Algorithm for creating the worst case median-of-three quicksort.

In order to find the worst case for the median-of-three quicksort, the following assumptions were made:

1. The leftmost point would be the pivot in the median-of-three every time.
2. The middle point would be the only number in the list that was less than the pivot every time.
3. The rightmost point would be the greatest number in the list.

Using these parameters we created a list and “sorted” it using the previous assumptions, ensuring that the recursive call would be made on the longest possible sublist.



In the example above of 25 elements, the green cells are the sorted segments, and the yellow cells are the pivots for each time step.

Each list element was labeled from 0 to 24 to track the order that the numbers were in originally. It was found that the original order was:



Several patterns were found in the worst case.

1. From the middle index (n/2) to the second to last index (n-2), fill each element counting from 0 by 2 (0, 2, 4, … , n-2 ).
2. The last index (n-1) is always the largest number in the list.
3. The first half of the list has the following pattern:

For each iteration , find the first space still unfilled and fill it with the next available odd number while jumping spaces until the middle of the list is reached.

The first open space for each iteration is given by the equation, for. The number of spaces that is jumped is given by the equations for the same.

Below is the code which implements this algorithm.

From these patterns the algorithm for creating a list of worst case for the quicksort was made.

Code snippet for the worst case for the front of the list.

int l = 0, start, step;

for (int p = 1; p < n; )  
{  
    start = pow (2, l) - 1;   
    step = pow(2, l+1);   
  
    if( start > (n-1)/2)  
      break;  
  
    while (start < (n-1)/2)  
    {  
       scrambled[start] = sorted[p];   
       p = p + 2;   
       start = start + step;   
    }  
    l++;   
  }

To determine the BigOh time two methods were used. The first was to plot points for different values of fit a line to it. The second was to take that approximation of the plot function and plot as discussed in class earlier this semester to see if the algorithm followed n^2 behavior.

To collect a large set of data, the number of comparisons was collected while running the quicksort for many cases with list sizes ranging from 10 to 1,000,000. This data was then plotted and analyzed.

Above is the graph for the median–of-three quicksort tested on three different lists. 1. an in-order list, 2. a random list, and 3. the worst-case median of three. As you can see an in-order list does not show up in comparison to the worst case median-of-three, and the random list appears to be at zero. A trendline was added to the worst case for the median of three to get an approximation for the BigOh time. From this our estimate of worst case time was .

From this it was possible to plot on . This gave the following graph.

Since for large cases of n the graph flattens out, and doesn’t trend upwards or downwards, there is evidence to believe that this is BigOh .

As extra assessment, the left-pivot quicksort was compared to the median-of-three quicksort. On an in-order list, the median-of-three was much quicker than the one-pivot quicksort as expected.

On a random list, something interesting was found. The median of three quicksort was slightly slower than the one pivot quicksort, but only slightly. Although the graph appear linear in nature, when the same analysis is done as in the worst case, it is found that it is not quite linear as expected from Data Structures.

In the worst case for the median of three list, it was found that the one-pivot quicksort is slightly slower than the worst-case median of three. Looking further into this, the worst case median of three has a list that is partially in order, and so the one-pivot quicksort will not perform well on this. Also the median-of-three quicksort is always guaranteed to move one piece past the pivot meaning it loses a size of two every time, where the one-pivot quicksort does not.

Many of these comparisons were made on the timings with similar results.

In conclusion, the worst case median-of-three algorithm found is BigOh . Also it is found that on a random list the one-pivot quicksort is slightly faster than the median-of-three quicksort, presumably because of the additional comparisons to determine which key value should be used with the median of three method. Even though the median of three will run marginally slower on a random-ordered list, if there is any risk of having an in-order list, it is wise to use the median-of-three quicksort.