NOVA School of science & Technology

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

Algoritmos e Estruturas de Dados

BdFI (Base de Dados de Filmes na Internet)

Ano Letivo 2021/2022

Guilherme Santana MIEI nº 60182
Pedro Fernandes MIEI nº 60694

Índice

Descrição dos TADs

TAD BDFI

Esta classe possui todos os métodos necessários para interagir com a base de dados. Nesta classe foram implementadas 4 variáveis distintas onde foram usadas 3 EDs diferentes como *SepChainHashTable, BinarySearchTree e OrderedDoubleList.*

Tanto a variável que guarda os programas – *shows* – como a variável que guarda os profissionais – *people* – foram guardadas numa *SepChainHashTable* (TAD *Dictionary*) com o intuito de diminuir a complexidade temporal dos métodos onde estas estão envolvidas. São utilizadas da seguinte forma, depois de serem adicionados ao sistema pelos métodos *addPerson* e *addShow*:

- Na retorna de qualquer informação que os programas e os profissionais nos métodos: *infoShow*, *infoPerson*, *listShowsPerson*, *listParticipations*
- Na eliminação de programas do sistema (*removeShow*).
- Adicionar um participante, uma palavra-chave e uma avaliação (*addParticipation*, *addTag* e *addRate* respetivamente).

Para guardar os programas já avaliados decidimos guardá-los num vetor de *OrderedDoubleList* (TAD *OrderedDictionary*). Inicialmente os dados estão guardados de acordo com o seu rating e depois são ordenados alfabeticamente, de acordo com o seu título diminuindo assim a complexidade temporal na procura do programa tanto na adição de um novo rating como na sua iteração durante os métodos *listBestShows* e *listShows*.

Em relação as palavras-chaves, ou tags, dos programas, à semelhança dos programas avaliados, implementou se uma estrutura *OrderedDoubleList (*TAD *OrderedDictionary*) dentro de uma *SepChainHashTable (*TAD *Dictionary)*. Primeiro, no método *addTag*, é guardado o valor da tag, independente da sua capitalização, e só depois são guardados os programas ordenados alfabeticamente, por ordem de título facilitando assim a procura das tags e posteriormente a sua iteração no método *listTaggedShows*.

TAD *Show*

A classe *Show* guarda a informação de cada programa, como o seu id, o seu título, o seu ano de produção, a sua avaliação e as palavras-chaves, ou tags, correspondentes e os seus participantes no filme. As tags são guardadas numa estrutura *DoubleList*(TAD *List*). Apesar de consumir mais tempo durante a procura da tag esta será compensada durante a sua iteração.

Para guardar os participantes usamos duas estruturas de dados diferentes. A variável participants, implementada com uma DoubleList (TAD List) é utilizada para a iteração dos participantes (*listParticipants*) num determinado show enquanto que a variável *people*, implementada com uma *SepChainHashTable* (TAD *Dictionary*), é utilizada quando tivermos a necessidade de remover o programa de todos os participantes (*removePeople*).

TAD Person

Nesta classe, para além das variáveis associadas à informação do profissional, como o seu id, o seu nome, o seu ano de nascimento, o seu género, o seu email e o seu telefone, para guardar os programas que cada profissional está inserido utilizou se uma *OrderedDoubleList* (TAD *OrderedDictionary*) para facilitar a sua iteração (*listParticipants*).

TAD Participant

Esta classe é especifica para um participante num determinado programa. Ela contém apenas um profissional – objeto *Person* – e a descrição que este tem num determinado programa.

Seria possível implementar as nossas variáveis com outras Estruturas de Dados mas avaliando toda a complexidade temporal dos método em que estas estavam envolvidas decidimos optar por estas.

Complexidade temporal

p – número de pessoas

s – número de programas (shows)

t – número de tags num programa

q – número de participantes num programa

Operação	Melhor caso	Caso esperado	Pior caso
addPerson	O(1)	O(1)	O(p)
addShow	O(1)	O(1)	O(s)
addParticipant	O(1)	O(1)	O(p + s)
premiere	O(1)	O(1)	O(s)
removeShow	O(s)	O(p + s + t)	O(p + s + t)
tagShow	O(1)	O(s)	O(t + s)
infoShow	O(1)	O(1)	O(s)
rateShow	O(1)	O(s)	O(s)
infoPerson	O(1)	O(1)	O(p)
listShowsPerson	O(1)	O(s)	O(p + s)
listParticipations	O(1)	O(q)	O(s + q)
listBestShows	O(1)	O(s)	O(s)
listShows	O(1)	O(s)	O(s)
listTaggedShows	O(1)	O(s)	O(s)
quit	O(p + s + t + q)	O(p + s + t + q)	O(p + s + t + q)

Complexidade espacial

p – número de pessoas

s – número de programas (shows)

t – número de tags num programa

q – número de participantes num programa

p * (O(p) + O(s)) corresponde à complexidade espacial da variável *people* na *BDFIClass*

s * (O(s) + O(t) + O(q)) corresponde à complexidade espacial da variável *shows* na *BDFIClass*

s é a complexidade espacial da variável *ratedShows* na *BDFIClass*

t * s é a complexidade espacial da variável *taggedShows* na *BDFIClass*

Assim a complexidade espacial da variável bdfi na main é

p * (O(p) + O(s)) + s * (O(s) + O(t) + O(q) + 1 + t)