Project Scope and Approach Document

Find source reliability by counting inversions during sorting

Team Members

The following are the group members who are part of the team

1.Sai Sathvick Chirakaka

2.Nikhila Ratakonda

3.Alekhya Satraboina

1.Interpretation of the problem

The main applications in the context of the Web includes creating search engines, selecting documents based on the user query. For instance, when we are using Google's, Yahoo or Naver search engine to find results of the user query, we know that the search engines will always give long list of search results. For example, now-a-days search engines are trying to integrate the information from different sources to get better search results. Initially every source is treated equally i.e. they calculate the total sum of the all ranks from different sources of each web page. Now in order to generate the combined rank we need to use the summation of the previous step.

In order to find the reliability of each and every source we need to assign highest weight to the most reliable source in the future rank combination by defining the reliability which is inversely proportional to the number of inversions between the ranks from source with the combined rank which therefore implies that the source is more reliable having fewer inversions.

The inversions of an array indicate; how many changes are required to convert the array into its sorted form. When an array is already sorted, it needs 0 inversions, and in another case, the number of inversions will be maximum, if the array is reversed.

2.Methodology of the solution

1. First we have read the five source files which are in the text file and listed them in an array using comma as a separator.

2. Then we have initialized the weights of all sources to one.

3. We then multiplied the first weight value with each element in the array of first source file and stored in a new array, this process is repeated for all the given five source files.

4. After getting the new array values of all the five source files we then added all the elements having same index value for all the five source files.

5. By doing the above step we got an array of elements which is not sorted

6. We then applied general sorting technique to sort the elements of the resulted sum array and then found the indices of the array and reordered all the five source arrays in the similar manner

7. We then sorted the five reordered arrays using three sorting techniques such as Quick, Merge and Insertion sort and calculated the number of inversions for each of the source files

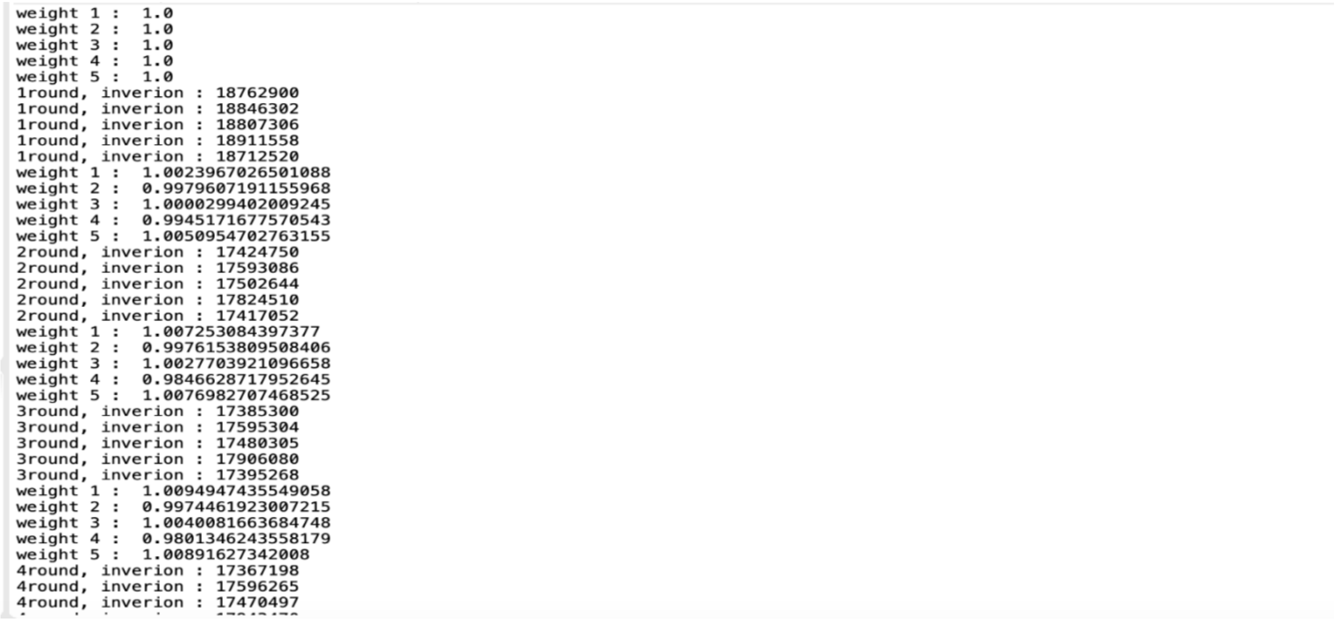
8. Now we calculated the "Qualities" of all the five arrays using their inversion values

9. We then calculated the "Normalization" [that is calculating the weights] of all the five using their respective qualities value

10. After getting the weights for all the five arrays we have replaced the initial weights which we have initialized to one to the above obtained respective weights and then the process repeats until we have the same inversion values for the previous and the current iterations for all the sorting techniques to get the stabilized value

