Lista 6
Árvores binárias

1. Faça um TAD de árvores binárias com números inteiros com as funções básicas apresentadas na aula. Para resolver as questões seguintes considere a existência do TAD arvores, cuja interface deverá estar definida em *arvore.h*.

2. Faça uma função que retorne a quantidade de nós de uma árvore binária. Esta função deve obedecer ao protótipo:
   `int nos(Arv *a);`

3. Faça uma função que retorne a quantidade de folhas de uma árvore binária. Esta função deve obedecer ao protótipo:
   `int folhas(Arv *a);`

4. Faça uma função que retorne a quantidade de nós de uma árvore binária que possuem apenas um filho. Esta função deve obedecer ao protótipo:
   `int um_filho(Arv *a);`

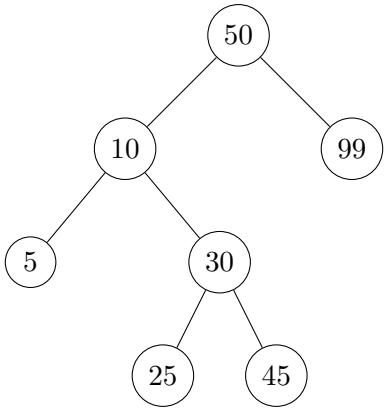
Árvores binárias de busca

Uma árvore binária de busca é um tipo especial de árvore binária. Nessa nova árvore cada nó possui um valor (chave) associado a ele, e esse valor determina a posição do nó na árvore.

Em uma árvore binária de busca, temos a seguinte regra para posicionamento dos valores na árvore: para cada nó pai:

- todos os valores da subárvore esquerda são **menores** do que o nó pai;
- todos os valores da subárvore direita são **maiores** do que o nó pai.

Um exemplo de árvore binária de busca é mostrado abaixo:



Inserção e **remoção** de nós na árvore binária de busca devem ser realizadas respeitando essa regra de posicionamento dos nós.

A árvore binária de busca é uma ótima alternativa ao uso de *arrays* para operações de busca binária, pois, além de permitir esse tipo de busca, ela possui a vantagem de ser uma estrutura dinâmica: é muito mais fácil acrescentar um nó na árvore seguindo a sua regra de posicionamento do que inserir um valor dentro de um vetor ordenado.

Na tabela a seguir podemos ver o custo para as principais operações em uma árvore binária de busca contendo N nós. Note que o pior caso ocorre quando a árvore não está balanceada.

Operação	Caso médio	Pior caso
Inserção	$O(\log n)$	$O(n)$
Remoção	$O(\log n)$	$O(n)$
Consulta	$O(\log n)$	$O(n)$

Inserindo um nó na árvore

Inserir um novo nó em uma árvore binária de busca é uma tarefa simples. Basicamente, o que temos que fazer é procurar a sua posição na árvore usando o seguinte conjunto de passos:

- primeiro compare o valor a ser inserido com a raiz;
- se o valor é **menor** do que a raiz: vá para a subárvore da esquerda;
- se o valor é **maior** do que a raiz: vá para a subárvore da direita;
- aplique o método recursivamente até chegar a um nó folha.

Também existe o caso em que a inserção é feita em uma árvore que está vazia. Nesse caso, a raiz da árvore, que inicialmente apontava para NULL, passa a apontar para o único elemento inserido até então.

Todo nó inserido em uma árvore binária de busca é um nó folha.

Prática Considerando os passos para inserir um elemento em uma árvore binária de busca, implemente uma função no TAD para inserir um elemento.

Consultando um nó da árvore

Consultar se determinado novo nó existe em uma árvore binária de busca é uma tarefa similar à inserção de um novo nó. Basicamente, o que temos que fazer é percorrer os nós da árvore usando o seguinte conjunto de passos:

- primeiro compare o valor buscado com a raiz;
- se o valor é menor do que a raiz: vá para a subárvore da esquerda;
- se o valor é maior do que a raiz: vá para a subárvore da direita;
- aplique o método recursivamente até que a raiz seja igual ao valor buscado.

Prática Considerando os passos para consultar um elemento em uma árvore binária de busca, implemente uma função no TAD para buscar um elemento.

Removendo um nó da árvore

Remover um nó de uma árvore binária de busca não é uma tarefa tão simples quanto a inserção. Isso ocorre porque precisamos procurar o nó a ser removido da árvore, o qual pode ser um nó folha ou um nó interno (que pode ser a raiz), com um ou dois filhos. Se este for um nó interno, é preciso reorganizar a árvore para que ela continue sendo uma árvore binária de busca. Além disso, precisamos verificar se a árvore é vazia (caso em que a remoção não é possível) e se a remoção desse nó não gera uma árvore vazia.

Segue o código que implementa a remoção de um nó de uma árvore binária de busca.

```
Arv* retira(Arv* a, int v){
  if (a == NULL){
    printf("Arvore vazia\n");
    return NULL;
  }
  else if (v < a->info){
    a->esq = retira(a->esq,v);
  }
  else if (v > a->info){
    a->dir = retira(a->dir,v);
  }
  else { //achou o elemento
    if (a->esq == NULL && a->dir == NULL){ //é uma folha
      free(a);
      a = NULL;
    }
    else if (a->esq == NULL) { //só tem filhos à direita
      Arv* temp = a;
      a = a->dir;
      free(temp);
    }
    else if (a->dir == NULL) { //só tem filhos à esquerda
      Arv* temp = a;
      a = a->esq;
      free(temp);
```

```
    }
    else { //tem os dois filhos
      Arv* temp = a->esq;
      while (temp->dir != NULL) { //busca elemento maior abaixo para trocar de posição
        temp = temp->dir;
      }
      a->info = temp->info; //troca
      temp->info = v;
      a->esq = retira(a->esq,v);
    }
    return a;
  }
}
```