Project 1 Report
Algorithms CSCI 3330
Fall 2018

Group Members:

Aung Kyaw Min

Hassan Khan

Chris Laxton

Interpretation of Project

We were given five different sources and we had to use three algorithms to sort all five sources. We also calculate the inversions by taking the sum of the rankings for the same documents but from different sources and sorted them. For each element that moves an index, an inversion is added. The purpose of this project is to evaluate the source reliability by counting inversion through sorting five sources. Most of the search engines, such as Yahoo, Bing, Google, Baidu, DuckDuckGo, use special algorithms to generate the search result. Google uses automated programs called spiders or crawlers. Google ranks the websites based on frequency and location of keywords of a web page. One of the methods to find the match results is to count the inversions of sorted list. An inversion can be defined as the number of swaps we need to try to get the sorted data. The source is more reliable if it has fewer inversions.

Methodology of the solution

The three algorithms we used in this project were quick sort, merge sort and insertion sort. The average running time for both quick sort and merge sort is $O(n \log n)$ and insertion sort takes $O(n^2)$.

- In the quick sort, we incremented the inversion by comparing the index of the pivot versus the index of the lesser or greater value.
- In merge sort, we incremented the inversion if the value of the right-side element is bigger than the value on the left side.
- In insertion sort, we incremented the inversion if the value we pick is higher than the previous value.

Methods we used:

For MergeSort:

public static ArrayList<Integer> MergeSortMain(ArrayList<Integer> a): to perform merge sort by dividing.

- Input: unsorted ArrayList
- Process: Divide the ArrayList into half until the smallest half becomes a single element.
- Output: Sorted ArrayLists

Public static ArrayList<Integer> MergeSortMerge(ArrayList<Integer> a, ArrayList<Integer> b): to conquer the merge sort, bottom up.

- Input: two unsorted small ArrayLists
- Process: Merge the two ArrayLists from MergeSortMain
- Output: Sorted ArrayList

For InsertionSort:

`public static ArrayList<Integer> InsertionSort(ArrayList<Integer> a): to perform insertion sort`

- Input: unsorted ArrayList
- Process: Pick the second element and compare with first element and if the second is less than the first, swap it and pick the third element, compare with first two elements and repeat the process. This process ends when the last element compares with all the previous elements.
- Output: Sorted ArrayList

For QuickSort:

public static ArrayList<Integer> QuickSortMain(ArrayList<Integer> a): to perform quicksort by pivoting

- input: unsorted ArrayList
- Process: Generate a random pivot then begin to sort the values between lessthan and greaterthan. Lessthan will include all values lessthan, greaterthan includes all values greaterthan or equal to. Recursively call QuickSort for each lessthan and greaterthan and return the final ArrayList.
- Output: Sorted ArrayList

Classes Created:

- Product_io.java: read the file
- Product.java: create the row of the table
- Table.java: contains all the methods for a table
- Sort.java: contains all the sorting methods

First, we inserted 5 sources into different ArrayLists. Then, we initiate and declare the product, and fill the product with data and we sorted the table according to the sum values of the products. After that, we created a new table to insert all the data which has been sorted according to the sum of the products. Later, we count the inversions of individual source by using three algorithms. We get the inversions by passing the data into the sorting functions.

Results

Table 1.1 (First Iteration)

Sources	Inversions by QuickSort	Inversions by MergeSort	Inversions by InsertionSort	Reliability
Source1	17441666	17457269	17457269	20.096%
Source2	17441666	17473327	17473327	20.077%
Source3	17403915	17431541	17431541	20.125%
Source4	17711318	17737121	17737121	19.770%
Source5	17589351	17610435	17610435	19.920%

Table 1.2 (10^{th} Iteration)

Sources	Inversions by QuickSort	Inversions by MergeSort	Inversions by InsertionSort	Reliability
Source1	16964499	16961050	16961050	20.633%
Source2	18288882	18276559	18276559	19.149%
Source3	17540640	17525968	17525968	19.969%
Source4	17762438	17761974	17761974	19.703%
Source5	17025119	17034236	17034236	20.545%

NOTE: Attached at the end of this report is a copy of the first 10 iterations from our data txt file we wrote into.

