### Algorithm: Prim's Algorithm

In Algoria, Sir Cedric and Ember needed to rebuild the Bridge of Light to cross the Abyss of Shadows and reach Princess Elara. The bridge required connecting several magical islands in the most efficient way.

#### Initialize Data Structures:

* Sir Cedric used a bag of enchanted ropes (priority queue) to connect the islands.
* He kept track of visited islands (set) to ensure no island was left unconnected.

#### Build the Bridge:

* Sir Cedric started from the first island, using the ropes to connect to the nearest unvisited island.

#### Repeat:

* He continued connecting islands until all were part of the bridge.

#### Implementation:

| **import** **heapq**  **def** prim(islands: Dict[int, List[Tuple[int, int]]], start: int) -> List[Tuple[int, int, int]]:  priority\_queue = [(0, start, -1)]  visited = set()  bridge = []  **while** priority\_queue:  weight, island, from\_island = heapq.heappop(priority\_queue)  **if** island **not** **in** visited:  visited.add(island)  **if** from\_island != -1:  bridge.append((from\_island, island, weight))  **for** next\_island, rope\_length **in** islands[island]:  **if** next\_island **not** **in** visited:  heapq.heappush(priority\_queue, (rope\_length, next\_island, island))  **return** bridge  *# Example usage:*  islands = {  1: [(2, 1), (3, 4)],  2: [(1, 1), (3, 2), (4, 5)],  3: [(1, 4), (2, 2), (4, 1)],  4: [(2, 5), (3, 1)]  }  start = 1  print(prim(islands, start)) *# Output: Minimum Spanning Tree (Bridge) edges* |
| --- |

#### Explanation:

Initialize:

* priority\_queue: A bag of enchanted ropes to connect the islands.
* visited: A set to track visited islands.

Build the Bridge:

* Sir Cedric connected the nearest unvisited island, ensuring the bridge was efficient.

Repeat:

### He continued until all islands were connected.