### Algorithm: A\* Search Algorithm

Sir Cedric and Ember faced the Inferno Plains, where they needed to find the shortest path to Princess Elara using the A\* Search Algorithm.

#### Initialize Data Structures:

* Sir Cedric used a magical compass (priority queue) to track the paths with the least fire.

#### Calculate Heuristic:

* Sir Cedric used a map (function) to estimate the distance to Princess Elara.

#### Find the Path:

* He moved through the plains, always choosing the path with the least estimated cost.

#### Retrieve the Result:

* The compass showed the shortest path to Princess Elara.

#### Implementation:

| **import** **heapq**  **def** a\_star(plains: Dict[int, List[Tuple[int, int]]], start: int, goal: int) -> List[int]:  **def** heuristic(a, b):  **return** abs(a - b) *# Example heuristic function (Manhattan distance)*  priority\_queue = [(0, start)]  came\_from = {}  cost\_so\_far = {start: 0}  **while** priority\_queue:  current\_cost, current = heapq.heappop(priority\_queue)  **if** current == goal:  **break**  **for** next\_node, weight **in** plains[current]:  new\_cost = cost\_so\_far[current] + weight  **if** next\_node **not** **in** cost\_so\_far **or** new\_cost < cost\_so\_far[next\_node]:  cost\_so\_far[next\_node] = new\_cost  priority = new\_cost + heuristic(next\_node, goal)  heapq.heappush(priority\_queue, (priority, next\_node))  came\_from[next\_node] = current  path = []  node = goal  **while** node != start:  path.append(node)  node = came\_from[node]  path.append(start)  path.reverse()  **return** path  *# Example usage:*  plains = {  1: [(2, 1), (3, 4)],  2: [(3, 2), (4, 5)],  3: [(4, 1)],  4: []  }  start, goal = 1, 4  print(a\_star(plains, start, goal)) *# Output: Shortest path from start to goal* |
| --- |

#### Explanation:

Initialize:

* priority\_queue: A magical compass to track the paths with the least fire.

Calculate Heuristic:

* heuristic: A map to estimate the distance to Princess Elara.

Find the Path:

* Sir Cedric moved through the plains, always choosing the path with the least estimated cost.

Retrieve the Result:

* The compass showed the shortest path to Princess Elara.