Secure Data Fragment Allocation Scenario

Important Note: While the scenario and task descriptions below use Python syntax for illustration, **please solve this challenge in the language specified by the recruiter or as per the requirements of the coding test**.

Scenario: In a dystopian cyberpunk future, data breaches are a pervasive threat. To counter this, an innovative data protection strategy has been devised: sensitive information is broken down into fragments and distributed among various data centers, each with its own inherent risk factor. The cumulative risk for storing a set of data fragments is determined by the product of the individual risks at each data center. The challenge lies in allocating these data fragments in such a manner that **the highest risk across all sets of fragments is as low as possible, thus minimizing potential data compromise**.

Risk Calculation Clarification: The risk associated with storing data fragments in a data center is calculated as the base risk of the data center raised to the power of the number of fragments stored in that center. For example, if a data center has a base risk factor of 2 and is holding 3 fragments, the total risk for that center would be $2^3 = 8$.

Task: Your objective is to develop an algorithm that efficiently distributes a specified number of data fragments across a designated number of data centers, ensuring that the maximum risk associated with any set of fragments is minimized. This algorithm must consider the unique risk factor of each data center during the distribution process.

Input:

- An array or list of integers, where each integer represents the risk factor associated with a data center.
- An integer that denotes the total number of data fragments that need to be distributed among the data centers.

Expected Output:

- An integer that represents the minimized maximum risk achievable through optimal distribution of the data fragments across the data centers.

Constraints:

- Please solve this challenge in the specified programming language. Avoid using external libraries that would obscure the core logic of your solution.

- Ensure that your solution is efficient and can handle a large number of data centers and data fragments.

Example:

```
# your implementation of the distribute_fragments function
data_centers = [10, 20, 30]
fragments = 5

min_risk = distribute_fragments(data_centers, fragments)
print(f"Minimized maximum risk: {min_risk}")
```

Evaluation Criteria:

- **Algorithmic Efficiency**: Your solution should employ an efficient algorithm that can scale with the number of data centers and fragments.
- **Correctness**: The output of your algorithm must accurately reflect the minimized maximum risk based on the given input.
- **Code Quality**: Your code should be well-organized, commented, and easily understandable, adhering to best practices.
- **Edge Case Handling**: Your solution should be robust, handling edge cases gracefully (e.g., large numbers of fragments, high-risk factors, etc.).