2. O(log n): This class represents algorithms whose running time grows logarithmically with the input size. Examples of such algorithms include binary search or finding the height of a balanced binary tree.
3. O(n): This class represents algorithms whose running time grows linearly with the input size. Examples of such algorithms include iterating through an array or performing a single pass over a linked list.
4. O(n log n): This class represents algorithms whose running time grows faster than linear but slower than quadratic, with the precise growth rate determined by the base of the logarithm. Examples of such algorithms include quicksort or mergesort.
5. O(n^c): This class represents algorithms whose running time grows polynomially with the input size, where c is a constant greater than 1. Examples of such algorithms include matrix multiplication or polynomial evaluation.
6. O(c^n): This class represents algorithms whose running time grows exponentially with the input size, where c is a constant greater than 1. Examples of such algorithms include exhaustive search or the traveling salesman problem.
7. O(n!): This class represents algorithms whose running time grows factorially with the input size. Examples of such algorithms include brute-force permutation search or the traveling salesman problem with arbitrary distances.

It is important to note that these classes represent upper bounds on the growth rate of an algorithm's running time, and that the actual running time of an algorithm may be lower than its upper bound for certain inputs. However, these classes provide a useful framework for comparing the relative efficiency of different algorithms and selecting the most appropriate algorithm for a given problem.