UNIVERSITY OF NOVI SAD
TECHNICAL FACULTY
"MIHAJLO PUPIN"
ZRENJANIN

ITROCONFERENCE[13]

INFORMATION TECHNOLOGY AND EDUCATION DEVELOPMENT

ITROCONFERENCE[13]
INFORMATION TECHNOLOGY AND EDUCATION DEVELOPMENT

PROCEEDINGS

ZRENJANIN, November 2022

UNIVERSITY OF NOVI SAD
TECHNICAL FACULTY ”MIHAJLO PUPIN”
ZRENJANIN
REPUBLIC OF SERBIA

XIII INTERNATIONAL CONFERENCE OF

INFORMATION TECHNOLOGY AND DEVELOPMENT OF EDUCATION ITRO 2022

PROCEEDINGS OF PAPERS

XIII MEĐUNARODNA KONFERENCIJA

INFORMACIONE TEHNOLOGIJE I RAZVOJ OBRAZOVANJA ITRO 2022

ZBORNIK RADOVA

ZRENJANIN, NOVEMBER 2022

Publisher and Organiser of the Conference:
University of Novi Sad, Technical faculty „Mihajlo Pupin", Zrenjanin, Republic of Serbia

For publisher:
Dragica Radosav, Ph. D, Professor,
Dean of the Technical faculty „Mihajlo Pupin", Zrenjanin, Republic of Serbia

Editor in Cheaf - President of OC ITRO 2022:
Dragana Glušac, Ph. D, Professor

Proceedings editor:
Marjana Pardanjac, Ph. D, Associate Professor

Technical design:
Snežana Jokić, Ph. D, Assistant Professor
Maja Gaborov MSc, Assistant
Nemanja Tasić MSc, Assistant

Circulation: **50**

ISBN: 978-86-7672-362-1

CIP - Каталогизација у публикацији
Библиотеке Матице српске, Нови Сад

37.01:004(082)(0.034.4)
37.02(082)(0.034.4)

INTERNATIONAL Conference of Information Technology and Development of Education ITRO (13 ; 2022 ; Zrenjanin)
Proceedings of papers [Elektronski izvor] / XIII International Conference of Information Technology and Development of Education ITRO 2022 = Zbornik radova / XIII međunarodna konferencija Informacione tehnologije i razvoj obrazovanja ITRO 2022, Zrenjanin, November 2022 ; [editor in chief Dragana Glušac]. - Zrenjanin : Technical Faculty "Mihajlo Pupin", 2023. - 1 elektronski optički disk (CD-ROM) : tekst, ilustr. ; 12 cm

Sistemski zahtevi: Nisu navedeni. - Nasl. sa naslovnog ekrana. - Elektronska publikacija u formatu pdf opsega XI, 221 str. - Tiraž 50. - Bibliografija uz svaki rad.

ISBN 978-86-7672-362-1

а) Информационе технологије -- Образовање -- Зборници б) Образовна технологија -- Зборници

COBISS.SR-ID 107092745

PARTNERS INTERNATIONAL CONFERENCE

South-West University „Neofit Rilski”
Faculty of Education, Blagoevgrad,
Republic of Bulgaria

Technical University of Košice
Faculty of Electrical Engineering and Informatics
Slovak Republic

University Goce Delcev Stip
Republic of Macedonia

THE SCIENCE COMMITTEE:

THE ORGANIZING COMMITTEE:

The editor and the publisher are not responsible either for the statements made or for the opinion expressed in this publication.

The author warrants that the article is original, written by stated author/s, has not been published before, contains no unlawful statements, does not infringe the rights of others, is subject to copyright that is vested exclusively in the author and free of any third party rights, and that any necessary written permissions to quote from other sources have been obtained by the author/s.

Authors retain the following rights:

- *copyright, and other proprietary rights relating to the article, such as patent rights,*
- *the right to use the substance of the article in future works, including lectures and books,*
- *the right to reproduce the article for own purposes, provided the copies are not offered for sale,*
- *the right to self-archive the article.*

The Proceedings have been published in a digital format on the Faculty web site.

