Graphs

Lecture 17

Instructor: Dr. Cong Pu, Ph.D.

cong.pu@okstate.edu

Adapted partially from Data Structures and Algorithms in Java, M.T. Goodrich, R.Tamassia and M. H. Goldwasser, Sixth Edition,Wiley; Data Structures and Algorithms in C++, Adam Drozdek, 4th Edition, Cengage Learning

Shortest Paths: Dijkstra Algorithm

- Finding the ***shortest path*** between two nodes,
  - the edges of the graph associated with values, e.g., distance, time, costs, amounts, etc.

- **Dijkstra's algorithm,**
  - find the **shortest path** between **source node** and **every other node**
  - if the path is **longer** than any other path from that point, it is **dropped**, and the other path is expanded
  - each vertex is visited, the new paths are started, and the vertex is then not used anymore
  - once all the vertices are visited, the algorithm is done

Dijkstra Algorith

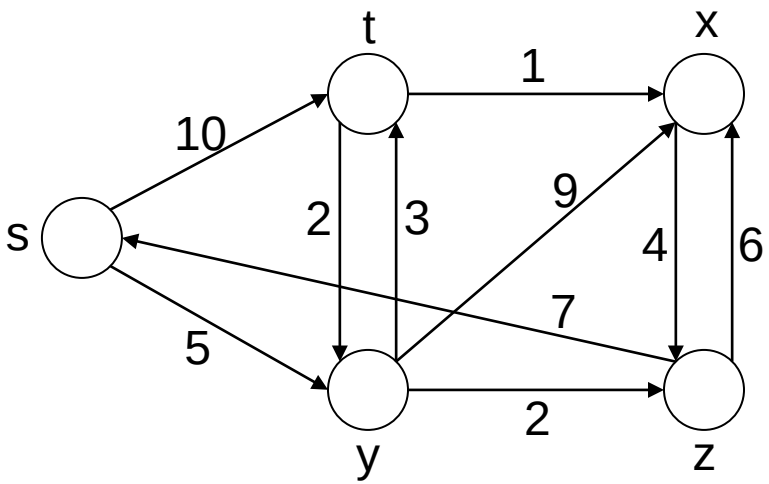
dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

Dijkstra Algorith



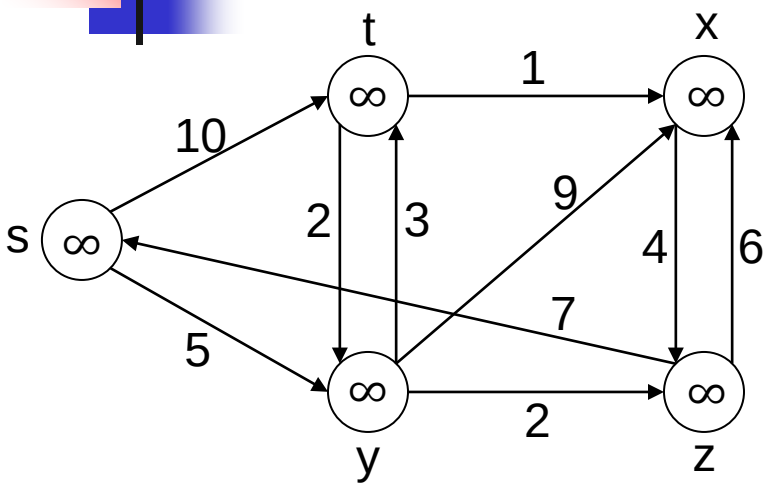

```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



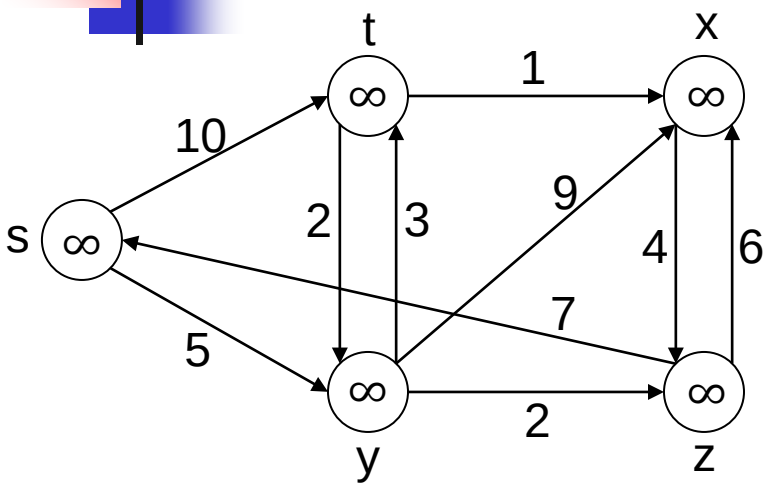

```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



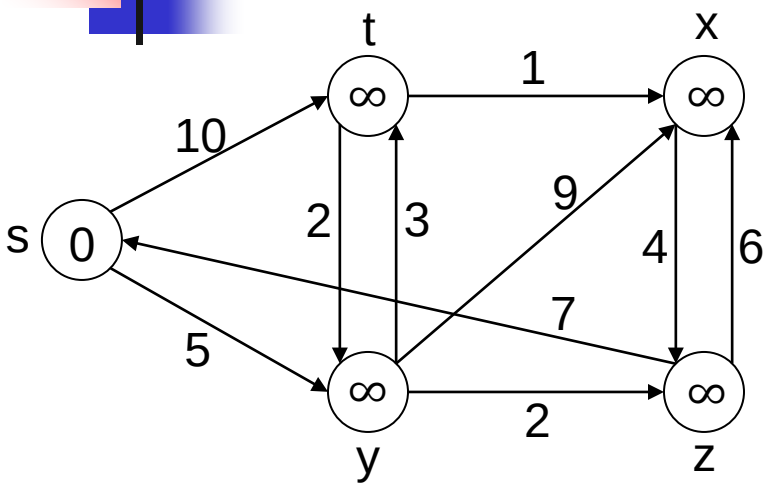

```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



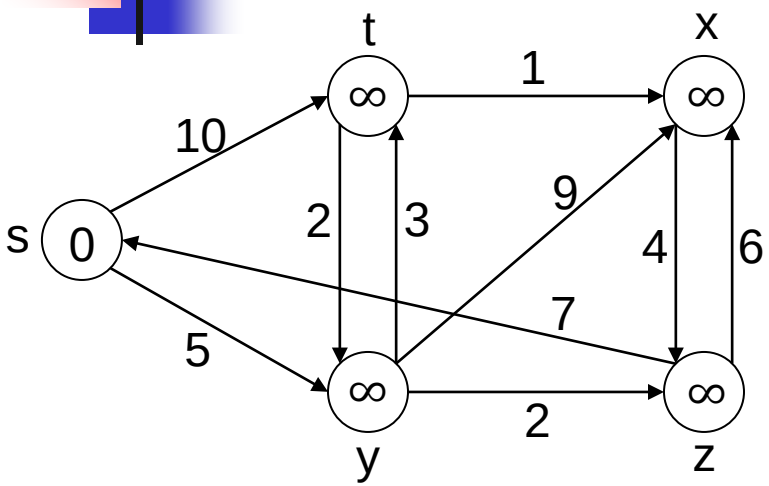

