CS2013 - Programación III

Práctica Calificada #2 (PC2)

2023 - 2

Profesor: Rubén Rivas

Librería Estándar – 6 puntos

Diseñar y desarrollar el template de función **generate_sub_ranges** que permita encontrar todos los subrangos posibles de un contenedor donde la suma de sus elementos este dentro del rango especificado (**sub_range**). La funcion retornara un vector (**std::vector**) de contenedores.

Los contenedores son secuenciales y el tipo del contenedor de origen no necesariamente coincide con el tipo de contenedor de destino aunque por default el contenedor de destino es igual a **std::vector**.

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

Casos de uso #1

```
std::vector<int> v1 = {1, 2, 3, 4, 5};
auto srs = generate_sub_ranges<std::vector>(v1, {3, 7});
std::cout << "Sub-rangos con sumas de 3 a 7:" << std::endl;
for (const auto& item: srs) cout << " " << item;
```

Casos de uso #2

```
std::list<int> lst1 = {-5, -2, -3, -1, 2, 4};
vector<std::list<int>> srs = generate_sub_ranges<std::list>(lst1,
                      {-3, 1});
std::cout << "Sub-rangos con sumas de -3 a 1:" << std::endl;
```

Casos de uso #3

```
std::deque<int> d1 = {10, 20, 30, 40, 50};
auto srs = generate_sub_ranges<std::deque>(d1, {60, 100});
std::cout << "Sub-rangos con sumas de 60 a 100:" << std::endl;
for (const auto& item: srs) cout << " " << item;
```

Casos de uso #4

```
std::vector<int> v1 = {1, 2, 3, 4, 5};
auto srs = generate_sub_ranges<std::vector>(v1, {0, 15});
std::cout << "Sub-rangos con sumas de 0 a 15:" << std::endl;
for (const auto& item: srs) cout << " " << item;
```

Complejidad Algorítmica – 6 puntos

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

```
void find_triplets(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) {
            int partial_sum = *it1 + *it2;
            if (std::binary_search(std::next(it2), copy.end(),
                        target_sum - partial_sum)) {
                std::cout << "Triplet found: ("
                          << *it1 << ", " << *it2 << ", "
                          << target_sum - partial_sum << ")\n";
            }
        }
    }
}
```

Programación Concurrente – 6 puntos

Escribir la función template **count_diphthongs** que cuente la aparición de un dictongo especifico en un conjunto de documentos utilizando la tecnica de data paralela.

Un dictongo es la combinacion de dos vocales en una misma silaba, dado un dictongo específico (ai, ea, etc) la tarea es contar cuantas veces aparece en el conjunto de documentos.

Caso de uso #1

```
std::vector<std::string> documents = {
    "This is a sample text with various ai combinations and ai.",
    "Another example where the ea diphthong is present.",
    "A third document with no specific diphthong focus."
};
std::string diphthong = "io";
int total_count = 0;
for (const auto &doc : documents) {
    total_count += count_diphthongs(doc, diphthong);
}
```

Caso de uso #2

```
std::vector<std::string> words = {
        "creation", "beautiful", "train", "plane", "good"};

for (const auto &word : words) {
    int count = count_diphthongs(word, "ai");
    std::cout << "The word '" << word
              << "' has 'ai' " << count
              << " times." << std::endl;
}
```

Caso de uso #3

```
std::wstring doc1 =
        L"The train arrives at eight and then leaves at nine.";
std::wstring doc2 =
        L"The creation of this beautiful artwork took years.";

int count_doc1 = count_diphthongs(doc1, L"ai");
int count_doc2 = count_diphthongs(doc2, L"ea");

std::wcout << L"ai: " << count_doc1 << L" times." << std::endl;
std::wcout << L"ea: " << count_doc2 << L" times." << std::endl;
```

Barranco, 3 de noviembre 2023.