INTRODUCTION

This Proceedings comprises papers from the International conference on Information technology and development of education that is held on line on November 25th 2022. The International conference on Information technology and development of education has had a goal to contribute to the development of education in Serbia and in the region, as well as, to gather experts in natural and technical sciences' teaching fields. The expected scientific-skilled analysis of the accomplishment in the field of the contemporary information and communication technologies, as well as analysis of state, needs and tendencies in education all around the world and in our country have been realized. The authors and the participants of the Conference have dealt with the following thematic areas: - Theoretical and methodological questions of contemporary pedagogy - Personalization and learning styles - Social networks and their influence on education - Children security and safety on the Internet - Curriculum of contemporary teaching - Methodical questions of natural and technical sciences subject teaching - Lifelong learning and teachers' professional training - E-learning - Education management - Development and influence of IT on teaching - Information communication infrastructure in teaching process All submitted papers have been reviewed.

The papers presented on the Conference and published in this Proceedings can be useful for teacher while learning and teaching in the fields of IT, informatics, technics and other teaching subjects and activities. At the end of the conference, and based on the papers of our participants, we conclude that the main focus points of this moment in education. Contribution to science and teaching development in this region and wider has been achieved in this way.

The ITRO Organizing Committee would like to thank the authors of papers, reviewers and participants in the Conference who have contributed to its tradition and successful realization.

Chairman of the Organizing Committee
Ph.D Dragana Glušac

IN MEMORIAM PROFESSOR DIJANA KARUOVIĆ 1978-2022.

We especially want to pay tribute to our late colleague professor Dijana Karuović PhD, as one of the founders of the ITRO conference.

To all of us who knew her, professor Dijana Karuović will be a symbol of professional attitude towards work, dedication and loyalty to the institution to which she belonged. Behind HER remain her wonderful children, her many scientific works, her goodness and her love.

We are grateful to have known her.

Also, we will always remember our dear colleague professor Ivan Tasić, PhD, who passed away in 2019.

Our team thus suffered an irreparable loss, and their names will forever remain on the pages of the conference proceedings.

Professor Dijana Karuović and professor Ivan Tasić

CONTENTS

INVITED LECTURE

SCIENTIFIC PAPERS

Application of Sorting Algorithms in Shopping Assistant Application

N.Stojkovikj, L.K.Lazarova, A.S.Ilievska, B. Tashkova
Goce Delcev University/Faculty of Computer Science, Stip, Republic of North Macedonia
natasa.stojkovik@ugd.edu.mk; aleksandra.stojanova@ugd.edu.mk; limonka.lazarova@ugd.edu.mk;

Abstract - Sorting algorithms are widely used in a broad variety of applications. The sorting is very important in the programming and in the real life. Also, is easier and faster to find some elements in a sorted array than in the unsorted array. For easier searching, sorting algorithms are necessary in the programming process. In this paper, we will present their use in shopping assistance application. The motivation for this application follows from everyday problems which arise while we are looking for some products in a shop.

The shopping assistant application uses Tim Algorithm for sorting of the products in order in which they are arranged in the shop.

I. Introduction

In the programming is needed to sort elements of the data structure into some specific order very often. The most common are numerical (from highest to lowest, or from lowest to highest) order or some alphabetical order. As examples of sorting are the following: lists of words can be sorted alphabetically or by the length, list of countries can be sorted by their population, by their area or by calling code. The data sorting is necessary because the sorted data are always easier for searching or analysis. Also, efficient sorting is important for optimization of the other algorithms that require sorted input data, [1, 2, 3].

Because many algorithms can be applied only on a sorted data, the data sorting is the first step for searching. For example, a binary search is faster that linear search, but binary search can be applied only to a sorted data.

Sorting is so important and potentially so time−consuming. This is reason because the sorting has been the subject of research in the computer science. As a result, very sophisticated methods for sorting have been developed [4].

The sorting has a wide application in all fields in science and in the real life [5].

- *Commercial computing*

Many organizations as financial, commercial or government institutions use sorting for organization of their data. Sorting algorithms can be used for sorting accounts by name or by number, for sorting transactions by place or time, for sorting mail by postal code or address, for sorting file by name or date.

