Transform Yourself

Optimal Gas Pipeline using Prim's algorithm

PROJECT GUIDE

Dr.S.Kayalvili

Associate Professor

PROJECT MEMBER

Deetshitha Sri K S - 23ADR027

Durga Devi E - 23ADR043

Hari Prasath C - 23ADR055

KONGU ENGINEERING COLLEGE
PERUNDURAI.
LOVE * KNOWLEDGE * SERVICE

Estd : 1984

PROBLEM STATEMENT

Design an optimal pipeline network to transport gas between various cities or nodes, minimizing the total cost of the pipeline construction while ensuring that every city is connected. The objective is to determine the minimal cost of constructing the pipeline network using **Prim's Algorithm**, which guarantees the minimum spanning tree (MST) of the graph representing the cities and potential pipelines.

OBJECTIVE

The main **objective** of using **Prim's Algorithm** for designing an optimal gas pipeline network is to **minimize the total construction cost** while ensuring that all cities are connected by pipelines

Prim's Algorithm helps achieve this by constructing a **Minimum Spanning Tree (MST)** of the graph representing cities and potential pipelines, which ensures that:

1.All cities are connected: Every city (node) in the network must be reachable from every other city, either directly or indirectly, through the selected pipelines (edges).

2.Minimization of cost: The total cost of constructing the pipelines between the cities is minimized. Each edge in the MST represents the cheapest connection between cities, ensuring the overall cost is the lowest possible

ALGORITHM USED

Prim's Algorithm is a **greedy algorithm** used to find the **Minimum Spanning Tree (MST)** of a graph, which ensures the minimum total cost for connecting all cities (nodes) with pipelines (edges). In the case of a gas pipeline network, cities are represented as nodes, and the pipelines connecting them have costs (weights). The goal is to connect all cities with the minimum possible cost for the pipeline construction.

KEY STEPS

- **Start** with an arbitrary node.
- **Initialize** the priority queue with edges from the starting node.
- **Select** the minimum weight edge from the priority queue.
- **Add** the corresponding node to the MST.
- **Update** the priority queue with edges from the new node.
- **Repeat** until all nodes are in the MST.
- **Terminate** when all nodes are included.
- **Output** the MST, which represents the optimal pipeline layout.

IMPLEMENTATION

```
import heapq
def prim(graph, start_node):
num_nodes = len(graph)
pq = []
heapq.heappush(pq, (0, start_node))
visited = [False] * num_nodes
mst_cost = 0
mst_edges = []
  while pq:
cost, node = heapq.heappop(pq)
if visited[node]:
        continue
visited[node] = True
mst_cost += cost
if cost != 0:  mst_edges.append((prev_node, node, cost))
for neighbor, edge_cost in graph[node]:
    if not visited[neighbor]:
         heapq.heappush(pq, (edge_cost, neighbor))
 prev_node = node
```

```
graph = {
        0: [(1, 2), (2, 3), (3, 4)],
        1: [(0, 2), (2, 1), (3, 5)],
        2: [(0, 3), (1, 1), (3, 6)],
        3: [(0, 4), (1, 5), (2, 6)]
 }
mst_cost, mst_edges = prim(graph, 0)
print("Total cost of the MST:", mst_cost)
print("Edges in the MST:", mst_edges)
```

RESULT

Total cost of the MST: 8
Edges in the MST: [(0, 1, 2), (1, 2, 1), (0, 3, 4)]

CONCLUSION

- The provided Python code implements **Prim's Algorithm** for finding the **Minimum Spanning Tree (MST)** of a connected graph, represented as an adjacency list.

- **Prim's Algorithm** is a greedy algorithm used to find the **minimum spanning tree (MST)** of a graph. The algorithm ensures that all nodes are connected while minimizing the total edge weight (cost).

- The algorithm ensures that all nodes are connected while minimizing the total edge weight (cost).

KONGU ENGINEERING COLLEGE

THANK YOU