MATH 282 Analysis of Algorithm’s Complexity

**Algorithm:** Binary search

**Factor to be analyzed:** Time (number of steps, speed)

**Situation to be analyzed:** Worst case

**Explanation of situation:** Item is not in the list (or is the last item that we look at)

**Key step (reflects work done):** Comparisons

**Parameter for analysis:** Size of list (number of data items in the list), *n*

*(what determines how the algorithm’s efficiency changes as the size of data increases?)*

**Questions/Process:**

* Are there any steps which are not simple steps (comparable to the key step)?
  + If so, what is the complexity of those steps (relative to the key step)? Factor into the analysis.
* If desired, count the number of times each step is carried out (or just the key step).
* Identify each loop and determine how many times the loop is carried out (in relation to *n*).
* How are the loops related?
  + If nested, multiply the steps.
  + If separate, add the steps.
* Eliminate any constants and any lower-level terms.

Focus on comparisons: (2C + 1C + 1C) \* *loops* + 1C

How many loops? Because the loop is cutting the data in half each time, that takes log2*n* iterations

4C \* log2*n* + 1C comparisons are being done, so since we can ignore the constants 4 and 1, the order of the algorithm is O(log *n*)

Alternative analysis: There is one loop. Every time that we go through that loop, we cut the number of items in half. Thus, the loop is done **log(*n*)** times. (More precisely, the logarithm is base-2, and the number of steps is **ceiling(log2*n* + 1)**, but this doesn’t affect the order of magnitude of the calculation, so it can be ignored.)

**Result:** Algorithm is O( log2*n* )