Estudo de Desempenho de Algoritmo de Estrutura de Dados da Árvore Binária de Busca e Árvore AVL

Vinicius da Silva Nunes

Resumo do Projeto:
Este projeto visa realizar um estudo de desempenho de algoritmos de estrutura de dados da Árvore Binária de Busca e Árvore AVL, o algoritmo foi desenvolvido para resolver o problema proposto pela professora, onde o objetivo era fazer um programa que atendesse as especificações do problema e em seguida realizar uma verificação do tempo de execução nas partes de inserção e busca dos dados nas árvores binária e avl

Introdução:

Árvore Binária binária é uma estrutura de dados muito importante para o área da computação. Árvore é um objeto matemático,definido como um conjunto finito de um ou mais nós tais que existe um nó denominado com raiz da árvore, a mesma contendo depois ponteiros para as suas subárvores da esquerda e da direita.

Como desafio, foi solicitado aos alunos que resolvessem um problema utilizando conceitos de árvore binária e árvore avl. As seções posteriores deste relatório dizem respeito aos seguintes tópicos, Seções Específicas em que apresenta os aspectos funcionais do programa; Resultados de Execução do Programa onde são dispostos os resultados de execução do programa e da medição de tempo de inserção e busca de um determinado dado na árvore;Conclusão, contém os dados do que foi possível observar durante a implementação e execução do programa.

Seções Específicas:
Esta seção contém os informações acerca das funções e algoritmos usados durante a execução do programa, foram desenvolvidos funções as funções de inserir dados nas árvores binárias e arvores avl, o algoritmo foi desenvolvido na linguagem C.

- **ArvoreSerie *criarArvoreSerie():** Retorna uma árvore de séries vazia.
- **ArvoreTemporada *criaAvoreTemporada():** Retorna uma árvore de temporadas vazia.

- **Participantes* criaListaParticipante(**): Retorna uma lista encadeada de participantes vazia.

- **ArvoreSerie*clerDadosSerie2(int codigo, char titulo[]):** Cria e retorna um nó da árvore de séries, passando como paramentro o codigo e o titulo da serie;

- **ArvoreSerie *lerDadosSerie():** Cria e retorna um nó da árvore de séries com dados lidos do usuário.

- **ArvoreTemporada *lerDadoTemporada(int codigo, int numtemporada, int quantEp, char titulo[], char ano[]):** Cria e retorna um nó da árvore de temporadas com os dados passados como parâmetros.
- **Participantes *lerDadosParticipante():** Lê os dados de um participante do usuário e retorna um nó da lista encadeada de participantes.

- **Participantes *insereParticipante(Participantes *lista, Participantes *dadoParticipante):** Insere um participante na lista encadeada, mantendo a ordem alfabética pelo nome do artista.

- **void buscaParticipantes(Participantes *lista,char nomePersonagem[])**: Busca e imprime participantes pelo nome do personagem.

- **ArvoreTemporada *insereTemporada(ArvoreTemporada **raiz, ArvoreTemporada *Dado)**: Insere uma temporada na árvore de temporadas.

- **void inserirSeries(ArvoreSerie **raiz, ArvoreSerie *DadoSerie):** Insere uma série na árvore de séries.

- **ArvoreSerie *BuscarSeries(ArvoreSerie *raizS, int codigo)**: Busca e retorna uma série na árvore de séries pelo código.

- **ArvoreTemporada *BuscaTemporada(ArvoreTemporada *raizT, int numTemporada):** Busca e retorna uma temporada na árvore de temporadas pelo número da temporada.

- **void imprimeParticipante(Participantes *lista):** Imprime os dados dos participantes na lista encadeada.

- **void imprimeArvoreSeries(ArvoreSerie *raiz):** Imprime os dados da árvore de séries de forma crescente.

- **void imprimeArvoreTemporada(ArvoreTemporada *raiz)**: Imprime os dados da árvore de temporadas de forma crescente.

- **void imprimeTemporadaDeUmaSerie(ArvoreSerie *raizS, int codigo):** Imprime os dados de todas as temporadas de uma série pelo código.

