BLG335E Aybike Zeynep Taşkın

HW1 150190004

a) Quicksort code and "How2Run" section:

I wrote the quicksort code in quicksort.cpp and provided the struct in books.h file.

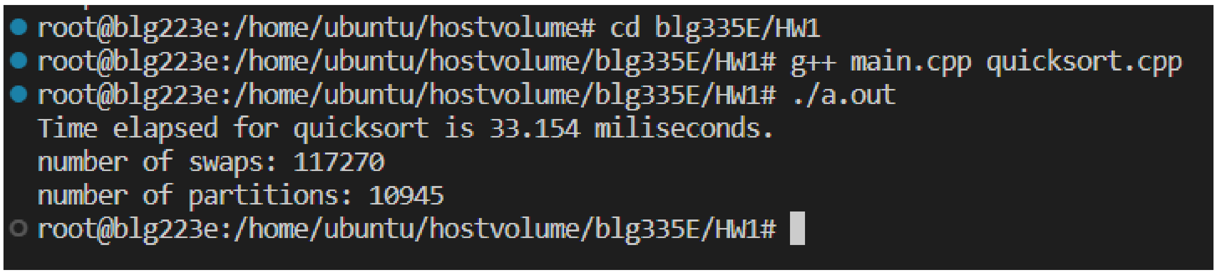
Quicksort.cpp file:

```cpp
#include <iostream>
#include <string>
#include "books.h"
using namespace std;


int num_of_swaps = 0;
int num_of_partitions = 0;

int Partition(Books array[], int low, int high)
{
    num_of_partitions++;
    float pivot = array[high].average_rating;
    int i = low;
    int j = low;
    while (j >= i)
    {
        while (array[i].average_rating <= pivot && array[j].average_rating <=
pivot && j < high)
        {
            j++;
            i++;
        }
        while (array[j].average_rating > pivot && j < high)
        {
            j++;
            swap(array[i], array[j]);
            num_of_swaps++;
            if (array[i].average_rating <= pivot)
            {
                i++;
            }

        }
        if (j == high)
        {
            swap(array[i], array[high]);
            num_of_swaps++;
            break;
        }

    }
    return i;
}

void quicksort(Books array[], int low, int high)
{
    if (low < high)
    {
        int i = Partition(array, low, high);
        quicksort(array, low, i - 1);
        quicksort(array, i+1, high);
    }
}
```

Books.h file:

```
#include <string>
using namespace std;
extern int num_of_swaps;
extern int num_of_partitions;

struct Books
{
    string bookId;
    string title;
    string authors;
    float average_rating;
    string isbn;
    string isbn13;
    string language_code;
    string num_pages;
    string ratings_count;
    string text_reviews_count;
    string publication_date;
    string publisher;
};
```

How2Run:
To run my code, it is needed to have all the main.cpp, quicksort.cpp, books.h, books.txt, books_half.txt and books_quarter.txt files as they will be provided in the zip file I will send to Ninova. Then after connecting to the docker environment and getting into the folder that contains the homework, writing "g++ main.cpp quicksort.cpp" and then "./a.out" to the terminal will give the output. I am giving the example terminal screenshot below.



