PROPOSAL to compare the costs and time of multiple sorting algorithms

İbrahim Türkmen 150140002, Kürşat Yaşar 150130064, Baran Kaya 150130032

Contents

Introduction

The following is a proposal to test the speed and efficiency of multiple sorting algorithms and then determine which is most suited to be used in a program that finds the closest desired establishments from any point on the map. We will use the most basic available technology to sort a size-varying array of distinct numbers which represent distances using multiple sorting algorithms which are Insertion Sort, Merge Sort and Quick Sort. We will compare the results and determine which sorting algorithm is most suited for this task. The algorithms will be written in C++ and the code will be compiled on the Windows platform with the aid of the dev-C++11 compiler. In addition to measuring the speed of the algorithms we will also compute the memory space required for the sorting.

In this report we will provide reasons why the development of this program would be beneficial and the importance of deciding on a suitable sorting algorithm, the experiments that we would be conducting as well as their conditions and setup, a summary of our proposal, the methods we would be using, the value of this project, the outcomes of these experiments, a schedule and a conclusive conclusion.

Problem: Which sorting algorithm will be the most suitable to be used in a program that calculates the closest desired points from a given location.

Background

We do not know the most suited algorithm to be used for finding the closest distances from a

specific location to several establishments. People by nature are lazy and disinclined to any sort of compromise. That is why it is so important that we make the best program suited to find the closest restaurants or other establishments in the shortest amount of time possible. According to Ass. Prof. Doshi, Jain and Shakwala; in the modern world, people are constantly on the move with their electronic devices that allow them to have access to nearly all knowledge known to man. By being aware of the user`s geographic location it is possible for computer programs to provide a variety of impressive services (2014) [1]. It is already a fact that smartphone users are in the majority of people. With the rise of customers, developing useful programs that can be used on mobile electronic devices would be a good business plan. Charuza wrote that the 'app-game' is a gold-rush that many corporations are investing in. For example, Android`s Car Locater app with a price of $3.99 has earned over 13,000 a month (2016).[2] Since Windows operated smartphones are rising in popularity, our program will be developed on the Windows platform.

The selling factor of the program will be the sorting algorithm that is used to give customers results in the shortest amount of time as possible. Akl said that performance of algorithm is based on two steps' time: computing and memory using. Different element size can change algorithms' calculation time and this affects the performance of program (1985) [3]. Since the memory cost is not an issue with our program we have no qualms with high memory demands, but the calculation time is what we are most focused on. The best sorting algorithms we could nominate for this task are Insertion Sort, Merge Sort and Quick Sort.

Sorting algorithms may perform differently for different input types. Alex Allain said (2011), "Many algorithms that have the same efficiency do not have the same speed on the same input." [4] From this we can state that there is no all-powerful sorting algorithm and that depending on the problem we have to choose to correct algorithm. According to Cook and Kim , there are many factors that are to be considered when choosing a sorting algorithm, so there is no one sorting method that is applicable in every task (1980).[5]

Woznlak, Marszalek, Gabriel and Nowicki stated that sorting algorithms that are used to sort large amounts of data, sometimes it can be difficult to find the right sorting especially for special entries. The most effective solutions for these problems are changing the input type or algorithm (2013).[6] This why we need to run these experiments so we can choose the correct algorithm. These three sorting algorithm will be tested to organize large amount of data. Our test results will give the most suitable sorting algorithm.

Proposal Summary

Ibrahim will input 20 arrays of floats ranging from 0 to 1000 with a 50 increment increase into an insertion sort algorithm and then take the algorithm`s run time (seconds) as an output.

Kürşat will input 20 arrays of floats ranging from 0 to 1000 with a 50 increment into a merge sort algorithm and then take the algorithm`s run time (seconds) as an output.

Baran will input 20 arrays of floats ranging from 0 to 1000 with a 50 increment into a quick sort algorithm and then take the algorithm`s run time (seconds) as an output.

The purpose of these experiments will be to determine the growth rates and speeds of these algorithms when the inputs are an array of floats (since the coordinates of the locations will be given in latitude and longitude, the data will most likely be a decimal number) we are not interested in the stability of the algorithms since the possibility of two identical coordinates are very unlikely. The conditions will be the same in all experiments. We will make sure to terminate any unwanted running processes from the system before running. We will make use of the library, "chrono" so that we can measure the speed in nanoseconds (although we will convert it to seconds when constructing the graphs).

