PROGRAM:

```
class Problem:
    def __init__(self, nodes, heuristic_values, goal_state):
        self.nodes = nodes
        self.heuristic_values = heuristic_values
        self.goal_state = goal_state
        self.node_to_neighbors = {node: [] for node in nodes}
    def add_edge(self, node1, node2):
        if node1 in self.nodes and node2 in self.nodes:
            self.node_to_neighbors[node1].append(node2)
            self.node_to_neighbors[node2].append(node1)
    def initial_state(self):
        return self.nodes[0]
   def get_neighbors(self, state):
        return self.node_to_neighbors[state]
  def evaluate(self, state):
        return self.heuristic_values[state]
def hill_climbing(problem):
    current = problem.initial_state()
    path = [current]
    while True:
        if current == problem.goal_state:
            return True, path
        neighbors = problem.get_neighbors(current)
        if not neighbors:
            break
        next_state = max(neighbors, key=problem.evaluate)
        if problem.evaluate(next_state) <= problem.evaluate(current):
            break
        current = next_state
```

```
        path.append(current)
    return current == problem.goal_state, path
nodes = ['A', 'B', 'C', 'D', 'E']
heuristic_values = {
    'A': 2,
    'B': 4,
    'C': 5,
    'D': 6,
    'E': 7
}
goal_state = 'E'
problem = Problem(nodes, heuristic_values, goal_state)
problem.add_edge('A', 'B')
problem.add_edge('B', 'C')
problem.add_edge('C', 'D')
problem.add_edge('D', 'E')
reachable, path = hill_climbing(problem)
if reachable:
    print(f"The goal state '{goal_state}' is reachable.")
else:
    print(f"The goal state '{goal_state}' is not reachable.")
print(f"Path taken: {' -> '.join(path)}")
```

OUTPUT:

```
[Running] python -u "d:\CSE-TCE\05SEM\22CS580\Graph.py"
The goal state 'E' is reachable.
Path taken: A -> B -> C -> D -> E
```