Week 4 Preparation

(Solutions)

Useful advice: The following solutions pertain to the preparation problems. You are strongly advised to attempt the problems thoroughly before looking at these solutions. Simply reading the solutions without thinking about the problems will rob you of the practice required to be able to solve complicated problems on your own. You will perform poorly on the exam if you simply attempt to memorise solutions to these problems. Thinking about a problem, even if you do not solve it will greatly increase your understanding of the underlying concepts.

Problem 1. What are the worst-case time complexities of Quicksort assuming the following pivot choices:

(a) Select the first element of the sequence.

(b) Select the minimum element of the sequence.

(c) Select the median element of the sequence.

(d) Select an element that is greater than exactly 10% of the others.

Describe a family of inputs that cause Quicksort to exhibit its worst case behaviour for each of these pivot choices.

Solution

If we select the first element of the sequence, the worst-case complexity will be $\Theta(n^2)$. To see this, consider applying Quicksort to a list that is already sorted. In this case, the pivot will always have no elements to its left, and every other element to its right. This means that we recurse on lists of size $n-1$, $n-2$, ..., which will add up to $\Theta(n^2)$ time.

Selecting the minimum element of the sequence is even worse. Not only will this have worst-case complexity $\Theta(n^2)$, it actually has **best-case** complexity $\Theta(n^2)$ since we will always have every element on the right side of the pivot. This means that any sequence will cause the worst-case behaviour.

Selecting the median is the optimal choice. The median is the element that splits the sequence into two equal halves, hence we will only require $\Theta(\log(n))$ levels of recursion, and the worst-case complexity will be $\Theta(n\log(n))$. This behaviour will occur for any input sequence.

Selecting the 10^{th} percentile element sounds bad since we split the list into sublists of size 10% and 90% which is rather unbalanced, but this actually still has good performance. After recursing to a depth of k, the list will be of size at most $0.9^k n$, hence we hit the base case after

$$0.9^k n = 1.$$

Solving this, we find

$$k = \log_{\frac{10}{9}}(n) = \Theta(\log(n)).$$

Even though the base of the logarithm is worse, we still achieve $\Theta(n\log(n))$ performance since the subproblem sizes are decreasing by a constant factor each time. Indeed, replace 10% with any percent, even 1% and you will still achieve $\Theta(n\log(n))$ performance (although the constant factor will be rather large). This performance occurs for any input sequence.