TECHNIQUES AND TRADEOFFS IN SORTING ALGORITHMS: A REVIEW

Anshuman Narayan
UG Scholar,
Departmentof CSE, BMSCE,
Bangalore, INDIA

Anurag Unnikrishnan
UG Scholar,
Departmentof CSE, BMSCE,
Bangalore, INDIA

Siddharth Benedict Dias
UG Scholar,
Department of CSE, BMSCE,
Bangalore, INDIA

H S Guruprasad
Professor and Head
Department of CSE, BMSCE
Bangalore, INDIA

Abstract: Towards the dawn of the 20th century, there have been numerous advancements in the field of computing. This has been possible because of the cumulative efforts of not only the intelligentsia but also due to the rise in the computational capabilities of machines. One of these advancements is the number of sorting algorithms that are available today. Numerous algorithms have been developed for different purposes, each one more unprecedented than the other. This papers deals with a heterogeneous combination of such sorting algorithms, where the purpose and idea behind each one is explained in a brief yet transparent manner.

Keywords: Sorting, Algorithms, Time Complexity, Space Complexity, Parallel Processing, Algorithm Analysis

I. INTRODUCTION

Sorting is defined as the act of grouping objects in a particular order based off of a certain characteristic common to these objects. A sorting algorithm is one whose purpose is to arrange the given set of elements in a vector in a certain order depending on either the lexicographic or numeric value of these elements.

Many sorting algorithms have been developed over the years. Some sorting algorithms follow the ideology of brute force, where every possibility is exhausted, for example bubble sort and selection sort. Some follow the ideology of divide and conquer, where the size of the problem is reduced to make it more manageable, as in merge sort or quick sort. These sorting algorithms have been heavily analyzed and scrutinized for their performance and reliability over the years. On seeing the drawbacks, newer algorithms or variants of the earlier algorithms have been proposed which have themselves been evaluated.

The sheer number of sorting algorithms has led them to be classified under many criteria. Sorting algorithms are grouped on the basis of their computational complexity, i.e. the performance of the algorithm in its best, worst and average case, their memory usage. The algorithms that do not require additional space for sorting are considered "in-place" and have a space complexity of O(1). They are classified on stability, where a stable algorithm is one which maintains relative ordering of the values from the original ordering. There are many other classifications that allow us to then define the nature of the sorting technique by mentioning some of the categories they fit into.

Sorting has a wide variety of applications in the field of computing, one being that sorted lists are needed for more efficient searching operations and better sorting techniques allow for data to be arranged so that it can be quickly searched through. Hence sorting is one of the most important problems in the field of computing. Since the dawn of computing, programmers around the world have developed many algorithms to propose better solutions to the sorting issue and continue to do so today. Many of the current sorting algorithms also lay into account the changing nature of computer hardware and now these newer algorithms are more suited to make the best use of the resources that are available on a computer, to allow for faster and more efficient work.

This paper is a literary survey of many of the current algorithms proposed which have added to the field of sorting. This paper is arranged into sections that contain a group of algorithms that share the same underlying theme. While the nuances within the different algorithms discussed may be different, they all share a common intent, be it enhancing an existing method, or developing something different. This paper summarizes some of the work done in the field of sorting and shows the current trends being followed and also intends to showcase some of the advancements that have propelled the field in the recent past.

II. LITERATURE SURVEY

Improving of pre-existing Algorithms

A lot of the work done in the field of sorting is to identify drawbacks of classical algorithms and improving on those by offering solutions to the drawbacks while still retaining the efficiency and simplicity of the pre-existing algorithm.

