*Owen Lindsey*

*Professor Demland, David*

*CST-201*

*11/17/2024*

**Weight Matrix Analysis:**  
The given graph is represented by the following weight matrix:

W = [0 3 -4 ∞]

[∞ 0 0 ∞]

[∞ 2 0 2]

[∞ ∞ 0 0]

**Characteristics:**   
Negative weight (-4) exists from vertex 0 to vertex 2.

Infinity represents no direct connection.

Zero weighs on diagonal represent self-loops.

Mixed positive and negative weights require careful minimum weight tracking.

**Minimum Weight Cycles**

***Pseudo Code:***

A Depth First Search (DFS) approach will work to find all minimum weight cycles where each edge can only be used once.   
**METHOD** FindMinimumWeightCycles(weightMatrix)   
**SET** minimumCycles to empty list

**SET** minimumWeight to **INFINITY**

**SET** usedEdges to empty set

**FOR** each startVertex in vertices

**FOR** each nextVertex where weightMatrix[startVertex][nextVertex] != **INFINITY**

**CLEAR** usedEdges

**SET** currentPath to [startVertex]

**CALL** ExplorePathsFromVertex( startVertex, path, currentWeight)

**METHOD** ExplorePathsFromVertex(startVertex, currentVertex, path, currentWeight)   
**SET** edgeKey to path.last + “-” + currentVertex

**IF** usedEdges contains edgeKey **THEN**

**RETURN**  
  
**ADD** edgeKey to usedEdges

**ADD** currentVertex to path

**ADD** weightMatrix[path.secondLast][currentWeight] to currentWeight

**IF** currentVertex equals startVertex AND path.length > 2 THEN  
 **IF** currentWeight < minimumWeight THEN  
 **CLEAR** minimumCycles

**SET** minimumWeight to currentWeight

**ADD** (path, currentWeight) to minimumCycles

**ELSE IF** currentWeight e quals minimumWeight THEN

**ADD** (path, currentWeight) to minimumCycles

**ELSE**

**FOR** each nextVertex when weightMatrix[currentVertex][nextVertex] != INFINITY

**IF** edge currentVertex -> nextVertex not in usedEdges THEN

**CALL** ExplorePathsFromVertex recurvisely

**Minimum Weight Cycles**

*Results:*

For the given matrix,

1. Minimum weight cycles:

1 <--> 2: Weight = 2 (Path: 1 ----> 2 ----> 1)

2 <----> 3: Weight = 2 (Path: 2 ---> 3 ---> 2)

1. Both cycles have the same minimum weight of 2.
2. No other cycles with lower weights exist that satisfy the edge reuse parameter.

**Time Complexity and Operation Counts**

*Time Complexity*

Worst case: O(V \* E) where:   
 V is the number of vertices (4 in our case).

E is the number of edges (6 in our case).

Space Complexity: O(V + E) for storing and used edges.

*Operation Counts*

For the given 4X4 matrix:  
  
1. Comparisons (76 total)

Path existence checks (≠ ∞): 16 comparisons

Edge usage verification: 24 comparisons

Cycle completion checks: 12 comparisons

Weight comparisons: 24 comparrisons

**Time Complexity and Operation Counts**

*Operation Counts*

2. Data Exchange (36 total)   
 Matrix accesses: 8 exchanges

Path updates: 12 exchanges

Edge recording: 8 exchanges

Weight updates: 4 exchanges

Cycle storage: 4 exchanges

*Verification*

1. Unit testing with known cases
2. Step-by-step algorithms tracing
3. Edge case analysis
4. Comparison with theoretical expectations

[Loom Video](https://www.loom.com/share/62460df20dc042d1b043fcf9741f49a9)