

BEE 1195

Árvore Binária de Busca

Prof. Edson Alves – UnB/FGA

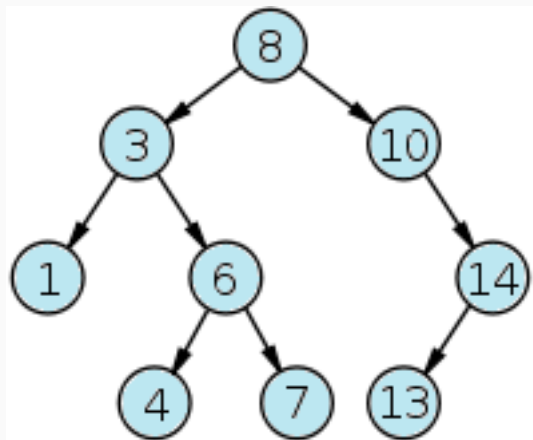
Problema

Em computação, as árvores binária de busca ou árvore binária de pesquisa é uma estrutura baseada em nós (nodos), onde todos os nós da subárvore esquerda possuem um valor numérico inferior ao nó raiz e todos os nós da subárvore direita possuem um valor superior ao nó raiz (e assim sucessivamente). O objetivo desta árvore é estruturar os dados de forma flexível, permitindo a busca binária de um elemento qualquer da árvore.

A grande vantagem das árvores de busca binária sobre estruturas de dados convencionais é que os algoritmos de ordenação (percurso infixo) e pesquisa que as utilizam são muito eficientes.

Para este problema, você receberá vários conjuntos de números e a partir de cada um dos conjuntos, deverá construir uma árvore binária de busca. Por exemplo, a sequência de valores: 8 3 10 14 6 4 13 7 1 resulta na seguinte árvore binária de busca:

Problema



Entrada

A entrada contém vários casos de teste. A primeira linha da entrada contém um inteiro C ($C \leq 1000$), indicando o número de casos de teste que virão a seguir. Cada caso de teste é composto por 2 linhas. A primeira linha contém um inteiro N ($1 \leq N \leq 500$) que indica a quantidade de números que deve compor cada árvore e a segunda linha contém N inteiros distintos e não negativos, separados por um espaço em branco.

Saída

Cada linha de entrada produz 3 linhas de saída. Após construir a árvore binária de busca com os elementos de entrada, você deverá imprimir a mensagem "Case n :", onde n indica o número do caso de teste e fazer os três percursos da árvore: prefixo, infixo e posfixo, apresentando cada um deles em uma linha com uma mensagem correspondente conforme o exemplo abaixo, separando cada um dos elementos por um espaço em branco.

Obs: Não deve haver espaço em branco após o último item de cada linha e há uma linha em branco após cada caso de teste, inclusive após o último.

Exemplo de entradas e saídas

Exemplo de Entrada

```
2
3
5 2 7
9
8 3 10 14 6 4 13 7 1
```

Exemplo de Saída

Case 1:

Pre.: 5 2 7

In..: 2 5 7

Post: 2 7 5

Case 2:

Pre.: 8 3 1 6 4 7 10 14 13

In..: 1 3 4 6 7 8 10 13 14

Post: 1 4 7 6 3 13 14 10 8

Solução com complexidade $O(N^2)$

- A solução do problema tem início com a codificação de uma árvore binária de busca
- Além do construtor, é preciso implementar a rotina de inserção (a qual tem complexidade $O(S)$ no pior caso, onde S é o tamanho da árvore)
- Além disso, é preciso implementar as três travessias por profundidade notáveis (cuja complexidade de cada travessia também é $O(S)$)
- Por fim, para cada caso de teste, basta instanciar uma árvore, inserir os elementos indicados e produzir a saída usando as travessias indicadas

Solução AC com complexidade $O(N^2)$

```
1 #include <iostream>
2
3 struct BST {
4     struct Node {
5         int info;
6         Node *left, *right;
7     };
8
9     Node *root;
10
11     BST() : root(nullptr) {}
12
13     void inorder(const Node* node) const
14     {
15         if (node) {
16             inorder(node->left);
17             std::cout << ' ' << node->info;
18             inorder(node->right);
19         }
20     }
```

Solução AC com complexidade $O(N^2)$

```
22 void preorder(const Node* node) const
23 {
24     if (node)
25     {
26         std::cout << ' ' << node->info;
27         preorder(node->left);
28         preorder(node->right);
29     }
30 }
31
32 void postorder(const Node* node) const
33 {
34     if (node)
35     {
36         postorder(node->left);
37         postorder(node->right);
38         std::cout << ' ' << node->info;
39     }
40 }
```


Solução AC com complexidade $O(N^2)$

```
42 void insert(int info)
43 {
44     Node **node = &root;
45
46     while (*node)
47     {
48         if ((*node)->info == info)
49             return;
50         else if (info < (*node)->info)
51             node = &(*node)->left;
52         else
53             node = &(*node)->right;
54     }
55
56     *node = new Node { info, nullptr, nullptr };
57 }
58 };
```

Solução AC com complexidade $O(N^2)$

```
60 int main()
61 {
62     std::ios::sync_with_stdio(false);
63
64     int C;
65     std::cin >> C;
66
67     for (int test = 1; test <= C; ++test)
68     {
69         int N;
70         std::cin >> N;
71
72         BST tree;
73
74         while (N-- > 0) {
75             int info;
76             std::cin >> info;
77
78             tree.insert(info);
79         }
```

Solução AC com complexidade $O(N^2)$

```
81     std::cout << "Case " << test << ":\n";
82
83     std::cout << "Pre.:";
84     tree.preorder(tree.root);
85     std::cout << "\n";
86
87     std::cout << "In..:";
88     tree.inorder(tree.root);
89     std::cout << "\n";
90
91     std::cout << "Post:";
92     tree.postorder(tree.root);
93     std::cout << "\n\n";
94 }
95
96 return 0;
97 }
```