# Unit 7: Searching and Sorting

## Searching

The two main methods for searching an array or list of elements are sequential search and binary search.

### Sequential Search

The method for sequential search works on any list, even if the list is unsorted. It is to simply look at each element, either from the front or from the back, and return if you find what you are looking for.

Pseudocode:

set index to 0

while index is less than list.length

if element at index equals target

return index

increment index

return -1 (sentinel value to say target is not found)

The sequential search algorithm’s best-case scenario is that the first item is target, at which point the algorithm is done after taking only 1 step.

The worst-case scenario is that the target is not in the list, at which point we had to look at every single element, taking N steps.

If the target is in the list, then it is equally likely to be in the first half as the second half, so it will take on average N/2 steps.

### Binary Search

The algorithm for binary search only works on sorted lists. It cuts the list into successive halves, each half getting more precise, honing in on the target value. After n steps, the list will have been cut in half n times, meaning binary search requires significantly fewer comparisons than sequential search.

For example, after 10 steps, binary search on a list of 1,000,000 elements will have cut the list down to fewer than 1000 elements. After 20 steps, binary search would be done. Contrasted with sequential search, the sequential search would have 999,979 more elements to look at.

The best case scenario for binary search is that the first guess is correct (middle element) taking only 1 step.

The average and worst-case scenarios take roughly log2(N) steps.

|  |  |
| --- | --- |
| N | ~log2(N) |
| 102=100 | 7 |
| 103=1000 | 10 |
| 104=10000 | 14 |
| 105=100000 | 17 |
| 106=1000000 | 20 |
| 107=10000000 | 24 |

Pseudocode:

set low to 0

set high to list.length - 1

set middle to list.length/2