```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

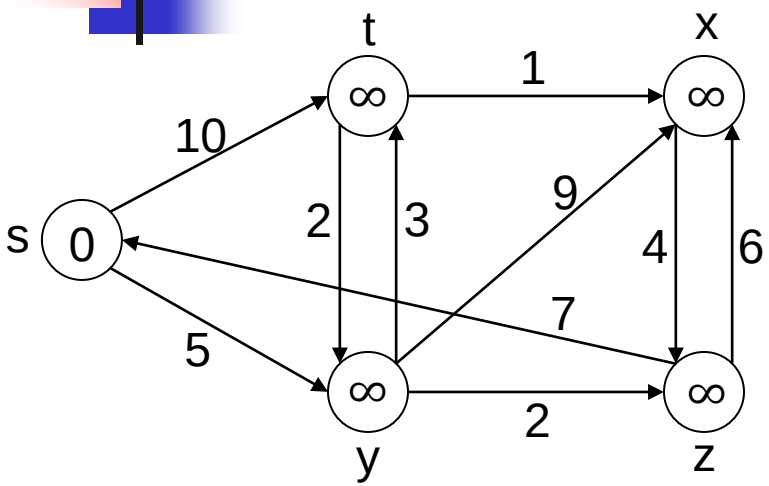
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

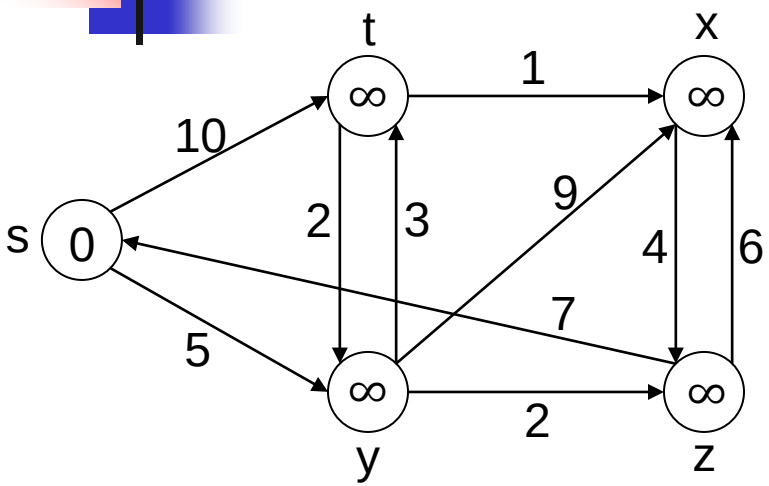
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

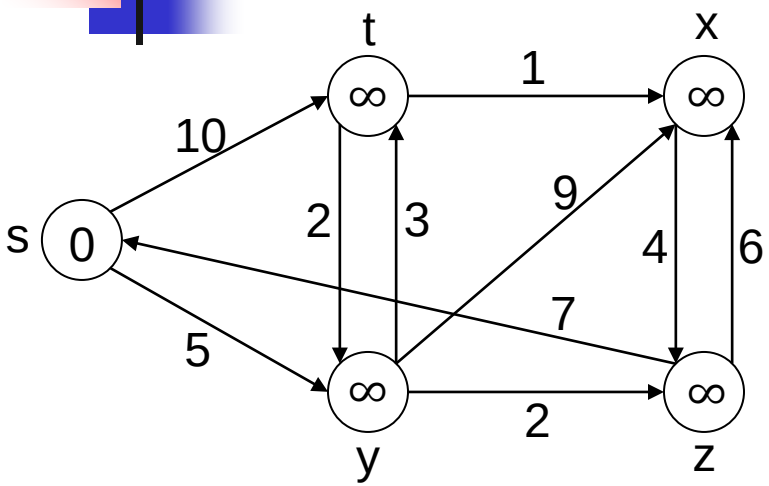
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[**s**, t, y, x, z]

u = s

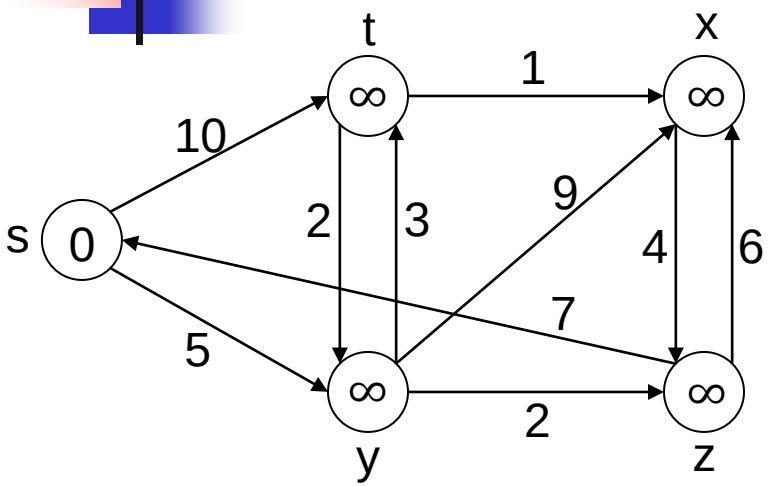
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

u = s

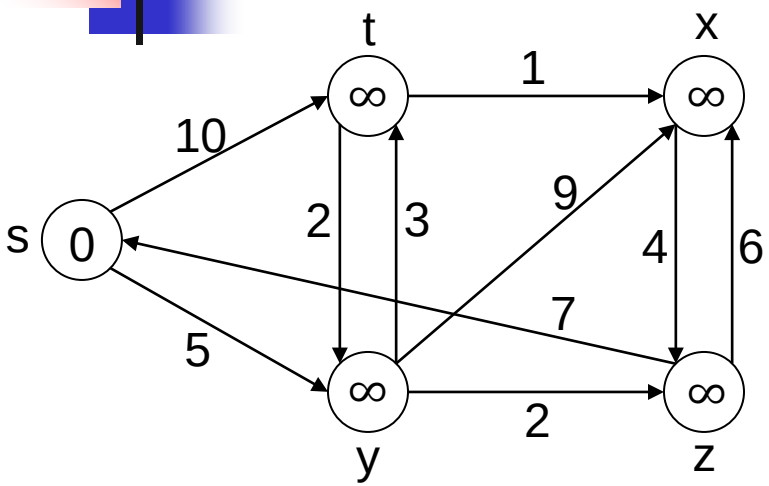
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

u = s

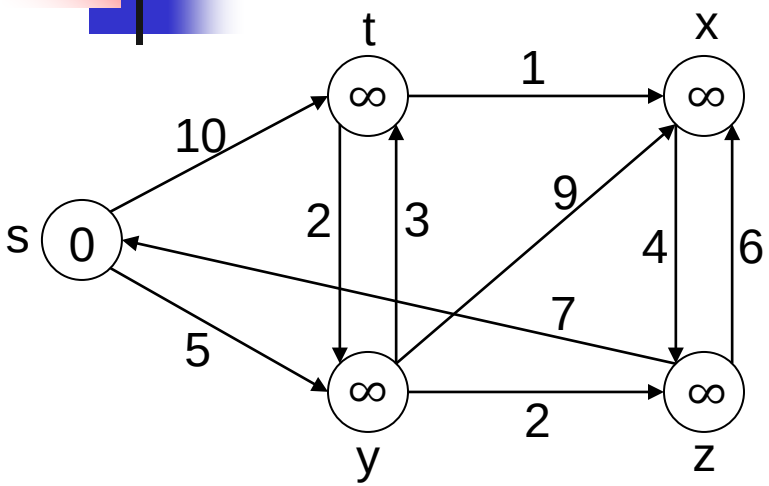
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