The setup will be:

- Operating System: Windows 10
- RAM: 6GB
- Compiler: Dev C++ 11, with option :
- Language: C++

We wish to develop a program that will find the closest establishments to a given location, the main issue is that we do not know which sorting algorithm to use once we calculate the distances to the establishments. All computer programmers naturally know the general performance of these algorithms however performance may change for different inputs. These experiments will help us determine which algorithm to use. The importance of discovering the most suitable algorithm for this application is unquestionable. From each experiment we will construct a (time – input size) line graph and then select the most suitable one for this program. We predict/expect Insertion sort to produce a parabolic graph, Merge sort and quick sort to produce a logarithmic graph.

Outcome

The results of the trials will be processed and compared by our prestigious minds and published along with our professional opinions on which algorithm is suited for this particular task. The results of the experiments will be published on 23/11/2100 (not a typo, we checked with every science magazine and this was the soonest available date) in '*The Mars News'.* The write-ups and more detailed information will be written by Baran and disclosed at a later date.

Value

Finding the correct sorting algorithm for the task of finding the closest eating establishment form a user`s location is an utmost priority. Since this is a new field of study that hasn't been fully explored yet, the success of this experiment will beget civil happiness and stimulate the country`s economy, particularly the food industry.

Methods

With only %19 of the budget we believe that we can present satisfactory results with 50 days. There are three members in our group and three sorting algorithms to be tested, each of our group members will experiment on one of the algorithms. Insertion Sort, Merge Sort and Quicksort will be tested by Ibrahim, Kürşat and Baran respectively. In our experiments the user location will be dependent and the locations of the establishments will be independent. Meaning that we will define 1000 pairs of numbers that represent the coordinates of the establishments and then the user will input a random location that will be used to calculate the distance from each establishment. These distances will form the array that we wish to sort. This way we can guarantee a random array of unordered floats. At first we will only calculate 50 distances and then 100 and so on, until 1000. At each run we will record the results and then use them to produce the graphs.

Schedule

TASKS	Ibrahim	Kürşat	Baran
Experiment Setup	R	I	A
Meeting Setup	I	R	A
Presentation Making	R	C	A
Final Report	R	R	R
Sustenance Providing	I	I	R

	Ibrahim Turkmen	Baran Kaya	Kürşat Yasar
22/03/2017	MEETING		
24/03/2017	PROPOSAL DEFENCE		
25/03/2017	SPRING BREAK		
12/04/2017	Insertion Sort experiment Start	Merge Sort experiment Start	Quick Sort experiment Start
14/04/2017	Progress Report	Progress Report	Progress Report
5/05/2017	PRESENTATION		
19/05/2017	FINAL REPORT		

Conclusion

The launching of this project will undoubtedly bring good to the future and to our inflated egos. The financial gain is also not to be overlooked. With this discovery we will rise to the top of the food chain by gaining power over local establishments. Any questions, concerns, complaints or love-confessions should be directed to our project manager, Baran Kaya.

References

[1] Doshi, P. & A. S, Location Based Services and Integration of Google Maps in Android. *International Journal Of Engineering And Computer Science*, 2014, 3(3), 5072-5075.

[2] Charuza, P. (10 January 2016). How Much Money Can You Earn With An App. Retrieved from https://fueled.com/blog/much-money-can-earn-app/

[3] Selim G. Akl, (1985). *Parallel Sorting Algorithms*. Orlando, Florida: Academic Press Inc. *https://books.google.com.tr/books?hl=tr&lr=&id=jhHjBQAAQBAJ*

[4] Allain, A. (2011). *Sorting Algorithm Comparison.* Retrieved from http://www.cprogramming.com/tutorial/computersciencetheory/sortcomp.html

[5] Cook, C. R. & Kim, D. J, Best Sorting Algorithm for Nearly Sorted Lists, *Communications of the ACM,* (November 1980), 23(11), 620

[6] Woznlak, M., Marszaleck, Z., Gabryel, M. & Nowicki, R. 2016. Preprocessing Large Data Sets by the Use of Quick Sort Algorithm. In *Advances in Intelligent Systems and Computing. KICSS-8th international conference on Knowledge, Information, and Creativity Support Systems.* Krakow, 2013, Krakow, Poland: Springer