

## Binary Search Tree □ Insert (không dùng đệ quy)

---



```

1
2  #include <bits/stdc++.h>
3  using namespace std;
4
5
6  class Node {
7      public:
8          int data;
9          Node *left;
10         Node *right;
11         Node(int d) {
12             data = d;
13             left = NULL;
14             right = NULL;
15         }
16     };
17
18     class Solution {
19     public:
20
21         void preOrder(Node *root) {
22
23             if( root == NULL )
24                 return;
25
26             std::cout << root->data << " ";
27
28             preOrder(root->left);
29             preOrder(root->right);
30         }
31
32         /* you only have to complete the function given below.
33         Node is defined as
34
35         class Node {
36         public:
37             int data;
38             Node *left;
39             Node *right;
40             Node(int d) {
41                 data = d;
42                 left = NULL;
43                 right = NULL;
44             }
45         };
46
47         */
48
49         Node * insert(Node * root, int data) {
50
51             // no recursion
52             Node* cur = root;
53             Node* prev = NULL;
54             Node* newNode = new Node(data);
55             while(cur != NULL) {
56                 prev = cur;
57                 if(data <= cur->data) {
58                     cur = cur->left;
59                 } else {

```

```

60         cur = cur->right;
61     }
62 }
63 if(prev == NULL) {
64     return newNode;
65 } else if(data <= prev->data) {
66     prev->left = newNode;
67 } else {
68     prev->right = newNode;
69 }
70 return root;
71 }
72 };
73
74 int main() {
75
76     Solution myTree;
77     Node* root = NULL;
78
79     int t;
80     int data;
81
82     std::cin >> t;
83
84     while(t-- > 0) {
85         std::cin >> data;
86         root = myTree.insert(root, data);
87     }
88
89     myTree.preOrder(root);
90
91     return 0;
92 }

```

## Binary Search Tree ☐ Insert

---





```

60         root->right = cur;
61     }
62
63     return root;
64 }
65 }
66 };
67
68 int main() {
69
70     Solution myTree;
71     Node* root = NULL;
72
73     int t;
74     int data;
75
76     std::cin >> t;
77
78     while(t-- > 0) {
79         std::cin >> data;
80         root = myTree.insert(root, data);
81     }
82
83     myTree.preOrder(root);
84
85     return 0;
86 }

```

## Binary Search Tree☒ Nút chung gần nhất

---





```

1
2  #include <bits/stdc++.h>
3
4  using namespace std;
5
6  class Node {
7      public:
8          int data;
9          Node *left;
10         Node *right;
11         Node(int d) {
12             data = d;
13             left = NULL;
14             right = NULL;
15         }
16     };
17
18     class Solution {
19     public:
20         Node* insert(Node* root, int data) {
21             if(root == NULL) {
22                 return new Node(data);
23             } else {
24                 Node* cur;
25                 if(data <= root->data) {
26                     cur = insert(root->left, data);
27                     root->left = cur;
28                 } else {
29                     cur = insert(root->right, data);
30                     root->right = cur;
31                 }
32
33                 return root;
34             }
35         }
36
37         /*The tree node has data, left child and right child
38         class Node {
39             int data;
40             Node* left;
41             Node* right;
42         };
43
44         */
45
46         Node *lca(Node *root, int v1,int v2) {
47             if (root == NULL) {
48                 return NULL;
49             }
50             if (root->data == v1 || root->data == v2) {
51                 return root;
52             }
53             Node* left = lca(root->left, v1, v2);
54             Node* right = lca(root->right, v1, v2);
55             if (left != NULL && right != NULL) {
56                 return root;
57             }
58             return (left != NULL) ? left : right;
59         }

```

```

60
61     }; //End of Solution
62
63     int main() {
64
65         Solution myTree;
66         Node* root = NULL;
67
68         int t;
69         int data;
70
71         std::cin >> t;
72
73         while(t-- > 0) {
74             std::cin >> data;
75             root = myTree.insert(root, data);
76         }
77
78         int v1, v2;
79         std::cin >> v1 >> v2;
80
81         Node *ans = myTree.lca(root, v1, v2);
82
83         std::cout << ans->data;
84
85         return 0;
86     }
87