u = s

v = t
alt = 0 + 10 = 10

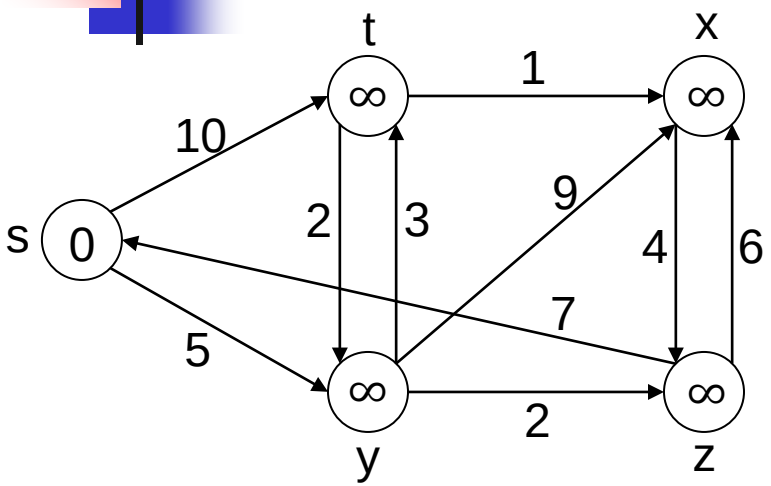
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

u = s

v = t

alt = 0 + 10 = 10 < ∞

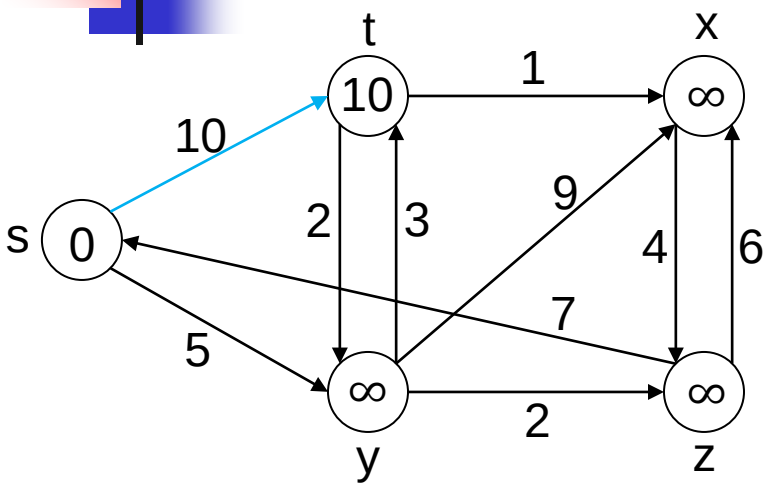
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

u = s

v = t

alt = 0 + 10 = 10 < ∞

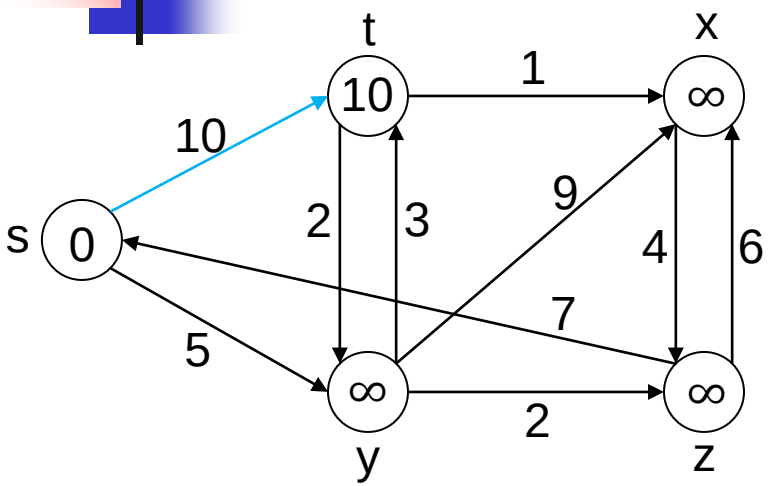
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

u = s

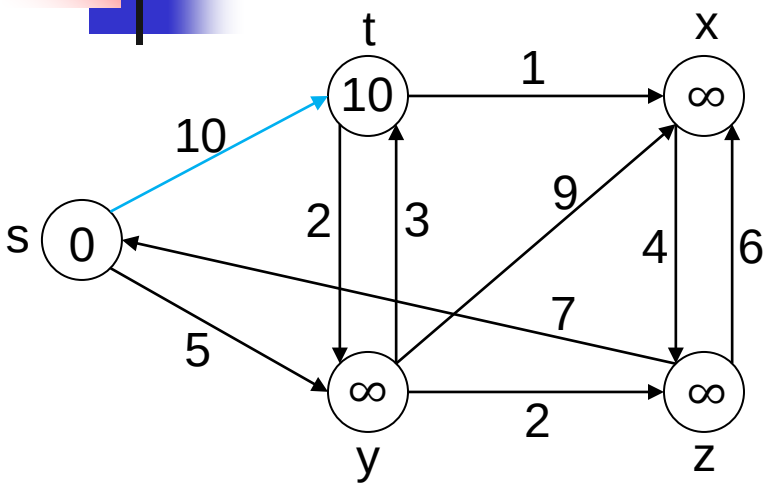
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

u = s

v = y
alt = 0 + 5 = 5

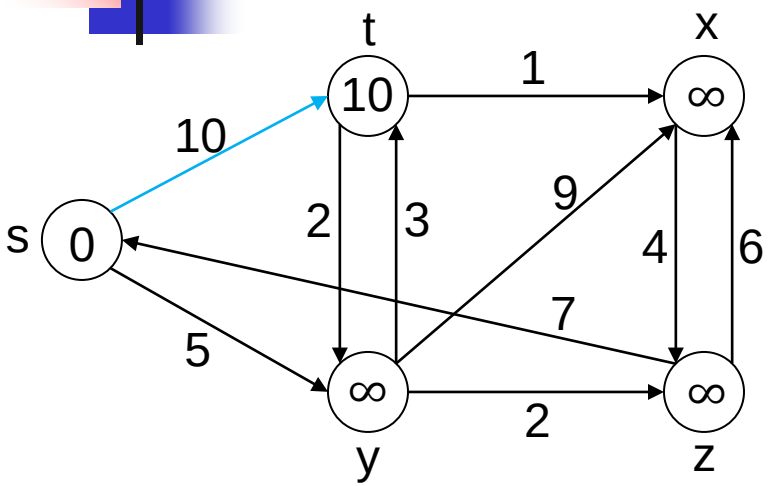
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

u = s

v = y

alt = 0 + 5 = 5 < ∞

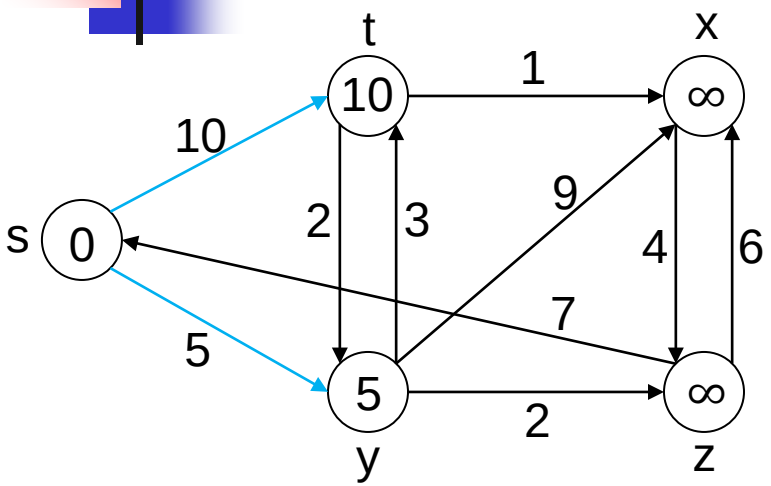
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

u = s

v = y

alt = 0 + 5 = 5 < ∞

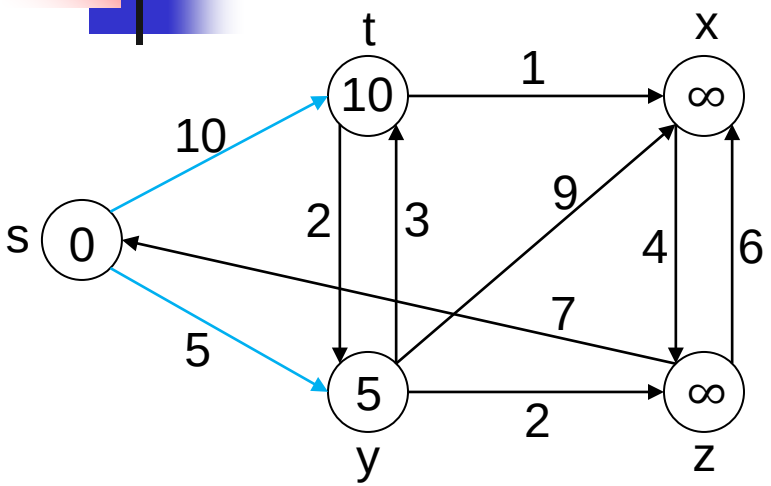
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

