Program:

```
from queue import PriorityQueue
def ucs(start, goal, graph): queue = PriorityQueue() queue.put((0,
  start)) visited = set()
while not queue.empty():
  cost, node = queue.get()
  if node == goal:
      return cost
    if node not in visited:
      visited.add(node)
      for neighbor, weight in graph[node].items():
          if neighbor not in visited:
            queue.put((cost + weight, neighbor))
  return -1
graph = {}
noOfVertices = int(input("Enter the number of vertices: "))
for i in range(noOfVertices):
  vertex = input(f"\nEnter vertex {i+1}: ")
  numOfNeighbors = int(input("Enter the number of neighbors: "))
  neighbors = {}
  for j in range(numOfNeighbors):
    neighbor = input(f"Enter neighbor {j+1}: ")
    weight = int(input(f"Enter the weight of edge ({vertex},
    {neighbor}): "))
    neighbors[neighbor] = weight
  graph[vertex] = neighbors
start = input("\nEnter the start node: ")
goal = input("Enter the goal node: ")
cost = ucs(start, goal, graph)
if cost == -1:
  print("\nPath not found.")
else:
  print(f"\nThe minimum cost from {start} to {goal} = {cost}")
```

Output:

```
Enter the number of vertices: 6

Enter vertex 1: A
Enter the number of neighbors: 2
Enter neighbor 1: B
Enter the weight of edge (A, B): 5
Enter neighbor 2: C
Enter the weight of edge (A, C): 1

Enter vertex 2: B
Enter the number of neighbors: 2
Enter neighbor 1: C
Enter the weight of edge (B, C): 2
Enter neighbor 2: D
Enter the weight of edge (B, D): 1

Enter vertex 3: C
Enter the number of neighbors: 2
Enter neighbor 1: B
Enter the weight of edge (C, B): 2
Enter neighbor 2: E
Enter the weight of edge (C, E): 8

Enter vertex 4: D
Enter the number of neighbors: 3
Enter neighbor 1: F
Enter the weight of edge (D, F): 6
Enter neighbor 2: B
Enter the weight of edge (D, B): 1
Enter neighbor 3: E
Enter the weight of edge (D, E): 3

Enter vertex 5: F
Enter the number of neighbors: 0
```

```
Enter vertex 6: E
Enter the number of neighbors: 2
Enter neighbor 1: C
Enter the weight of edge (E, C): 8
Enter neighbor 2: D
Enter the weight of edge (E, D): 3

Enter the start node: A
Enter the goal node: F

The minimum cost from A to F = 10
```