**Product Search**

Big O notation is a mathematical notation that describes the complexity of an algorithm, which is the amount of time or space it requires as the input size grows. It's used to classify algorithms based on their performance, helping developers understand how efficient they are. Big O notation gives an upper bound on the number of steps an algorithm takes to complete, relative to the size of the input. It's usually expressed as a function of the input size, typically represented as 'n'.

- O(1) - constant time complexity (the algorithm takes the same amount of time regardless of input size)

- O(log n) - logarithmic time complexity (the algorithm takes time proportional to the logarithm of the input size)

- O(n) - linear time complexity (the algorithm takes time proportional to the input size)

- O(n log n) - linearithmic time complexity (the algorithm takes time proportional to the product of the input size and its logarithm)

- O(n^2) - quadratic time complexity (the algorithm takes time proportional to the square of the input size)

- O(2^n) - exponential time complexity (the algorithm takes time proportional to 2 raised to the power of the input size)

**Best-Case Scenario:**

The missing person or item is quickly and easily found, with minimal risk or harm to searchers. The operation is well-coordinated and efficient, with a successful outcome.

**Average-Case Scenario:**

The search operation takes longer and requires more resources, but the missing person or item is eventually located. Some challenges and risks are encountered, but searchers are able to overcome them.

**Worst-Case Scenario:**

The search operation is prolonged and extensive, with significant risks and resources involved, but the missing person or item is not found or is located in a dire condition. Poor coordination and communication lead to a failed or complicated outcome.