AppData\Local\Temp\431ada4f-a97b-40c3-99f9-bc2f6b475331_OSAP_003_7_최종보고서(소스코드 포함).zip.331\src\utilities\balance_node.cc

```
1
2  /*
3  MIT License
4  This file is part of the INHA_OSAP_003_7 project.
5  Copyright (c) 2024 tbmyong
6
7  Permission is hereby granted, free of charge, to any person obtaining a copy
8  of this software and associated documentation files (the "Software"), to deal
9  in the Software without restriction, including without limitation the rights
10 to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
11 copies of the Software, and to permit persons to whom the Software is
12 furnished to do so, subject to the following conditions:
13
14 The above copyright notice and this permission notice shall be included in all
15 copies or substantial portions of the Software.
16
17 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
18 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
19 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
20 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
21 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
22 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
23 SOFTWARE.
24
25 작성자: 이현진
26 작성일(파일 생성일): 2024-11-27
27 작성일(파일 최종 수정일): 2024-12-17
28 */
29
30 /*
31 <Rotate 기능 구현>
32 Node* RotateLeft(Node* y)
33 : y를 기준으로 오른쪽 자식 x와 x의 왼쪽 자식 T1을 회전시킨다.
34     y                      x
35    / \                    / \
36   a   x        ->        y   c
37      / \                / \
38    T1   c              a  T1
39
40 Node* RotateRight(Node* y)
41 : y를 기준으로 왼쪽 자식 x와 x의 오른쪽 자식 T2를 회전시킨다.
42     y                      x
43    / \                    / \
44   x   c        ->        a   y
45  / \                        / \
46 a   T2                     T2  c
47
48 <balance 기능 구현>
49 int BalanceFactor(Node* current_node)
50 : current_node의 balance factor를 계산한다.
51
```

```
52  void Balance(Node*& current_node)
53  : current_node의 balance factor를 확인하고, LL, RR, LR, RL 중 어떤 회전을 할지
54    결정한다.
55  */
56
57  #include "../../base/balance_node.h"
58
59  Node* BalanceNode::RotateLeft(Node* y) {
60    Node* x = y->get_right();
61    Node* T1 = x->get_left();
62
63    x->set_left(y);
64    y->set_right(T1);
65
66    Node* parent_node = y->get_parent();
67    // 부모 노드가 있는 경우
68    if (parent_node != nullptr) {
69      // y가 부모의 왼쪽 자식인 경우
70      if (parent_node->get_left() == y) {
71        parent_node->set_left(x);
72        // y가 부모의 오른쪽 자식인 경우
73      } else {
74        parent_node->set_right(x);
75      }
76    }
77
78    x->set_parent(y->get_parent());
79    y->set_parent(x);
80    // T1이 존재하는 경우
81    if (T1 != nullptr) {
82      T1->set_parent(y);
83    }
84
85    updater_.Update(y);
86    updater_.Update(x);
87
88    return x;
89  }
90
91  Node* BalanceNode::RotateRight(Node* y) {
92    Node* x = y->get_left();
93    Node* T2 = x->get_right();
94
95    x->set_right(y);
96    y->set_left(T2);
97
98    Node* parent_node = y->get_parent();
99    // 부모 노드가 있는 경우
100   if (parent_node != nullptr) {
101     // y가 부모의 왼쪽 자식인 경우
102     if (parent_node->get_left() == y) {
103       parent_node->set_left(x);
104       // y가 부모의 오른쪽 자식인 경우
105     } else {
```

```
106       parent_node->set_right(x);
107     }
108   }
109
110   x->set_parent(y->get_parent());
111   y->set_parent(x);
112   // T2가 존재하는 경우
113   if (T2 != nullptr) {
114     T2->set_parent(y);
115   }
116
117   updater_.Update(y);
118   updater_.Update(x);
119
120   return x;
121 }
122
123 void BalanceNode::Balance(Node*& current_node) {
124   int balance_factor = BalanceFactor(current_node);
125
126   // LL case
127   //        z                                      y
128   //       / \                                   /  \
129   //      y   T4   Right Rotate(z)             x     z
130   //     / \          ============>           /  \  /  \
131   //    x   T3                               T1  T2 T3  T4
132   //   / \
133   // T1   T2
134   if (balance_factor > 1 && BalanceFactor(current_node->get_left()) >= 0) {
135     current_node = RotateRight(current_node);
136   }
137   // RR case
138   //     z                                y
139   //   /  \                             /  \
140   // T1    y     Left Rotate(z)       z     x
141   //      /  \    ============>     / \   / \
142   //     T2   x                    T1  T2 T3  T4
143   //         / \
144   //       T3   T4
145   else if (balance_factor < -1
146            && BalanceFactor(current_node->get_right()) <= 0) {
147     current_node = RotateLeft(current_node);
148   }
149   //      z                                z                          x
150   //     / \                              / \                       /   \
151   //    y   T4  Left Rotate(y)           x   T4  Right Rotate(z)    y     z
152   //   / \      ============>          / \      ============>      / \   / \
153   // T1   x                          y   T3                     T1  T2 T3  T4
154   //     / \                        / \
155   //    T2  T3                    T1   T2
156   // LR case
157   else if (balance_factor > 1 && BalanceFactor(current_node->get_left()) < 0) {
158     current_node->set_left(RotateLeft(current_node->get_left()));
159     current_node = RotateRight(current_node);
```

```
  }
  // RL case
  //   z                                z                               x
  //  / \                              / \                            /   \
  // T1   y   Right Rotate(y)       T1   x    Left Rotate(z)    z     y
  //     / \    ============>           / \    ============>    / \   / \
  //    x   T4                        T2   y                  T1  T2 T3  T4
  //   / \                               / \
  // T2   T3                           T3   T4
  else if (balance_factor < -1
           && BalanceFactor(current_node->get_right()) > 0) {
    current_node->set_right(RotateRight(current_node->get_right()));
    current_node = RotateLeft(current_node);
  }
}

int BalanceNode::BalanceFactor(Node* current_node) {
  // 노드가 없는 경우
  if (current_node == nullptr) return 0;

  /* 왼쪽 자식 높이 계산, 왼쪽 자식 노드가 존재하면 높이를 가져오고 존재하지
  않으면 0으로 설정 */
  int left_height = (current_node->get_left())
                        ? current_node->get_left()->get_height()
                        : 0;

  /* 오른쪽 자식 높이 계산, 오른쪽 자식 노드가 존재하면 높이를 가져오고 존재하지
  않으면 0으로 설정 */
  int right_height = (current_node->get_right())
                         ? current_node->get_right()->get_height()
                         : 0;

  return left_height - right_height;
}
```