JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-9, Sept. 2018 E-ISSN: 2347-2693

Execution Time of Quick Sort on Different C Compilers: A Benchmark

Mehzabeen Kaur[1*] , Surender Jangra[2]

[1]Dept. of Computer Science & Engineering, BBSBEC, Fatehgarh Sahib, Punjab, India
[2]Dept. of Computer Science, Guru Teg Bahadur College, Bhawanigarh, Sangrur, Punjab, India

[*]*Corresponding Author: jangra.surender@gmail.com, Tel.: +91-96715-02537*

Available online at: www.ijcseonline.org

Accepted: 24/Sept./2018, Published: 30/Sept./2018

***Abstract*—** Sorting is a process of arranging the elements in specific order. Computer systems use many sorting algorithms to arrange the numbers in ascending or descending order and 'quicksort' is one of the better performing algorithms. This algorithm follows divide-and-conquer approach by compiling the large data set to partition the list of elements and then exchange the numbers after scanning the list. In today's ever expanding world of technology, users find themselves in a situation where they have so many choices in selecting the best compilers. However, most of the time, technically the users are not able to identifying which translator is the best one for the completion of a particular assignment. The main aim of this paper is to find out the best compiler for 'quick sort' to reduce the execution time and automation through analyzing the performance of different compilers.

***Keywords*—**Borland, Digital Mars, Tiny C, Bloodshed, CC386

I. INTRODUCTION

Although technological advancements are bringing in new options of new technological changes but some programming languages like C++, java etc. stand the test of time as their flexibility and portability continue to be important even in the constantly changing technological scenario. Compiler is a computer program that transforms source code from high level language into lower level language or machine language. Compiler includes better detection mechanisms, higher performance in terms of execution and enhances optimization. The quality of the resulting code and compilation time are not only two aspects for measuring the efficiency of the compiler. That's get tricky as well, because as well, because there are so many compilers options that can skew the results. To decide the best compiler some factors come into the main role: time to compiled code, size of compiled code, memory usage of compiled code bugs etc.

Computer systems use many sorting algorithms to arrange the numbers in ascending or descending order and quicksort is one of them. Quicksort is a sorting algorithm that follows divide-and-conquer approach to partition the list of elements and hence, exchange the numbers after scanning the list. The performance of different C compilers is measured to get the least execution time of quicksort in case of arrays containing large number of elements in it. This paper presents a review of different C language compiler's execution time of quick sort.

The analysis is made on quicksort algorithm that selects a pivot and thus compares that pivot with all elements present in the list. The list is first scanned from right end towards the left to get the number smaller than the pivot. On getting the smaller number, the two numbers are swapped and similarly a number is swapped when it is greater than pivot on scanning the list from left to right. The list is then divided into sub-lists till the pivot reaches its immovable original position.

```
quicksort(x,first,j-1);
quicksort(x,j+1,last);
```

In this paper, performance analysis of some of the compilers has been examined and such kind of analytical exercise brings a kind of easy option that helps the common men to choose and buy the best compilers. This kind of exercise facilitates the computer to run and work faster.

The rest of the paper is organized as follows. Overview of different compilers is presented in section II, Experimental setup in section III. Results and Analysis are presented in section IV and at last concluding remarks are given in section V.

II. OVERVIEW OF COMPILERS

In today's fast changing technological scenario, users are left with so many choices and it is a kind of challenge in selecting the most suitable compiler. There are so many factors like size of RAM, faster hard drives (including SSDs), and CPUs that enhance its performance and speed and thus adding more features to its ever expanding capacity.

There are lots of compilers which are used for converting the source code in to object code. In this paper, we have used Borland C++ 5.5, Tiny C, CC386, C-Free, Bloodshed Dev C++, Digital Mars and Turbo C, compilers for evaluating the performance of different compilers through quicksort.