```

**khangtd.HoaNhac**

---

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  #define fi first
5  #define se second
6  const int N = 1e6 + 9;
7  const int N2 = N * 10;
8  const int mod = 1e9 + 7;
9  const int inf = LLONG_MAX;
10
11
12
13  signed main() {
14      ios::sync_with_stdio(false);
15      cin.tie(NULL);
16      if (fopen("TASK.INP", "r")) {
17          freopen("TASK.INP", "r", stdin);
18          freopen("TASK.OUT", "w", stdout); }
19
20      int n;
21      cin >> n;
22      vector<int> v(n);
23      for (int i = 0; i < n; ++i) {
24          cin >> v[i];
25      }
26      stack<pair<int, int>> st;
27      int ans = 0;
28
29      for (int i = 0; i < n; ++i) {
30          int c = 1;
31          while (!st.empty() && st.top().fi <= v[i]) {
32              if (st.top().fi < v[i]) {
33                  ans += st.top().se;
34                  st.pop();
35              } else {
36                  ans += st.top().se;
37                  c += st.top().se;
38                  st.pop();
39              }
40          }
41          if (!st.empty()) {
42              ans += 1;
43          }
44          st.push(make_pair(v[i], c));
45      }
46      cout << ans;
47
48  }

```

## khangtd.HuffmanCoding

---

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  #define fi first
5  #define se second
6  const int N = 1e6 + 9;
7  const int N2 = N * 10;
8  const int mod = 1e9 + 7;
9  const int inf = LLONG_MAX;
10
11 int huffman(string s){
12     int n = s.size();
13     vector<int> cnt(256, 0);
14     for (char c : s) cnt[c]++;
15     priority_queue<int, vector<int>, greater<int>> pq;
16     for (int i = 0; i < 256; i++) if (cnt[i]) pq.push(cnt[i]);
17
18     if (pq.size() == 1) return n;
19
20     int ans = 0;
21     while (pq.size() > 1){
22         int a = pq.top(); pq.pop();
23         int b = pq.top(); pq.pop();
24         ans += a + b;
25         pq.push(a + b);
26     }
27     return ans;
28 }
29
30 signed main(){
31     ios::sync_with_stdio(false);
32     cin.tie(NULL);
33     if (fopen("TASK.INP", "r")){
34         freopen("TASK.INP", "r", stdin);
35         freopen("TASK.OUT", "w", stdout);}
36
37     int n;
38     cin >> n;
39     string s;
40     cin >> s;
41     cout << huffman(s);
42
43 }

```

**khangtd.KetBan**

---

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  #define fi first
5  #define se second
6  const int N = 1e6 + 9;
7  const int N2 = N * 10;
8  const int mod = 1e9 + 7;
9  const int inf = LLONG_MAX;
10
11
12
13  signed main() {
14      ios::sync_with_stdio(false);
15      cin.tie(NULL);
16      if (fopen("TASK.INP", "r")) {
17          freopen("TASK.INP", "r", stdin);
18          freopen("TASK.OUT", "w", stdout); }
19
20      int n, m;
21      cin >> n >> m;
22      vector<int> prev(n + 1, 0);
23      vector<int> next(n + 1, 0);
24
25      for (int i = 1; i <= n; i++) {
26          prev[i] = i - 1;
27          next[i] = i + 1;
28      }
29      prev[1] = n;
30      next[n] = 1;
31
32      while (m--) {
33          int x, y;
34          cin >> x >> y;
35
36          int a = prev[x];
37          int b = next[x];
38
39          next[a] = b;
40          prev[b] = a;
41
42          next[x] = next[y];
43          prev[next[y]] = x;
44          prev[x] = y;
45          next[y] = x;
46
47      }
48      int tmp = 1;
49      for (int i = 1; i <= n; i++) {
50          cout << tmp << " ";
51          tmp = next[tmp];
52      }
53  }

```

## Tree Hieght of Tree



```

1
2  #include <bits/stdc++.h>
3
4  using namespace std;
5
6  class Node {
7      public:
8          int data;
9          Node *left;
10         Node *right;
11         Node(int d) {
12             data = d;
13             left = NULL;
14             right = NULL;
15         }
16     };
17
18     class Solution {
19     public:
20         Node* insert(Node* root, int data) {
21             if(root == NULL) {
22                 return new Node(data);
23             } else {
24                 Node* cur;
25                 if(data <= root->data) {
26                     cur = insert(root->left, data);
27                     root->left = cur;
28                 } else {
29                     cur = insert(root->right, data);
30                     root->right = cur;
31                 }
32
33                 return root;
34             }
35         }
36
37         /*The tree node has data, left child and right child
38         class Node {
39             int data;
40             Node* left;
41             Node* right;
42         };
43
44         */
45         int height(Node* root) {
46
47             Node* cur = root;
48             queue<pair<Node*, int>>> q;
49             q.push({cur, 1});
50             int maxHeight = 0;
51             while(!q.empty()) {
52                 pair<Node*, int> p = q.front();
53                 maxHeight = max(maxHeight, p.second);
54                 q.pop();
55                 Node* node = p.first;
56                 int height = p.second;
57                 if (node->left != NULL) {
58                     q.push({node->left, height + 1});
59                 }
60                 if (node->right != NULL) {

```