Sumanthiet. al. [2] discusses a new type of selection sort, a double ended version of the original sorting algorithm. In this algorithm, the largest and smallest elements from the given input are taken and are exchanged with the first and last element respectively. The size of the array is reduced by two in the next iteration till the array is completely sorted. Almihoudet. al. [3] proposes the idea of Enhancing the Bubble sort and Selection sort algorithms which are similar to the algorithm proposed by [1]. Yashwant Singh Patel et al. [4] introduce a new algorithm based on Divide and Conquer approach called Fuse-sort with O (n log log n) time and linear space complexity. It is a comparison-based, stable, not in-place sorting algorithm. When compared with Merge sort, which has similar characteristics and is the preferred choice when used to sort large data-sets, Fuse sort uses lesser amount of space. The array is divided into √n number of sub-arrays of size √n. This step is performed recursively until the size of the sub-arrays is lesser than or equal to 2. This is the base case where the 2 elements are sorted manually. The sorted sub-arrays are repeatedly "fused" together to get the final sorted array. Wang Min [5] designs and analyses a Bidirectional Selection Sort. The bidirectional version of selection sort finds both the maximum and the minimum from the unsorted array elements and places them at positions i and n-i+1 respectively. Thus, sorting the array requires n/2 passes rather than n-1 passes. KamleshNenwani et al. [6] demonstrate an adaptive version of Insertion sort that utilizes the existing ordering of elements and is able to expand in both directions. The proposed algorithm utilizes O (n) of space to implement a 2-way expansion of ordered elements in order to reduce the number of shift operations performed in the traditional insertion sort. The number of comparisons to find the position of insertion of the element in the ordered list is reduced by performing Binary Search.

Wang Min [7] describes an improved version of Insertion Sort Algorithm by inserting 2 elements in a pass instead of one. This algorithm moves the two elements to be positioned to temporary locations 0 and n+1 with r [0] < r [n+1]. Then the sorted part is iterated through from the right and elements are shifted to make place for the elements to be inserted. A track is kept as to whether the 2 elements have been placed or not. The iteration through the sorted part can be stopped when both have been positioned. If the number of elements is even then an extra iteration is required to position the leftover element. Wainwright [8] proposes a class of algorithms based on the Quick sort algorithm. One such algorithm, B sort overcomes the worst case complexity of quick sort which occurs in a nearly sorted data set. In B sort, the pivot is chosen in the middle and two pointers are used to maintain the left and right sub file. Every time a new value is added to the either left or right sub file, compare and exchange operations are done to ensure that the rightmost element of the left sub file is the largest and the leftmost element in the right sub file is the smallest. This is continued until the two pointers reach a common point. It is after this that the partition is created, thus smaller instances of the problem are solved locally leading to improved nearly-sorted list performance. Cantoneet. al. [9] proposes QuickHeapSort that combines the Quick sort and Heap sort algorithm to achieve greater efficiency. QuickHeapSort involves partitioning the array into two halves around a pivot with the larger work area and the smaller heap area. A heap is formed by the elements in the heap area, and the root is extracted and put in its place in the work area, the key replaced by the root is stored in the special child location. The elements shifted into the work area are sorted and the unsorted elements are partitioned to form the areas again till the array is sorted.

Parallel Processing

A major development in the field of computing came when parallel processing was invented, processors with more than one core allowed for multiple tasks to be performed together. With this discovery, algorithms were developed that took advantage of this added power. Vivek Kale et. al. [10] explain that an unordered sequence could be sorted using parallel processors, where in each processor handles a part of the unordered sequence and sorts it using a capable sorting algorithm.

Tang et. al. [11] discusses parallelizing the pre-existing sequential merge sorting network algorithm. In the merge sorting network algorithm, the given array is recursively split into equal halves till they contain just one element and then a bitonic sort on the two sub arrays is performed following which they are merged. It is observed that the merge and bitonic sort functions and the sorting functions have independent recursive calls. These independent tasks are therefore assigned to individual processors to improve the performance. However, if the no of parallel processes are more than the no of processors, the tasks are executed sequentially to avoid synchronization delay. David Bader et. al [12] talk about a practical parallel algorithm for personalized communication and integer sorting by adopting a new standard in message passing called Message Passing Interface (MPI). This provides

a more coherent and platform independent parallel algorithm. ZehraYildiz et al. [13] describes methods to parallelize Bitonic and Radix sort algorithms on many cores GPU's using the Compute Unified Device Architecture (CUDA) platform used by NVIDIA's GPUs. GPUs support the SIMD (single instruction, multiple data), model of parallel processing. The program that is executed in parallel is called kernel which in this case is written in CUDA which is basically an extended version of ANSI C. Threads are a way for a program to divide itself into two or more simultaneously running tasks. A group of threads constitute a block and a group of blocks constitute a grid. There are three types of memory in the CUDA programming model. These are local, shared and global memories. This three-level memory model is used to increase efficiency of data access. Radix sort is parallelized by splitting the initial problem set into smaller problems and assigning it to different processors which compute the bucket index for all elements assigned to it and also counts the number of elements in each bucket and stores this in the global memory. Then the keys within each processor are moved to appropriate buckets. Bitonic sort is one of the fastest parallel sorting algorithms even though its time complexity is O $((\log n)^2)$. It is implemented in parallel by performing each split and compare operation on different cores. Each core works on different parts of an array. The results of each sub array are independent of the other, thus facilitating in fast parallel execution. Experimental analysis found that the parallel versions of these sorting algorithms implemented on GPUs were 20 times faster than their serial counterparts. The parallel radix sort performed better than parallel bitonic sort.

