AppData\Local\Temp\c03595e2-c960-4df1-800b-85256a5ad29b_OSAP_003_7_최종보고서(소스코드 포함).zip.29b\src\rank.cc

```
/*
MIT License
This file is part of the INHA_OSAP_003_7 project.
Copyright (c) 2024 tbmyong

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

작성자: 이현진
작성일(파일 생성일): 2024-12-04
작성일(파일 최종 수정일): 2024-12-17
*/

/*
<Rank 기능 구현>
Rank(int key)
: root부터 key와 크기를 비교하며 순위를 찾아 반환.
  없으면 0 반환.
*/

#include "../base/avl.h"

std::pair<int, int> AVL::Rank(int key) {
  // rank: key보다 작은 원소의 수 + 1
  int rank{1};

  // root부터 내려가면서 비교할 것
  Node* current_node{get_root()};
  while (current_node) {
    // 현재 노드의 키가 찾는 키와 같을 때
    if (key == current_node->get_key()) {
      // 왼쪽 subtree의 노드 수를 rank에 더하고 반환
      if (current_node->get_left() != nullptr) {
        rank += current_node->get_left()->get_subtree_size();
      }
      return {CalculateDepthHeightSum(get_root(), key), rank};
```

```
    }

    // 찾는 key가 현재 노드의 key보다 작은 경우
    if (key < current_node->get_key()) {
      current_node = current_node->get_left();
    }
    // 찾는 key가 현재 노드의 key보다 큰 경우
    else {
      rank += ((current_node->get_left() != nullptr)
                  ? current_node->get_left()->get_subtree_size() + 1
                  : 1);
      current_node = current_node->get_right();
    }
  }

  return {0, 0};
}
```