Conclusion:

Based on the results, QuickSort gave us a little bit different data, but the error could be in the inversion counting in the function. Insertion and Merge gave us the same inversions. The fastest sorting algorithm is quick sort, then second is the merge sort, and then the insertion sort. While evaluating the dataset we acquired through iterating we noticed that the inversions seem to act like a sine curve. They will become better and more reliable, and then get worse and less reliable repeatedly. The 10^{th} iteration is where many of the sources hit their lowest number of inversions. The first 10 iterations were represented in this dataset to represent the highs and lows of the curve. It appears Source1 kept the most consistent reliability. This would make it the most reliable source out of the datasets we have tested.

Discussion and Contributions by team members

We divided the team into two groups. We had a programming team and report team. Hassan Khan worked on MergeSort and QuickSort. Chris Laxton worked on QuickSort and InsertionSort. Aung Kyaw Min worked on the report. Future work could include experimenting with several different sorting algorithms to find the best runtime for sorting through the rankings. We would also like to experiment and track if the inversions ever steady out, or they will continuously be in the sine curve that was observed through our data. This project brought us a deeper understanding of sorting algorithms and the true power they hold.

```
Iteration number: 0
Printing inversions as calculated using MergeSort:
Source 1 inversions: 17457269
Source 2 inversions: 17473327
Source 3 inversions: 17431541
Source 4 inversions: 17737121
Source 5 inversions: 17610435
Printing inversions as calculated using InsertionSort:
Source 1 inversions: 17457269
Source 2 inversions: 17473327
Source 3 inversions: 17431541
Source 4 inversions: 17737121
Source 5 inversions: 17610435
Printing inversions as calculated using QuickSort:
Source 1 inversions: 17436579
Source 2 inversions: 17454256
Source 3 inversions: 17406683
Source 4 inversions: 17712135
Source 5 inversions: 17585700
Reliability for Iteration number: 0
Reliability for Source 1: 20.0961329423124%
Reliability for Source 2: 20.077664582834025%
Reliability for Source 3: 20.12579373241002%
Reliability for Source 4: 19.779061041010035%
Reliability for Source 5: 19.921347701433515%
Iteration number: 1
Printing inversions as calculated using MergeSort:
```

```
Source 1 inversions: 17170776
Source 2 inversions: 17973133
Source 3 inversions: 17304529
Source 4 inversions: 17740075
Source 5 inversions: 17472323
Printing inversions as calculated using InsertionSort:
Source 1 inversions: 17170776
Source 2 inversions: 17973133
Source 3 inversions: 17304529
Source 4 inversions: 17740075
Source 5 inversions: 17472323
Printing inversions as calculated using QuickSort:
Source 1 inversions: 17181824
Source 2 inversions: 17983106
Source 3 inversions: 17308423
Source 4 inversions: 17751383
Source 5 inversions: 17498324
Reliability for Iteration number: 1
Reliability for Source 1: 20.415333878146903%
Reliability for Source 2: 19.503952143360507%
Reliability for Source 3: 20.2575363446569%
Reliability for Source 4: 19.760182842633007%
Reliability for Source 5: 20.062994791202684%
Iteration number: 2
Printing inversions as calculated using MergeSort:
Source 1 inversions: 17108788
Source 2 inversions: 18078503
```

```
Source 3 inversions: 17428385
Source 4 inversions: 17714966
Source 5 inversions: 17373640
Printing inversions as calculated using InsertionSort:
Source 1 inversions: 17108788
Source 2 inversions: 18078503
Source 3 inversions: 17428385
Source 4 inversions: 17714966
Source 5 inversions: 17373640
Printing inversions as calculated using QuickSort:
Source 1 inversions: 17113766
Source 2 inversions: 18093325
Source 3 inversions: 17430060
Source 4 inversions: 17722512
Source 5 inversions: 17392375
Reliability for Iteration number: 2
Reliability for Source 1: 20.4978506826448%
Reliability for Source 2: 19.39836406169868%
Reliability for Source 3: 20.121966674531755%
Reliability for Source 4: 19.796446828429083%
Reliability for Source 5: 20.185371752695698%
Iteration number: 3
Printing inversions as calculated using MergeSort:
Source 1 inversions: 17499744
Source 2 inversions: 17759268
Source 3 inversions: 17351440
Source 4 inversions: 17581426
```

```
Source 5 inversions: 17515358
Printing inversions as calculated using InsertionSort:
Source 1 inversions: 17499744
Source 2 inversions: 17759268
Source 3 inversions: 17351440
Source 4 inversions: 17581426
