Advanced Algorithms - Randomized Algorithms

Yao Yao (1739440)

In this report, I will analyze the lazy select algorithm, quick select algorithm and the median of medians algorithm. After studying these three selection algorithms, I will describe the findings.

Quick select is using the idea of A quick sort algorithm, so let's say we have an array of A [left...right]

1. Select A pivot element M in array A first

2. Go through the array (from left to right), put elements larger than pivot element M on the right of pivot element, and elements less than or equal to pivot element on the left of pivot element, and then the pivot element in the array has the position I. So， pivot M is the i-th order statistic in the array

3. Compare pivot location I and target order statistic k. If I =k, then return pivot M directly. If k< I, update right= I −1, go to the next action to continue running; If k> I, update left= I +1, k= I −left, go to the next action and continue.

In this way, we can find the k-th target order statistic, the expected running time of this operation is O(n).

When I run the coding:

Input array is: {96, 47, 95, 38, 53, 45, 3, 92, 20, 73}

The 1st max number is 3.

The 2nd max number is 20.

The 3rd max number is 38.

The 6th max number is 53.



Another test:

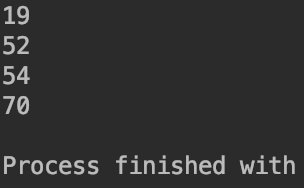
Input array is: {62, 66, 70, 54, 74, 98, 83, 52, 80, 19}

The 1st max number is 19.

The 2nd max number is 52.

The 3rd max number is 54.

The 6th max number is 70.



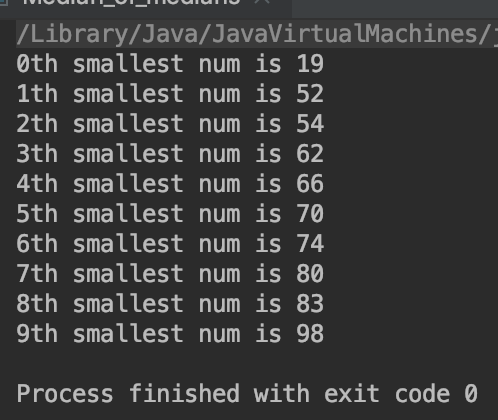
In the process of selecting an array, again and again, the data has gradually become orderly. In the beginning, the input array was 62, 66, 70, 54, 98, 83, 52, 80, 19. After four times of selection, it was found that the array had been updated to 19, 52, 54, 62, 66, 70, 74, 98, 80, 83, which was nearly orderly. Ordered arrays are lethal for fast algorithms, and if you don't do anything to optimize the fast algorithm, the fast algorithm will reach the worst time complexity O(n2), because the ordered data will cause the grouping of quicksort to be extremely unbalanced.

Quick select algorithm is the same, should avoid grouping input orderly array of extreme imbalance, so he made the following optimization, before a quick choice, first of all, from an array of the head, middle, tail to select the three elements, find out the second element in these three elements, and compared with the last element of an array of exchange, so I can avoid grouping of extreme imbalance, but just can guarantee to avoid grouping extremely unbalanced situation, it is possible to group is not balanced, the median of medians algorithm can very good balance.

This algorithm relies on a well-designed pivot selection method, in which the median of the median is selected as the pivot element, thus ensuring linear time complexity in the worst case, and beating the quicksort algorithm with average O(nlogn) and worst O(n2) complexity.

After I run the coding by the median of the median algorithm, the result is like this:

Input array is: {62, 66, 70, 54, 74, 98, 83, 52, 80, 19}



For the median of the median algorithm, each group of five elements, to group the array and end up with n over 5 groups, excluding the groups with less than five elements and including pivot element x, at least 3×(n/ 5-2)≥3n/10-6 elements are greater than x, similarly, at least 3n/10-6 elements are less than xx, so in each group, the most unbalanced case of the two groups is 7:3, so the recurrence is obtained, where 130 is a random constant.

T (n) = {O (1), n < 130

T (n) = {T (n / 5) + T (7 n / 10 + 6) + O (n), n≥130

By calculating the above equation, we can know that the time complexity is O(n).

When I test the same array, the quick select algorithm takes less time than the median of medians algorithm. But the time difference is not obvious because the data is not large enough.

Test input array is: {62, 66, 70, 54, 74, 98, 83, 52, 80, 19}

 （the quick select algorithm）（the median of medians algorithm）

In practice, however, the median of medians algorithm is usually 1 to 2 times slower than quick select, so it's not commonly used.

For lazy select algorithm, the probability of the k-th largest element in S is 1-O(n-1/4), which means the probability of the k-th largest element in S is 1-o (n-1/4), which requires 2n+O(nlgn) comparisons. So, testing by lazy select algorithm, the running time is: 

Based on the comparison of the three algorithms, the quick select algorithm is widely used because of its high efficiency and good average case time complexity. By contrast, the lazy select algorithm is used less often. However, the median of medians algorithm can achieve O(n) time complexity by searching for the k-th largest or k-th smallest element in a set of data. Therefore, these randomized algorithms should be mastered and can be applied.