

Name: **Kashif Ali**
Roll No: **20P-0648**
Section: **3D**

Lab-10 Tasks
AVL Tree

home > kashiii > Documents > 10_AVL > C 02_avl.cpp > ...

```
1  #include<iostream>
2  using namespace std;
3
4  class Node
5  {
6      public:
7      int key;
8      Node *left;
9      Node *right;
10     int height;
11 };
12
13
14 int max(int a, int b);
15
16
17 int height(Node *N)
18 {
19     if (N == NULL)
20         return 0;
21     return N->height;
22 }
23
24
25 int max(int a, int b)
26 {
27     return (a > b)? a : b;
28 }
29
30
31 Node* newNode(int key)
32 {
33     Node* node = new Node();
34     node->key = key;
35     node->left = NULL;
36     node->right = NULL;
37     node->height = 1;
38     return(node);
39 }
```

Code part-a

```
41
42 Node *rightRotate(Node *y)
43 {
44     Node *x = y->left;        // root left
45     Node *T2 = x->right;       // root ke left ke right
46
47
48     x->right = y;
49     y->left = T2;
50
51
52     y->height = max(height(y->left),
53                     height(y->right)) + 1;
54     x->height = max(height(x->left),
55                     height(x->right)) + 1;
56
57
58     return x;
59 }
60
61
62 Node *leftRotate(Node *x)
63 {
64     Node *y = x->right;
65     Node *T2 = y->left;
66
67
68     y->left = x;
69     x->right = T2;
70
71
72     x->height = max(height(x->left),
73                     height(x->right)) + 1;
74     y->height = max(height(y->left),
75                     height(y->right)) + 1;
76
77
78     return y;
79 }
```

Code part-b

```
81
82 int getBalance(Node *N)
83 {
84     if (N == NULL)
85         return 0;
86     return height(N->left) - height(N->right);
87 }
88
89
90 Node* insert(Node* node, int key)
91 {
92
93     if (node == NULL)
94         return(newNode(key));
95
96     if (key < node->key)
97         node->left = insert(node->left, key);
98     else if (key > node->key)
99         node->right = insert(node->right, key);
100 else
101     return node;
102
103
104     node->height = 1 + max(height(node->left),
105                          height(node->right));
106
107
108
109     int balance = getBalance(node);
110
111
112
113     // Left Left Case
114     if (balance > 1 && key < node->left->key)
115         return rightRotate(node);
116
117     // Right Right Case
118     if (balance < -1 && key > node->right->key)
119         return leftRotate(node);
120
```

Code part-c

```
121 // Left Right Case
122 if (balance > 1 && key > node->left->key)
123 {
124     node->left = leftRotate(node->left);
125     return rightRotate(node);
126 }
127
128 // Right Left Case
129 if (balance < -1 && key < node->right->key)
130 {
131     node->right = rightRotate(node->right);
132     return leftRotate(node);
133 }
134
135 /* return the (unchanged) node pointer */
136 return node;
137 }
138
139
140
141
142 // Search
143
144 bool avlSearch( Node *root, int key)
145 {
146     if (root == NULL)
147         return false;
148     else if (root->key == key)
149         return true;
150
151     else if (root->key > key) {
152         bool val = avlSearch(root->left, key);
153         return val;
154     }
155
156     else {
157         bool val = avlSearch(root->right, key);
158         return val;
159     }
```

Code part-d

Searching in AVL code

```
161
162
163 void preOrder(Node *root)
164 {
165     if(root != NULL)
166     {
167         cout << root->key << " ";
168         preOrder(root->left);
169         preOrder(root->right);
170     }
171 }
172
173 // Inorder traversal
174
175 void InOrder(Node *root)
176 {
177     if(root != NULL)
178     {
179         InOrder(root->left);
180         cout << root->key << " ";
181         InOrder(root->right);
182     }
183 }
184
185
186 // PostOrder Traversal
187
188 void PostOrder(Node *root)
189 {
190     if(root != NULL)
191     {
192         InOrder(root->left);
193         InOrder(root->right);
194         cout << root->key << " ";
195     }
196 }
197
198
199 int main()
200 {
```