Source 5 inversions: 17515358
Printing inversions as calculated using QuickSort:
Source 1 inversions: 17509539
Source 2 inversions: 17774062
Source 3 inversions: 17355652
Source 4 inversions: 17583036
Source 5 inversions: 17510337
Reliability for Iteration number: 3
Reliability for Source 1: 20.046522927616785%
Reliability for Source 2: 19.753574281123125%
Reliability for Source 3: 20.217861984485737%
Reliability for Source 4: 19.953388275590324%
Reliability for Source 5: 20.02865253118404%
Iteration number: 4
Printing inversions as calculated using MergeSort:
Source 1 inversions: 17333284
Source 2 inversions: 17809043
Source 3 inversions: 17357892
Source 4 inversions: 17482146
Source 5 inversions: 17797677
Printing inversions as calculated using InsertionSort:
```

```
Source 1 inversions: 17333284
Source 2 inversions: 17809043
Source 3 inversions: 17357892
Source 4 inversions: 17482146
Source 5 inversions: 17797677
Printing inversions as calculated using QuickSort:
Source 1 inversions: 17359917
Source 2 inversions: 17825076
Source 3 inversions: 17408622
Source 4 inversions: 17488470
Source 5 inversions: 17811722
Reliability for Iteration number: 4
Reliability for Source 1: 20.25414959155621%
Reliability for Source 2: 19.71307091515285%
Reliability for Source 3: 20.225435616124457%
Reliability for Source 4: 20.08168374874536%
Reliability for Source 5: 19.725660128421104%
Iteration number: 5
Printing inversions as calculated using MergeSort:
Source 1 inversions: 17061694
Source 2 inversions: 17952843
Source 3 inversions: 17522996
Source 4 inversions: 17408724
Source 5 inversions: 17586437
Printing inversions as calculated using InsertionSort:
Source 1 inversions: 17061694
Source 2 inversions: 17952843
```

```
Source 3 inversions: 17522996
Source 4 inversions: 17408724
Source 5 inversions: 17586437
Printing inversions as calculated using QuickSort:
Source 1 inversions: 17068784
Source 2 inversions: 17988098
Source 3 inversions: 17528744
Source 4 inversions: 17431263
Source 5 inversions: 17597393
Reliability for Iteration number: 5
Reliability for Source 1: 20.515918021428643%
Reliability for Source 2: 19.49754233516534%
Reliability for Source 3: 19.975825820584177%
Reliability for Source 4: 20.10694843686824%
Reliability for Source 5: 19.9037653859536%
Iteration number: 6
Printing inversions as calculated using MergeSort:
Source 1 inversions: 17091946
Source 2 inversions: 17864067
Source 3 inversions: 17584442
Source 4 inversions: 17786732
Source 5 inversions: 17443686
Printing inversions as calculated using InsertionSort:
Source 1 inversions: 17091946
Source 2 inversions: 17864067
Source 3 inversions: 17584442
Source 4 inversions: 17786732
```

```
Source 5 inversions: 17443686
Printing inversions as calculated using QuickSort:
Source 1 inversions: 17129340
Source 2 inversions: 17872402
Source 3 inversions: 17584030
Source 4 inversions: 17788779
Source 5 inversions: 17453054
Reliability for Iteration number: 6
Reliability for Source 1: 20.53580312458688%
Reliability for Source 2: 19.648204351207877%
Reliability for Source 3: 19.960646954121515%
Reliability for Source 4: 19.733632849150734%
Reliability for Source 5: 20.121712720932987%
Iteration number: 7
Printing inversions as calculated using MergeSort:
Source 1 inversions: 17239257
Source 2 inversions: 18011418
Source 3 inversions: 17305013
Source 4 inversions: 17557889
Source 5 inversions: 17417619
Printing inversions as calculated using InsertionSort:
Source 1 inversions: 17239257
Source 2 inversions: 18011418
Source 3 inversions: 17305013
Source 4 inversions: 17557889
Source 5 inversions: 17417619
Printing inversions as calculated using QuickSort:
```

```
Source 1 inversions: 17244949
Source 2 inversions: 18019930
Source 3 inversions: 17314442
Source 4 inversions: 17565580
Source 5 inversions: 17427278
Reliability for Iteration number: 7
Reliability for Source 1: 20.304808850253888%
Reliability for Source 2: 19.43432876722318%
Reliability for Source 3: 20.22765415908997%
Reliability for Source 4: 19.9363271105019%
Reliability for Source 5: 20.096881112931055%
Iteration number: 8
Printing inversions as calculated using MergeSort:
Source 1 inversions: 17076155
Source 2 inversions: 18125911
Source 3 inversions: 17212490
Source 4 inversions: 17678916
Source 5 inversions: 17448569
Printing inversions as calculated using InsertionSort:
Source 1 inversions: 17076155
Source 2 inversions: 18125911
Source 3 inversions: 17212490
Source 4 inversions: 17678916
Source 5 inversions: 17448569
Printing inversions as calculated using QuickSort:
Source 1 inversions: 17093488
Source 2 inversions: 18141389
```

