When analyzing algorithms, it is common to consider the worst-case, best-case, and average-case scenarios. Let’s explore each of these types of analysis:

1. Worst-Case Analysis:

- The worst-case analysis determines the upper bound on the running time or space requirements of an algorithm for the input that results in the maximum execution time or space usage.
- It assumes that the input is chosen in such a way that it leads to the most inefficient behavior of the algorithm.
- It provides a guarantee on the performance of the algorithm, ensuring that it will not exceed a certain time complexity or space complexity for any input of size ‘n’.
- The worst-case analysis helps identify scenarios in which the algorithm may perform poorly and assists in assessing its efficiency in the worst possible scenario.

2. Best-Case Analysis:

- The best-case analysis determines the lower bound on the running time or space requirements of an algorithm for the input that results in the minimum execution time or space usage.
- It assumes that the input is chosen in such a way that it leads to the most efficient behavior of the algorithm.
- The best-case analysis is often used to showcase the algorithm’s potential when the input is already in an ideal or sorted state.
- However, the best-case scenario is not always realistic or representative of the algorithm’s typical performance.

3. Average-Case Analysis:

- The average-case analysis considers the expected or average running time or space requirements of an algorithm for all possible inputs of size ‘n’.
- It takes into account the likelihood of different inputs occurring and assigns probabilities to them.
- It requires a probability distribution of the inputs and calculates the average running time or space usage by considering the weighted sum of the running times or space usages for different inputs.
- The average-case analysis is generally more informative and practical than the worst-case analysis alone, as it provides insight into the expected behavior of the algorithm under typical conditions.

Difference between Worst-case, best-case, and average-case analysis:

	Worst-Case Analysis	Best-Case Analysis	Average-Case Analysis
Purpose	Determines the upper bound on running time or space requirements.	Determines the lower bound on running time or space requirements.	Considers the expected or average performance under typical conditions.
Input	Input that leads to the most inefficient behavior.	Input that leads to the most efficient behavior.	Probability distribution of inputs.
Assumption	Input is chosen to maximize algorithm inefficiency.	Input is chosen to minimize algorithm inefficiency.	Considers a distribution of inputs based on probability.

	Worst-Case Analysis	Best-Case Analysis	Average-Case Analysis
Guarantees	Provides a guarantee on algorithm performance.	Not a guarantee but showcases algorithm potential.	Provides insight into the expected behavior under typical conditions.
Practicality	Helps identify scenarios where the algorithm may perform poorly.	Not always realistic or representative of actual performance.	More informative and practical, reflects real-world scenarios.
Decision-Making	Helps in assessing algorithm efficiency and worst-case behavior.	Useful in understanding algorithm capabilities in ideal scenarios.	Helps in comparing algorithms' performance under typical conditions.
Performance Range	Upper bound on time complexity or space requirements.	Lower bound on time complexity or space requirements.	Average running time or space usage based on probability distribution.