- *Search for information*

If the data are sorted, then searching through it is faster and more efficient.

- *Operation research*

Sorting is used in job-shop scheduling, when the goal is to complete all jobs (job j requires t_j seconds of processing time) with minimized average time. If the shortest processing time is the first used rule, then the jobs are scheduled in increasing order of the processing time. If the longest processing time is the first used rule, then the jobs are scheduled in decreasing order of processing time. In the two cases for ordered processing times (either increasing order or decreasing order) sorting algorithm must be used.

- *Numerical computations*

Many numerical algorithms use sorting to control accuracy in calculations.

- *Graph theory*

Dijkstra's algorithm is the most famous algorithm for finding the shortest path between the nodes in the graph. For implementation of Dijkstra's algorithm, priority queue can be used. In a priority queue, elements with higher priority have priority before elements with lower priority. For this reason, elements in priority queue must be sorted. Also, Prim's algorithm is used for finding minimum spanning tree. And for implementation on this algorithm, priority queue can be used. Other algorithm for finding minimum spanning tree is Kruskal's algorithm. This algorithm uses sorting algorithm for sorting of all edges in non-decreasing order of their weight.

- *Data compressing*

Huffman compression is a classic data compression algorithm. For implementation of this algorithm, priority queue is used.

- String processing

All algorithms for string processing are based on sorting.

In this paper, we used Tim sort algorithm in Shopping Assistant Application. We used this sorting algorithm for sorting products in the shop by category and by the order in which they are arranged (organized) in the shop.

II. Sorting Algorithms

There are many algorithms for sorting, a few of them stand out as the most effective. All the algorithms that will be presented here have advantages and disadvantages, depending on the size of the data that needs to be sorted. The algorithms that will be presented are the following: Insertion Sort (insertion method), Selection Sort (selection method), Bubble Sort, Merge Sort, Quick Sort and Tim Sort [6, 7, 8, 9].

We will consider all these algorithms.

- Insertion Sort

Insertion Sort is a simple sorting algorithm that provides data sorting like sorting in playing cards. The array is visually divided into a sorted and an unsorted subarray. Elements of unsorted subarray are selected and placed in the correct position in the sorted section. If the array is sorted in increasing order, it is necessary to find the position in the array with a value greater than the value of the element that is inserted. If such position does not exist, then the position of the new element is after the last element in the sorted array. (This element is greater than all elements in the array). The new element in the position which is found is inserted, and all elements of the array are shifted from one place to the right. The shifting is made, to release space for new elements.

A. Time complexity of insertion Sort

Best case time complexity: $O(n)$.

If the array is already sorted, then the inner loop will not be executed and the outer loop will be executed $n-1$ times.

Worst case time complexity: $O(n^2)$.

If the array is reversely sorted, then the inner loop will be executed maximum times.

Average case time complexity: $O(n^2)$.

- Selection Sort

This algorithm sorts the array by repeatedly finding the minimal element of the unsorted subarray and putting it at the beginning of the array. The algorithm maintains two subarrays, the subarray which is already sorted and unsorted subarray. In every iteration of the algorithm, from the unsorted subarray the minimal element is selected and moved to the sorted subarray.

B. Time complexity of Selection Sort

The time complexity of algorithm is $O(n^2)$ as there are two nested loops. One loop to select an element of the array (one by one) and another loop to compare that element with all other elements from the array.

- Bubble Sort

Bubble Sort is the simplest sorting algorithm that works on the principle by swapping the adjacent elements if they are in the wrong order. To sort an array with this algorithm, it is necessary to make passes through the array. In each pass, pairs of adjacent elements are compared, and if it is necessary, the elements will be swapped. After each iteration, the smallest element among the unsorted elements is placed at the beginning.

C. Time complexity of Bubble Sort

Best case time complexity: $O(n)$.

If the array is already sorted in increasing order, the algorithm will determine in the first iteration that no pairs of elements need to be swapped and then will terminate immediately. The bubble sort algorithm must perform $n-1$ comparisons.

Worst case time complexity: $O(n^2)$.

