A white paper with black text

Description automatically generated

1.<https://github.com/OnetoTommy/Program-Structure-Algorithms-/tree/main/Course/Course_10/PA_1>  
  
**2. Describe Algorithm (Dijkstra's Algorithm)**

1. **Graph Representation:**

The graph is stored as an adjacency list using a dictionary where keys are nodes, and values are lists of tuples representing neighbors and edge weights.

1. **Function dijkstra\_shortest\_cycle(start):**

* This function uses **Dijkstra’s Algorithm** to find the shortest cycle starting and ending at start.
* It initializes a shortest\_cycle dictionary where all nodes have infinite distance except the start node (set to 0).
* A **priority queue (min-heap)** is used to always process the node with the smallest distance.
* For each node, its neighbors are checked, and distances are updated if a shorter path is found.
* **If a neighbor is the start node and it's not a direct back edge**, a cycle is detected, and its length is returned.

1. **Function finding\_shortest\_cycle():**

* This function calls dijkstra\_shortest\_cycle(start) for every node in the graph.
* It keeps track of the shortest cycle found.
* If no cycle exists, it returns 0.

1. **Reading Input Graph from File:**

* The input file contains graph edges and each row represents a source vertex followed by pairs of (neighbor, weight).
* This is parsed into the adjacency list representation.

1. **Final Output:**

The algorithm prints the length of the shortest cycle found like “**The length of the shortest cycle is: 8**”

**3. Testcase(s) on which you validated your program.**

**Case-1: Acyclic**

**0 : 1 1**

**1 : 2 2**

**2 : 3 3**

**Case-2: Single Cycle**

**0 : 1 1 2 4**

**1 : 3 2**

**2 : 3 1**

**3 : 0 3**

**Case-3: Double Cycle**

**0 : 1 1**

**1 : 2 2**

**2 : 3 3**

**3 : 1 4 0 5**

**Case-4: Empty**

Link: https://github.com/OnetoTommy/Program-Structure-Algorithms-/tree/main/Course/Course\_10/PA\_1