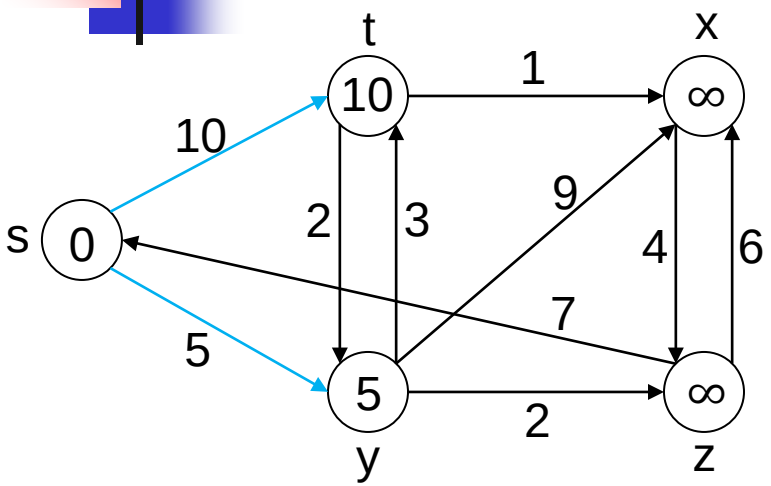
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

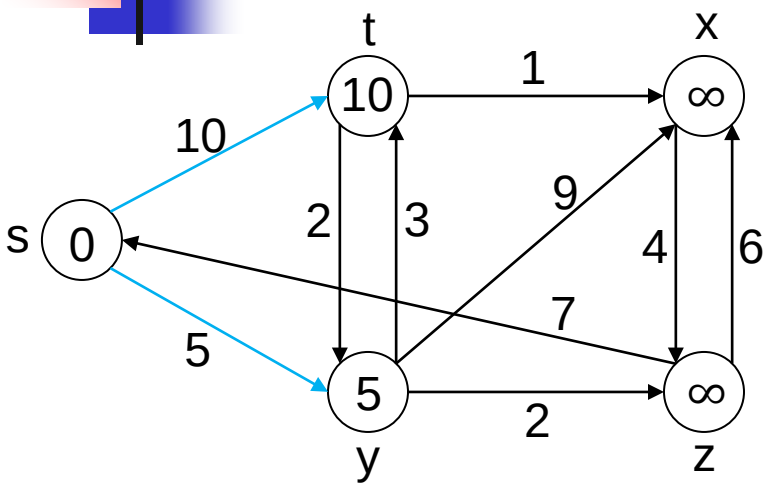
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

u = y

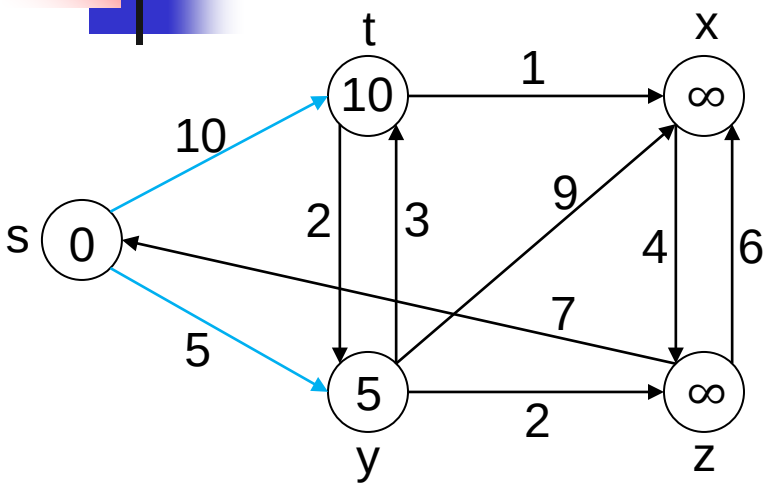
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, z]

u = y

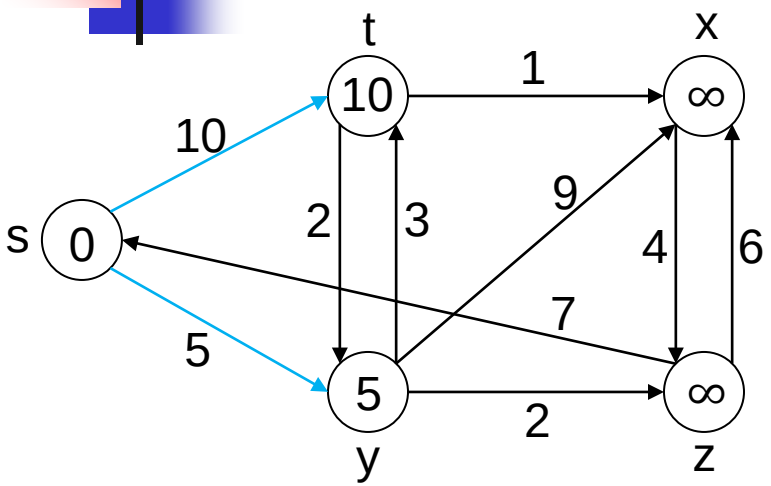
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

u = y

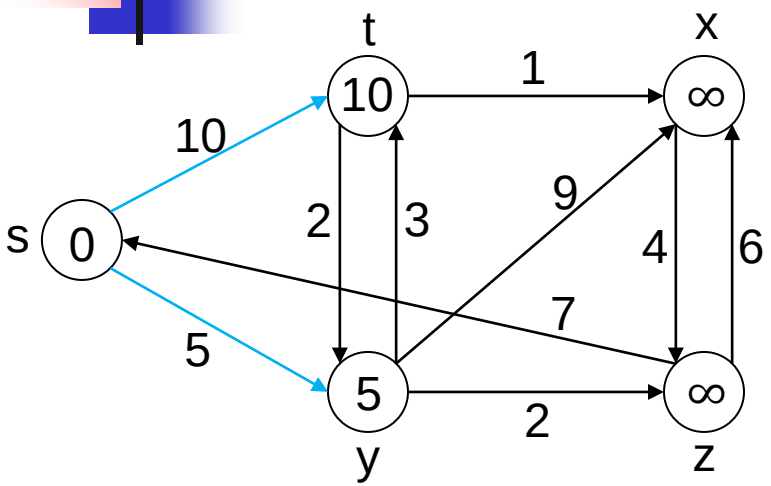
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, z]

