1. O(n^2)
2. O(n^4)
3. O(n^2)
4. O(n^3)
5. O(1)
6. O(1)
7. O(n)
8. O(n)
9. O(nlog(n))
10. O(n^3)
11. O(1)
12. O(n)
13. O(n) is the time complexity used when removing an item from the middle of an ArrayList, this allows for one to adjust to the shifting of n-1 after a number is removed.
14. O(1)
15. O(n) if you must reallocate. O (1) if you don’t have to reallocate.
16. An arraylist would come in handy in situations where not many items are being added or if one knows how many items will be added.
17. The Rudrata path problem involves finding a path through a graph that visits each vertex exactly once. This path corresponds to the ‘Map & Maze puzzle’ since every point is visited once without repetition until a path is devised that allows one to exit.
18. If the number of starts is defined by n, then you would use (n – 1)! To find how many paths an algorithm would check to find the solution.
19. O((n+1)!)
20. Because Bogosort’s worst-case runtime accounts that it just keeps randomizing the order of the list so it’s not guaranteed that it will stop randomizing and become sorted
21. O(n!)
22. Because there are no more or no less than 52 cards in a deck given this one could see the outcome is 52!

Unique: O(n)

A computer screen with text

Description automatically generated

All Multiples: O(n)A computer screen shot of a code

Description automatically generated

All Strings of Size: O(n)A computer code with text

Description automatically generated

Is Permutation: O(n^2)A screenshot of a computer program

Description automatically generated

List of words: O(n^2 ) 🡨 ?A computer screen shot of a computer code

Description automatically generated

Remove all instances: O(n)A screenshot of a computer code

Description automatically generated