Code part-e

```

C 02_avl.cpp ×
home > kashiii > Documents > 10_AVL > C 02_avl.cpp > ...
198
199 int main()
200 {
201     Node *root = NULL;
202     // as { 55, 66, 77, 11, 33, 22, 35, 25, 44, 88,99},
203
204     root = insert(root, 55);
205     root = insert(root, 66);
206     root = insert(root, 77);
207     root = insert(root, 11);
208     root = insert(root, 33);
209     root = insert(root, 22);
210     root = insert(root, 35);
211     root = insert(root, 25);
212     root = insert(root, 44);
213     root = insert(root, 88);
214     root = insert(root, 99);
215
216
217
218
219
220     cout<<"Height of the Tree: "<<height(root)<<endl;
221     cout<<endl;
222
223     cout << "Preorder traversal of the "
224     |       "constructed AVL tree is \n";
225     preOrder(root);
226
227     //
228     cout<<"\n"<<endl;
229
230     cout << "InOrder traversal of the "
231     |       "constructed AVL tree is \n";
232     InOrder(root);
233
234     //
235     cout<<"\n"<<endl;
236
237     cout << "PostOrder traversal of the "

```

Main driver

```
202 root = insert(root, 55);
203 root = insert(root, 66);
204 root = insert(root, 77);
205 root = insert(root, 11);
206 root = insert(root, 33);
207 root = insert(root, 22);
208 root = insert(root, 35);
209 root = insert(root, 25);
210 root = insert(root, 44);
211 root = insert(root, 88);
212 root = insert(root, 99);
213
214 cout<<"Height of the Tree: "<<height(root)<<endl;
215 cout<<endl;
216
217 cout << "Preorder traversal of the "
218 |      | "constructed AVL tree is \n";
219 preorder(root);
220
221 //
222 cout<<"\n"<<endl;
223
224 cout << "InOrder traversal of the "
225 |      | "constructed AVL tree is \n";
226 inorder(root);
227
228 //
229 cout<<"\n"<<endl;
230
231 cout << "PostOrder traversal of the "
232 |      | "constructed AVL tree is \n";
233 postorder(root);
234
235 cout<<"Searching Element: "<<endl;
236 cout<<avlSearch(root, 32)<<endl;           //returns true of false boolean value
237 cout<<avlSearch(root, 55)<<endl;
238 return 0;
239 }
```

Main driver part-b


```
kashiii@kashiii: ~/Documents/10_AVL
kashiii@kashiii:~/Documents/10_AVL$ g++ 02_avl.cpp -o 02_avl.exe
kashiii@kashiii:~/Documents/10_AVL$ ./02_avl.exe
Height of the Tree: 4

Preorder traversal of the constructed AVL tree is
33 22 11 25 66 44 35 55 88 77 99

InOrder traversal of the constructed AVL tree is
11 22 25 33 35 44 55 66 77 88 99

PostOrder traversal of the constructed AVL tree is
11 22 25 35 44 55 66 77 88 99 33 Searching Element:
0
1
kashiii@kashiii:~/Documents/10_AVL$ |
```

Output-1

```
kashiii@kashiii: ~/Documents/10_AVL
kashiii@kashiii:~/Documents/10_AVL$ g++ 02_avl.cpp -o 02_avl.exe
kashiii@kashiii:~/Documents/10_AVL$ ./02_avl.exe
Height of the Tree: 4

Preorder traversal of the constructed AVL tree is
11 6 1 8 7 9 33 22 25 35 44

InOrder traversal of the constructed AVL tree is
1 6 7 8 9 11 22 25 33 35 44

PostOrder traversal of the constructed AVL tree is
1 6 7 8 9 22 25 33 35 44 11

Searching Element:
0 Value not found
1 Value found
kashiii@kashiii:~/Documents/10_AVL$ |
```

Output-2

Thank You

...