- **void imprimirPersonagensDeUmaTemporada(ArvoreSerie *raizS, int codigo, int NumeroTemporada)**:Imprime os personagens de uma temporada pelo código da série e número da temporada.

- **void imprimirNomeDeUmArtista(ArvoreSerie *raizS, char nomePersonagem[]):** Imprime o nome de um artista.

- **void imprimeTemporada(ArvoreTemporada temporada):** Imprime os dados de uma temporada.
- **void imprimeSeries(ArvoreSerie serie):** Imprime os dados de uma série.

- **void imprimeSeriesPeloCodigo(ArvoreSerie *raiz, int codigo):** Imprime os dados de uma série pelo código.

- **void realizarTesteDeInsercaoDaSerie(ArvoreSerie *raizS, int quant):** Realiza um teste de inserção de séries na árvore e calcula o tempo de execução.
- **void realizarTesteDeBusca(ArvoreTemporada **raizT, int quant):** Realiza um teste de busca de temporadas na árvore e calcula o tempo de execução

Resultados da Execução do Programa:

Com a execução do programa e teste de inserção obteve-se os seguintes resultados é importante ressaltar que os tempo de execução está em microsegundos, os dados de plataforma em que o código foi executado são, memória ram 16 gb, 240 gb, processador core i3:

Teste de Execução com a árvore Binária de Busca

Teste	Teste de Execução (microsegundos)	Media
Inserção da série	15.000000	0.500000
Busca temporada	16.00000	0.600000

Teste de Execução com a árvore AVL

Teste	Teste de Execução (microsegundos)	Media
Inserção da série	5.000000	0.300000
Busca temporada	8.00000	0.400000

Conclusão:

Em geral, a inserção em uma árvore binária não balanceada (sem regras específicas para manter o equilíbrio) pode ter um desempenho pior em comparação com uma árvore AVL, especialmente quando a árvore atinge uma altura significativa. A árvore AVL é projetada para manter o equilíbrio automaticamente, o que pode resultar em melhor desempenho em operações de busca, inserção e remoção.

O tempo exato em microsegundos depende da implementação específica do código, do hardware do sistema em que o código está sendo executado e do número de elementos a serem inseridos.

Apêndice:

A baixo algumas das funções criadas para a execução do programa:

Códigos de Isere temporada

```
ArvoreTemporada*insereTemporada(ArvoreTemporada **raiz, ArvoreTemporada
*Dado){
   Participantes  *lista =  criaListaParticipante();
   Participantes *dadoParicipante;
   int sco, i = 0;
   if(*raiz == NULL){
       (*raiz) = (ArvoreTemporada*)malloc(sizeof(ArvoreTemporada));
       (*raiz) = Dado;

       (*raiz)->esq = NULL;
       (*raiz)->dir = NULL;
   }else{
       if((*Dado).numTem < (*raiz)->numTem)
           insereTemporada(&((*raiz)->esq), Dado);
       else
           insereTemporada(&((*raiz)->dir), Dado);
   }

   return *raiz;
}
```

Código de Inserir Serie

```
void inserirSeries(ArvoreSerie **raiz, ArvoreSerie *DadoSerie){

   int sco, i = 0;
   if(*raiz == NULL){
```

```
        (*raiz) = (ArvoreSerie*)malloc(sizeof(ArvoreSerie));
        (*raiz) = DadoSerie;
        (*raiz)->numeroDeTemporada = 0;
        (*raiz)->esq = NULL;
        (*raiz)->dir = NULL;
    }else{
        if((*DadoSerie).codigo < (*raiz)->codigo)
            inserirSeries(&((*raiz)->esq),DadoSerie );
        else
            inserirSeries(&((*raiz)->dir),DadoSerie );
    }
}
```

Busca serie

```
ArvoreSerie *BuscarSeries( ArvoreSerie *raizS, int codigo){
    ArvoreSerie *serieBusca;
    serieBusca =  NULL;
    if(raizS !=NULL){
        if(codigo == raizS->codigo){
            serieBusca =  raizS;
        }else if(codigo < raizS->codigo){
            serieBusca = BuscarSeries(raizS->esq, codigo);
        }else{
            serieBusca = BuscarSeries(raizS->dir, codigo);
        }
    }
    return serieBusca;

}
```