1. 1. 37
   2. 2 / N,
   3. Sqrt N
   4. N
   5. N log log N ,
   6. N log N,
   7. N log (N2),
   8. N1.5
   9. N2
   10. N2Log(N)
   11. N3
   12. 2n/2
   13. 2n
   14. Big O- T(w) = O(f(n)) if there are positive constants C and N0 such that T(n) <= cf(n)
   15. Big Ω - T(n) = Ω(g(b)) if there are positive constants C and N0 such that T(N) >= cg(n) when N >= n0
   16. Big Theta – T(n) = Theta(h(n)) if and only if T(N) = O(h(N)) and T(N) = Ω(h(N))
   17. Little o – T(w) = o(p(n)) if there are positive constants c and n0 such that T(n) < cp(n) when n > n0
2. Big O notation represents the upper bound while Big Theta means the tight bound.
   1. True
   2. True
   3. True
   4. True
   5. O(xn)
   6. O(log n)
   7. The growth of a binary search on a sorted array would be O(log n)
   8. A binary search works by always splitting the data in half at each pass, in every instance the next size to handle will be n/2 smaller than the past size. This leads to the analysis that the growth of the algorithm is log n, very small. A sequential search on the other hand does not cut down the data at any point, and is at the mercy of the size of the data set. As a sequential search goes from x0 to xn-1 it’s growth rate is *n*
   9. The running time of inserting a node inside of a linked list is of O(1), or a constant time regardless of the size of the data set. This is because inserting a node inside a linked list is not dependent of it’s size, it is only required that pointers are to be reorganized such that the new data is added into the list.
   10. O(n)
   11. O(n2)
   12. O(n3)
   13. O(n5)
   14. O(n4)