CS2013 - Programación III

Práctica Calificada #2 (PC2)

2023 - 2

Profesor: Rubén Rivas

Librería Estándar – 7 puntos

Diseñar y desarrollar el template de función **consolidate_ranges** que reciba un contenedor secuencial que contenga pares de datos numericos del mismo tipo, el template deberá unir los rangos consecutivos que se translapan, ejemplo {1, 3} {2, 5} y los consolidara en {1, 5}, si hubiesen más de dos rangos consecutivos también se pueden consolidar. El resultado podra retornarse en un contenedor de un tipo diferente al contenedor de original.

Los contenedores originales podrian ser: **std::list**, **std::vector**, **std::deque**, **std::forward_list**.

Casos de uso #1

```
std::vector<std::pair<int, int>> v1 = {{1, 3}, {2, 4}, {5, 7}, {6, 8}};
auto crs = consolidate_ranges<std::list>(begin(v1), end(v1));
for (const auto& [f, s]: crs) cout << '{' << f << ", " << s << "} ";
   // El resultado es {1, 4} {5, 8}
```

Casos de uso #2

```
std::list<std::pair<int, int>> v1 = {
    {2, 4}, {3, 6}, {3, 5}, {1, 7}, {8, 10} };
std::deque<std::pair<int, int>> crs = consolidate_ranges<std::deque>(
    begin(v1), end(v1));
for (const auto& [f, s]: crs) cout << '{' << f << ", " << s << "}";
   // El resultado es {1, 7} {8, 10}
```

Casos de uso #3

```
std::deque<std::pair<size_t, size_t>> v1 = {
    {1, 3}, {1, 2}, {1, 4}, {5, 6}};
auto crs = consolidate_ranges<std::vector>(begin(v1), end(v1));
for (const auto& [f, s]: crs) cout << '{' << f << ", " << s << "} ";
   // El resultado es {1, 4} {5, 6}
```

Casos de uso #4

```
std::forward_list<std::pair<double, double>> v1 = {
         {1.0, 3.6}, {3.5, 4.0}, {5.1, 5.9}, {4.5, 15.9}};
auto srs = consolidate_ranges<std::vector>(begin(v1), end(v1));
for (const auto& [f, s]: crs) cout << '{' << f << ", " << s << "} ";
   // El resultado es {1.0, 4.0} {5.1, 15.9}
```

Complejidad Algorítmica – 6 puntos

Dada la siguiente función determinar la complejidad algoritmica, incluir en su respuesta el procedimiento.

```
void find_quartets(const std::vector<int>& numbers, int target_sum) {
    std::vector<int> copy = numbers;
    std::sort(copy.begin(), copy.end());

    for (auto it1 = copy.begin(); it1 != copy.end(); ++it1) {
        for (auto it2 = std::next(it1); it2 != copy.end(); ++it2) {
            for (auto it3 = std::next(it2); it3 != copy.end(); ++it3) {
                int partial_sum = *it1 + *it2 + *it3;
                if (std::binary_search(std::next(it3), copy.end(),
                            target_sum - partial_sum)) {
                    std::cout << "Quartet found: ("
                              << *it1 << ", " << *it2 << ", " << *it3 << ", "
                              << target_sum - partial_sum << ")\n";
                }
            }
        }
    }
}
```

Programación Concurrente – 7 puntos

En base a la primera pregunta, desarrollar una funcion **parallel_consolidate_ranges** similar pero, que realice la consolidación de rangos de forma concurrente, debe incluirse adicionalmente, como parametro, el número de hilos (**nt**) de ejecución.