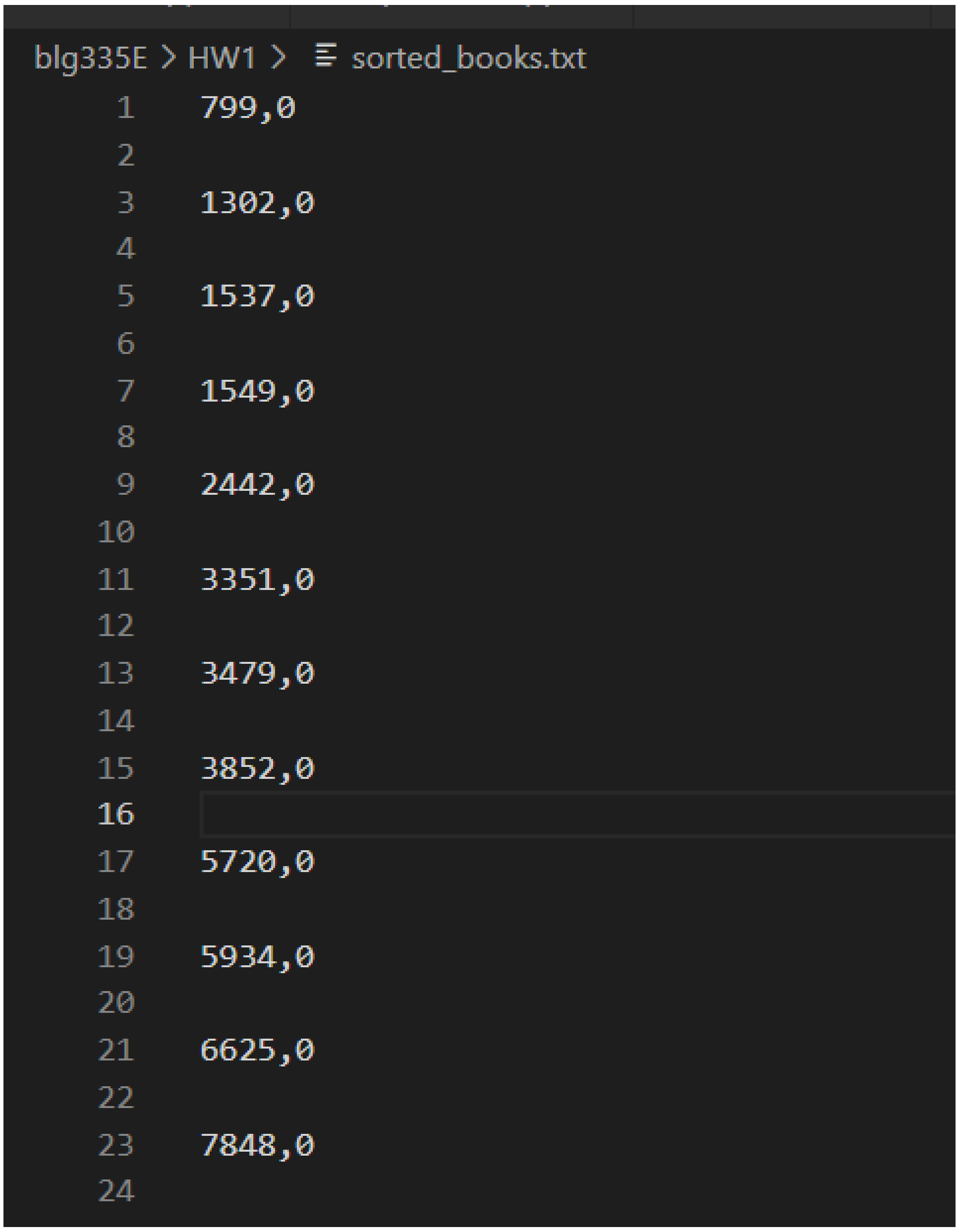
```
root@blg223e:/home/ubuntu/hostvolume# cd blg335E/HW1
root@blg223e:/home/ubuntu/hostvolume/blg335E/HW1# g++ main.cpp quicksort.cpp
root@blg223e:/home/ubuntu/hostvolume/blg335E/HW1# ./a.out
Time elapsed for quicksort is 33.154 miliseconds.
number of swaps: 117270
number of partitions: 10945
root@blg223e:/home/ubuntu/hostvolume/blg335E/HW1# 
```

