CS1103

Programación Orientado a Objetos II

Práctica Calificada #2 (PC2)

Sección 101

2021 - 1

Profesor: Rubén Rivas:

6.5 puntos: Heap

1. Escribir la función **mininum_range** que procese un número variado de contenedores secuenciales ordenados y que retorne de forma eficiente el rango más pequeño que incluya al menos un elemento de cada lista.

```
// contenedores
vector <int> v1 = {2, 3, 4};
vector <int> v2 = {3, 4, 6, 11};
vector <int> v3 = {6, 7, 9, 13};
vector <int> v4 = {5, 8, 10};
vector <int> v5 = {1, 3, 5};
vector < vector <int>> vr = {v1, v2, v3, v4, v5};
// buscar rango
pair<int, int> res = mininum_range (vr);
// muestra el resultado
cout << res.first << " " << res.second << endl; // 3 6
```

```
// contenedores
vector <int> v1 = {3, 4};
vector <int> v2 = {3, 4, 6, 11};
vector <int> v3 = {2, 7, 9, 13};
vector <int> v4 = {4, 8, 10};
vector <int> v5 = {5};
// buscar rango
pair<int, int> res = mininum_range (v1, v2, v3, v4, v5);
// muestra el resultado
cout << res.first << " " << res.second << endl; // 2 5
```

6.5 puntos: hash

Escribir la función **rank_values**, que permita generar a partir de un contenedor otro contenedor que obtenga para cada valor del contenedor original el número orden del valor.

Ejemplo #1:

```
vector<int> vec1 = {9, 10, 2, 5, 8, 4, 3, 1};
auto res = rank_values(vec1);
for (auto row: res)
   cout << row << " ";
cout << endl;
// 9, 10, 2, 5, 8, 4, 3, 1
```

Ejemplo #2:

```
vector<int> vec1 = {10, 12, 3, 4, 15, 8, 1, 18};
auto res = rank_values(vec1);
for (auto row: res)
   cout << row << " ";
cout << endl;
// 5 6 2 3 7 4 1 8
```

Ejemplo #3:

```
deque<double> deq1 = {5.5, 1, 2, 13, 4, 17, 8, 29, 40.5, 17, 13};
auto res = rank_values(deq1);
auto res = rank_values(vec1);
for (auto row: res)
   cout << row << " ";
cout << endl;
// 4 1 2 6 3 7 5 8 9 7 6
```

5 puntos: Tree

Escribir la función **is_max_heap** que permite verificar si un árbol binario es un heap

Ejemplo #1:

```
binary_tree<int> bt1(10);
// Izquierda
auto left_branch = bt1.add_left(bt1.get_root(), 8);
bt1.add_left(left_branch, 3);
bt1.add_right(left_branch, 4);
// Derecha
auto right_branch = bt1.add_right(bt1.get_root(), 7);
bt1.add_left(right_branch, 8);
bt1.add_right(right_branch, 4);
// Verificar si es heap
cout << boolalpha << is_max_heap (bt1) << endl; // false
```

Ejemplo #2:

```
binary_tree<int> bt2(20);
// Izquierda
auto left_branch = bt2.add_left(bt2.get_root(), 12);
bt2.add_left(left_branch, 5);
bt2.add_right(left_branch, 8);
// Derecha
auto right_branch = bt1.add_right(bt1.get_root(), 7);
bt2.add_left(right_branch, 4);
// Verificar si es heap
cout << boolalpha << is_max_heap (bn1) << endl; // true
```

Barranco, 13 de junio del 2021