QuickPivot

A quick pseudo-deterministic pivot selection algorithm

P. Roldan, S. Schwartz
Yeshiva College

Introduction

This work represents an improvement on the classical QuickSort algorithm introduced by Hoare in 1959. There have been many improvements to the core algorithm, including a series of improvements introduced by Bentley and McIlroy. One such improvement focuses on a subroutine of quicksort, namely the process of selecting a pivot.

As a matter of background, it is worth restating that the ideal pivot in a quicksort subroutine is the true median of the array to be sorted. Picking the true median as the pivot results in exactly half of the array being pivoted to either side of the pivot and the most efficient work at every level. While selecting the true median has a probability of 1/length of the array, an approximation of the median will result in nearly identical performance. For our purposes, a good approximation of the pivot will be a pivot between the 25th and 75th percentiles.

There are, broadly speaking, two classes of algorithms for selecting a pivot. The first class of algorithms randomly selects a pivot using O(1) work. The trivial example would involve selecting an array index at random which would result in a good pivot 50% of the time. The second class of algorithms deterministically selects a pivot using O(n) work. The median-of-median algorithm is a good example of the second class of algorithms. The median-of-medians divides the input (a list of length n) into groups of at most five elements, computes the median of each of those groups using some subroutine, and then recursively computes the true median of the n/5 medians found in the previous step.

Each class of pivot selection algorithms has obvious tradeoffs that are empirically observed. There have been improvements made to the non-deterministic class of algorithms such as median-of-3 but they have yet to improve the worst-case for quicksort which remains $O(n^2)$.

Of course, the optimal algorithm would be a deterministic, O(1) pivot selection algorithm. This would also imply a worst-case $O(nlogn)$ for quicksort.

Part 1 - Single Pivot Selection

Algorithm

I propose an algorithm that falls into the third class of algorithms, one that uses only O(1) work while deterministically selecting a good pivot. The algorithm is relatively simple: we take

m indices in the array (or sub-array) and average them. We then select the index of those *m* indices that is closest to the average as our pivot.[1]

The total number of operations is *m* additions and one division operation or *m*+1 work. In practice, we chose small values of *m* relative to the size of the array and achieved O(1) performance. This would guarantee a worst-case $O(nlogn)$ for quicksort using the quickpivot selection algorithm.

Proof

The algorithm may be familiar as it effectively describes employing the Central Limit Theorem. Indeed, applying the terminology of the Central Limit Theorem, we would say that our pivot is the element in our sub-array closest to the sample mean.

However, this algorithm is not a standard reduction to the Central Limit Theorem as we are seeking the sample median, not the sample mean. For a uniform distribution, the sample median is well-approximated by the sample mean.

To further illustrate the effectiveness of the algorithm, we will describe the distribution of our approximation given a uniformly distributed integer array of size 1,000,000 with each index holding a randomly generated integer with a range (0, 100,000,000).[2] If we were to follow the trivial way to pick a random index, our chances of selecting an integer in the 25th to 75th percentile would be 50% in this uniform distribution. Say we choose a small *n* value, such as 7. By the CLT, the distribution of our sample mean would approximate a normal distribution with a mean of 50,000,000 and a Standard Deviation of$28,867,513/\sqrt{7} \simeq 10,910,894$. Because our distribution is normal, 95% of the time we will obtain pivots between 28,178,212 and 71,821,788.

This represents a significant improvement over both classes of selection algorithms, achieving a quasi-deterministic approximation of the pivot for only O(1) work.

Empirical Results

In real-world testing, QuickPivot marginally outperforms Tukey's Ninther and falls short of the performance found in Java's implementation of Dual-Pivot Quicksort.

Part 2 - Dual Pivot Selection

Summary: To select a pivot for use in QuickSort, QuickSelect, and related algorithms, we average *m* indices in the array and then select the index from that group that is closest to

[1] While in theory any value of *n* can be used, I found *n* values near 7 to be fastest.
[2] It can be shown that skewed distributions, and really any distribution, will reduce to a similar result

the average. With *m* values larger than 11, QuickPivot selects a pivot between the 25th and 75th percentile of the values in the array with more than 99% probability, while only performing O(1) work.

Citations

Hoare, C. A. (1962). Quicksort. The Computer Journal, 5(1), 10–16.

Engineering a Sort Function by Jon L. Bentley and M. Douglas McIlroy. Software-Practice and Experience, Vol. 23 (11), 1249-1265 (November 1993)

Blum, M.; Floyd, R. W.; Pratt, V. R.; Rivest, R. L.; Tarjan, R. E. (August 1973). "Time bounds for selection". Journal of Computer and System Sciences. 7 (4): 448–461.