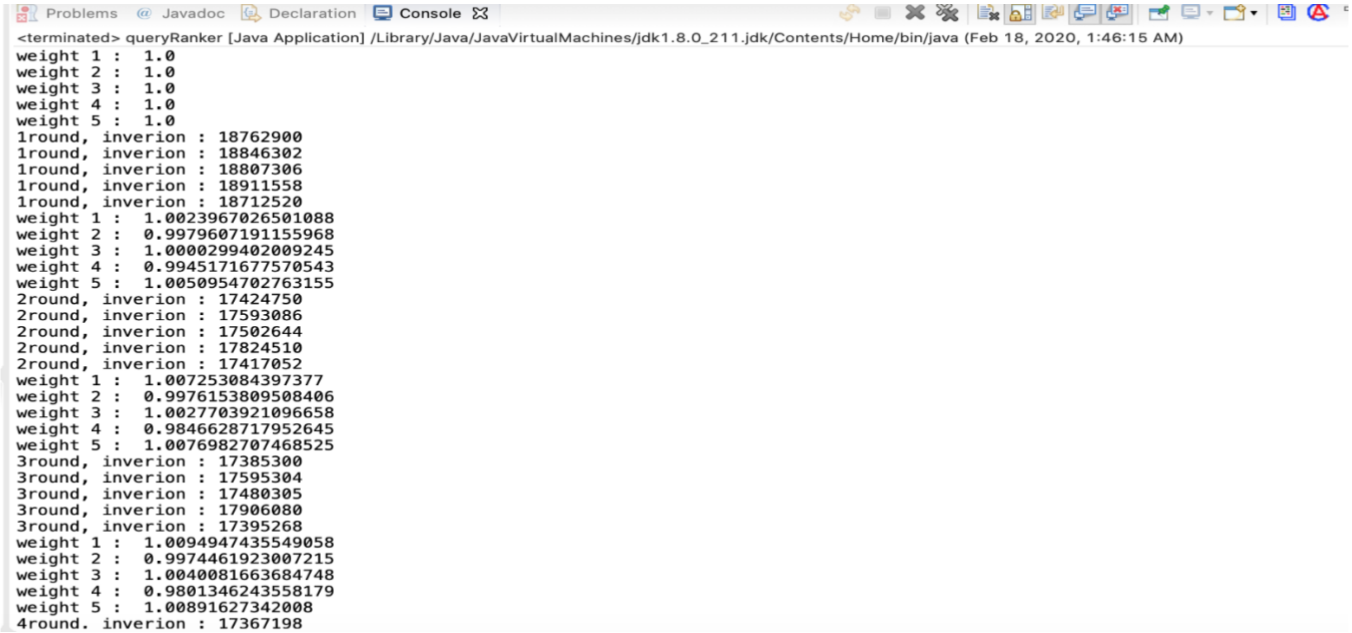
3.Experimental Results

Number of Inversions and weights of Merge Sort Algorithm



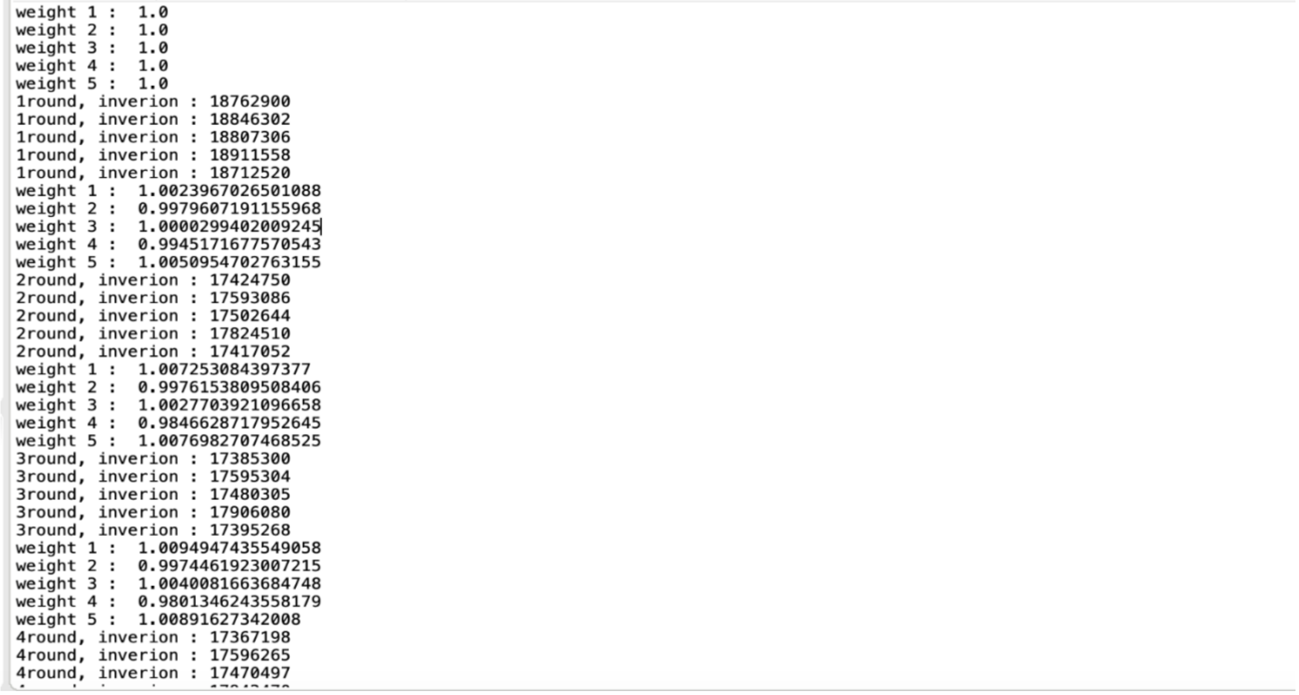
```
weight 1 :  1.0
weight 2 :  1.0
weight 3 :  1.0
weight 4 :  1.0
weight 5 :  1.0
1round, inverion : 18762900
1round, inverion : 18846302
1round, inverion : 18807306
1round, inverion : 18911558
1round, inverion : 18712520
weight 1 :  1.0023967026501088
weight 2 :  0.9979607191155968
weight 3 :  1.0000299402009245
weight 4 :  0.9945171677570543
weight 5 :  1.0050954702763155
2round, inverion : 17424750
2round, inverion : 17593086
2round, inverion : 17502644
2round, inverion : 17824510
2round, inverion : 17417052
weight 1 :  1.007253084397377
weight 2 :  0.9976153809508406
weight 3 :  1.0027703921096658
weight 4 :  0.9846628717952645
weight 5 :  1.0076982707468525
3round, inverion : 17385300
3round, inverion : 17595304
3round, inverion : 17480305
3round, inverion : 17906080
3round, inverion : 17395268
weight 1 :  1.0094947435549058
weight 2 :  0.9974461923007215
weight 3 :  1.0040081663684748
weight 4 :  0.9801346243558179
weight 5 :  1.00891627342008
4round, inverion : 17367198
4round, inverion : 17596265
4round, inverion : 17470497
```