u = y

v = t

alt = 5 + 3 = 8

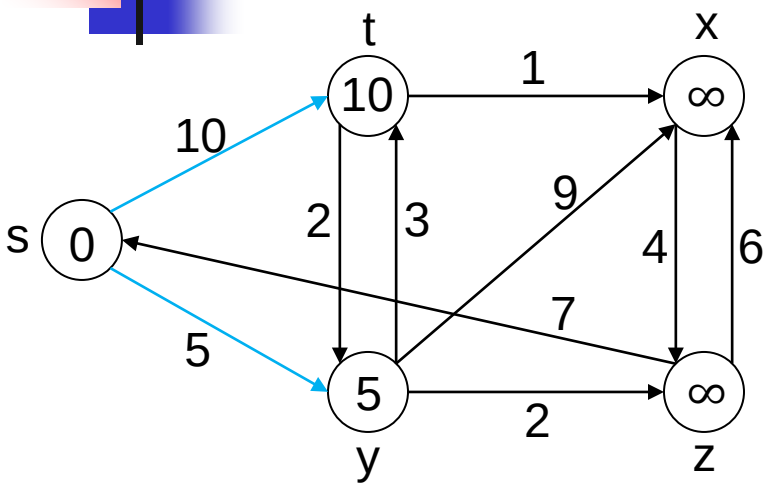
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

u = y

v = t

alt = 5 + 3 = 8 < 10

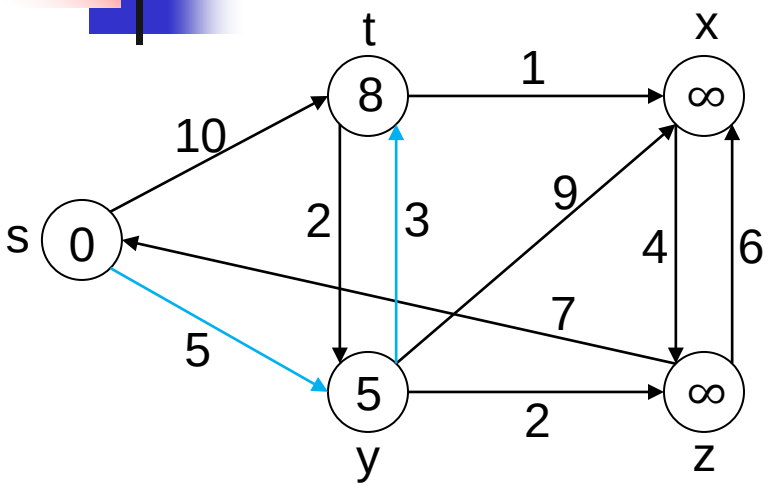
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

u = y

v = t

alt = 5 + 3 = 8 < 10

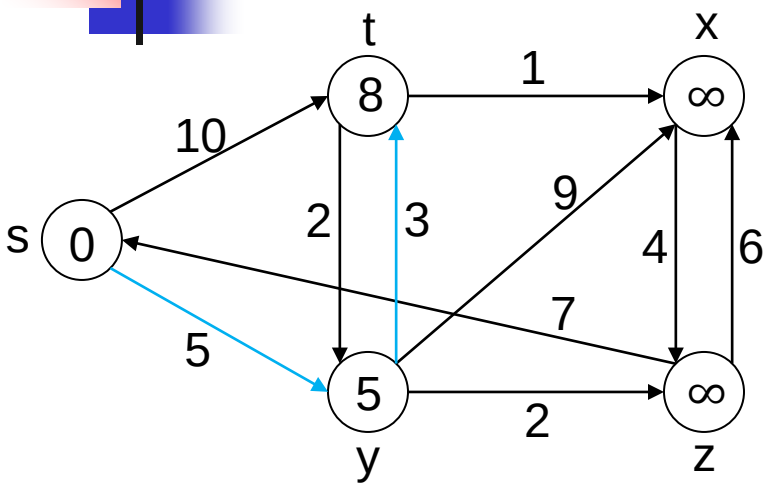
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, z]

u = y

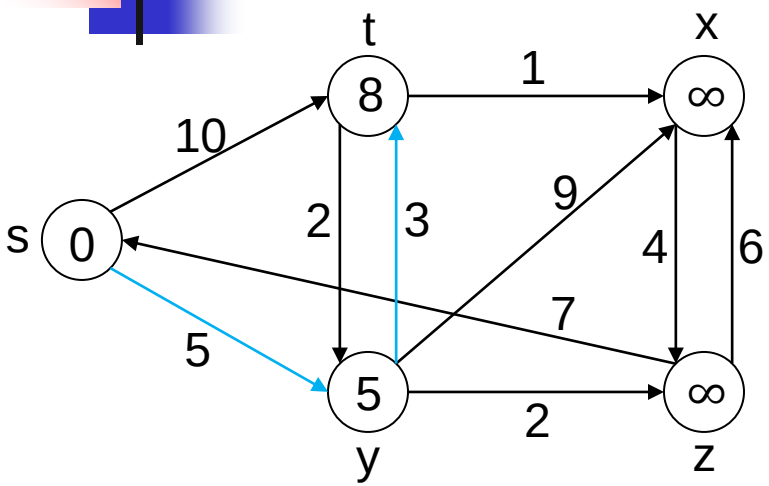
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, z]

u = y

v = x

alt = 5 + 9 = 14

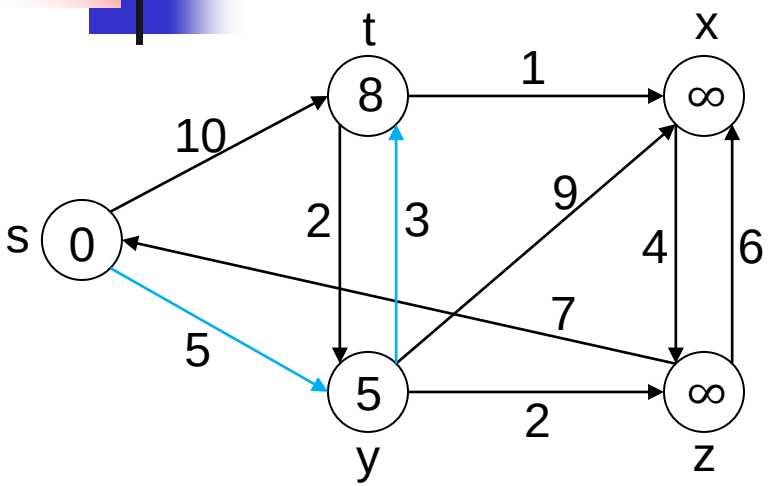
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, z]

u = y

v = x

alt = 5 + 9 = 14 < ∞

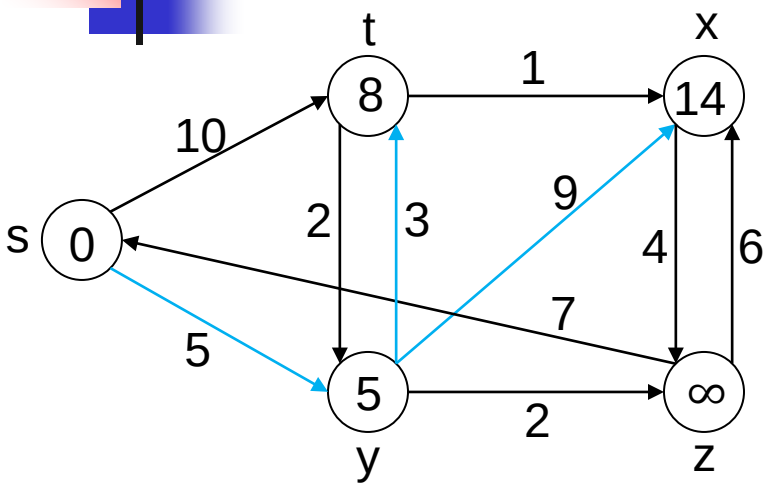
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, z]

