Activity 1.1

José Carlos Martínez Núñez, Tania Sayuri Guizado Hernández

August 23, 2022

Contents

Algorithm and Implementation

There are two main files in this project:

- `generate_numbers.cpp`: Generates random numbers (real) between 1 and 100.
- `main.cpp`: Receives via stdin N real numbers separated by a newline, sorts them in descending, using merge sort, and prints the result and the amount of time it took to perform the sort.

The algorithm that's used to sort the numbers is the following:

```
void mergeSort(vector<double> &vect)
{
    if (vect.size() > 1)
    {
        vector<double> left(vect.begin(), vect.begin() + vect.size() / 2);
        vector<double> right(vect.begin() + vect.size() / 2, vect.end());
        mergeSort(left);
        mergeSort(right);
        merge(left, right, vect);
    }
}
```

```
void merge(vector<double> &left, vector<double> &right, vector<double> &vect)
{
    vector<double>::iterator v = vect.begin();
    vector<double>::iterator l = left.begin();
    vector<double>::iterator r = right.begin();
    while (v != vect.end())
    {
        while (l != left.end() && (r == right.end() || *l >= *r))
        {
            *v = *l;
            l++;
            v++;
        }
        while (r != right.end() && (l == left.end() || *r > *l))
        {
            *v = *r;
            r++;
```

```
            v++;
        }
    }
}
```

Test Cases

For this project, we will use the following test cases:

- `10_in_reverse_order.txt`: 10 random real numbers in ascending order.
  - This test case was chosen because it's the one with the most amount of moves the algorithm has to make.
- `50_in_order.txt`: 50 random real numbers in descending order.
  - This test case was chosen because it's the one with the least amount of moves the algorithm has to make.
- `333_random.txt`: 333 random real numbers in random order.
  - This test case was chosen because it's an odd number and the algorithm has to divide it by half.
- `50000_random.txt`: 50000 random real numbers in random order.
  - This test case was chosen because it's very large and it's a good representation of the efficiency of the algorithm.

The results of each test case are located in the `results` folder.

Hardware

For this activity I used a MacBook Pro (13-inch, 2018, Four Thunderbolt 3 Ports) with the following specs:

- **Processor:** 2.3 GHz Quad-Core Intel Core i5
- **Memory:** 8 GB 2133 MHz LPDDR3
- **Graphics:** Intel Iris Plus Graphics 655 1536 MB