Analyzing the Performance of CS 430 Project

WenLong Li ,Fan Guo , Hao Liu

Supervised by:

Lan Yao

Abstract

Quicksort is one of the most widely used sorting algorithms. QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quickSort that pick pivot in different ways.One is pick a random element as pivot.the other is pick median as pivot.

Median is an algorithm for selecting the k^{th} largest element in an unordered list, having worst case linear time complexity [1]. It finds the approximate median in linear time which is then used as pivot in the quickselect algorithm. This approximate median can be used as pivot in Quicksort, giving an optimal sorting algorithm that has worstcase complexity O(n log n) [2].

Graphs were also plotted for the usual Quicksort function and the Quicksort function that uses the proposed median as pivot for relatively large values of size of array and results were compared. These confirm that proposed algorithm indeed has worst-case complexity O(n log n) for Quicksort.

keywords：Median , Quicksort, Partition, Median Selection

Introduction

We know select one number as the pivot from unordered list , If unordered list is partially ordered, the fixed selection of pivots makes quicksort less efficient. To alleviate this situation, pick a random element as pivot is a good idea.

Median is the middle value in a data set. Median selection is a problem that can be considered a special case of selecting the ith smallest element in an ordered set of n elements, when $i = \lceil n/2 \rceil$. An approach to solve this problem could be to sort the list and then choose the ith element. This could be using any sorting algorithm such as - Heapsort that has the worst case upper bound as O(n log n), Quicksort that has an expected running time O(n log n) though its running time is $O(n^2)$ in the worst case. Once the data values are sorted, it takes O(1) time to find the ith order statistics. Using an optimal sorting algorithm, the aforesaid approach gives complexity of O(n log n) as upper bound for selecting ithorder statistics [2].,A better method is pick median element as pivot .

Methodology

This report is to answer those questions:

1、In different data conditions, the performance difference between Quicksort with median as pivot and Quicksort which chooses a random element in the array as the pivot .

2、In different data conditions, the performance difference between order statistics median finding algorithm that use randomized and not use randomized.

3、Time complexity verification of using median Quicksort algorithm.

This report main method used:

In the same environment, different algorithms are tested with different data of different orders of magnitude and different characteristics, and the actual running time of the program is recorded.

The main data sets include:

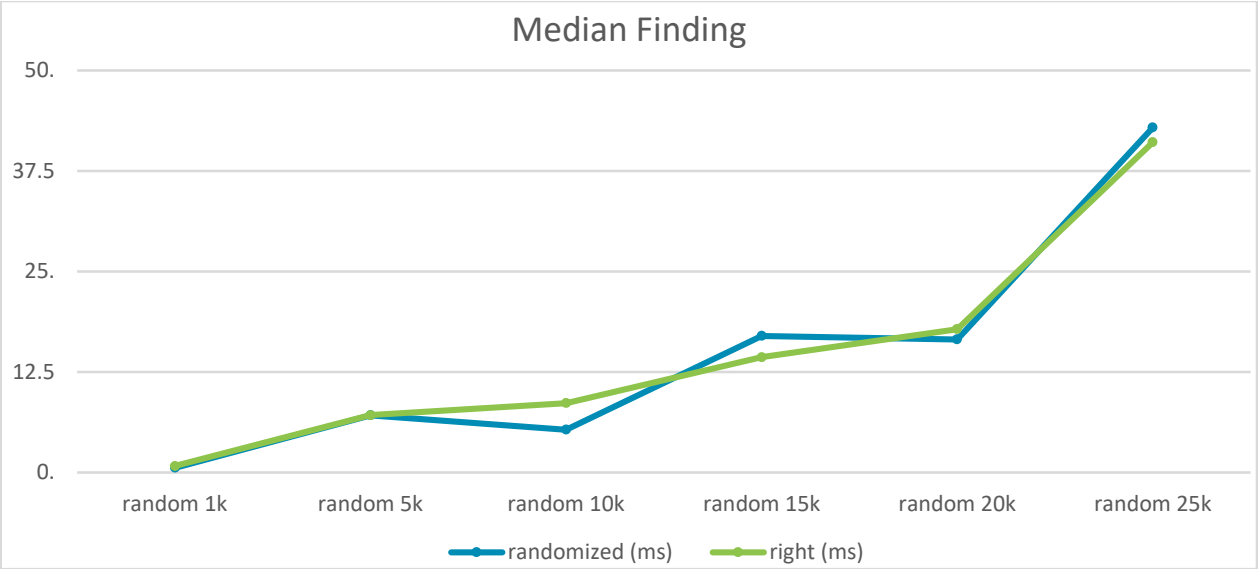
positive order data, reverse order data, data generated with randomized, data with a lot of number of same elements, data of different orders of magnitude.