Average case time complexity: $O(n^2)$

- Merge Sort

The Merge Sort is recursive sorting algorithm that is based on the Divide and Conquer principle. With this algorithm, the array is initially divided into two equal halves. Each subarray will be independently sorted of the other. With the algorithm, the array is recursively divided in two equal halves until it cannot be further divided. The empty array or array with one element is a base case of the recursion. Finally, when the two subarrays are sorted, the merging of two subarrays is made.

D. Time complexity of Merge Sort

Time complexity of this algorithm is $O(n\log(n))$.

Merge Sort is a recursive algorithm, and its time complexity can be expressed by the following relation:

$T(n)=2T(n/2)+O(n)$.

- Quick Sort

The algorithm Quick Sort is Divide and Conquer algorithm. The algorithm solves problems, so the problem is first divided into smaller subproblems, then the subproblems are solved and their solutions are combined. In this way, the algorithm solves the basic problem. At the beginning, the algorithm selects an element as pivot. The pivot can be selected in different ways: always select the first (or last) element as a pivot, select a random element, select median. The pivot is put at the correct position in the sorted array. All smaller elements are put before the pivot and all greater elements are put after the pivot. In this way, in the first subarray are put the smaller elements than pivot, and the second subarray consists of the greater elements than the pivot. The algorithm Quick Sort recursively is applied more times until the subarray is empty or it has only one element.

E. Time complexity of Quick Sort

Best case time complexity: $O\left(n\log\left(n\right)\right)$.

In each step of dividing, if partitioning algorithm always chooses the middle element as pivot (two subarray with equal length are obtained), best case scenario will be occurred.

Best time complexity can be expressed by the following relation: $T\left(n\right)=2T\left(n/2\right)+O\left(n\right)$.

Worst case time complexity: $O\left(n^2\right)$.

The worst-case is when the partitioning algorithm selects the largest or smallest element as the pivot element every time (one subarray has only one element, and all other elements are in the second subarray).

The worst time complexity can be expressed by the following relation:

$$T\left(n\right)=T\left(0\right)+T\left(n-1\right)+O\left(n\right)$$

Average case time complexity: $O\left(n\log\left(n\right)\right)$.

Can be expressed by the following relation:

$$T\left(n\right)=T\left(n/9\right)+T\left(9n/10\right)+O\left(n\right).$$

III. A SORTING ALGORITHM USED IN PYTHON

Tim sort algorithm is a sorting algorithm designed and implemented by Tim Peters in the programming language Python in 2002. This algorithm is derived from the Insertion sort algorithm and Merge sort algorithm. First is analyzed the array that needs to be sorted. Depending on the results of this analysis, the most appropriate approach is chosen.

If the array has less than 64 elements, then Tim Sort chooses Insertion Sort algorithm for sorting. This algorithm Insertion Sort is the most effective algorithm for sorting small arrays, [10].

If the array has more than 64 elements then the algorithm will make a first pass through the array, looking for the parts of the array that are strictly increasing or decreasing. These parts are known as runs. If some of the parts are in decreasing order, then the algorithm will reverse those parts [10,11].

These runs, one by one, are sorted by using of the Insertion Sort algorithm. In the second step, the runs are merged by using the Merge Sort algorithm.

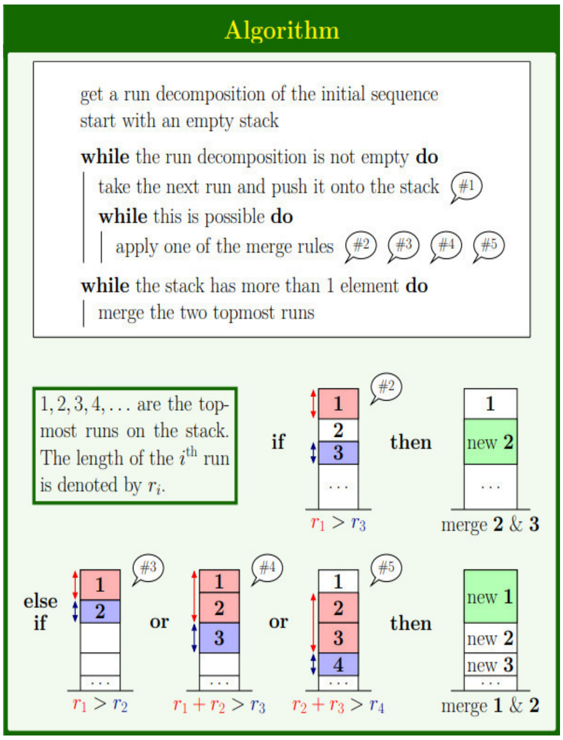
If the size of the array is less than the run, the array will be sorted by Insertion Sort algorithm. The size of the run may be from 32 to 64 depending on the size of the array. We can note that the merge function performs well when the size of the runs are powers of 2, [10, 11].