Turbo C and Borland C++ 5.5 includes the compiler bcc32, 32 bit linker (tlink32), Borland Resource Compiler / Binder (brc32, brcc32), C++ Win32 Pre-processor (cpp32) [1] etc. Borland is one of the major manufacturers of compilers and Turbo C++ is widely used product of Borland which is compatible for c and c++ programming environment. Turbo C++ supports IDE features for MS DOS and Microsoft windows and has a better debugging tool named Turbo Debugger. Turbo C was an integrated development environment (IDE) for programming in the C language.

Tiny C: it is a small fast C compiler which is self-relying and designed especially for slow computer with low disk size. This do not required an external assembler or linker and can be used as a backend code generator with the aid of another library. This compiler is very fast and can compile large projects in minimum time [1].

CC386: This is freeware Win 32 C compiler and one of the older one that work for many years. It also includes an IDE which provides compilation, editing and debugging. A very impress achievement for one individual [1].The Run time library in this package has WIN32 headers and an import library, many windows programs will compile with it although there are a few incompatibilities.

C-Free: It is also an Integrated Development Environment (IDE) likes CC386 for C and C++ programming language. It includes MinGW 5 package in C-Free, as an IDE [2].

Bloodshed Dev C++: This is a full-featured Integrated Development Environment (IDE) for the C/C++ programming language. Dev-C++ is generally considered a Windows-only program [3]. Dev-C++ can also be used in combination with Cygwin or any other GCC based compiler.
Digital Mars: This is Walter Bright owned company that makes high performance compiler for the C, C++ and D programming languages as well as DMD Script and related IDE based packages for Win32, Win16, DOS32 and DOS. This compiler possesses fastest compile/link times, powerful optimization technology and is designed by complete library source, HTML browsable documentation [4] and terms as Integrated Development and Debugging Environment (IDDE). Comparative analysis of different compilers are shown in Table 1.

III. Experimental Setup

A. Hardware and Software Requirement:

RAM: 2GB
Processor: CORE i3(2.53 GHz)
No. of Elements: 100

Table 1: Execution Time (sec) of Quicksort for 100 elements on different Compiler

Compiler	**Version**	**Program Execution Environment**	**CPU Usage (%)**	**Execution Time (Sec)**	**Average Execution Time (Sec)**
BORLAND C/CPP	Borland C++ 5.5	CUI	0-16	1.077	1.141142857
			0-19	1.155	
			0-19	1.155	
			0-19	1.139	
			0-25	1.154	
			0-25	1.154	
			0-23	1.154	
TINY C	TCC 0.9.26	CUI	0-20	1.138	1.183714286
			0-20	1.185	
			0-18	1.201	
			0-25	1.185	
			0-23	1.185	
			0-20	1.201	
			0-20	1.201	
C-FREE	C-Free 5	GUI	0-25	3.837	4.071142857
			0-28	4.134	
			0-25	4.726	
			0-28	4.196	
			0-25	3.447	
			0-25	3.79	
			0-25	4.368	
DEV C/CPP	DEV C++ 5.0	CUI	0-17	5.553	1.969714286
			0-25	2.355	
			0-18	1.17	
			0-16	1.17	
			0-20	1.185	
			0-16	1.17	
			0-21	1.185	
DIGITAL MARS	DIGITAL MARS 8.56	CUI	0-16	1.077	1.141142857
			0-19	1.155	
			0-19	1.155	
			0-19	1.139	
			0-25	1.154	
			0-23	1.154	
			0-20	1.154	

Table 2: Execution Time (sec) of Quicksort for 20000 elements on Turbo C Compiler

Compiler	Version	No. of Elements	CPU Usage (%)	Execution Time (Sec)	Average Execution Time (Sec)
TURBO C	GUI	100	24-26	0	0
		300	24-26	0	0
		350	24-26	0	0
		359	24-26	0	0
		360	24-26	0.0549	0.001098
		400	24-26	0.10989	0.0021978
		500	24-26	0.549	0.01098
		1000	24-26	0.549	0.01098
		5000	24-26	0.7692	0.015384
		10000	24-26	1.0989	0.021978
		20000	24-26	2.0879	0.041758

