11.9.2 Big O of the Binary Search

- Worst-case scenario: Searching a sorted array of 1023 elements takes only 10 comparisons with binary search
- Repeatedly dividing 1023 by 2 (because after each comparison we can eliminate half the array) and rounding down (because we also remove the middle element) yields 511, 255, 127, 63, 31, 15, 7, 3, 1 and 0
- 1023 (which is $2^{10} – 1$) is divided by 2 only 10 times to get the value 0, which indicates that there are no more elements to test
- Dividing by 2 is equivalent to one comparison in the binary search algorithm
- An array of 1,048,575 ($2^{20} – 1$) elements takes a maximum of 20 comparisons

11.9.2 Big O of the Binary Search

- An array of about one billion elements takes a maximum of 30 comparisons
- Tremendous improvement over linear search
  - For a one-billion-element array, this is a difference between an average of 500 million comparisons for the linear search and a maximum of only 30 comparisons for the binary search!
- Maximum number of comparisons needed for the binary search of any sorted array is the exponent of the first power of 2 greater than the number of elements in the array, which is represented as $\log_2 n$
- From a Big O perspective, all logarithms grow at roughly the same rate, so in Big O, the base can be omitted
- O(log ***n***) for a binary search, which is also known as logarithmic run time and pronounced as “order log n.”
- Assumes the array is sorted, which could take time