Results and Discussions

Median Finding	randomized (ms)	right (ms)
positive order 1k	0.48	93.26
reverse order 1k	0.86	87.58
random 1k	0.63	0.35

The median finding algorithm based on random numbers is not affected by whether the input sequence is sorted, the median finding algorithm based on the right boundary value is greatly affected by whether the input sequence is sorted.

	randomized (ms)	right (ms)
random 1k	0.59	0.79
random 5k	7.1	7.12
random 10k	5.32	8.62
random 15k	16.97	14.34
random 20k	16.54	17.82
random 25k	42.9	41.07



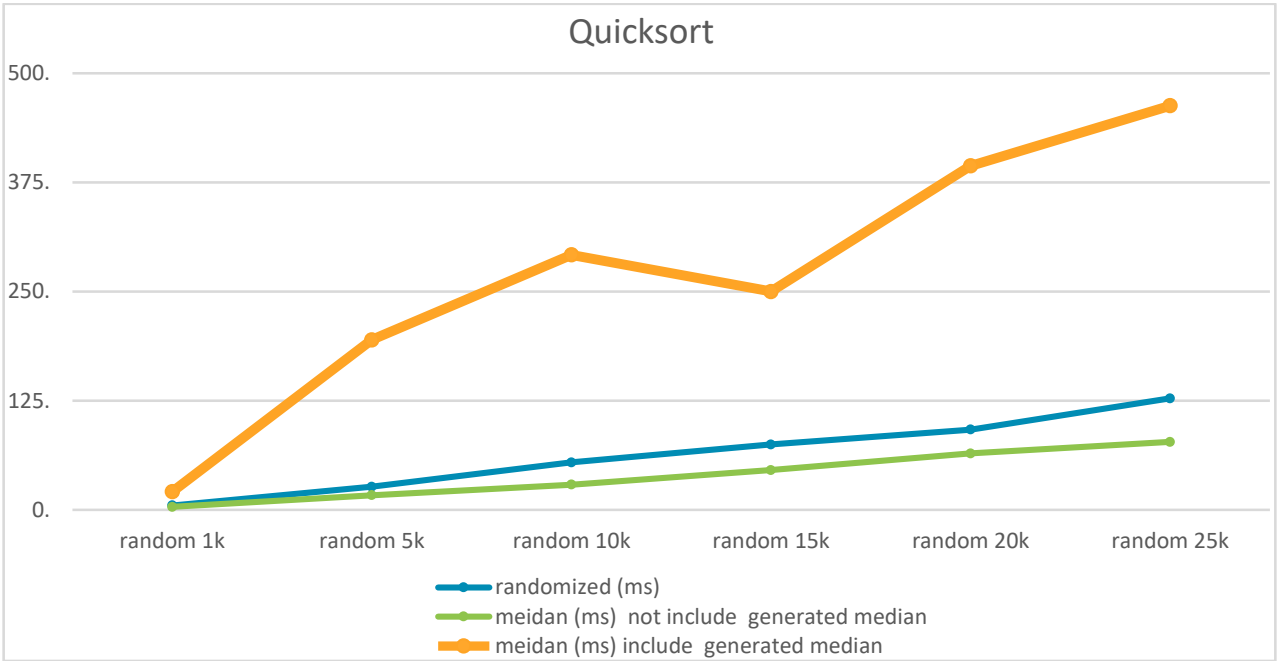
the median finding algorithm based on random numbers and the median finding algorithm based on the right boundary value have the same execution efficiency for the input random sequence.

In the case of random data, the two generate median algorithms are almost the same, and the average time complexity is O(n).

Quicksort	randomized(ms)	median(ms) not include generated median	median(ms) include generated median
positive order 1k	4.61	2.93	24.12
reverse order 1k	4.35	2.69	25.17
random 1k	4.43	3.15	26.06

	randomized (ms)	meidan (ms) not include generated median	meidan (ms) include generated median
random 1k	4.82	3.22	20.55
random 5k	26.28	16.66	194.55

	randomized (ms)	meidan (ms) not include generated median	meidan (ms) include generated median
random 10k	54.18	28.76	291.98
random 15k	74.74	45.29	250
random 20k	91.81	64.45	393.84
random 25k	127.44	77.72	462.68



The base median Quicksort algorithm and the random number-based Quicksort algorithm have no effect on whether the input sequence is sorted, The Quicksort algorithm based on random numbers is more efficient than the Quicksort algorithm based on the median value. For inputting different sequences of the same order of magnitude(positive order, reverse order , random)

```
def partition(self, n, p, r):

    if str(self.type_) == "median":
        t_median_=time.time()
        x = median_finding_randomized(n, p, r, int((r - p + 1) / 2))
        self.t_median_+=time.time()-t_median_
        i = p - 1

        for j in range(p, r):

            if n[j] <= x:
                i += 1
                temp = n[i]
                n[i] = n[j]
                n[j] = temp

        return i

def _quickSort(self, n, p, r):
    if p < r:
        q = self.partition(n, p, r)
        self._quickSort(n, p, q - 1)
        self._quickSort(n, q + 1, r)
```

In the Quicksort algorithm which median that use order statistics algorithm generate as the pivot .

Because every time a partition function splits n, it is necessary to re acquire the median time complexity O(n) and add the circulate within the partition , the recurrence is

T(n)=2T(n/2)+O(2n) ==>. T(n)=O(nlgn)

Quicksort algorithm which median that use order statistics algorithm generate as the pivot , it 's time complexity of the algorithm is the same as that Quicksort which chooses a random element in the array as the pivot.

But because the Quicksort using the median algorithm has a higher constant C,so it takes more run time on average than Quicksort with randomized

The Quicksort algorithm based on random numbers is more efficient than the base median Quicksort algorithm

Reference

1.Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. Time bounds for selection. Journal of Computer and System Sciences, 7:448{461, 1973.
[2] Introduction to Algorithms, Second Edition, Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, The MIT Press, ISBN 0- 07-013151-1 (McGraw-Hill)