Source 3 inversions: 17226838

Source 4 inversions: 17700093

Source 5 inversions: 17466059

Reliability for Iteration number: 8

Reliability for Source 1: 20.49715349961639%

Reliability for Source 2: 19.310067858400476%

Reliability for Source 3: 20.334801669055075%

Reliability for Source 4: 19.798304993195877%

Reliability for Source 5: 20.059671979732176%

Iteration number: 9

Printing inversions as calculated using MergeSort:

Source 1 inversions: 17167618

Source 2 inversions: 17943585

Source 3 inversions: 17378206

Source 4 inversions: 17821173

Source 5 inversions: 17419959

Printing inversions as calculated using InsertionSort:

Source 1 inversions: 17167618

Source 2 inversions: 17943585

Source 3 inversions: 17378206

Source 4 inversions: 17821173

Source 5 inversions: 17419959

Printing inversions as calculated using QuickSort:

Source 1 inversions: 17178477

Source 2 inversions: 17953745

Source 3 inversions: 17377811

Source 4 inversions: 17842190

```
Source 5 inversions: 17427397
Reliability for Iteration number: 9
Reliability for Source 1: 20.435360794855654%
Reliability for Source 2: 19.5516374627468%
Reliability for Source 3: 20.187726400866268%
Reliability for Source 4: 19.685935856617473%
Reliability for Source 5: 20.1393394849138%
Iteration number: 10
Printing inversions as calculated using MergeSort:
Source 1 inversions: 16961050
Source 2 inversions: 18276559
Source 3 inversions: 17525968
Source 4 inversions: 17761974
Source 5 inversions: 17034236
Printing inversions as calculated using InsertionSort:
Source 1 inversions: 16961050
Source 2 inversions: 18276559
Source 3 inversions: 17525968
Source 4 inversions: 17761974
Source 5 inversions: 17034236
Printing inversions as calculated using QuickSort:
Source 1 inversions: 16965711
Source 2 inversions: 18291419
Source 3 inversions: 17532236
Source 4 inversions: 17768059
Source 5 inversions: 17042360
Reliability for Iteration number: 10
```

Reliability for Source 1: 20.633870830099998%

Reliability for Source 2: 19.148687470549074%

Reliability for Source 3: 19.968775220174038%

Reliability for Source 4: 19.703447137873937%

Reliability for Source 5: 20.545219341302957%