b) I read the data from books.txt file and then rewrite the sorted result into sorted_books.txt file in main.cpp file. The sorted result can be accessed from the sorted_txt file. A small portion (first few bookId and average_ratings) of the sorted data is given below.

```cpp
#include <iostream>

#include <iomanip>
#include <string>
#include <fstream>
#include <stdlib.h>
#include <cstdlib>
#include <ctime>
#include "books.h"
using namespace std;

void quicksort(Books array[], int low, int high);

int main(int argc, char*argv[]){
    int num=0;
    string lines;
    int linecount=0;
    char c;
    ifstream books1file;
    books1file.open("books.txt");
    Books* array;

    if (!books1file){
        cerr << "books.txt file could not be opened!";
        exit(1);
    }
    while(!books1file.eof()){
        //count num of lines to create array of size lines
        getline(books1file, lines);
        linecount++;
    }
    array = new Books[linecount];
    books1file.close();
    ifstream booksfile;
    booksfile.open("books.txt");

    //get data that are separated by ","
    char separate = ',';
    float float_data;
    string data[12];
    getline(booksfile, lines); //first line
    for(int i=0;i<linecount-2;i++){
        // bookId,title, authors,
average_rating,isbn,isbn13,language_code,num_pages
        //ratings_count,test_reviews_count,publication_date,publisher
        getline(booksfile,lines);
        stringstream line(lines);

        for(int j=0;j<12;j++){
            getline(line,data[j],separate);
        }
        array[i].bookId=data[0];
        array[i].title=data[1];
        array[i].authors=data[2];
        float_data = stof(data[3]);
        array[i].average_rating=float_data;
        array[i].isbn=data[4];
        array[i].isbn13=data[5];
        array[i].language_code=data[6];
        array[i].num_pages=data[7];
        array[i].ratings_count=data[8];
        array[i].text_reviews_count=data[9];
        array[i].publication_date=data[10];
        array[i].publisher=data[11];
        getline(line, lines, '\n'); //go to next row and continue iterating
    }

    booksfile.close();

    clock_t time = clock();
    quicksort(array, 0 , linecount-1);
    time = clock() - time;


    //create new sorted file
    ofstream outfile("sorted_books.txt");
    if (!outfile){
        cerr << "sorted_books.txt file could not be opened!";
        exit(1);
    }
    for(int i=0;i<linecount-1;i++){
        //outfile << array[i].bookId<<","<< array[i].title<<","<<
array[i].authors<<",";
        //outfile << array[i].average_rating<<","<< array[i].isbn<<","<<
array[i].isbn13<<","<<array[i].language_code<<","<< array[i].num_pages<<",";
```

```cpp
        //outfile << array[i].ratings_count<<","<<
array[i].text_reviews_count<<","<< array[i].publication_date<<","<<
array[i].publisher<<"\n";
        outfile<< array[i].bookId<<","<<array[i].average_rating<<"\n"<<endl;
    }
    outfile.close();

    float total_time = 1000*(float)time / CLOCKS_PER_SEC;
    cout<<"Time elapsed for quicksort is "<<total_time<<" miliseconds."<<endl;
    cout << "number of swaps: " << num_of_swaps << endl;
    cout << "number of partitions: " << num_of_partitions << endl;
    return 0;
}
```

First few rows of sorted data (bookId, average_rating):

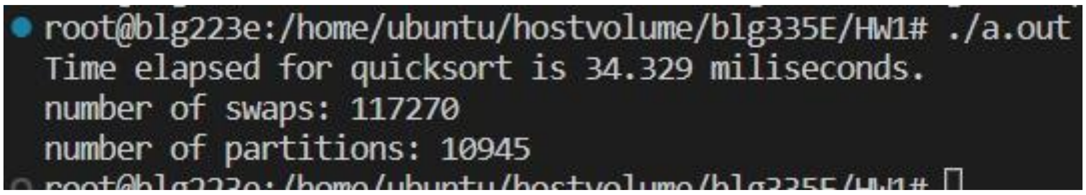


blg335E > HW1 > sorted_books.txt

```
799,0

1302,0

1537,0

1549,0

2442,0

3351,0

3479,0

3852,0

5720,0

5934,0

6625,0

7848,0

```

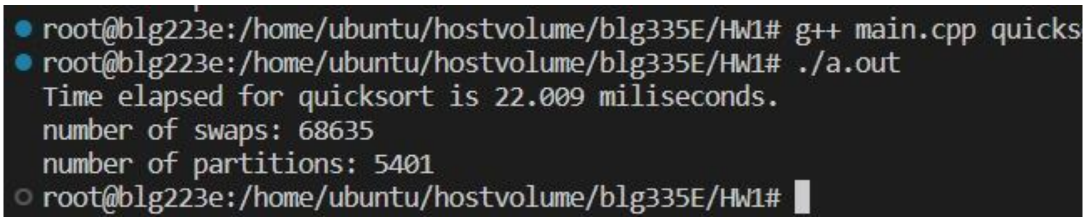
c) I calculated the time elapsed using the clock() function. And also calculated number of swaps and number of partitions and outputted them in my code. Also to compare the calculations and observe the change in execution time I also run the same code for books_half.txt and books_quarter.txt files by changing the file names in the main.cpp file. As it can be seen, there was significant change in time depending on the number of inputs.

