:** **Notas:**

5. Conclusão

Tendo em vista o exposto, este trabalho foi realizado para complementar a segunda nota da disciplina, como já foi apresentado anteriormente, e também como um meio de exercitar os ensinamentos aprendidos em aula, colocando os em prática, com esses exercícios. O objetivo foi alcançado completamente, ficou bem claro o modo de utilizar os paradigmas e métodos, pelos quais a professora nos ensinou.

Foram utilizados também as funções de alta ordem, que são algumas soluções do próprio haskell para o auxílio dos programadores, o que foi sim de muita ajuda, tendo em vista que diminuía o código e também era bem mais eficiente do que fazê las manualmente.

Com esse trabalho ficou bem claro como é o funcionamento do paradigma funcional e da linguagem haskell já que para a resolução do mesmo, foram necessárias a utilização de variadas funções e técnicas presentes na linguagem, reforçando e aumentando assim o meu conhecimento sobre programação.