THU_VasiliyKlyosov

Solution for A3 : Concurrency:

First we create a template that takes vector by reference, a value to find and iterators specifying a range withing the vector to search. Returns a vector of pointerf to the found element

```
template <typename T>
std::vector<T*> find_all(std::vector<T>& v, const T& val, typename
std::vector<T>::iterator begin, typename std::vector<T>::iterator end) {
    std::vector<T*> res;
    for (auto it = begin; it != end; ++it) {
        if (*it == val) {
            res.push_back(&(*it));
        }
    }
    return res;
}
```

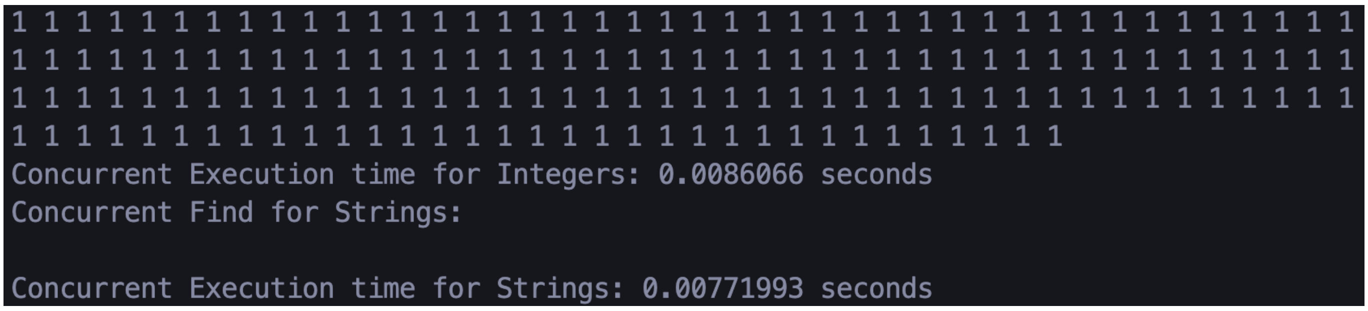
nothing interesting, after that we create an overloaded version of the function find_all and SynchronizedQueue for thread-safe operations in a queue with mutexes to avoid conflicts in concurrency

```
std::vector<int> generateRandomIntVector(size_t size, int minVal, int
maxVal) {
    // ... function implementation ...
}

std::vector<std::string> generateRandomStringVector(size_t size) {
    // ... function implementation ...
}
```

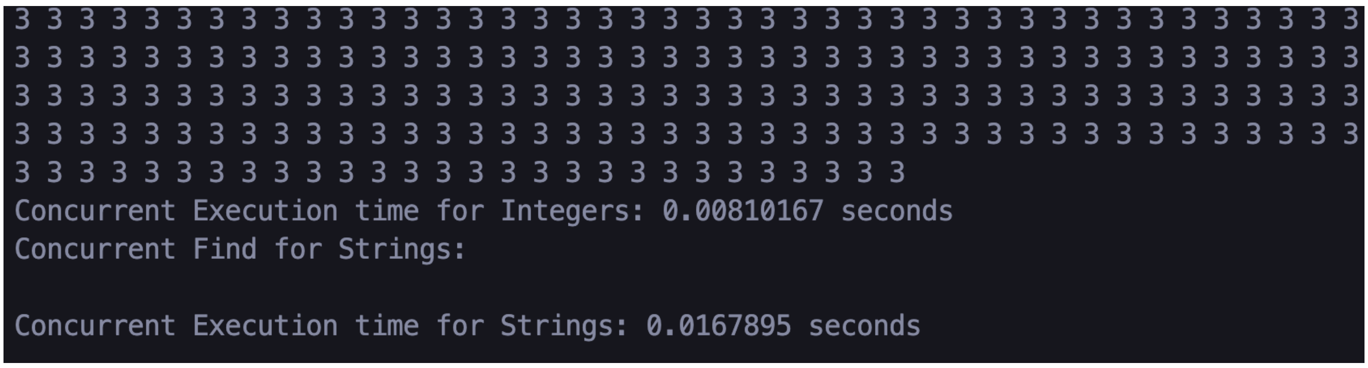
These are functions to generate random input vectors of ints and strings. For their implementation pls look at the new_find_all.cpp file

```
int main() {
    const size_t numValues = 1000000;

    // Generate input vectors
    std::vector<int> intVector = generateRandomIntVector(numValues, 1,
10);
    std::vector<std::string> stringVector =
generateRandomStringVector(numValues);

    // Test sequential find_all for integers
    auto start = std::chrono::high_resolution_clock::now();
    auto resultInt = find_all(intVector, 4);
    auto end = std::chrono::high_resolution_clock::now();

    // ... Output results and measure execution time ...

    // Test sequential find_all for strings
    start = std::chrono::high_resolution_clock::now();
    auto resultString = find_all(stringVector, std::string("Denmark"));
    end = std::chrono::high_resolution_clock::now();

    // ... Output results and measure execution time ...

    // Test concurrent find_all for integers
    start = std::chrono::high_resolution_clock::now();
    auto resultIntParallel = find_all_parallel(intVector, 4);
    end = std::chrono::high_resolution_clock::now();

    // ... Output results and measure execution time ...

    // Test concurrent find_all for strings
    start = std::chrono::high_resolution_clock::now();
    auto resultStringParallel = find_all_parallel(stringVector,
std::string("DENMARK"));
    end = std::chrono::high_resolution_clock::now();

    // ... Output results and measure execution time ...

    return 0;
}
```

