Árvores AVL Genéricas

Disciplina: Laboratório de Programação II

Professor Luciano Brum
Email: lucianobrum18@gmail.com

Assunto da aula de hoje:

Árvores AVL Genéricas

Árvores AVL Genéricas

- Vimos que é possível declararmos **ponteiros** e **estruturas** que podem armazenar **qualquer tipo de informação**.

- O meio para isso é através da declaração: ***void**.

- Vimos que podemos armazenar qualquer informação em **listas encadeadas**, fazendo com que seu campo **info** seja do tipo ***void**.

Árvores AVL Genéricas

- Agora vamos aplicar os conceitos de estruturas genéricas em árvores AVL.
- Objetivo: criar uma árvore AVL com informações genéricas.
- Para simplificar, vamos criar uma AVL que contenha informações do tipo int.

Árvores AVL Genéricas

- Vamos reaproveitar todas funções anteriormente apresentadas e feitas na disciplina Estruturas de Dados;

- A seguir, uma pequena revisão da implementação dessas funções;

Árvores AVL Genéricas: Estrutura

```
struct avl{
    int altura;
    int info; //void* info;
    struct avl *esq;
    struct avl *dir;
};
typedef struct avl Avl;
```

Árvores AVL Genéricas: Criação e Inserção

```
Avl *criavazia(){

    return NULL;

}
```

```
Avl *inserearv(Avl *a, int i){
  if(a==NULL){
    a=(Avl*)malloc(sizeof(Avl));
    a->info=i;
    a->altura=0;
    a->esq=NULL;
    a->dir=NULL;}
  else{
    if(i<a->info){
       a->esq=inserearv(a->esq, i);}
        else{
          if(i>a->info){
            a->dir=inserearv(a->dir, i);}}}
a=Balanceamento(a);
return a;
}
```

Árvores AVL Genéricas: Remoção

```
Avl *excluiarv(Avl *a, int i){
  if(a==NULL){
     return NULL;
  }
    else{
       if(i<a->info){
             a->esq=excluiarv(a->esq, i);
       }
      else{
            if(i>a->info){
               a->dir=excluiarv(a->dir, i);
          }
           else{
            if(a->esq==NULL && a->dir==NULL){
             free(a);
             return NULL;}
            else{
            if(a->dir==NULL){
                 Avl *aux=a;
                 a=a->esq;
                 free(aux);
             }
                else{
                    if(a->esq==NULL){
                      Avl *aux=a;
                      a=a->dir;
                      free(aux);
                      }
                      else{
                            Avl *temp=a->esq;
                            while(temp->dir!=NULL){
                                  temp=temp->dir;
                            }
                            a->info=temp->info;
                            temp->info=i;
                            a->esq=excluiarv(a->esq, i);}}}}}
  a=Balanceamento(a);
  return a;
}
```

Árvores AVL Genéricas: Impressão e Libera

```
void imprimearv_preordem(Avl *a){
    printf("<");
    if(a!=NULL){
       printf("%d", a->info);
       imprimearv_preordem(a->esq);
       imprimearv_preordem(a->dir);
    }
    printf(">");
}
```

```
Avl *libera_arv(Avl *a){
    if(a!=NULL){
       a->esq=libera_arv(a->esq);
       a->dir=libera_arv(a->dir);
       free(a);
    }
    return NULL;
}
```

Função Balanceamento

```
AvlGen *BalanceamentoGenerico(AvlGen *a){
    if(retornaaltura(a->esq)-retornaaltura(a->dir)==2){
        if(retornaaltura(a->esq->esq)-retornaaltura(a->esq->dir)==1){
            a=rotacionadireita(a); }
        else{
            a=rotacionaesquerdadireita(a); }
    }
    else{
        if(retornaaltura(a->esq)-retornaaltura(a->dir)==-2){
            if(retornaaltura(a->dir->esq)-retornaaltura(a->dir->dir)==-1){
                a=rotacionaesquerda(a);}
            else{
                a=rotacionadireitaesquerda(a);
            }
        }
    }
    a->altura=1+max(retornaaltura(a->esq),retornaaltura(a->dir));
    return a;
}
```

Funções de Rotação

```
Avl *rotacionadireita(Avl *a){
  Avl *aux;
  aux=a->esq;
  a->esq=aux->dir;
  aux->dir=a;
  a->altura=1+max(retornaaltura(a->esq),retornaaltura(a->dir));
  aux->altura=1+max(retornaaltura(aux->esq),retornaaltura(aux->dir));
return aux;
}

Avl *rotacionaesquerda(Avl *a){
  Avl *aux;
  aux=a->dir;
  a->dir=aux->esq;
  aux->esq=a;
  a->altura=1+max(retornaaltura(a->esq),retornaaltura(a->dir));
  aux->altura=1+max(retornaaltura(aux->esq),retornaaltura(aux->dir));
return aux;
}
```

```
Avl *rotacionaesquerdadireita(Avl *a){
  a->esq=rotacionaesquerda(a->esq);
  a=rotacionadireita(a);
  return a;
}

Avl *rotacionadireitaesquerda(Avl *a){
  a->dir=rotacionadireita(a->dir);
  a=rotacionaesquerda(a);
  return a;
}
```

Exercício

- Implemente funções de um TAD de árvore AVL genérica:
  - Inserção;
  - Remoção;
  - Impressão nos 3 percursos;
  - Cria;
  - Libera;

- Considere que os dados sejam do tipo int.
- Crie um atributo tipo na estrutura da árvore que identifique que o dado armazenado é do tipo int. Isso deve ser considerado na inserção, remoção, impressão e liberação.

Referências Bibliográficas

- CELES, W.; CERQUEIRA, R.; RANGEL, J. L. **Introdução a Estruturas de Dados**. 1ª. Ed. Rio de Janeiro: Campus, 2004.

Dúvidas?

Professor Luciano Brum
email: lucianobrum18@gmail.com
https://sites.google.com/view/brumluciano