In the best case, the time complexity of Tim Sort algorithm has complexity $O\left(n\right)$ and in the worst and average case, time complexity is $O\left(n\log\left(n\right)\right)$.

Tim Sort is better than the other sorting algorithms because it uses a combination of multiple sorting algorithms and manages to maintain sorting stability. This algorithm finds wide application and is used as the default sorting algorithm in the Python programming language.

IV. SHOPPING ASSISTANT APPLICATON

The motivation for this application follows from everyday problem with which every citizen encounters when is shopping in a shop. Namely, the customers waste a lot of time finding all the products that want to buy from the shop. They often pass the same path more times looking for desired products. The solution offered by this application is simple and effective. The application contains all product that are found at the shop. The products are sorted by category. Customers can choose desired products that are stored in the shopping list. After this, the products from shopping list are sorted according to the order in which they are arranged in the shop. By help of this application, the customer can create his shopping list and take the products from the list in a given order. On this way, the time that the customers spend for shopping will be significantly reduced.



```
get a run decomposition of the initial sequence
start with an empty stack
while the run decomposition is not empty do
    take the next run and push it onto the stack (#1)
    while this is possible do
        apply one of the merge rules (#2) (#3) (#4) (#5)
while the stack has more than 1 element do
    merge the two topmost runs
```

Figure 1.Time Sort algorithm [12]

The Shopping Assistant App is created using Python in combination with the framework Kivy. The interface is simple and user-friendly, with products organized in categories placed on the left side of the screen, whereas the right side of the screen is reserved for the shopping cart, as well as the buttons for sorting, removing a product or resetting the list.

Upon opening the application, the user can select various items from the given categories which appear in the shopping cart in the order that the user selects them. After choosing all the desired products, the user can click on the button "Sort" and the list of products in the shopping cart gets sorted according to the order in which they would be placed in the shop that the application is modeled on. By clicking the button "Remove", the user can remove the last added item from the shopping cart. The "Reset" button gives the user the option to delete the whole list of products and start anew. An extra feature is the counter which shows the user the number of products currently present in the shopping cart.

The code behind works in a simplistic and straightforward manner. The products are given a number that refers to their placement in the shop and are sorted using the default Python function for sorting that utilizes the algorithm Tim Sort. The code is split into two files, one being the Kivy file that contains the details of the interface and the other being the Python file with all the functions and necessary building blocks that create the application.

