CSCI 4452.2 2015 Assignment #5

Show all your work.

(1) (a) We saw in class a proof that the worst case performance for QuickSort in terms of the number of comparisons is $O(N^2)$ for a list of size N. Explain the circumstances under which the worst case arises.

(b) The worst case is substantially worse than the average case for QuickSort - the average case is $O(N \log_2 N)$. For a list of size 10 million and assuming that each comparison takes one microsecond, compare the time required for the average case with that required for the worst case.

(c) Based on your answer to part (a), consider how likely it is that the worst case will arise? Develop an argument to explain how likely it is that the worst case will arise.

(d) The key idea in assessing worst case conditions involves counting the number of times a worst case partitioning arises. What is a worst case partitioning? For a list of size N, how many worst case partitioning scenarios are there? Experimentally count the number of times that a worst case partitioning arises vs. the number of times that a non-worst case partitioning arises. As well experimentally determine the average number of comparisons performed by QuickSort.

This will involve implementing QuickSort. **Use the specific algorithms for QuickSort and the Partitioning algorithm given in the class notes.** Try out your code on the detailed example given in class to verify that it works. For the experimental analysis, try out your code on a large collection of randomly generated lists of integers of size N. Summarize your results. How likely is a worst case partitioning going to arise compared to the number of times that a non-worst case partitioning arises? How do the experimentally determined comparisons counts compare with the the theoretical count for the average number of comparisons?

(2) Another way of looking at radix sort is to consider the key as just a bit pattern. So, if the keys are 4-byte integers, they are just considered as 32 bits, and if the keys are strings of 15 alphanumeric characters (15 bytes), they are just considered as 120 bits. These bit streams are then subdivided into pieces, which determine the number of passes and the number of buckets. So, if we have 120-bit keys, we might do 12 passes with 10-bit pieces, 10 passes with 12-bit pieces, or 5 passes with 24-bit pieces.

(a) If the key is a number in the range of 0 to 264, choose two options (one smaller and one larger) for the number of bits that will be used on each pass and indicate how many buckets and passes will be needed.

(b) If the key is a string of 40 characters, choose two options (one smaller and one larger) for the number of bits that will be used on each pass and indicate how many buckets and passes will be needed.

(c) Based on your answers to parts (a) and (b), can you give any general recommendations for how to make the choice of passes and key subdivisions?