An Experimental Analysis for Comparison of Searching Algorithms

OKAN YILDIRIM (150160537)
FATİH YILMAZ (150160531)
ÖZGÜR AKTAŞ (150160530)

1. Introduction

We live in information era. With developing technology, it is inevitable that there is a huge mass of data circling in our daily life. However, we have chance to find information looking for, in seconds by just googling. How it could be possible? When we want to search or surfing the Internet, find the meaning of words in a dictionary, find old texting among millions of word, searching algorithms are worked in the background. In other words, searching for data is one of the fundamental field of digital world and we will analyze searching algorithms in this project. Search is one of the most frequently used problem solving methods in artificial intelligence (AI), and search methods are gaining interest with the increase in activities related to modeling complex systems [1].

We will cover analysis of three search algorithms following linear search, binary search trees and hash tables. Each team member is responsible for a search algorithm. We would like to implement these algorithms by using C++ programming language. We create different sized string dictionaries. Dictionaries are data structure that support search, insert, and delete operations [2]. We will test these algorithms on these dictionaries. We used Linux, Windows and OS-X as an experimental platform. For instance, we run these algorithms for different sized string words and evaluate their running times for each operating system. We will create various tables and graphics for result. Finally, determine which algorithm's performance better than the others.

Remainder of this proposal include background that gives information about searching algorithms, methods that how we do experimental analysis, schedule that breaks up by small task packages. Finally, share result of our project with conclusion.

2. Background

2.1 Searching Algorithms

Search algorithms aim to find solutions or objects with specified properties and constraints in a large solution search space or among a collection of objects [1]. Searching is common and vital in computer programs. In computer science, when searching for data, the use of different appropriate search algorithms between a fast application and a slower application.

2.2 Linear Search

A linear search algorithm (LSA) is an abstraction of a random access machine (RAM). Although the inputs of the RAMs we consider are assumed to be integers, inputs for LSAs are real numbers [3]. The linear search algorithm is to look for the elements of the array one by one. So if I want to call number 15 between the following numbers: 4 6 12 8 5 15 25 34. One by one, all the numbers are looked at. For example; Starting from the beginning, 4 is the number we look for? Not. 6 is the number we call? Not. 12 is the number we are looking for? Not. In this way all numbers are retrieved and the number searched continues until it is found.

As can be understood from the example, when a search is performed on a set of n numbers, the case in which the searched number is not found in this cluster may be determined by looking at the whole number n. Therefore, the complexity of the algorithm is O (n).

2.3Binary Search Tree

The idea behind binary search is that each time we make a comparison, we eliminate half of the list, until we either find the search term or determine that the term is not in the list. We do this by looking at the middle item in the list, and determining if our search term is higher or lower than the middle item. If it is lower, we eliminate the upper half of the list and repeat our search starting at the point halfway between the first item and the middle item. If it's higher, we eliminate the lower half of the list and repeat our search starting at the point halfway between the middle item and the last item[4]. For randomly inserted data, search time is O(lg n). Worst-case behavior occurs when ordered data is inserted. In this case the search time is O(n)[2].

2.4 Hash Table

Searching with hashing based on simple function structure. First all items enters the function and hashing function produces related index numbers so items placed those indexes. While searching, item again enters the function and that index is obtained. The key point here is function must produce unique index numbers per item. If it is not possible to produce this function, then function produces same indexes for some items, which called collision. To overcome collision some additional operations should made. While searching if searched index is not the searched item then additional operation done. The aim of the hash is finding item instantly.

3. Methods

In our project, we will analyze the efficiency of the three algorithms from various qualities and compare them with each other. Among the features that can be used to compare algorithms according to Brownlee are "efficiency, robustness, effectiveness, complexity, innovation, impact and generalization" [5]. In our project, we will try to examine these three algorithms in terms of efficiency using different data sets and different operating systems.

First, we will analyze the variability of the search speed of the algorithms according to the number of inputs, by changing the number of elements in the data sets. We will set the element numbers of the data groups to 1K, 10K, 100K, 1M respectively and compare the running times we have achieved. Later, we will obtain the results by experimenting in Windows operating system, then Linux operating system and finally OS-X operating system to determine the working times related to the operating systems of these three algorithms. We will display this data graphically and try to understand the results more clearly. Eventually we will have achieved our result by determining which algorithm is more efficient in which operating system.

4. Schedule

4.1 Research

In research part of our experiment, we will conduct research about linear search, hash and binary search tree. Main questions for our research will be how to implement them, what are their time complexities, which of them will give best responses on which operating system.

4.2 Implementation

In coding part, we will implement these three searching algorithms in C++ language to work on windows, Linux and Mac OS.

4.3 Experiment

In this stage, we will test these three searching algorithms, with the same dataset and record the results then continue with changing the size of dataset. After running all test on one platform, we will conduct same experiments on the other operating systems.

4.4 Visual - Representation

In this stage, we will give meaning to the numbers. We will graphically show the results, taking percentages, averages and maximum, minimum responses.

4.5 Submission

In submission part, we will create the draft after comprehensive reading error finding and editing. And make ready to submission.

5. Conclusion

In summary, we will analyze three searching algorithms and try to decide which of them works better on which platform. To do this, we will implement these searching algorithms and run them on different platforms and with different datasets. We will compare the results and visualize them.

6. References

[1] Nashat Mansour, Fatima Kanj, and Hassan Khachfe. 2011. Enhanced Genetic Algorithm for Protein Structure Prediction based on the HP Model. Search Algorithms and Applications (2011). DOI:http://dx.doi.org/10.5772/15796

[2] Niemann, T. (2017) Sorting and Searching Algorithms: A Cookbook. N.p., n.d. Web. 24 Feb.

[3] Friedhelm Meyer Auf Der Heide. 1983. A polynomial linear search algorithm for the n-dimensional knapsack problem. Proceedings of the fifteenth annual ACM symposium on Theory of computing - STOC '83 (1983). DOI:http://dx.doi.org/10.1145/800061.808734

[4] Amy Csizmar Dalal. 2004. Searching and Sorting Algorithms. Supplementary Lecture Notes. DOI:http://dx.doi.org/10.1109/glocomw.2012.6477777

[5] Cosmin Parvulescu. Jason Brownlee - Clever Algorithms. Retrieved March 5, 2017 from https://www.scribd.com/document/47728903/Jason-Brownlee-Clever-Algorithms.