Caso de uso #1

```
std::vector< std::pair<double, double>> v1 = {
   {-2.2, 7.6}, {-8.9, -3.6}, {9.3, 17.1}, {0.4, 8.6}, {-2.9, -0.1},
   {2, 10}, {3.6, 4.8}, {-6, 3}, {-7.9, 1.7}, {2.4, 8.5}, {2.7, 9},
   {8.9, 17.5}, {-1, -0.9}, {-10, -5.6}, {1.2, 3.1}, {-8.7, -0.9},
   {-7.7, 2.1}, {-6.3, -4.2}, {-2.9, 5.8}, {6.5, 7}, {7.5, 12.5},
   {-0.4, 6.6}, {-6.2, -1.5}, {3.7, 9.4}, {-4, -1.8}, {7, 8.5}, {-5.5, 0.6},
   {2.5, 11.2}, {6.3, 6.7}, {5.2, 14.2}, {9.2, 14.9}, {-4.8, 0.1},
   {8.4, 12.4}, {2.3, 8.7}, {9.5, 9.7}, {2.2, 9.5}, {-1.1, 3.3}, {-2.2, 5.5},
   {-4, -2.2}, {-5.3, 4.1}, {-1.7, 3.8}, {-9.8, -4.2}, {-8.8, 1}, {1.1, 3},
   {7.2, 10.7}, {-7.7, 0.5}, {-4, 0}, {-6.7, 3.1}, {-5.9, -3.1},
   {-2.7, 0.3}, {-1.4, 4.1}, {5.4, 14.3}, {-8.9, -2.6}, {5.1, 7.9},
   {-8.3, -5.8}, {4.4, 10.1}, {7.9, 15.8}, {3.7, 12.5}, {-2.7, 1.5},
   {8.2, 9.2}, {-2.1, 6.6}, {9, 18}, {8.9, 16.5}, {-5.4, -5.2}, {4.4, 8.3},
   {6.9, 16.4}, {-5, 4.5}, {6.4, 15.3}, {3.3, 6.3}, {5.6, 15.5}, {1.9, 7.5},
   {0.7, 5.8}, {-2, 2.6}, {-4.5, 2.5}, {-9.2, -1.1}, {7.5, 15.8},
   {-8.3, -4.9}, {-4.8, 2.1}, {6.5, 8.1}, {-5.7, 0.7}, {-4.4, -3.1},
   {9, 11.7}, {4.6, 12.8}, {-9.8, -2.2}, {-1.2, 8.6}, {5.9, 13.3},
   {-4.4, -4.2}, {-2.1, 7.5}, {-3.1, 1}, {1.8, 8.6}, {-5.9, -1.9},
   {1.4, 10.8}, {5.7, 11.5}, {-1.8, 6.3}, {6.5, 11}, {7.7, 16.1},
   {-1.7, 4.9}, {8.1, 8.9}, {6, 7.3}, {-6.7, 0.1}, {-8.8, -0.9},
   {-1.6, -0.8}, {10, 14}, {8.4, 16.7}, {5.5, 7.5}, {8, 11.8}, {9.8, 16},
   {1.4, 6.6}, {-0.5, 3}, {3, 6.2}, {4.2, 13.1}, {0.9, 5.2}, {-1.7, -0.7},
   {6.1, 12}, {0.2, 1.4}, {-6.6, 0.5}, {7.8, 15.2}, {5.3, 13.3},
   {-4.4, -2.6}, {-6.3, -1.7}, {-6.9, -2.6}, {5.1, 7}, {1.3, 4.1},
   {-2.2, 0.6}, {-1.9, 5.9}, {-9.3, -8.5}, {-0.6, 5}, {8.4, 16.7},
   {2.9, 11.5}, {8.4, 16.2}, {-3.3, 4.5}, {6.9, 16.7}, {8.9, 9},
   {-7.9, -6.1}, {-10, -7.2}, {9.4, 19.4}, {7.2, 8.5}, {-2.7, -1.6},
   {0.1, 1.3}, {9.6, 9.8}, {-3.6, 5.1}, {8.4, 16.6}, {-4, -2.1},
   {3.4, 13.1}, {7.8, 13.6}, {0.4, 9.6}, {-8.5, -3}, {-4.2, -0.6},
   {8.1, 14.8}, {4.5, 8}, {-7, -4.9}, {5.3, 9.4}, {1.8, 9.5}, {-9.3, -3.7},
   {-5, -1.6}, {8.8, 9.7}, {-8.2, -3.5}, {0.6, 2.1}, {1.4, 3.9},
   {1.7, 3.3}, {5.9, 6.2}, {-7.9, -7.5}, {-3.9, 2.8}, {4.6, 10.3},
   {-0.4, 9.1}, {2.8, 11.5}, {4.6, 6.1}, {-9.3, -2.3}, {3.6, 8.4},
   {2.1, 11.5}, {-6.8, 1.4}, {7.5, 9.2}, {6.8, 12}, {4.1, 6.4}, {-7.3, -3.7},
   {2.7, 5.9}, {1.6, 7.3}, {7.3, 9.6}, {6.6, 15.2}, {-2.3, -0.5},
   {-3.7, -1.9}, {-2.9, 3}, {-9.1, -2.9}, {-9.1, -3.6}, {-2.2, 6.5},
   {2.8, 10.8}, {-5.6, 3.4}, {5.9, 15.4}, {8.2, 8.6}, {-1, 7.8},
   {-0.2, 5.2}, {8.5, 13.8}, {2.4, 5.4}, {9.8, 19.3}, {-8.3, -3.6},
   {8, 13.4}, {-4.5, 0.1}, {5.9, 15.1}, {2.6, 4.6}, {-5.6, 3.1}
};
size_t nt = thread::hardware_concurrency();
auto crs = parallel_consolidate_ranges <std::list>(begin(v1), end(v1), nt);
for (const auto& [f, s]: crs) cout << '{' << f << ", " << s << "}";
```