u = y

v = x

alt = 5 + 9 = 14 < ∞

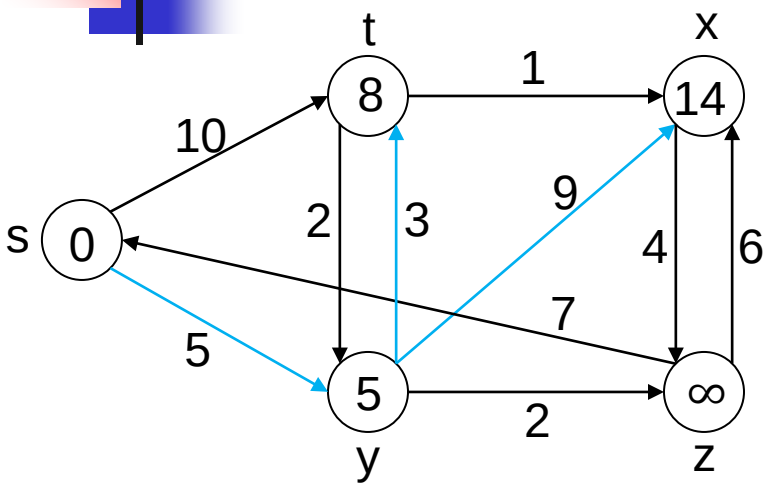
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, z]

u = y

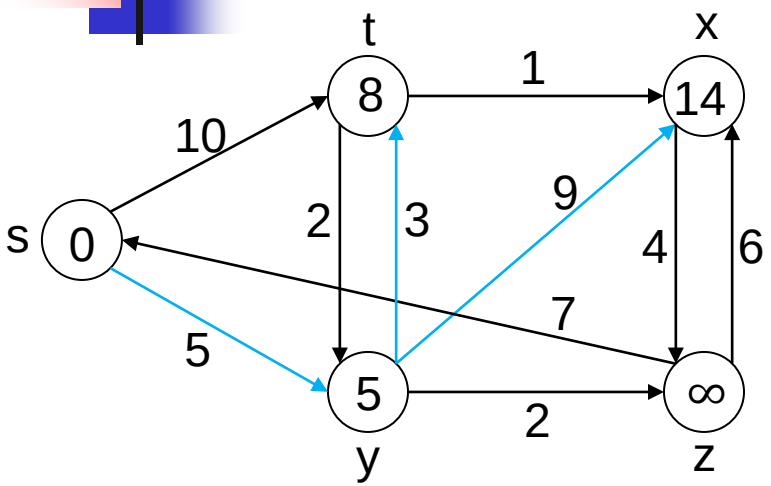
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

u = y

v = z

alt = 5 + 2 = 7

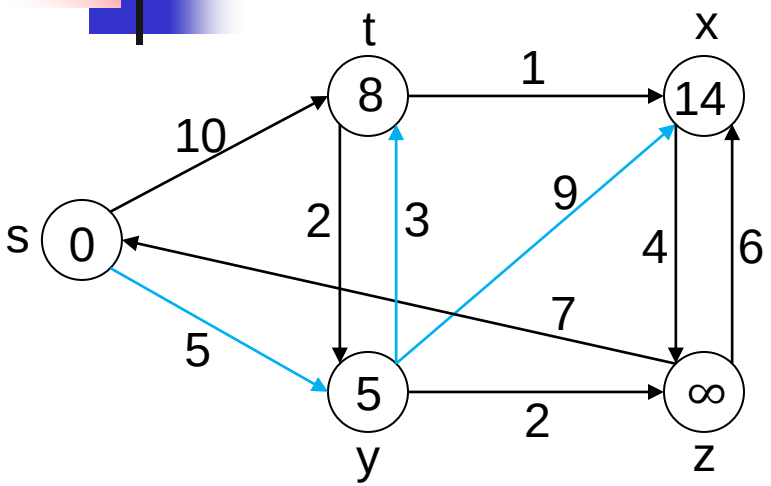
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, z]

u = y

v = z

alt = 5 + 2 = 7 < ∞

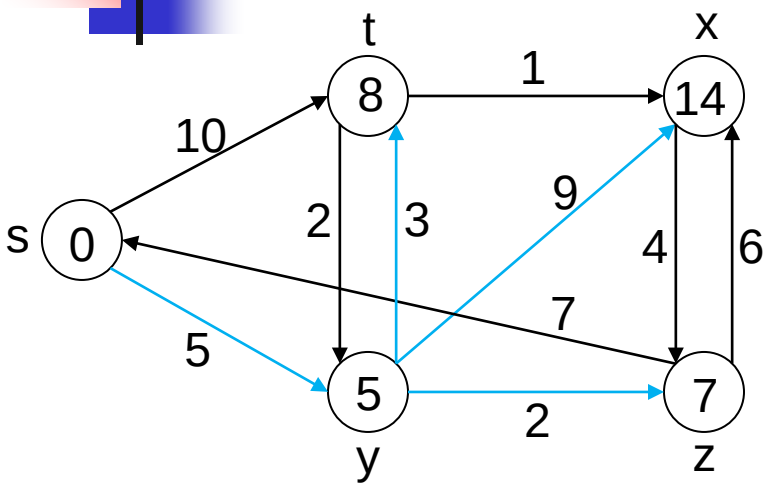
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, z]

u = y

v = z

alt = 5 + 2 = 7 < ∞

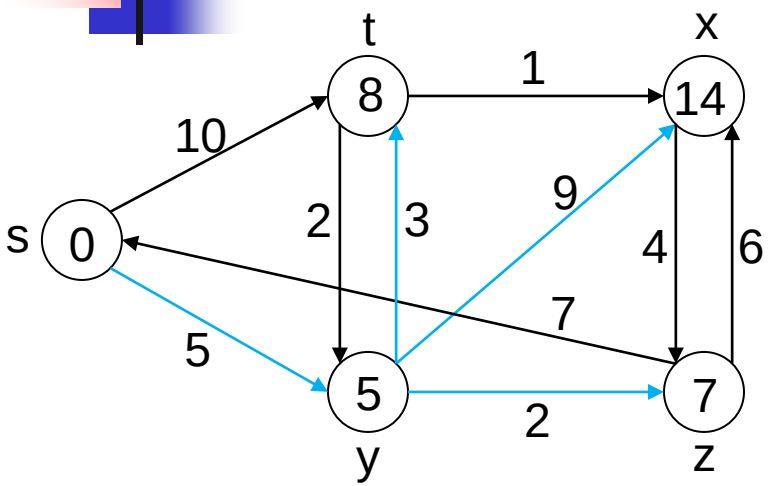
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, z]

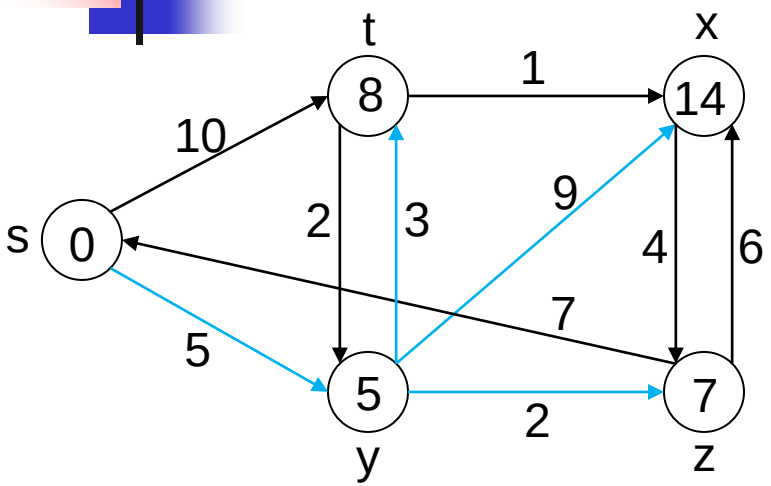
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[s, t, y, x, z]

