BCC - Estruturas de Dados
Lab 10 - Árvore Binária de Busca

Prof. Dr. Paulo César Rodacki Gomes
IFC - Instituto Federal Catarinense - Campus Blumenau

14 de junho de 2022

1 Objetivo

O objetivo desta atividade prática em laboratório é implementar uma estrutura de árvore binária de busca para armazenar e buscar números inteiros.

Você pode usar sua implementação de árvores binárias como ponto de partida para a resolução deste exercício. Voce deverá implementar os arquivos `arvbb.c` e `arvbb.h` com as estruturas, tipos de dados e funções da árvore binária de busca, e um arquivo `principal.c`, com a função main demonstrando a sua implementação.

Cada nó de sua árvore binária de busca deve armazenar um número inteiro. O tipo de dados para representar um nó deve ser implementado da seguinte maneira:

```
struct arv {
   int info;
   struct arv* esq;
   struct arv* dir;
};

typedef struct arv Arv;
```

2 Descrição das funções a serem implementadas

1. `Arv* abb_cria(void)`: cria uma árvore binária vazia (i.e., simplesmente retorna NULL);

2. `void abb_imprime(Arv *a)`: percorre a árvore em ordem simétrica para imprimir seu conteúdo (números) em ordem crescente;

3. `Arv* abb_busca(Arv* r, int v)`: esta é a operação para buscar um elemento na árvore, com desempenho computacional proporcional à sua altura ($O(\log n)$), portanto, a função deve se valer do fato de a àrvore estar ordenada. Caso o valor v seja encontrado, a função retorna o endereço do nó que armazena v. Caso contrário, a função deve retornar NULL;

4. `Arv* abb_insere(Arv* a, int v)`: adiciona um novo elemento na árvore na posição correta para que a propriedade fundamental de ordenação seja mantida. Importante notar a necessidade de atualizar os ponteiros para sub àrvores à esquerda ou à direita quando da chamada recursiva da função, pois a inserção pode alterar o valor do ponteiro para a raiz da (sub)árvore;

5. `Arv* abb_retira(Arv* r, int v)`: operação para retirar um elemento com valor `v` da árvore. A função tem como valor de retorno a eventual nova raiz da (sub)árvore.

6. `void abb_imprimeDecrescente(Arv* a)`: implementação semelhante à função `abb_imprime`, porém deve imprimir o elementos da árvore em ordem decrescente.