Caso de uso #2

```
    std::deque< std::pair<double, double>> d1 = {
       {33, 72}, {-93, -35}, {79, 158}, {32, 76}, {-15, 12}, {-61, 2},
       {-80, -76}, {-25, 54}, {-24, 35}, {13, 80}, {-93, -79}, {18, 61},
       {14, 108}, {-51, 1}, {51, 72}, {7, 34}, {-86, -74}, {-100, -10},
       {-36, 22}, {81, 91}, {-16, 2}, {-98, -26}, {-28, 69}, {54, 61},
       {-49, -33}, {-87, -8}, {-4, 34}, {-83, -56}, {-32, 19}, {-56, -37},
       {-39, 27}, {-62, -25}, {-19, 58}, {-91, -44}, {91, 110}, {40, 110},
       {56, 130}, {-64, 6}, {-91, -75}, {-81, -50}, {-40, -27}, {-8, 3},
       {39, 43}, {34, 76}, {-65, 26}, {59, 122}, {94, 130}, {20, 27},
       {70, 163}, {-6, 37}, {-83, -41}, {84, 120}, {74, 85}, {-81, -61},
       {95, 113}, {-27, -8}, {-86, -68}, {-60, -17}, {81, 112}, {64, 131},
       {43, 142}, {-85, -2}, {4, 98}, {-78, -71}, {-86, -37}, {85, 124},
       {-25, 51}, {-51, -48}, {-9, -6}, {-74, 1}, {-63, 34}, {64, 144},
       {-36, 8}, {-19, 6}, {-22, 5}, {-14, 10}, {94, 183}, {76, 122},
       {-31, 1}, {-13, 69}, {88, 137}, {71, 110}, {-19, 19}, {13, 26},
       {-59, 31}, {24, 49}, {28, 40}, {82, 119}, {-96, 4}, {-85, -45},
       {28, 103}, {-56, 37}, {-29, 44}, {50, 97}, {-83, -18}, {-64, -1},
       {43, 122}, {-23, 65}, {-26, 26}, {-86, -20}, {60, 73}, {30, 53},
       {12, 58}, {11, 57}, {-94, -5}, {44, 114}, {-89, -48}, {-27, 5},
       {85, 161}, {92, 191}, {45, 85}, {-30, -14}, {-91, -34}, {44, 78},
       {32, 65}, {-91, -5}, {90, 166}, {96, 142}, {33, 49}, {65, 158},
       {97, 129}, {8, 63}, {4, 56}, {-19, 60}, {-90, -10}, {-21, 6},
       {16, 43}, {-30, 7}, {-48, -19}, {37, 50}, {58, 114}, {49, 120},
       {8, 39}, {-63, -11}, {42, 80}, {68, 85}, {23, 44}, {-10, 56},
       {96, 108}, {1, 11}, {96, 105}, {52, 114}, {-23, 61}, {-91, -46},
       {56, 70}, {100, 122}, {22, 32}, {-46, -38}, {-88, -24}, {-55, -23},
       {-98, -33}, {-57, -12}, {-58, 23}, {30, 97}, {-20, 26}, {64, 150},
       {-8, 69}, {39, 114}, {76, 140}, {34, 36}, {-64, -22}, {14, 28},
       {12, 13}, {-80, -61}, {85, 98}, {54, 62}, {66, 68}, {87, 183},
       {-72, -21}, {68, 88}, {-49, 20}, {-69, -63}, {-28, 60}, {-48, -37},
       {15, 71}, {2, 41}, {-13, 8}, {-79, -48}, {-17, -1}, {88, 144},
       {-82, -31}, {-67, -27}, {-51, -7}, {48, 98}, {-75, -57}, {91, 190},
       {-64, -41}, {-4, 68}, {-53, -4}, {21, 92}, {-100, -97}, {-75, 2},
       {-6, 7}, {86, 112}, {39, 101}, {52, 145}, {25, 76}, {-76, 11},
       {22, 110}, {-87, 6}
    };
    size_t nt = 10;
    auto crs = parallel_consolidate_ranges <std::list>(begin(d1), end(d1), nt);
    for (const auto& [f, s]: crs) cout << '{' << f << ", " << s << "}";
```

Barranco, 3 de noviembre 2023.