Time elapsed, number of swaps and number of partitions for the whole data:



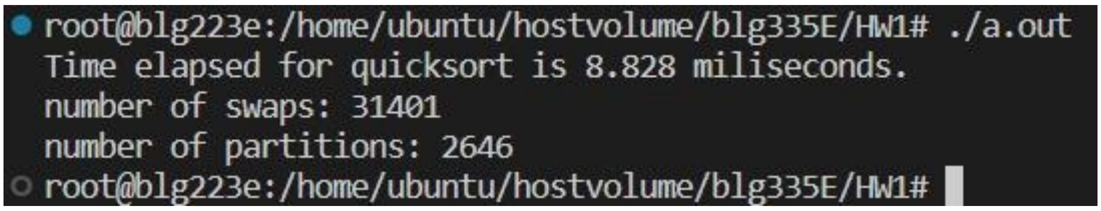
```
● root@blg223e:/home/ubuntu/hostvolume/blg335E/HW1# ./a.out
  Time elapsed for quicksort is 34.329 miliseconds.
  number of swaps: 117270
  number of partitions: 10945
```

Time elapsed, number of swaps and number of partitions for the half of the data:



```
● root@blg223e:/home/ubuntu/hostvolume/blg335E/HW1# g++ main.cpp quicks
● root@blg223e:/home/ubuntu/hostvolume/blg335E/HW1# ./a.out
  Time elapsed for quicksort is 22.009 miliseconds.
  number of swaps: 68635
  number of partitions: 5401
○ root@blg223e:/home/ubuntu/hostvolume/blg335E/HW1# █
```

Time elapsed, number of swaps and number of partitions for the quarter of the data:

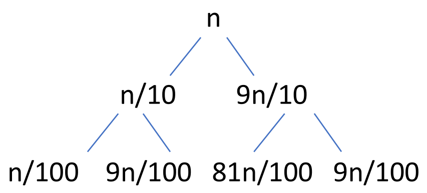


```
● root@blg223e:/home/ubuntu/hostvolume/blg335E/HW1# ./a.out
  Time elapsed for quicksort is 8.828 miliseconds.
  number of swaps: 31401
  number of partitions: 2646
○ root@blg223e:/home/ubuntu/hostvolume/blg335E/HW1# █
```

d) Quicksort asymptotic upper bound for the best case: O(n.lgn)
Quicksort asymptotic upper bound for the average case: O(n.lgn)
Quicksort asymptotic upper bound for the worst case: $O(n^2)$

Best case happens when pivot is in the middle and partitioned arrays will have n/2 elements. For the best case, recurrence equation is: $T(n) <= 2T(n/2) + \Theta(n)$
Solving by the master method:
b=2 a=2
$n^{\log_b a} = n^{\log_2 2} = n = f(n)$
therefore, as a result of case 2 of master method, upper bound for best case is
T(n) = O(nlgn)

Average case happens when pivot is at a random ith position of the array and array will be partitioned into length i and n-i-1. Suppose we take the case where split is 9 to 1. Then the recurrence would be $T(n) <= T(9n/10) + T(n/10) + \Theta(n) = T(9n/10) + T(n/10) + cn$
We solve this using recursion tree.



Tree goes on as n, 9n/10+n/10, n/100+9n/100+81n/100+9n/100… So, at each level we have ‘n’ for lgn steps. Therefore, the upper bound for average case becomes T(n) = O(nlgn)

Worst case happens when pivot is the highest or lowest element when array is sorted or reverse sorted. For the worst case recurrence equation is
$T(n) = T(n-1) + T(0) + \Theta(n) = T(n-1) + \Theta(n)$
By solving it with iteration:
$T(n) = T(n-1) + \Theta(n)$
$T(n) = \Theta(n) + \Theta(n-1) + \Theta(n-2) + \ldots + \Theta(1) = (n(n+1)/2) = \Theta(n^2)$
Therefore $T(n) = O(n^2)$

e) Worst case scenario for our Quicksort method would be when the array is given as sorted or reverse sorted. To reduce the time complexity in those scenarios we can use a randomized quicksort method where we choose pivot randomly. That would reduce the time complexity. Or another method could be to randomly mix the given input array before sorting so that we can get rid of the sorted array situation and we would get the average time complexity as a result.