### Algorithm: Bellman-Ford Algorithm

Sir Cedric and Ember flew to the Land of Sorrows, where Princess Elara was held captive by an evil sorcerer. The land was cursed, with negative energy draining travelers. To save the princess, Sir Cedric needed to find the shortest path through the treacherous terrain.

#### Initialize Data Structures:

* Sir Cedric used a mystical compass (list) to keep track of the shortest path to each location.
* He recorded the previous location visited on his map (list).

#### Traverse the Land of Sorrows:

* Sir Cedric started at the entrance, marking it with a distance of 0.
* He moved through each location, updating the shortest path.

#### Repeat:

* He continued until he had visited all locations or detected a negative cycle.

#### Implementation:

| **def** bellman\_ford(land: List[Tuple[int, int, int]], V: int, start: int) -> List[int]:  dist = [float('inf')] \* V  dist[start] = 0  **for** \_ **in** range(V - 1):  **for** u, v, w **in** land:  **if** dist[u] != float('inf') **and** dist[u] + w < dist[v]:  dist[v] = dist[u] + w  **for** u, v, w **in** land:  **if** dist[u] != float('inf') **and** dist[u] + w < dist[v]:  print("Land contains a negative energy cycle")  **return** []  **return** dist  *# Example usage:*  land = [(0, 1, -1), (0, 2, 4), (1, 2, 3), (1, 3, 2), (1, 4, 2), (3, 2, 5), (3, 1, 1), (4, 3, -3)]  V = 5  start = 0  print(bellman\_ford(land, V, start)) *# Output: Shortest path distances* |
| --- |

#### Explanation:

Initialize:

* dist: A mystical compass to track the shortest path to each location.
* dist[start]: Marked with a distance of 0 at the start.

Traverse the Land of Sorrows:

* Sir Cedric moved through each location, updating the shortest path.

Repeat:

* He continued until all locations were visited or a negative energy cycle was detected.