```

60         q.push({node->right, height + 1});
61     }
62 }
63 return maxHeight;
64 }
65
66 }; //End of Solution
67
68 int main() {
69
70     Solution myTree;
71     Node* root = NULL;
72
73     int t;
74     int data;
75
76     std::cin >> t;
77
78     while(t-- > 0) {
79         std::cin >> data;
80         root = myTree.insert(root, data);
81     }
82
83     int height = myTree.height(root);
84
85     std::cout << height;
86
87     return 0;
88 }

```

## Tree☐ Inorder Traversal (LNR) - Duyệt cây BST theo LNR

---





```

1
2  #include <bits/stdc++.h>
3  using namespace std;
4
5
6  class Node {
7      public:
8          int data;
9          Node *left;
10         Node *right;
11         Node(int d) {
12             data = d;
13             left = NULL;
14             right = NULL;
15         }
16     };
17
18     class Solution {
19     public:
20         Node* insert(Node* root, int data) {
21             if(root == NULL) {
22                 return new Node(data);
23             } else {
24                 Node* cur;
25                 if(data <= root->data) {
26                     cur = insert(root->left, data);
27                     root->left = cur;
28                 } else {
29                     cur = insert(root->right, data);
30                     root->right = cur;
31                 }
32
33                 return root;
34             }
35         }
36
37         /* you only have to complete the function given below.
38         Node is defined as
39
40         class Node {
41         public:
42             int data;
43             Node *left;
44             Node *right;
45             Node(int d) {
46                 data = d;
47                 left = NULL;
48                 right = NULL;
49             }
50         };
51
52         */
53
54         void inOrder(Node *root) {
55             if (root == NULL) {
56                 return;
57             }
58             inOrder(root->left);
59             cout << root->data << " ";

```

```

60         inOrder(root->right);
61     }
62
63 }; //End of Solution
64
65 int main() {
66
67     Solution myTree;
68     Node* root = NULL;
69
70     int t;
71     int data;
72
73     std::cin >> t;
74
75     while(t-- > 0) {
76         std::cin >> data;
77         root = myTree.insert(root, data);
78     }
79
80     myTree.inOrder(root);
81
82     return 0;
83 }

```

## Tree☒ Inorder Traversal (LNR) II - Duyệt cây BST theo LNR không đệ quy

---



```

1
2  #include <bits/stdc++.h>
3  using namespace std;
4
5
6  class Node {
7      public:
8          int data;
9          Node *left;
10         Node *right;
11         Node(int d) {
12             data = d;
13             left = NULL;
14             right = NULL;
15         }
16     };
17
18     class Solution {
19     public:
20         Node* insert(Node* root, int data) {
21             if(root == NULL) {
22                 return new Node(data);
23             } else {
24                 Node* cur;
25                 if(data <= root->data) {
26                     cur = insert(root->left, data);
27                     root->left = cur;
28                 } else {
29                     cur = insert(root->right, data);
30                     root->right = cur;
31                 }
32
33                 return root;
34             }
35         }
36
37         /* you only have to complete the function given below.
38         Node is defined as
39
40         class Node {
41         public:
42             int data;
43             Node *left;
44             Node *right;
45             Node(int d) {
46                 data = d;
47                 left = NULL;
48                 right = NULL;
49             }
50         };
51
52         */
53
54         void inOrder(Node *root) {
55             stack<Node*> s;
56             Node* curr = root;
57             while (curr != NULL || !s.empty()) {
58                 while (curr != NULL) {
59                     s.push(curr);

```

