DCC013 - Estrutura de Dados / DCC107 - Laboratório de Programação II

Árvore binária de busca

Você pode utilizar qualquer ambiente de programação para desenvolver sua atividade. Ao final, **copie e cole o seu código-fonte com a resposta aqui mesmo neste documento**, dentro dos espaços indicados para isso e **preservando a identação do código**. **Depois que terminar sua avaliação, não se esqueça de entregar sua atividade!** Fique atento ao relógio, pois as atividades entregues com atraso não serão aceitas.

Para resolver esta atividade, **clique aqui para baixar o projeto da aula de laboratório de programação 2, que contém a implementação do TAD ArvBinBusca para árvore binária de busca de números inteiros.** Na sua solução para a questão abaixo, você pode utilizar/chamar qualquer uma das operações que estejam disponíveis no projeto (exatamente do jeito que ele se encontra no site da disciplina). Outras operações que você venha a criar para resolver o seu exercício, **inclusive as operações auxiliares**, devem ser copiadas para sua resposta neste documento. **Não cole código desnecessariamente**.

Desenvolver uma operação `int ArvBinBusca::contaNos(int a, int b)` para, dado o intervalo `[a, b]`, composto por 2 numeros inteiros tal que `a<b`, calcular e retornar a quantidade de nós que tem um único filho com valor ímpar dentro desse intervalo. Usar a **propriedade de ABB** para visitar a quantidade mínima de nós.

```
int ArvBinBusca::contaNos(int a, int b)
{
    if (raiz == NULL)
        return 0;
    else
        return auxContaNos;
}

int ArvBinBusca::auxContaNos()
{
    if(a < b)
        return 1 + p->getEsq() + p->getDir();
}

bool ArvBinBusca::auxEhABB(NoArv* p, int *ultimo)
{
    if(p==NULL)
```

```
        return true;
    else
    {

        bool abbEsq = auxEhABB(p->getEsq(), ultimo);

        if(*ultimo > p->getInfo())
            return false;

        else
        {
            *ultimo = p->getInfo();

            bool abbDir = auxEhABB(p->getDir(), ultimo);

            return abbEsq && abbEsq;
        }
    }
}

bool ArvBinBusca::EhABB()
{
    int ultimo = noMaisEsquerda();
    return auxEhABB(raiz, &ultimo);
}
```