**Prove Prim’s Algorithm using the Loop Invariant technique.**

The loop invariant states that at the start of each iteration, the current set of vertices forms a valid partial minimum spanning tree. This guarantees termination and the production of a minimum spanning tree.

Initially, the empty set serves as a valid partial minimum spanning tree.

During each iteration, the algorithm selects the minimum weight edge connecting a vertex in the set to a vertex outside. Adding this edge creates a valid partial minimum spanning tree, as ensured by the loop invariant.

By consistently choosing the minimum weight edge, the algorithm guarantees the validity of the new subtree. Upon termination, all vertices are included in the minimum spanning tree, satisfying the loop invariant.

The loop invariant technique proves the correctness of Prim's algorithm by maintaining a valid partial minimum spanning tree throughout each iteration. The resulting set of vertices represents the minimum spanning tree for the graph when the algorithm finishes.

**Example A:**

**Time complexity: O(n^2)**

The time complexity is O(n^2) due to the presence of nested loops where each loop iterates over the length of the list "n," resulting in a quadratic relationship between the input size and the number of operations performed.

**Mystery2 block Description:**

The Mystery2 block performs a calculation that involves multiplying corresponding elements of a given list "n" and accumulating the sum of all these products into a variable called "s."

**Example B:**

**Time Complexity: O(logn)**

The Mystery4 block utilizes a binary search algorithm, resulting in a time complexity of O(logn) due to the efficient halving of the search range with each iteration.

**Mystery4 block Description:**

The Mystery4 block performs a binary search algorithm to find the index of a target element within a given list. It utilizes a logarithmic time complexity for efficient searching.