CSCI 230 -- Makeup Assignment

Name: Nero Li

You can use this assignment to earn up to 10 points for a missing pop quiz, a missing in-class exercise, or a missing lab. If you did not miss any assignment, you can use it to earn up to 5 extra credit points.

Which assignment do you need to make up (or EC if applicable)? EC

Given a data file of integer values, write a program to find the total number of inversions.
If value i comes before value j in the file and value i is larger than value j then it is an inversion. We need to count all such pairs in the file and output it to the screen. This count would tell us how closely the file is sorted. For example, the file with 1 9 6 4 5 has 5 inversions: (9, 6), (9, 4), (9, 5), (6, 4), (6, 5).

You need to provide two different algorithms to solve this problem, a simple O(n^2) algorithm with a nested loop and a fast divide-and-conquer algorithm. Once you confirm that they work on a small file like the 5 values above, run the two data files, small1k.txt and large100k.txt, used in PA 4.

What is the running time for your fast divide-and-conquer algorithm? You can either do a running time analysis or an experimental analysis by collecting run times.

If we are using simple inversion check:
Running time for small1k.txt is 5.0047 millisecond.
Running time for large100k.txt is 49339.1 millisecond.
By checking the gap between two experience, it meet the O(n^2) running time.

If we are using merge inversion check:
Running time for small1k.txt is 19.0178 millisecond.
Running time for large100k.txt is 219.197 millisecond.
By checking the gap between two experience, it meet the O(nlogn) running time.

Copy/paste your source code and output below.

makeup.cpp:

```
/*  Program: Makeup_assignment
    Author: Nero Li
    Class: CSCI 230
    Date: 06/02/2022
```

```
    Description:
        Given a data file of integer values, write a program to find
        the total number of inversions. If value i comes before value
        j in the file and value i is larger than value j then it is an
        inversion. We need to count all such pairs in the file and
        output it to the screen.  This count would tell us how closely
        the file is sorted

    I certify that the code below is my own work.

      Exception(s): N/A

*/

#include <iostream>
#include <fstream>
#include <vector>
#include <ctime>
#include <chrono>

using namespace std;

void merge(vector<int> &vec, vector<int> &a, vector<int> &b, long long
&count, bool withOutput)
{
    int i = 0, j = 0;

    while (i < a.size() && j < b.size())
    {
        if (a[i] > b[j])
        {
            if (withOutput)
                for (int k = i; k < a.size(); ++k)
                    cout << "(" << a[k] << ", " << b[j] << ")\n";
            count += a.size() - i;
            vec.push_back(b[j++]);
        }
        else
            vec.push_back(a[i++]);
    }

    while (i < a.size() || j < b.size())
    {
        if (i < a.size())
            vec.push_back(a[i++]);
        else
```

```
            vec.push_back(b[j++]);
    }
}

void mergeInversion(vector<int> &vec, long long &count, bool
withOutput)
{
    if (vec.size() <= 1)
        return;

    vector<int> a;
    vector<int> b;

    int i = vec.size() / 2;
    vector<int>::iterator mid = vec.begin() + i;

    a.assign(vec.begin(), mid);
    b.assign(mid, vec.end());

    mergeInversion(a, count, withOutput);
    mergeInversion(b, count, withOutput);
    vec.clear();
    merge(vec, a, b, count, withOutput);
}

long long simpleInversion(vector<int> vec, bool withOutput = false)
{
    long long count{0};

    for (int i = 0; i < vec.size(); ++i)
        for (int j = i + 1; j < vec.size(); ++j)
            if (vec[i] > vec[j])
            {
                if (withOutput)
                    cout << "(" << vec[i] << ", " << vec[j] << ")\n";
                ++count;
            }

    return count;
}

void testSimple(string str, bool checkTime = true, bool withOutput =
false)
{
    vector<int> vec;
    ifstream fin;
```

```
    long long count = 0;
    fin.open(str, ios::binary);

    if (!fin)
    {
        cout << "err\n";
        return;
    }

    while (!fin.eof())
    {
        int value;
        fin >> value;
        vec.push_back(value);
    }

    cout << "Simple Inversion for " << str << ":\n";

    auto start = chrono::high_resolution_clock::now();
    count = simpleInversion(vec, withOutput);
    auto end = chrono::high_resolution_clock::now();

    cout << "Inversion count: \t" << count << endl;

    if (checkTime)
        cout << "Time used:\t\t" <<
(chrono::duration_cast<chrono::nanoseconds>(end - start).count() *
(double)1e-6) << " ms" << endl;

    cout << endl;
    fin.close();
}

void testMerge(string str, bool checkTime = true, bool withOutput =
false)
{
    vector<int> vec;
    ifstream fin;
    long long count = 0;
    fin.open(str, ios::binary);

    if (!fin)
    {
        cout << "err\n";
        return;
    }
```

```
    while (!fin.eof())
    {
        int value;
        fin >> value;
        vec.push_back(value);
    }

    cout << "Merge Inversion for " << str << ":\n";

    auto start = chrono::high_resolution_clock::now();
    mergeInversion(vec, count, withOutput);
    auto end = chrono::high_resolution_clock::now();

    cout << "Inversion count: \t" << count << endl;

    if (checkTime)
        cout << "Time used:\t\t" <<
(chrono::duration_cast<chrono::nanoseconds>(end - start).count() *
(double)1e-6) << " ms" << endl;

    cout << endl;
    fin.close();
}

int main()
{
    testSimple("test.txt", false, true);
    testSimple("small1k.txt");
    testSimple("large100k.txt");

    testMerge("test.txt", false, true);
    testMerge("small1k.txt");
    testMerge("large100k.txt");

    cout << "Author: Nero Li\n";

    return 0;
}
```

I/O below:

```
Simple Inversion for test.txt:
(9, 6)
(9, 4)
(9, 5)
```

```
(6, 4)
(6, 5)
Inversion count:        5

Simple Inversion for small1k.txt:
Inversion count:        246372
Time used:              5.0047 ms

Simple Inversion for large100k.txt:
Inversion count:        2407913387
Time used:              49339.1 ms

Merge Inversion for test.txt:
(6, 4)
(6, 5)
(9, 4)
(9, 5)
(9, 6)
Inversion count:        5

Merge Inversion for small1k.txt:
Inversion count:        246372
Time used:              19.0178 ms

Merge Inversion for large100k.txt:
Inversion count:        2407913387
Time used:              219.197 ms

Author: Nero Li
```

Save as PDF format and submit this file.