Estrutura de Dados - Lista 5 - Árvore

José Gildásio Freitas do Ó - 473901

Junho 2023

1. Apresente o pseudocódigo de uma função ContarFolhas(r) que recebe como entrada o nó raiz r de uma BST e retorna a quantidade de folhas presentes nesta BST. Qual a complexidade desta função?

   - **Resposta:**
     - **Algoritmo:** ContarFolhas(**NO*** arv)
     - **Entrada:** NO raiz da arvore
     - **Saída:** Quantidades de folhas na arvore
     - **Complexidade:** O(n)

```
1– se arv == NULL então
2– |   return 0
3– se arv → esq == NULL && arv → dir == NULL então
4– |    return 1
5– return ContarFolhas(arv→esq) + ContarFolhas(arv→dir)
```

2. Apresente o pseudocódigo de uma função ContarNos(r) que recebe como entrada o nó raiz r de uma BST e retorna a quantidade de nós nesta BST que tem pelo menos uma sub-árvore vazia. Qual a complexidade desta função?

   - **Resposta:**
     - **Algoritmo:** ContarNos_SubArvoreVazia(**NO*** arv)
     - **Entrada:** NO raiz da arvore
     - **Saída:** Quantidade de nós com sub arvove vazia
     - **Complexidade:** O(n)

```
1– se arv == NULL então
2– |   return 0
3– int result = 0
4– se arv → esq == NULL || arv → dir == NULL então
5– |    result = 1
6– result += ContarNos_SubArvoreVazia(arv → esq)
7– result += ContarNos_SubArvoreVazia(arv → dir)
8– return result
```

3. Apresente o pseudocódigo de uma função RemoveTodos(r, x) que recebe como entrada o nó raiz r de uma BST e um valor x e remove todos os nós desta BST que tem chave igual a x. Qual a complexidade desta função?

- **Resposta:**
  - **Algoritmo:** RemoveTodos(**NO*** arv, int x)
  - **Entrada:** NO raiz da arvore e x que é o valor que deve ser removido todos os nós com chave x.
  - **Saída:** Arvore com os elementos com chave x removidos.
  - **Complexidade:** O(h) e no pior caso O(n+h)

```
1– NO* no = buscar(arv, x)
2– se no == NULL então
3– |   return
4– se no → esq == NULL && no → dir == NULL então
5– |   se no → pai == NULL então
6– |   |   free(no)
7– |   |   return
8– |   se no → pai → esq == no então
9– |   |   no → pai → esq = NULL
10– |   senão
11– |   |   no → pai → dir = NULL
12– |   free(no)
13– |   return
14– se no → esq == NULL então
15– |   se no → pai == NULL então
16– |   |   arv = no → dir
17– |   |   free(no)
18– |   |   return
19– |   se no → pai → esq == no então
20– |   |   no → pai → esq = no → dir
21– |   senão
22– |   |   no → pai → dir = no → dir
23– |   free(no)
24– |   return
25– se no → dir == NULL então
26– |   se no → pai == NULL então
27– |   |   arv = no → esq
28– |   |   free(no)
29– |   |   return
30– |   se no → pai → esq == no então
31– |   |   no → pai → esq = no → esq
32– |   senão
```

33– | | no → pai → dir = no → esq
34– **NO*** aux = no → esq
35– **enquanto** aux → dir ≠ NULL **então**
36– | aux = aux → dir
37– no → chave = aux → chave
38– **se** aux → pai → esq == aux **então**
39– | aux → pai → esq = aux → esq
40– **senão**
41– | aux → pai → dir = aux → esq
42– **free**(aux)
43– return

4. Em classe, vimos uma função para preencher os campos alt (altura) de cada um dos nós de uma BST. Porém, assim como no caso do campo p (pai), o campo alt pode ser atualizado no momento da inserção. Apresente um pseudocódigo da função de inserir que atualiza os campos alt corretamente no momento da inserção.

- **Resposta:**
  - **Algoritmo:** Inserir_com_altura(**NO* arv**, int x)
  - **Entrada:** NO raiz da arvore e x que é o valor a ser inserido
  - **Saída:** arvore com o valor x inserido e alterado a altura dos nós caso necessário
  - **Complexidade:** O(n+h)

1– **se** arv == NULL **então**
2– | **NO*** novo = (**NO***) malloc(sizeof(NO))
3– | novo → chave = x
4– | novo → esq = NULL
5– | novo → dir = NULL
6– | novo → pai = NULL
7– | novo → altura = 1
8– | return novo
9– **se** x < arv → chave **então**
10– | arv → esq = **inserir_com_altura**(arv→ esq, x)
11– | arv → esq → pai = arv
12– **senão se** x > arv → chave **então**
13– | arv → dir = **inserir_com_altura**(arv → dir, x)
14– | arv → dir → pai = arv
15– **senão**
16– | return arv
17– **alterarAltura**(arv)
18– return arv

```
void alterarAltura(NO* arv){
    int esqAltura = Altura(arv → esq)
    int dirAltura = Altura(arv → dir)
    arv → altura = 1 + max(esqAltura, dirAltura)
}
```

5. Apresente o pseudocódigo de uma função ImprimeNos(r) **não-recursiva** que recebe como entrada o nó raiz r de uma BST e imprime os nós desta BST em ordem simétrica.
   Dica: utilize as funções MinimoBST(r) (que retorna o nó com a menor chave na árvore) e SucessorBST(v) (que retorna o nó sucessor de v).

   - **Resposta:**
     - **Algoritmo:** imprimir_interativo(**NO*** arv)
     - **Entrada:** NO raiz da arvore
     - **Saída:** Imprimir os NOs da arvore
     - **Complexidade:** O(n)

```
1– NO* aux = minimo(arv)
2– enquanto aux ≠ NULL faça
3– |    cout << aux → chave << endl
4– |    aux = sucessor(aux)
```