

UNIVERSIDADE FEDERAL RURAL DE PERNAMBUCO

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

JOÃO VICTOR MARQUES DOS SANTOS

LUIZ CARLOS BRITO DE ANDRADE LIMA FILHO

VINICIUS PAES

RELATÓRIO DO PROJETO DE ALGORITMOS E ESTRUTURA DE DADOS

**RECIFE 2017**

Para nosso projeto de AED escolhemos a Biblioteca, pois parecia a opção que conseguimos visualizar com mais facilidade todo o projeto, mas antes da explicação de como fizemos o projeto, só que antes uma pequena explicação sobre nossa árvore.

Para que houvesse melhor aproveitamento para o grupo, nós não dividimos partes das árvores para cada fazer e sim cada um fez a sua e os outros integrantes olhavam para no final decidir qual delas seria melhor para o uso e lucramos com melhor conhecimento e ajudamos uns aos outros em alguns bugs que apareceram inicialmente.

Utilizamos a árvore red-black, que é uma árvore binária que tem um extra bit no armazenamento por nó que é sua cor, que pode ser vermelha ou preto. Esse tipo de árvore restringe as cores dos nós em qualquer percurso, logo as árvores red-black garantem que nenhum caminho é duas vezes maior que qualquer outro caminho, por isso a arvore é aproximadamente balanceada.

As árvores red-black são árvores binárias que satisfazem as seguintes propriedades red-black:

1. Cada nó ou é vermelho ou é preto.
2. A raiz é preta.
3. Cada folha (nula) é preta.
4. Se o nó é vermelho, então seus filhos são pretos.
5. Para cada nó, todos os caminhos simples do nó para as folhas descendentes contém o mesmo número de nós pretos.

Seguindo as propriedades acima conseguimos ter uma árvore binária red-black. A das grandes dificuldades que encontramos foi nas suas rotações, pois precisamos parar para pensar e imaginar todo os casos para a rotação ocorrer sem a árvore ficar desbalanceada, voltando principalmente a atenção a função inserirFix e deleteFix, pois dependendo do caso ela vai usar as funções rotEsq (que faz as rotações para esquerda) e rotDir (que faz as rotações para direita), pois se utilizamos as funções inserir e delete apenas podem violar nossas propriedades que foram comentadas acima.

Em relação à nossa função \_\_init\_\_ fizemos um nó nulo de cor preta, pois primeiramente sempre nossa raiz inicial vai ser preta e nula e o motivo principal é para que no final de cada caminho tenha um nó nulo preto, pois nas rotações ele vai pegar um nó para frente ou seja nada, já que se não existir esses nós nulos não será possível rotacionar a árvore.

Em nossa árvore utilizamos também as funções minimoArvore e maximoArvore para facilitar nosso regate de nós tanto no início como no final de alguns percursos e também as funções sucessorArvore e predecessorArvore, cujo quais são para pegar o sucessor e o predecessor dos nós que tais informações.

Temos a função buscar que seu nome já dá uma noção que ela vai buscar o nó que contém a chave que queremos e temos o percorrerEmOrdem para mostrar nossa árvore item por item em ordem crescente em nosso terminal que irei explicar nosso terminal mais à frente.

No projeto criamos 5 classes que são: Usuario, Livro, No, ArvoreRB, Ui\_MainWindow

Classe Usuário:

É utilizado para o cadastro dos usuários para colocar em nossa árvore de usuários. Nessa classe utilizamos as seguintes variáveis: self.chave que vai receber o CPF do usuário que tal utilizaremos como chave, pois cada usuário tem um CPF diferente logo nunca teremos chaves iguais facilitando na hora que fomos buscar o nó dele. As self.anterior e self.proximo (ambos inicialmente = None) servem para quando precisamos saber o anterior e sucessor do nó do usuário que queremos para facilitar nossa vida. Self.pai (que é inicialmente = None) para salvar o pai do nó pelo menos motivo falado anteriormente. Self.cor que inicialmente é preto, pois facilita na hora verificar se a árvore red-black está balanceada. Self.nome que é para botar o nome do usuário e o self.livros\_alugados que é uma lista para saber quais livro o usuário alugou e botamos os gets e sets das variáveis do \_\_init\_\_ menos do self.nome(que será já colocado e não será mudado depois) e o self.livros\_alugados (ele é será modificado nas funções alugarLivro e devolverLivro).

Classe Livro:

É bem parecida com a classe usuário só que se difere nas variáveis que são: self.chave (que vamos utilizar para achar o nó e essa variável será o nome do livro em questão), self.quantidade (que é a quantidade de exemplares desse livro), self.reservado( que inicialmente é False, pois inicialmente ninguém reservou o livro), self.alugado (inicialmente igual a False, pois tem que estar todos alugados para ele se tornar True), self.usuarios\_alugou( é uma lista vazia e será colocado os nomes dos usuários que alugou o livro do nó em questão) e o self.usuarios\_reservou(que é uma lista vazia e será colocado os usuários que reservaram o livro do nó em questão). Tem os gets e sets das variáveis do \_\_init\_\_ , a função mudarQuantidade (que vai pegar a quantidade e vai somar com o numero de livros que vai ter a mais ou a menos), a função isReservado para retornar se está reservado o livro já, o adicionarAluguel(vai colocar o nome do usuário na lista do self.usuarios\_alugou), removeAluguel( que retira o nome do usuário da lista self.usuarios\_alugou), adicionarReserva (adiciona o nome do usuário na lista usuários\_reservou) , removeReserva( tira o nome do usuário na lista de usuários\_reservou)

Classe No:

Serve para colocar

Classe ArvoreRB:

É a nossa árvore red-black com as funções que foram ditas acima.