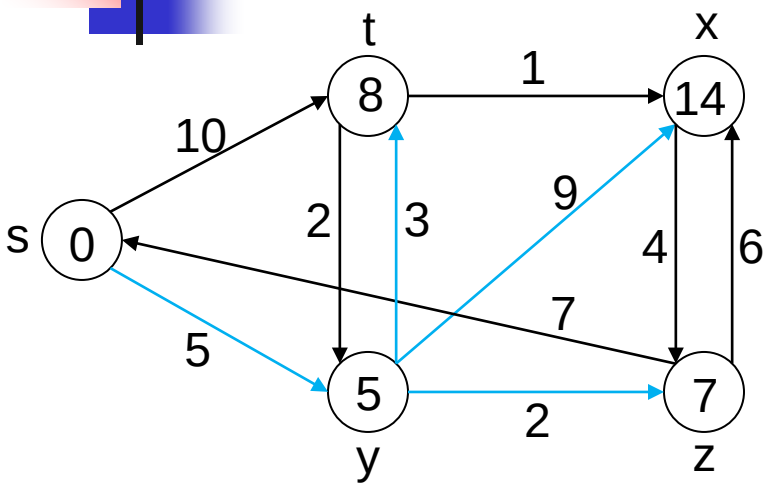
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, z]

u = z

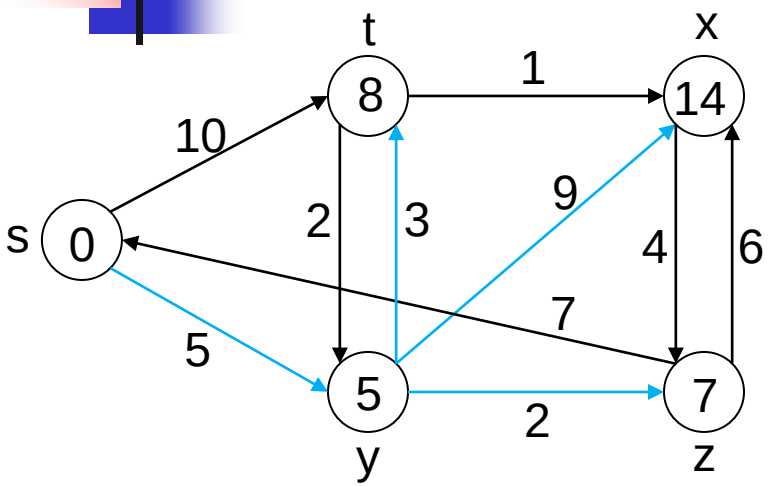
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, ~~z~~]

u = z

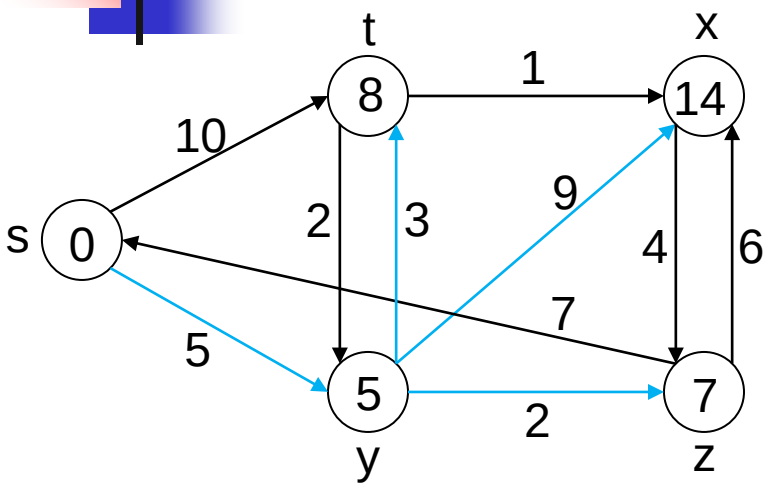
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, ~~z~~]

u = z

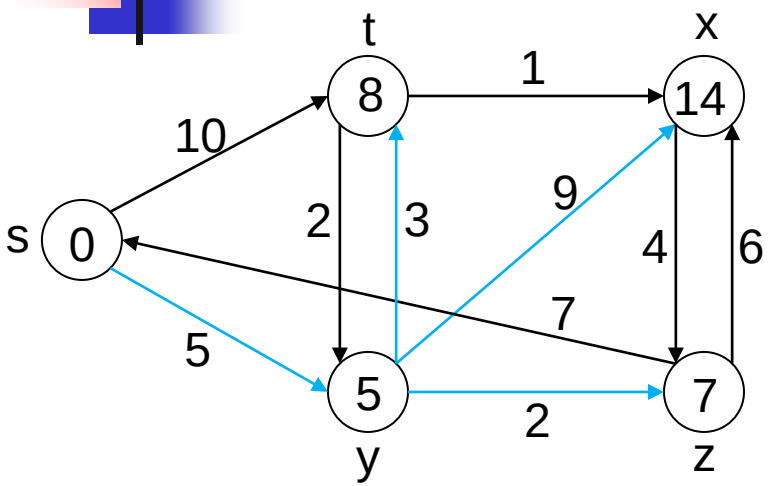
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, ~~z~~]

u = z

v = s

alt = 7 + 7 = 14

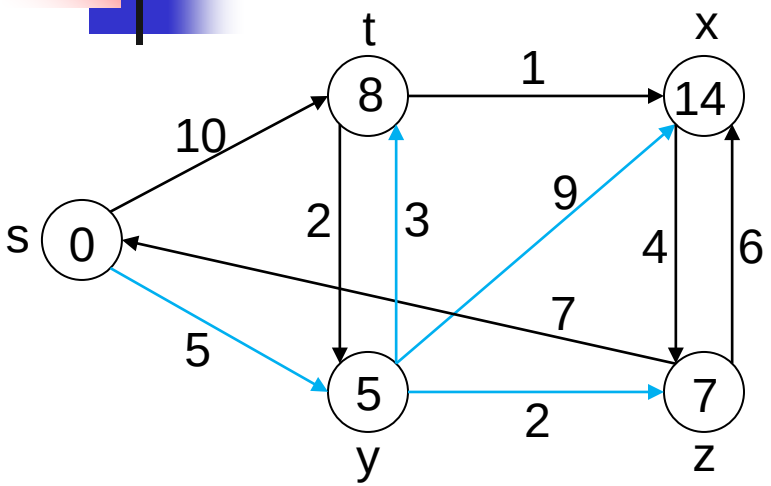
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, ~~z~~]

u = z

v = s

alt = 7 + 7 = 14 > 0

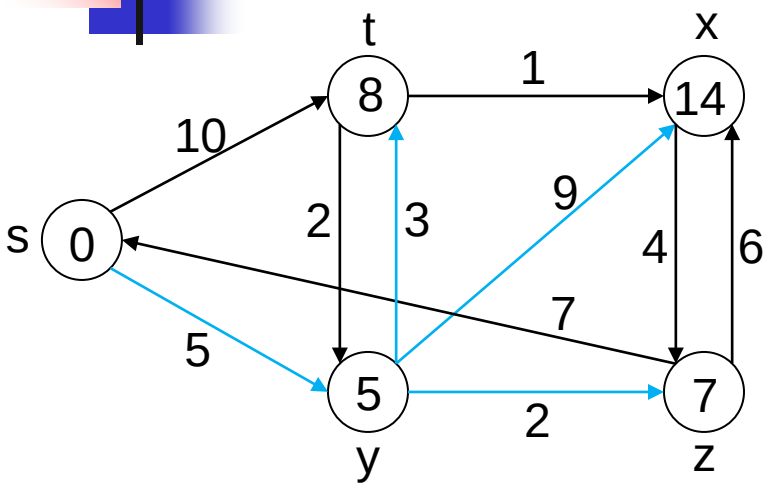
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, ~~z~~]

