DETERMINING WHICH SORTING ALGORITHM IS SUITABLE BASE ON THE GIVEN DATA

PARHAM HOUSHMAND[*], YASAMIN VAZIRI[*]
MAHYA MOTTAGHI[*] & ELHAM SOLEIMANI[*]

July 2022

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1   ABSTRACT

source neveshtam akharesh yademoon nare Sometimes the cost of sorting algorithms is very different, and finding the best sorting algorithm is a great help in reducing the time and space used for sorting. There are linear sorting algorithms that need to be preprocessed before they can be used. The quick sort algorithm also needs the data to be unsorted and sometimes according to the structured data we have (heap, linked list) they need another special algorithm (heap and merge sort) also sometimes the data is very voluminous, and We have to use an algorithm that performs in place sorting. According to these interpretations, we perform linear pre-processing on the primary data to find the best sorting algorithm with the features we mention from each sorting algorithm.

* Department of Mathematical science, Sharif University of Technology, Tehran, Iran

2 INTRODUCTION

inja bayad kamel konim kholase ravesh khodemoono As there are different sorting algorithms, we know they have bold differences with each other and we are going to give a short summary about them. But the main reason of this paper is to construct the best sorting algorithm with given input which has the minimum time complexity no matter what the size of input is.Our algorithm firstly checks each number behind the head whether it is ascending or descending, and checks whether the number is negative or not, then it finds the largest length of the number and the largest number, then it checks its inversion and then it finds whether the inversion is larger than nlogn or not.After that the ascending should not be less than nlogn and negative.

3 METHODS

inja zanak ax doos dare model konim ax nemoodar jadval. link githubam bayad bezarim comment gozari shode bashe. tahlil azmayesham bayad benevisim (khoobia va badiaye algoritm khodemoon)

1. different methods of sorting
2. different techniques
3. Analysis of sorting techniques
4. best sorting algorithm analysis

3.1 different sorts time complexity

1. Bubble sort and Insertion sort
   The average and worst case time complexity of both is n^2 while the best time complexity is n. This happens when the array is already sorted, instead the worst case is when the array is completely reversed. and as a use, these algorithms work better for the smaller data and insertion sort is of order of n when the array is sorted.
2. Merge sort
   Merge sorts best, average and worst case time complexity is $nlogn$ and again similarly independent of distribution of data. for this algorithm we can say it works the best when the array is reversed sort
3. Heap sort
   like merge sort, its best, average and worst case time complexity is $nlogn$ which is independent of distribution of data. heap sort's property is that the space is of order of 1.
4. Quick sort
   It is a divide and conquer approach with recurrence relation:

   $$T(n) = T(k) + T(n - k - 1) + cn$$

   Its worst case is when the array is sorted or reverse sorted, the partition algorithm divides the array in two sub arrays with 0 and $n - 1$ elements. Therefore,

   $$T(n) = T(0) + T(n - 1) + cn$$

   Solving this we get:

   $$T(n) = O(n^2)$$

   On an average, the partition algorithm divides the array in two subarrays with equal size. Therefore,

   $$T(n) = 2T(\frac{n}{2}) + cn$$

and finally

$$T(n) = O(n \log n)$$

the algorithm is of order of n^2 when the array is revered. Non-comparison based sorting:
In non-comparison based sorting, elements of array are not compared with each other to find the sorted array.

5. Radix sort
   Best, average and worst case time complexity for radix sort is nk where k is the maximum number of digits in elements of array.

6. Count sort
   For this algorithm best, average and worst case time complexity is $n + k$ where k is the size of count array.

DIFFERENT PARAGRAPH DESCRIPTION

3.2 common techniques

1. in place & out place technique
   A sorting technique is inplace if it does not use any extra memory to sort the array. Among the comparison based techniques discussed, only merge sort is outplaced technique as it requires an extra array to merge the sorted subarrays. Among the non-comparison based techniques discussed, all are outplaced techniques. Counting sort uses a counting array and bucket sort uses a hash table for sorting the array.
2. online & offline technique
   A sorting technique is considered Online if it can accept new data while the procedure is ongoing i.e. complete data is not required to start the sorting operation. Among the comparison based techniques discussed, only Insertion Sort qualifies for this because of the underlying algorithm it uses i.e. it processes the array (not just elements) from left to right and if new elements are added to the right, it doesn't impact the ongoing operation.
3. Stable & unstable technique
   A sorting technique is stable if it does not change the order of elements with the same value. Out of comparison based techniques, bubble sort, insertion sort and merge sort are stable techniques. Selection sort is unstable as it may change the order of elements with the same value. For example, consider the array $4, 4, 1, 3$. In the first iteration, the minimum element found is 1 and it is swapped with 4 at 0th position. Therefore, the order of 4 with respect to 4 at the 1st position will change. Similarly, quick sort and heap sort are also unstable. Out of non-comparison based techniques, Counting sort and Bucket sort are stable sorting techniques whereas radix sort stability depends on the underlying algorithm used for sorting.

3.3 sorting techniques analysis

1. When the array is almost sorted, insertion sort can be preferred.
2. When order of input is not known, merge sort is preferred as it has worst case time complexity of $n \log n$ and it is stable as well.
3. When the array is sorted, insertion and bubble sort gives complexity of n but quick sort gives complexity of n^2.

3.4 Constructed Algorithm

firstly it checks the inversion number of the array then it checks that if it is ascending or descending and if it has negative element and it finds max value and max value digits then it checks if number of inversions are less than nlogn and number of ascending and descending numbers are equal to the length of the array or the length of the array is (0,20] the we sort with insertion. if it is only descending (reversed sorted)we sort with merge sort and if it is completely ascending we use insertion. now if it doesn't contain any negative number we check if the max value is less than nlogn we use count sort and if max number of digits is less than nlogn we return radix sort.if none of these algorithms were not the answer we choose between quick and merge sort. if number of ascending and descending numbers are less than n we use quick sort, else we return merge sort.
python code of our algorithm is:

4 RESULTS AND DISCUSSION

here is the plot of the growth of different algorithms with same giving input

5 REFERENCES