Written Homework #4

Richard Dizon
CS 4102: Algorithms
University of Virginia

October 28th, 2016

1

There is a 50% chance that selecting a pivot point will lie in the middle 50th-percentile of the list for the first selection. For every subsequent selection, it's another 50% chance that it will be in the middle 50th-percentile of that subset but stacked with the first selection, it will be 50% of the probability of the previously incurred pivot selection, in this case, now 25% to get in the middle 50th-percentile twice in a row. This regressively approaches 0 at a large n given that the probability reduces by two at each level down of pivot selection.

If this behavior is guaranteed, the worst-case scenario would be that a pivot point would be selected such that it would split the list into 75% and 25% parts and will recursively continue to do so until the list is sorted. This worst-case behavior will result in a run-time of $O(n^2)$ reflective of that of the normal worst-case run time of the general quicksort algorithm despite choosing a pivot within the middle 50th-percentile because it doesn't quite reduce to half which would subsequently get it down to the normal average time at $O(nlog(n))$.

2

When the gun is fired, kills are calculated by the following:

$$\begin{cases} C[i] & C[i] < Z[i] \\ Z[i] & C[i] \geq Z[i] \end{cases}$$

Since we have all the data ahead of time we can improve from the brute force $O(2^n)$ run-time solution where we check each situation from every minute by having all combinations of firing and not firing. An optimization would be to cache some values to determine when the optimal firing time would be considering all possible C(i) values and the kills that result from them.

3

A line is visible if an only if it dominates all other lines for some given x-value. For us to compare each line to each other from a given x range of length m, this would result in a run time of $O(m * n)$ where we try to find the max L value within a set of n lines at each x value. We can optimize this by playing with the second clarification that no three lines in the input meet at any one shared point. For some range of x-values where a line is dominant, the extremities of this range indicate where one dominant line ends and one dominant line begins. An intersection is considered when two lines have the same L-value for some a and b at the same x.

This switch only happens when the magnitude of the slope a decreases on the lower end of the range of x or increases on the higher end of the range of x so we'd only have to check the subset of lines where the value of a fulfills either of those requirements which results in a $log(n)$ for the range m resulting in a $O(m * log(n))$ run-time.