Number of Inversions and weights of Bubble Sort Algorithm



```
<terminated> queryRanker [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_211.jdk/Contents/Home/bin/java (Feb 18, 2020, 1:46:15 AM)
weight 1 :  1.0
weight 2 :  1.0
weight 3 :  1.0
weight 4 :  1.0
weight 5 :  1.0
1round, inverion : 18762900
1round, inverion : 18846302
1round, inverion : 18807306
1round, inverion : 18911558
1round, inverion : 18712520
weight 1 :  1.0023967026501088
weight 2 :  0.9979607191155968
weight 3 :  1.0000299402009245
weight 4 :  0.9945171677570543
weight 5 :  1.0050954702763155
2round, inverion : 17424750
2round, inverion : 17593086
2round, inverion : 17502644
2round, inverion : 17824510
2round, inverion : 17417052
weight 1 :  1.007253084397377
weight 2 :  0.9976153809508406
weight 3 :  1.0027703921096658
weight 4 :  0.9846628717952645
weight 5 :  1.0076982707468525
3round, inverion : 17385300
3round, inverion : 17595304
3round, inverion : 17480305
3round, inverion : 17906080
3round, inverion : 17395268
weight 1 :  1.0094947435549058
weight 2 :  0.9974461923007215
weight 3 :  1.0040081663684748
weight 4 :  0.9801346243558179
weight 5 :  1.00891627342008
4round, inverion : 17367198
```

Number of Inversions and weights of Quick Sort Algorithm



```
weight 1 :  1.0
weight 2 :  1.0
weight 3 :  1.0
weight 4 :  1.0
weight 5 :  1.0
1round, inverion : 18762900
1round, inverion : 18846302
1round, inverion : 18807306
1round, inverion : 18911558
1round, inverion : 18712520
weight 1 :  1.0023967026501088
weight 2 :  0.9979607191155968
weight 3 :  1.0000299402009245
weight 4 :  0.9945171677570543
weight 5 :  1.0050954702763155
2round, inverion : 17424750
2round, inverion : 17593086
2round, inverion : 17502644
2round, inverion : 17824510
2round, inverion : 17417052
weight 1 :  1.007253084397377
weight 2 :  0.9976153809508406
weight 3 :  1.0027703921096658
weight 4 :  0.9846628717952645
weight 5 :  1.0076982707468525
3round, inverion : 17385300
3round, inverion : 17595304
3round, inverion : 17480305
3round, inverion : 17906080
3round, inverion : 17395268
weight 1 :  1.0094947435549058
weight 2 :  0.9974461923007215
weight 3 :  1.0040081663684748
weight 4 :  0.9801346243558179
weight 5 :  1.00891627342008
4round, inverion : 17367198
4round, inverion : 17596265
4round, inverion : 17470497
```

We have obtained the same inversion results for all the sorting techniques by running the program for two iterations

4.Conclusions

By this project experimental results, we can conclude that the number of inversions on all the given set of source files is same irrespective of the three different sorting techniques used.

Team Contributions

The complete project is all members effort equally. So, the contributions of each members would be the following:

1.Sai Sathvick Chirakaka – Coding, Testing and Verification

2.Nikhila Ratakonda – Implementation, Testing and Verification

3.Alekhya Satraboina – Project Implementation, Testing and Documentation