Inoue et. al. [14] proposes Aligned Access Sort that is meant to run on multiple processors using SIMDs (Single Instruction Multiple Data) commands. This algorithm contains two parts, inside core algorithm and outside of the core algorithm. The given data set is divided into a group of elements of blocks of the size of the cache of the processor that are sorted using the inside core algorithm by one core in parallel. These sorted blocks are merged using the outside of the core algorithm. The inside core algorithm involves sorting keys in ascending order, transposing them using comb sort and then reordering them in the earlier order. The outside of the core algorithm uses odd even merge sort and SIMD instructions. SwetaKumari et al. [15] implement a hybrid sorting algorithm that is a combination of parallel Radix Sort and Selection Sort on GPU. It is based on 'Split and Concurrent Selection' strategy. The data to be sorted is split among the cores that are available. Then Radix Sort is applied on the split data in parallel. Binary Search is used to find number of elements which are smaller than the element encountered. This is used while performing parallel Selection Sort to position the elements correctly to obtain the final sorted array. Rahim Rashidyet. al. [16] implements a parallel version of Bubble Sort using Stream Programming model. The code for the implementation is written in simple Java code and takes the help of the JStream package in Java library. This algorithm uses the pipeline communication pattern. A sorter filter is constructed which basically compares 2 data elements. Sorting n numbers would require n sorter filters connected to each other in a pipeline such that output of one is channelized to the input of the next. There are no explicit loops in the code, other than in the creation of the sorter filters. S.Rajasekaran [17] provides a novel solution to the parallel disk sorting problem, with the (l,m)-mergesort. The (l,m) mergesort is an algorithm that is a generalization of the odd even mergesort and the bitonic mergesort. In this sorting technique the given sequence is divided into l subparts done recursively till each part contains a single value. These l sequences are merged back by shuffling them into m groups and then merging those using routine comparisons and exchanges.

Independent Algorithms

These are some algorithms that generally are based on completely new ideas. Unlike the algorithms that were encountered earlier, these algorithms bring in fresh ways of thinking about the problem of sorting data.

Quintal et. al. [18] proposed Bit-index Sort, which is a fast non-comparison sorting algorithm for sequences of integer numbers without repeated elements, i.e. for partial permutations. Bit-index sort uses a bit-array, implemented as an array of unsigned integers, to classify input data. In its classification stage, a bit-array is obtained for the given array. In the retrieval stage, the bit-array is treated as an unsigned integer and the sorted array is obtained through a series of bit-wise operations. Wang [19] proposes a new sorting technique, Self-Indexed Sort, a non-compare based sorting algorithm which has a time complexity of O(n) and a space complexity of O(n+m),where n is the size of the array and m is the auxiliary space required to sort the data. The approach implemented is this algorithm consists of 3 steps. It starts of by initializing the sorting space. A self-indexed arrangement which is a linear mapping is then assigned based on the value of the keys in the sorted data. This is followed by compressing through preserving the order at the time of placing the sorted but scattered data in its intended position. Sehgal et al[20] proposed Use-Me Sort, which provides a trade-off between space and time complexity with a performance that is better than the existing sorting algorithms belonging to the $O(n^2)$ class.

