# Kruskal’s Algorithm and Cycle Checking

## Introduction

Kruskal's algorithm is a popular method used to find a Minimum Spanning Tree (MST) for a connected, weighted graph. The algorithm operates by sorting all edges in the graph in order of increasing weight and then adding them to the MST one by one, provided they don't form a cycle.

## Circuit (Cycle) Checking in Kruskal's Algorithm

To ensure that adding a new edge doesn't form a cycle, Kruskal's algorithm uses a data structure known as a "Union-Find" or "Disjoint Set". This data structure allows us to efficiently manage a partition of a set into disjoint subsets. It supports two primary operations:

1. Find: Determine which subset a particular element is in. This helps in determining if two elements are in the same subset.

2. Union: Join two subsets into a single subset.

These operations are used to check and ensure that adding a new edge does not form a cycle.

## Example

Consider a simple graph with four vertices {A, B, C, D} and the following edges:

- A-B with weight 1  
- B-C with weight 3  
- C-D with weight 4  
- A-D with weight 2  
- B-D with weight 5

Steps to apply Kruskal’s Algorithm:

1. Sort the edges by weight:  
 - A-B (1), A-D (2), B-C (3), C-D (4), B-D (5)

2. Apply Kruskal’s Algorithm:

- A-B: No common subset, include it in MST. Union(A, B).  
- A-D: No common subset, include it in MST. Union(A, D) which also merges B into the same subset due to previous union.  
- B-C: No common subset, include it in MST. Union(B, C) merges all nodes into a single subset.  
- C-D: Check finds that C and D are already connected through previous unions, including this edge would form a cycle, so skip it.  
- B-D: Already connected, skip it.

Thus, the MST includes the edges A-B, A-D, and B-C with a total weight of 1 + 2 + 3 = 6.

# A/Definition and Conditions of Eulerian Circuit and Eulerian Path

## Eulerian Circuit

An Eulerian circuit is a circuit that traverses each edge of the graph exactly once, and a graph is called an Eulerian graph if it contains an Eulerian circuit.

## Eulerian Path

An Eulerian path is a path that traverses each edge of the graph exactly once.

## Conditions

For the given illustrated graph, it is an undirected graph, so the appropriate conditions are:

### Eulerian Circuit

A graph has an Eulerian circuit if and only if:  
- The vertices with non-zero degree are connected.  
- All vertices in the graph have even degrees.

### Eulerian Path

A graph has an Eulerian path if and only if:  
- The vertices with non-zero degree are connected.  
- The graph has exactly 0 or 2 vertices with odd degrees.

## Checking the Degrees of Vertices

Vertex A: Connected to E, F, I, J, K, L. Degree of A is 6.

Vertex B: Connected to E, G, M, N, S, Y. Degree of B is 6.

Vertex C: Connected to F, H, R, X, d, j. Degree of C is 6.

Vertex D: Connected to G, H, n, i, k, m. Degree of D is 6.

Vertex E: Connected to A, B, I, M. Degree of E is 4.

Vertex F: Connected to A, C, K, R. Degree of F is 4.

Vertex G: Connected to B, D, S, k. Degree of G is 4.

Vertex H: Connected to C, D, X, i. Degree of H is 4.

For the inner vertices like I, J, K, L, R, M, N, O, P, Q, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, k, l, m, n: Each of these vertices is connected to 4 other vertices, so their degrees are 4.

## Summary

Vertices with odd degrees: There are no vertices with odd degrees; all vertices have even degrees.

## Final Conclusion

Since all vertices in the graph have even degrees and the graph is connected, this graph has an Eulerian circuit.

# Eulerian Circuit Using Hierholzer’s Algorithm

**B/Hierholzer's Algorithm to Find an Eulerian Circuit**

**Introduction**

The Hierholzer algorithm is a classic method for finding Euler circuits in graphs. This algorithm is efficient and works with both directed and undirected graphs, as long as they satisfy the necessary conditions for an Euler cycle, which is that the graph needs to be connected and have vertices of even degree

.**Steps of Hierholzer's Algorithm**

1. **Initialization**:
   * Start with an empty circuit and an arbitrary vertex 𝑣*v*.
2. **Constructing the Circuit**:
   * Begin from 𝑣*v* and follow edges one by one until you return to 𝑣*v*, forming a cycle.
   * While traversing, remove the edges from the graph to avoid revisiting them.
3. **Expanding the Circuit**:
   * While the current circuit does not cover all edges:
     + Find a vertex 𝑤*w* in the current circuit that has unused edges.
     + Construct a new cycle starting from 𝑤*w*, following unused edges until returning to 𝑤*w*.
     + Integrate this new cycle into the existing circuit.
4. **Output the Circuit**:
   * Once all edges are used and integrated into the circuit, the resulting circuit is an Eulerian circuit.

**Conclusion**

Hierholzer's algorithm is a straightforward and efficient method to find an Eulerian circuit in a graph, provided the graph meets the necessary conditions. The algorithm's step-by-step edge traversal and integration of cycles ensure that all edges are covered exactly once, resulting in a valid Eulerian circuit.

## C/Determining the Initial Circuit (R1)

My student ID 52200024, the last four digits form the number 0024. Therefore, (abcd) mod 4 = 0024 mod 4 = 0.  
  
Thus, the initial circuit R1 is EINME.

## Applying Hierholzer's Algorithm to the Graph

To determine an Eulerian circuit starting with R1 = EINME, we proceed as follows:  
  
1. Start with Initial Circuit R1:  
 R1 = E → I → N → M → E  
  
2. Identify and Extend Subtours:  
 - Starting from vertex E (EINME), we follow the unused edges.  
 - Continue from the endpoints of E and incorporate all edges.  
  
3. Constructing the Full Eulerian Circuit:  
 Let's use a step-by-step approach, given the complexity of the graph and its structure:

* Start at E.
* Traverse E → I → N → M → E.
* Extend from E: E → B → G → k → e → Y → S → M.
* Extend from M: M → N → Z → T → U.
* Continue U → V → b → c → X.
* Continue X → W → R → L → K → J → I.
* Finally, return I → E.

### Resulting Eulerian Circuit:

The final Eulerian circuit for the graph, starting with R1 = EINME, would be:  
  
E → I → N → M → E → B → G → k → e → Y → S → M → N → Z → T → U → V → b → c → X → W → R → L → K → J → I → E