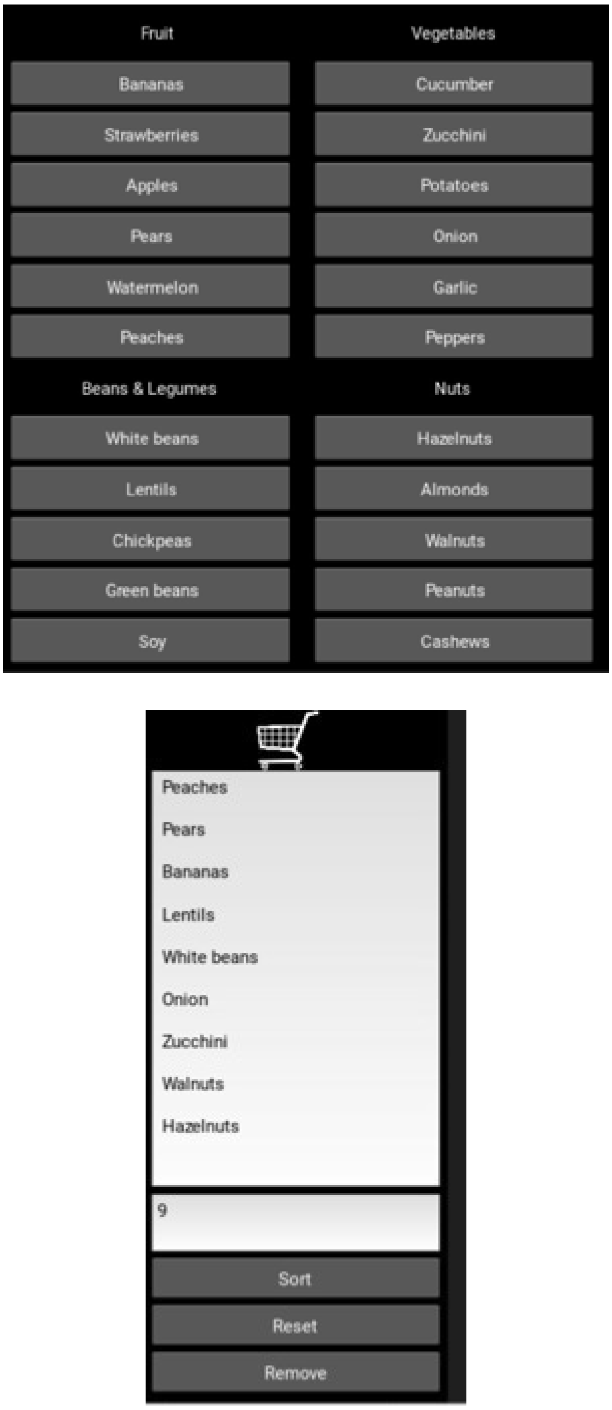


Figure 2. Graphic-user interface

The Shopping Assistant App is an easy-to-use application that provides users with a smoother and less time-consuming shopping experience.

CONCLUSION

Tim Sort is an efficient and fast sorting algorithm that is used in many applications and solves different types of problems. It implements the idea that many real data sets contain ordered subsequences. It has a very important role in the application because it enables quick sorting of the products selected by the user. On the other hand, the application solves a big problem with which every customer is encountered and represents an innovative solution that has the potential to become a necessary part of everyday shopping in many shops in the region and wider.

REFERENCES

[1] B. Miller, D. Ranum "Problem Solving with Algotihms and Data Structure Using Pyhton," Franklin Beedle ve Associates, 2005.

[2] P. P. Puschner, "Real-Time Performance of Sorting Algorithms", Real-Time Systems 16, 63–79 (1999). https://doi.org/10.1023/A:1008055919262

[3] T. D. Fryar, "Analysis of Sorting Algorithms",(1988). Honors Theses. Paper 230.

[4] R. Lafore, "Data structure and algorithms in Java", Waite Group Press, Macmillan Computer Publishing, 1998

[5] R.Sedgewick and K.Wayne, "Algorithms 4th Edition", Amazon,

[6] D. Mittermeir, P. P. Puschaner, "Which sorting algorithms to choose for hard real-time applications", Proceedings Ninth Euromicro Workshop on Real Time Systems, (1997).

[7] T. Cormen, C. Leiserson, R. Rivest, C.Stein "Introduction to Algorithms 3rd Edition", MIT Press, 2009.

[8] G. V. Shubham, G.R.S. Vishwas, A.Kumar," Analysis of Sorting Algorithms Using Time Complexity", International Journal of Engineering Research & Technology.Vo;l5, 1 Issue 21, 2017, pp.1-3

[9] R. K. Gogoi, "Time Complexity Analysis of Sorting Algorithms", High Technology Letters, Volume 23, Issue 12, 2017.

[10] Y. B. Jmaa, R. B. Atitallah, D. Duvivier, M. B. Jemaa, "A Comparative Study of Sorting Algorithms with FPGA Acceleration by High Level Synthesis", Computacion y Sistemas 23(1), (2019).

N. Auger, V. Juge, C. Nicaud, and C. Pivoteau, "Analysis of Tim Sort Algorithm", Labaratorie D'Informatique Gaspard-Monge,2019