Rupayan Dutta et al. [21] introduce a new sorting algorithm which is most useful in cases where the cost of moving each element is not the same. In this algorithm,

element of i^{th} cost is moved to its correct position by making a maximum of i+1 exchanges/movements. Thus, element with the highest cost (i=0) will only be moved once. The algorithm's time complexity is O (cn^2) where c denotes the number of categories. Its experimental efficiency lies between that of Selection sort and Insertion sort. Dominik Schultes [22] proposes Rainbow Sort. Inspired by the natural phenomenon of light rays being arranged in a spectrum by their wavelength, Schultes proposes a sorting technique that involves assuming each key with a corresponding wavelength, passing a ray formed by wavelengths through a prism and collecting those rays and obtaining the key associated to each ray with an inverse mapping function. The larger wavelength rays arrive first and hence the order of arrival of rays provides the sorted set of values. Diosanet. al. [23] proposes Friction based sorting, which is based on the property that frictional force, or drag, acting on a freely falling object is dependent on it weight. The heavier objects will fall faster. The keys are associated with a uniform ball with proportional weight. These balls are then dropped from a fixed height at the same time. The collector receives the balls and weight on the collector gives us the weight of the ball that has fallen and thus the order of the fallen balls sorts the keys.

Variations of Traditional Algorithms

- Bubble sort.

Bubble sort is a classic sorting algorithm. Following the brute force ideology, and is very important in the field of computing. However, as explained by Astrachan [24] Bubble sort being an extremely popular $O(n^2)$ sort is actually a misnomer. Bubble sort is a $O(n^2)$ algorithm. As compared to any traditional sort, the process of exchanging values requires twice as many redundant operations as compared to the others. It is expedient to use this $O(n^2)$ algorithm for sorting only a handful of items. Bubble sort needs no auxiliary space i.e other than the input vector, no other vector is needed. The disadvantage of the bubble sort algorithm is that it does not have many real-world applications other than teaching beginners the basics of sorting. Yet, there are useful and more efficient variants of Bubble sort. Avinash Bansal [25] presents a sorting algorithm called Avi Sort, based on Bubble Sort where elements at alternate positions are compared rather than adjacent positions. There are 2 sets of alternate positions, (1,3,5,…) and (2,4,6,…). The largest elements from both these sets of positions are being pushed to the end. Then the 2 elements which were pushed are compared to finalize their relative positions. The execution time is equivalent to that of Insertion sort. Shagun Bhardwaj [26] introduces a new bidirectional sorting algorithm called Mark-Set Sort that improves efficiency of Bubble sort by taking into account partial ordering of data. The largest and smallest among the unsorted elements are moved to the beginning and the end respectively. The position where the last swap occurred is saved and the next iterations begin from this position which skips over the elements which have fallen into place. Experimental analysis confirmed that for sorting large data-sets, Mark-set sort algorithm is better than Insertion sort.

- Shellsort

Shellsort is an improvement on bubble sort where there is a offset sequence that specifies the offset between the two keys being compared. The efficiency of Shellsort lies in the correct specification of the offset sequence. In the algorithm Randomized Shellsort proposed by Goodrich [27], a variant of Shellsort, it employs a fixed offset sequence containing decreasing powers of 2. Regions are created in the array on the basis of the offset sequence and then three region compare and exchange operations are performed on the corresponding values in each region, namely the shaker pass, the shaker-brick pass and odd even comparisons. After this a cleanup phase is performed for the nearly sorted array performing two passes up and down the array. Goodrich [28] raises another variant of Shellsort by proposing Spin-the-Bottle sort and Annealing sort. Spin-the-Bottle sort and Annealing sort are two algorithms which fall under the category of random round robin comparisons. Spin-the-Bottle sort is an algorithm where two numbers are chosen at random and compared as long as the array is unsorted. This algorithm is highly inefficient with a worst case complexity more than that of Bubble sort. However, if the algorithm is modified slightly such that we restrict the range within which a number can be chosen at random and restrict the number of such random exchanges for each range, we get Annealing sort. Annealing sort has three phase sorting process with three sets of Range restriction sequences and repetition sequences. On implementing these sequences, a highly efficient sorting algorithm is attained. Goodrich [29] proposes ZigZag Sort, a variant of the Shellsort family of algorithms that runs in O(n log n). ZigZag is the first Shellsort algorithm that is deterministic and non-rigid. ZigZag sort is based on the concept of a ε-halver. In the ZigZag sort algorithm, first a pass is performed on the array that creates smaller sub arrays which is followed by a shellsort pass down the array. In each iteration, the sub arrays are first swapped, compared and then exchanged. The ε-halver is used in the inner iteration to perform the compare exchange operations.

III. CONCLUSION

In this paper, a variety of sorting algorithms were discussed, some of which were novel. A few existing algorithms have also been discussed in order to broadcast a supplementary comprehension on these algorithms. The

algorithms differed not only in perception and implementation but also in application and space-time complexities.