```

60         curr = curr->left;
61     }
62     curr = s.top();
63     s.pop();
64     cout << curr->data << " ";
65     curr = curr->right;
66 }
67 }
68
69 }; //End of Solution
70
71 int main() {
72
73     Solution myTree;
74     Node* root = NULL;
75
76     int t;
77     int data;
78
79     std::cin >> t;
80
81     while(t-- > 0) {
82         std::cin >> data;
83         root = myTree.insert(root, data);
84     }
85
86     myTree.inOrder(root);
87
88     return 0;
89 }

```

## Tree☒ levelOrder Traversal - Duyệt cây BST theo chiều rộng

---



```

1
2  #include <bits/stdc++.h>
3  using namespace std;
4
5
6  class Node {
7      public:
8          int data;
9          Node *left;
10         Node *right;
11         Node(int d) {
12             data = d;
13             left = NULL;
14             right = NULL;
15         }
16     };
17
18     class Solution {
19     public:
20         Node* insert(Node* root, int data) {
21             if(root == NULL) {
22                 return new Node(data);
23             } else {
24                 Node* cur;
25                 if(data <= root->data) {
26                     cur = insert(root->left, data);
27                     root->left = cur;
28                 } else {
29                     cur = insert(root->right, data);
30                     root->right = cur;
31                 }
32
33                 return root;
34             }
35         }
36
37         /* you only have to complete the function given below.
38         Node is defined as
39
40         class Node {
41         public:
42             int data;
43             Node *left;
44             Node *right;
45             Node(int d) {
46                 data = d;
47                 left = NULL;
48                 right = NULL;
49             }
50         };
51
52         */
53
54         void levelOrder(Node *root) {
55             if (root == NULL) {
56                 return;
57             }
58             queue<Node*> q;
59             q.push(root);

```



```

60         while (!q.empty()) {
61             Node* node = q.front();
62             cout << node->data << " ";
63             q.pop();
64             if (node->left != NULL) {
65                 q.push(node->left);
66             }
67             if (node->right != NULL) {
68                 q.push(node->right);
69             }
70         }
71     }
72
73 }; //End of Solution
74
75 int main() {
76     Solution myTree;
77     Node* root = NULL;
78
79     int t;
80     int data;
81
82     std::cin >> t;
83
84     while(t-- > 0) {
85         std::cin >> data;
86         root = myTree.insert(root, data);
87     }
88
89     myTree.levelOrder(root);
90
91     return 0;
92 }
93

```

## Tree☐ Postorder Traversal (LRN) - Duyệt cây BST theo LRN

---



```

1
2  #include <bits/stdc++.h>
3  using namespace std;
4
5
6  class Node {
7      public:
8          int data;
9          Node *left;
10         Node *right;
11         Node(int d) {
12             data = d;
13             left = NULL;
14             right = NULL;
15         }
16     };
17
18     class Solution {
19     public:
20         Node* insert(Node* root, int data) {
21             if(root == NULL) {
22                 return new Node(data);
23             } else {
24                 Node* cur;
25                 if(data <= root->data) {
26                     cur = insert(root->left, data);
27                     root->left = cur;
28                 } else {
29                     cur = insert(root->right, data);
30                     root->right = cur;
31                 }
32
33                 return root;
34             }
35         }
36
37         /* you only have to complete the function given below.
38         Node is defined as
39
40         class Node {
41             public:
42                 int data;
43                 Node *left;
44                 Node *right;
45                 Node(int d) {
46                     data = d;
47                     left = NULL;
48                     right = NULL;
49                 }
50         };
51
52         */
53
54         void postOrder(Node *root) {
55
56             if (root == NULL) {
57                 return;
58             }
59             postOrder(root->left);

```

```

60         postOrder(root->right);
61         cout << root->data << " ";
62     }
63 }
64
65 }; //End of Solution
66
67 int main() {
68     Solution myTree;
69     Node* root = NULL;
70
71     int t;
72     int data;
73
74     std::cin >> t;
75
76     while(t-- > 0) {
77         std::cin >> data;
78         root = myTree.insert(root, data);
79     }
80
81     myTree.postOrder(root);
82
83     return 0;
84 }
85

```

## Tree☐ Postorder Traversal (LRN) II - Duyệt cây BST theo LRN không đệ quy

---