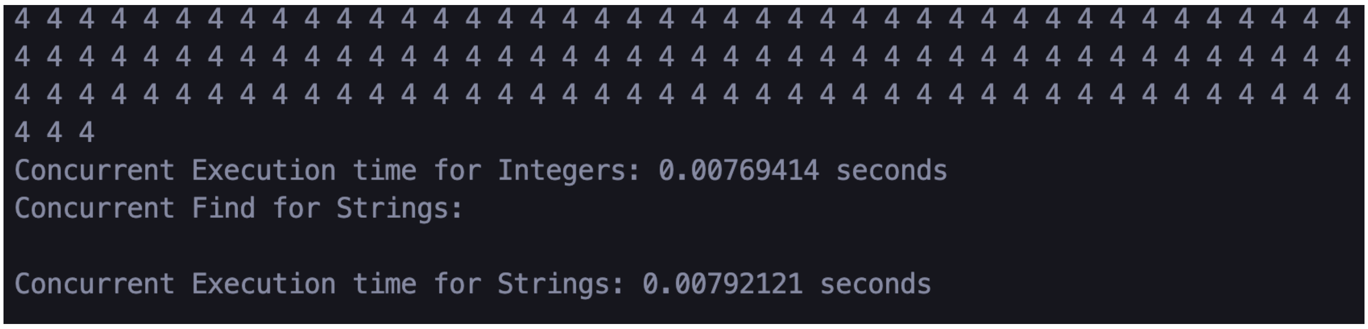
in main we generate random input vectors for ints and strings then we test it and output the results with measuring time.



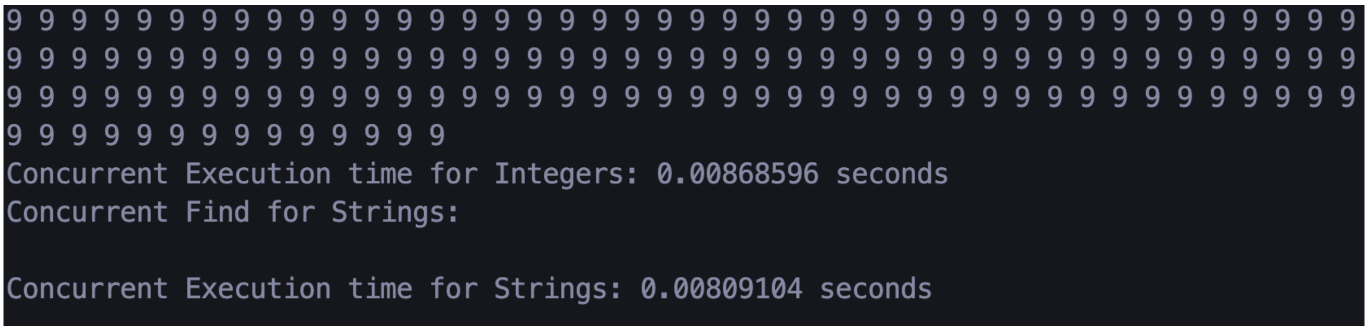
```
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
Concurrent Execution time for Integers: 0.0086066 seconds
Concurrent Find for Strings:

Concurrent Execution time for Strings: 0.00771993 seconds
```



```
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
Concurrent Execution time for Integers: 0.00810167 seconds
Concurrent Find for Strings:

Concurrent Execution time for Strings: 0.0167895 seconds
```



```
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
4 4 4
Concurrent Execution time for Integers: 0.00769414 seconds
Concurrent Find for Strings:

Concurrent Execution time for Strings: 0.00792121 seconds
```



```
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9 9
9 9 9 9 9 9 9 9 9 9 9 9 9 9
Concurrent Execution time for Integers: 0.00868596 seconds
Concurrent Find for Strings:

Concurrent Execution time for Strings: 0.00809104 seconds
```

My comments:

- In terms of efficiency the concurent implementation is better than sequenctial due to parallel processing
- The output neven showed DENMARK, obviously due to the randomness of generated input vectors and the probability of my string to be present
- but result on screenshot is efficient as expected, very nice