IV. ACKNOWLEDGMENT

The work reported in this paper is supported by the college [BMSCE, Bangalore] through the TECHNICAL EDUCATION QUALITY IMPROVEMENT PROGRAMME [TEQIP-II] of the MHRD, Government of India.

V. REFERENCES

[1] SonalBeniwal, Deepti Grover, "Comparison Of Various Sorting Algorithms: A review", International Journal of Emerging Research in Management and Technology, Volume 2, Issue 5, pp 83-86, May 2013, ISSN:2278-9359.

[2] P.Sumanthi, V.V Karthikeyan, "A New Approach for Selection Sorting", International Journal of Advanced Research in Computer Engineering and Technology(IJARCET), Volume 2, Issue 10, pp 2720-2724, October 2013, ISSN:2278-1323.

[3] JehadAlnihoud, Rami Mansi, "An Enhancement of Major Sorting Algorithms", The International Arab Journal of Information Technology, Volume 7, January 2010.

[4] Yashwant Singh Patel, Nitish Kumar Singh, Lalit Kumar Vashishtha, "Fuse Sort Algorithm: A Proposal of Divide and Conquer based Sorting Approach with O (nloglog n) Time and Linear Space Complexity", International Conference on Data Mining and Intelligent Computing (ICDMIC), New Delhi, India, 5-6 Sept. 2014, DOI:10.1109/ICDMIC.2014.6954240.

[5] Wang Min, "Design and Analysis on Bidirectional Selection Sort Algorithm", 2nd International conference on Education Technology and Computer (ICETC), Shanghai, China, 22-24 June 2010, pp V4-380 - V4-383, DOI:10.1109/ICETC.2010.5529660.

[6] KamleshNenwani, Vanita Mane, SmitaBharne, "Enhancing Adaptability of Insertion Sort through 2-Way Expansion", 5th International Conference Confluence The Next Generation Information Technology Summit (Confluence), Noida, India, 25-26 Sept. 2014, DOI:10.1109/CONFLUENCE.2014.6949294.

[7] Wang Min, "Analysis on 2-Element Insertion Sort Algorithm", International Conference on Computer Design and Applications (ICCDA), Qinhuangdao, China, 25-27 June 2010, DOI:10.1109/ICCDA.2010.5541165.

[8] Roger L. Wainwright, "A Class of Sorting Algorithms Based on Quick sort", Communications of the ACM, Volume 28, Issue 4, April 1985, pp 396-402, DOI:10.1145/3341.3348.

[9] DomenicoCantone, GianlucaCIncotti, "QuickHeapSort, an Efficient Mix of Classical Sorting Algorithms", Algorithms and Complexity 4th Italian Conference, CIAC 2000 Rome, pp 150-162, DOI:10.1007/3-540-46521-9_13.

[10] Vivek Kale, Edgar Solomnik, "Parallel sorting pattern", Proceedings of the 2010 Workshop on Parallel Programming Patterns, Article No. 10, Appears in ICPS: ACM International Conference Proceeding Series, ISBN: 978-1-4503-0127-5, DOI:10.1145/1953611.1953621.

[11] Peiyi Tang, Doug Serfass, "Parallelizing the merge sorting algorithm on a multi-core computer using Go and Clik", Proceedings of the 49th Annual Southeast Regional Conference, 2011, pp 144-149, DOI:10.1145/2016039.2016080.

[12] David A. Bader, David R. Helman, Joseph Jaja, "Practical parallel algorithms for personalized communication and integer sorting", Journal of Experimental Algorithmics (JEA), Volume 1, January 1996, ACM New York, NY, USA, ISSN: 1084-6654, EISSN:1084-6654, DOI :10.1145/235141.235148.

[13] ZehraYildiz, Musa Aydin, GürayYilmaz, "Parallelization of Bitonic and Radix Sort Algorithms on many core GPUs", International Conference on Electronics, Computer and Computation (ICECCO), Ankara, Turkey, November 7-9, 2013, DOI:10.1109/ICECCO.2013.6718294.