IV. RESULTS AND DISCUSSION

The execution time of this algorithm is computed for different compilers discussed above and following are the observations made on executing quicksort for an array of 100 elements. But when I observed the same for Turbo C compiler, the observations were as that shown in Graphical representation of Execution Time on Turbo C

EC-Free>EDev-C/CPP>ETiny C>E CC386 > (EBorland =E Digital Mars)>E Turbo C

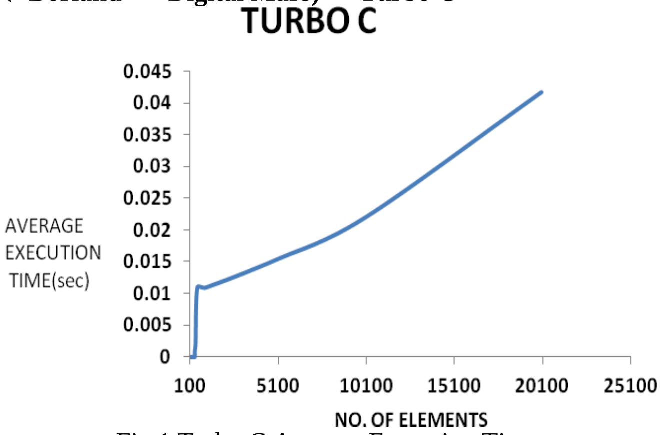

Fig.1 Turbo C Average Execution Time

Turbo C executes the same algorithm at such a fast pace that it takes less than 1sec to sort more than 20000 elements in an array that every other compiler consumes for just 100 elements. No changes take place in the time till the sorting of 359 elements. The first reading occurred when 360 elements were taken into consideration. Then, there occurs a very slight change of few milliseconds as the number of elements in an array starts increasing.

CONCLUSION

Our digital computer systems use many sorting algorithms to arrange the numbers in ascending or descending order and quicksort is one of them. Quicksort is one of the important soring algorithms which is follows divide-and-conquer approach to partition the list of elements. Quick short have large application area which is best suitable for case of large data sets. So, there is the need is to reduce the execution time of this algorithm for automation. In this paper firstly, the review of different C compiler is done and then performance of different C compilers is measured to get the least execution time of quicksort in case of arrays containing large number of elements in it.

Turbo C executes the same algorithm at such a fast pace that it takes less than 1sec to sort more than 20000 elements in an array that every other compiler consumes for just 100 elements. There is no changes take place in the time till the sorting of 359 elements and after that first reading occurred when 360 elements were taken into consideration. Then, there occurs a very slight change of few milliseconds as the number of elements in an array starts increasing.

REFERENCES

[1] Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks", in Mobile Computing and Networking, 2000, pp. 243–254.
[2] https://en.wikipedia.org/wiki/List_of_compilers
[3] https://en.wikipedia.org/wiki/Tiny_C_Compiler
[4] http://ladsoft.tripod.com/cc386_compiler.html

Authors Profile

Mehzabeen Kaur has received his Bachelor's Degree in Computer science and Engineering from Punjab Technical University, Jalandhar (Punjab), India. She is Final Year student of M.Tech in Computer and Engineering at Baba Banda Singh Bahadur Engineering College Fatehgarh Sahib, Punjab, India. Her main research interests are in image processing, artificial n eural networks genetic algorithms and fuzzy logic.

Dr. Surender completed his M.Tech degree in Computer Science and Engineering from Ch. Devi Lal University Sirsa (Hry) in 2006,, Ph.D in Computer Science and Application from Kurukshetra University, Kurukshetra in 2011.. He has more than 10 years teaching experience to teach B.Tech, M.Tech., BCA and MCA Classes. Recently he is working as an Assistant Professor, in the Department of Computer Science, at GTB College, Bhawanigarh (Sangrur), Punjab, India. He has published over 50 publications in different International Journals and Conferences of repute. His research interests lies in Fault Tolerance in Mobile Distributed Systems, Adhoc N/W, Data Mining, Cloud Computing, System Security and Cryptography.