Sadhan Gowda
8/4/21

CN Lab Internal

1) Design a topology of 3 routers which accept packets from any address and forwards it through the default route. Entire topology should consist of 2 switches and 6 PCs

→ In the given network, a static route is configured to enable sharing of data between multiple networks

Initially, pinging a PC on a different subnet network throws up an error.

Adding static route to the routers using ip route <dest. network> <subnet mask> <next hop> in configure terminal mode.

Upon adding static routes, pinging a PC on a different network (i.e) PC2 from PC0 works as expected.

2) Consider you are in Bangalore and wish to travel to Hyderabad, Mumbai, Delhi, Kolkata. You need to travel to all other places and identify the shortest path with minimal cost from Bengaluru to other destinations

```
import math
def dijkstra(graph, n, src):
    distance = [math.inf] * n
    distance[src] = 0
    final_selected = [(src, distance[src])]
    curr_vertex = src

    while len(final_selected) < n:
        min_vertex, min_dist = -1, math.inf
        for neighbor in graph[curr_vertex]:
            vertex, weight = neighbor
            distance[vertex] = min(distance[curr_vertex] + weight, distance[vertex])
```

```
for vertex in range(n):
    if distance[vertex] <= min_dist and (vertex, distance[vertex]) not in
      final selected:
      min vertex, min_dist = vertex, distance[vertex]

  final_selected.append((min_vertex, min_dist))
  curr_vertex = min_vertex

Print("Vertex\t distance")
[print(f"{v}\t{d}") for v, d in final_selected]
if __name__ == "__main__":
  n = int(input("Enter no. of vertices:"))
  e = int(input("enter no. of edges:"))
  graph_dict = {}
  print("Enter the edges as follows: [start] [end] [weight]")
  for i in range(e):
  start, end, weight = [int(j) for j in input().split()]
    if not graph_dict.get(start):
        graph_dict[start] = [(end, weight)]
    else
      graph_dict[start].append((end, weight))
    if not graph_dict.get(end):
        graph_dict[end] = [(start, weight)]
    else:
      graph_dict[end].append((start, weight))
    for i in range(n):
      print(f"source{i}:")
      dijkstra(graph_dict, n, i)
```

Output :

Enter no. of vertices : 5

Enter no. of edges : 7

Enter edges as follows : [start] [end] [weight]

0 1 234
0 2 456
3 4 245
2 3 567
2 4 678
1 4 986
0 4 345

Source 0 :

| Vertex | distance |
|---|---|
| 0 | 0 |
| 1 | 234 |
| 4 | 345 |
| 2 | 456 |
| 3 | 590 |

Source 1 :

| Vertex | distance |
|---|---|
| 1 | 0 |
| 0 | 234 |
| 4 | 579 |
| 2 | 690 |
| 3 | 824 |

Source 2 :

| Vertex | Distance |
|---|---|
| 2 | 0 |
| 0 | 456 |
| 3 | 567 |
| 4 | 678 |
| 1 | 690 |

Source 3 :

| Vertex | Distance |
|---|---|
| 3 | 0 |
| 4 | 245 |
| 2 | 567 |
| 0 | 590 |
| 1 | 824 |

Source 4 :

| Vertex | Distance |
|---|---|
| 4 | 0 |
| 3 | 245 |
| 0 | 345 |
| 1 | 579 |
| 2 | 678 |

Assigning Bengaluru to 0, Hyderabad to 1, Mumbai to 2, Delhi to 3 and Kolkata to 4,

Drawing the graph,