```

1
2  #include <bits/stdc++.h>
3  using namespace std;
4
5
6  class Node {
7      public:
8          int data;
9          Node *left;
10         Node *right;
11         Node(int d) {
12             data = d;
13             left = NULL;
14             right = NULL;
15         }
16     };
17
18     class Solution {
19     public:
20         Node* insert(Node* root, int data) {
21             if(root == NULL) {
22                 return new Node(data);
23             } else {
24                 Node* cur;
25                 if(data <= root->data) {
26                     cur = insert(root->left, data);
27                     root->left = cur;
28                 } else {
29                     cur = insert(root->right, data);
30                     root->right = cur;
31                 }
32
33                 return root;
34             }
35         }
36
37         /* you only have to complete the function given below.
38         Node is defined as
39
40         class Node {
41             public:
42                 int data;
43                 Node *left;
44                 Node *right;
45                 Node(int d) {
46                     data = d;
47                     left = NULL;
48                     right = NULL;
49                 }
50         };
51
52         */
53
54         void postOrder(Node *root) {
55
56             stack<Node*> s;
57             Node* curr = root;
58             Node* prev = NULL;
59             while (curr != NULL || !s.empty()) {

```

```

60         while (curr != NULL) {
61             s.push(curr);
62             curr = curr->left;
63         }
64         curr = s.top();
65         if (curr->right == NULL || curr->right == prev) {
66             cout << curr->data << " ";
67             s.pop();
68             prev = curr;
69             curr = NULL;
70         } else {
71             curr = curr->right;
72         }
73     }
74 }
75 }; //End of Solution
76
77
78
79 int main() {
80
81     Solution myTree;
82     Node* root = NULL;
83
84     int t;
85     int data;
86
87     std::cin >> t;
88
89     while(t-- > 0) {
90         std::cin >> data;
91         root = myTree.insert(root, data);
92     }
93
94     myTree.postOrder(root);
95
96     return 0;
97 }

```

## Tree Top view

---





```

1  #include <bits/stdc++.h>
2  #include <regex>
3  using namespace std;
4  #define int long long
5  #define fi first
6  #define se second
7  const int N = 1e6 + 9;
8  const int N2 = N * 10;
9  const int mod = 1e9 + 7;
10 const int inf = LLONG_MAX;
11
12 map<int, int> check;
13
14 struct Node{
15     int data;
16     Node *left, *right;
17     Node(int x) : data(x), left(NULL), right(NULL) {}
18 };
19
20 struct BST{
21     Node *root;
22
23     BST() : root(NULL) {}
24
25     void insert(int x){
26         if (root == NULL){
27             root = new Node(x);
28         } else {
29             insertHelper(root, x);
30         }
31     }
32
33     void insertHelper(Node *node, int x){
34         if (x <= node->data){
35             if (node->left == NULL){
36                 node->left = new Node(x);
37             } else {
38                 insertHelper(node->left, x);
39             }
40         } else {
41             if (node->right == NULL){
42                 node->right = new Node(x);
43             } else {
44                 insertHelper(node->right, x);
45             }
46         }
47     }
48
49     void printRoot(){
50         if (root != NULL){
51             cout << root->data << endl;
52         } else {
53             cout << "Tree is empty" << endl;
54         }
55     }
56
57     void traverseFromRoot(){
58         if (root == NULL) return;
59         queue<pair<Node*, int>> q;

```

```

60     q.push({root, 0});
61     while (!q.empty()) {
62         pair<Node*, int> p = q.front();
63         Node *current = p.first;
64         int level = p.second;
65         if (check[level] == 0) {
66             // cout << current->data << " " << level << "\n";
67             check[level] = current -> data; // Mark this level as visited
68         }
69         q.pop();
70         if (current->right != NULL) q.push({current->right, level + 1});
71         if (current->left != NULL) q.push({current->left, level - 1});
72     }
73 }
74
75 };
76
77 signed main() {
78     ios::sync_with_stdio(false);
79     cin.tie(NULL);
80     if (fopen("TASK.INP", "r")) {
81         freopen("TASK.INP", "r", stdin);
82         freopen("TASK.OUT", "w", stdout);
83     }
84     int n;
85     cin >> n;
86
87     BST tree;
88     for (int i = 0; i < n; i++) {
89         int x;
90         cin >> x;
91         tree.insert(x);
92     }
93     // tree.printRoot();
94     tree.traverseFromRoot();
95     vector<int> levels;
96     for (auto it : check) {
97         levels.push_back(it.second);
98     }
99     sort(levels.begin(), levels.end());
100    for (int i = 0; i < levels.size(); i++) {
101        cout << levels[i] << " ";
102    }
103 }

```