[14] Hiroshi Inoue, Takao Moriyama, Hidaeki Komatsu, Toshio Nakatani, IBM Tokyo Research Laboratory "AA-Sort:A New Parallel Sorting Algorithm for Multi-Core SIMD Processors", 16th International Conference on Parallel Architecture and Compilation Techniques, 15-19 Sept. 2007, DOI:10.1109/PACT 2007.4332611.

[15] SwetaKumari, DhirendraPratap Singh, "A Parallel Selection Sorting Algorithm on GPUs Using Binary Search", IEEE International Conference on Advances in Engineering and Technology Research (ICAETR), 2014, Unnao, India, 1-2 Aug, DOI:10.1109/ICAETR.2014.7012819.

[16] Rahim Rashidy, SaeidYousefpour, MohamadKoohi, "Parallel bubble sort using stream programming paradigm", 5th International Conference on Application of Information and Communication Technologies (AICT), Baku, Azerbaijan, 12-14 Oct. 2011, DOI:10.1109/ICAICT.2011.6111024.

[17] S.Rajasekaran,"A Framework for Simple Sorting Algorithms on Parallel Disk Systems", SPAA '98 Proceedings of the tenth annual ACM symposium on Parallel algorithms and architectures, 1998, pp 88-97, ISBN:0-89791-989-0, DOI:10.1145/277651.277675.

[18] Curi-Quintal, L.F, Cadenas, J.O, Megson, G.M, "Bit-index sort: A fast non-comparison integer sorting algorithm for permutations", International Conference on Technological Advances in Electrical, Electronics and Computer Engineering, pp 83-87, 9-11 May 2013, DOI:10.1109/TAEECE.2013.6557200.

[19] Sunny Y. Wang, "A new sort algorithm: self-indexed sort", ACM SIGPLAN Notices, Volume 31, Issue 3, March 1996, pp 28-36, DOI:10.1145/227717.227725.

[20] MohitSehgal, Nihal Kumar, "Use-Me Sort: A new Sorting Algorithm", International Journal of Applied Information Systems (IJAIS), Foundation of Computer Science FCS, New York, USA, Volume 7, No.8, Sept 2014, ISSN:2249-0868.

[21] Rupayan Dutta, SayanNayak, "A Novel In-Place Sorting Algorithm with Element wise Restricted Swap", Third

International Conference on Computer, Communication, Control and Information Technology (C3IT), 2015, Hooghly, India, 7-8 Feb. 2015, DOI:10.1109/C3IT.2015.7060123.

[22] Dominik Schultes,"Rainbow Sort: Sorting at the speed of the light", Natural Computing (2006), Volume 5, Issue 1, March 2006, pp 67-82, DOI: 10.1007/s11047-004-3379-3.

[23] Laura Diosan, MihaiOltean,"Friction-based sorting", Volume 10, Issue 1, pp 527-539, March 2011, DOI: 10.1007/s11047-010-9201-5.

[24] Owen Astrachan, "Bubble Sort: An Archaeological Algorithmic Analysis", ACM SIGCSE Bulletin, Volume 35, Issue 1, January 2003, Pages 1-5, DOI: 10.1145/611892.611918.

[25] Avinash Bansal, "Avi Sort Algorithm", International Conference on Human Computer Interactions (ICHCI), 2013, Chennai, India, 23-24 Aug 2013, DOI: 10.1109/ICHCI-IEEE.2013.6887800.

[26] Shagun Bhardwaj, "Mark-Set{} Sort: An Eidetic Sorting Technique", Third International Conference on Advanced Computing and Communication Technologies (ACCT), 2013, Rohtak, India, 6-7 April 2013, DOI:10.1109/ACCT.2013.40.

[27] Goodrich, M.T, "Randomized shellsort: A simple data-oblivious sorting algorithm", J.ACM, Article 27, Volume 58, Issue 6, December 2011, DOI:10.1145/2049697.204970.

[28] Michael T. Goodrich, "Spin-the-Bottle Sort and Annealing Sort: Oblivious Sorting via Round-Robin Random Comparisons", Algorithmica 68, Volume 68, Issue 4, pp 835-858, April 2014, DOI:10.1007/s00453-012-9296-5.

[29] Michael T. Goodrich, "Zig-Zag Sort: a simple deterministic data-oblivious sorting algorithm running in O (n log n) time", Proceeding of the 46th Annual ACM Symposium on Theory of Computing, 2014, pp 684-693, DOI:10.1145/2591796.2591830.