u = z

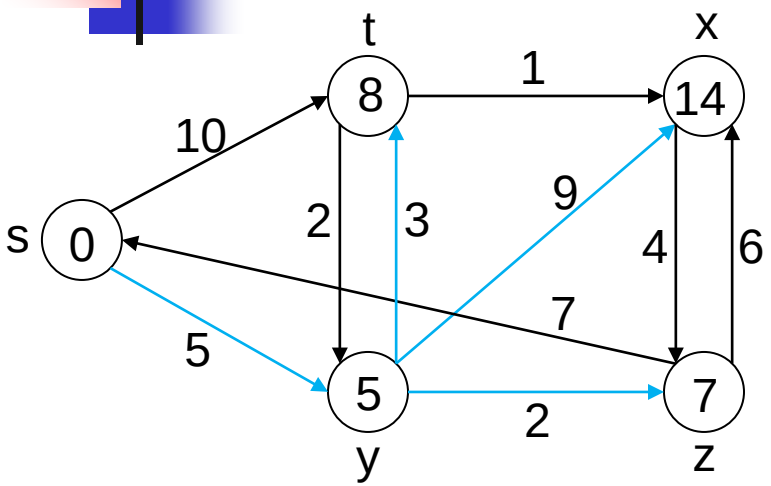
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, ~~z~~]

u = z

v = x

alt = 7 + 6 = 13

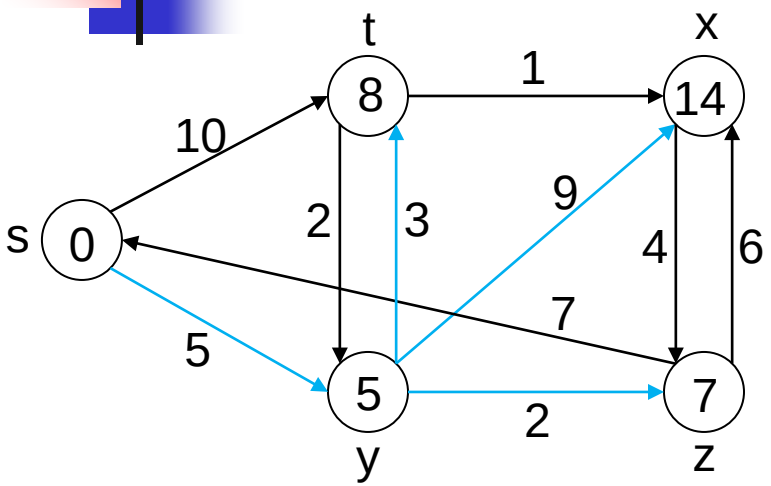
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, ~~z~~]

u = z

v = x

alt = 7 + 6 = 13 < 14

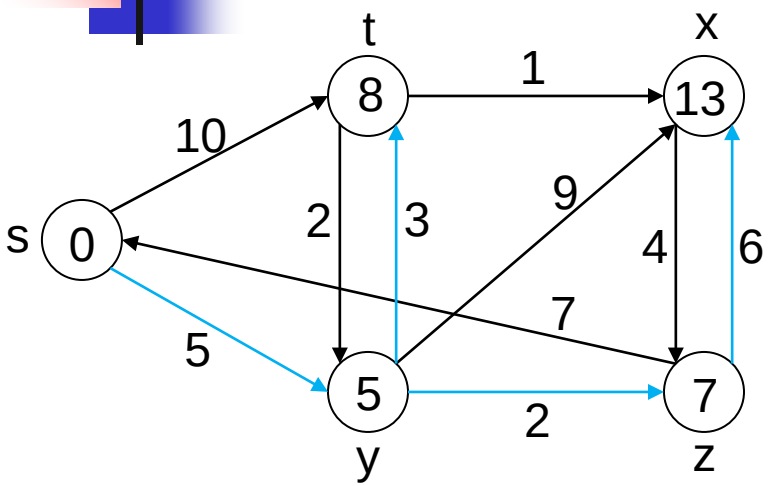
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, ~~z~~]

u = z

v = x

alt = 7 + 6 = 13 < 14

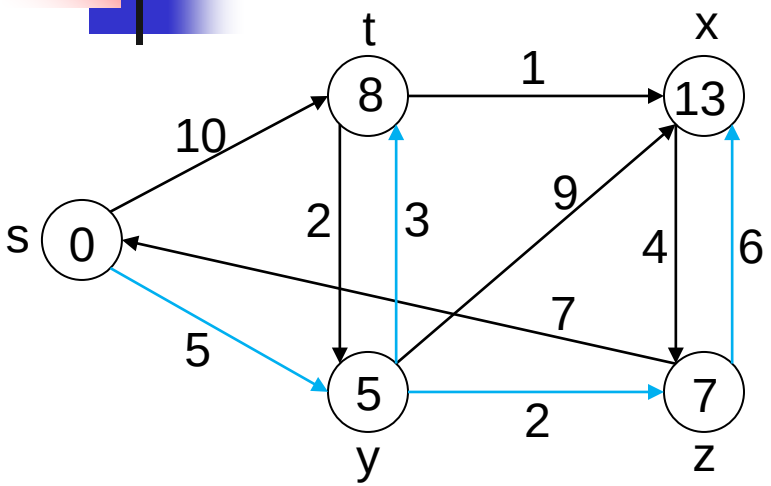
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, ~~z~~]

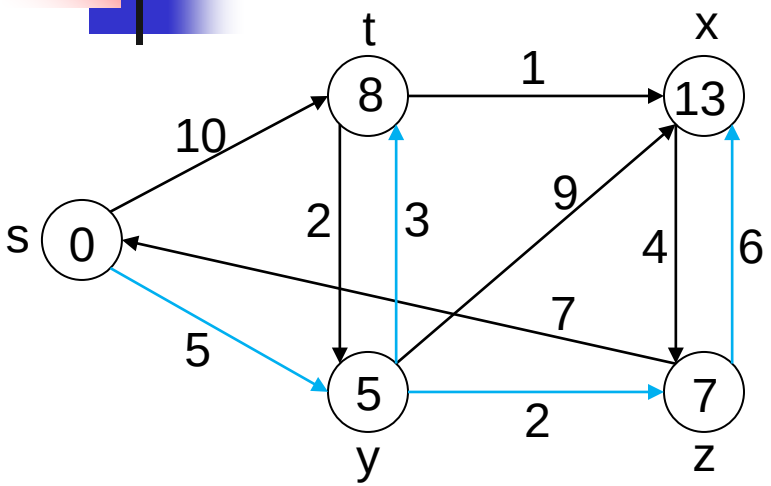
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, ~~z~~]

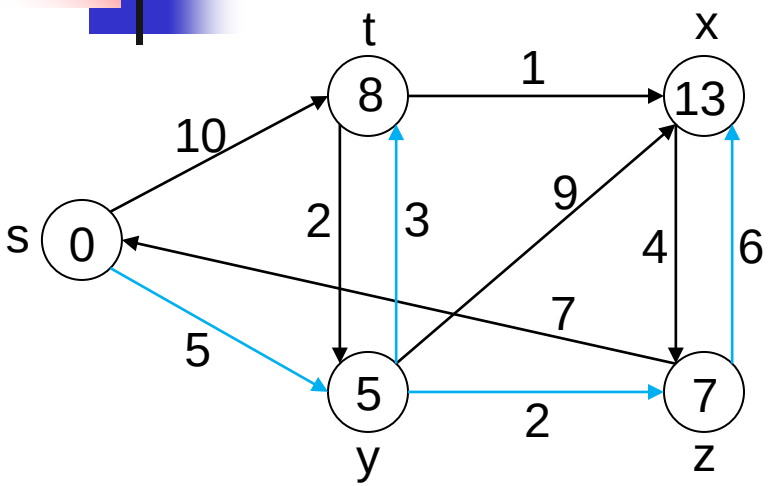
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, t, ~~y~~, x, ~~z~~]

u = t

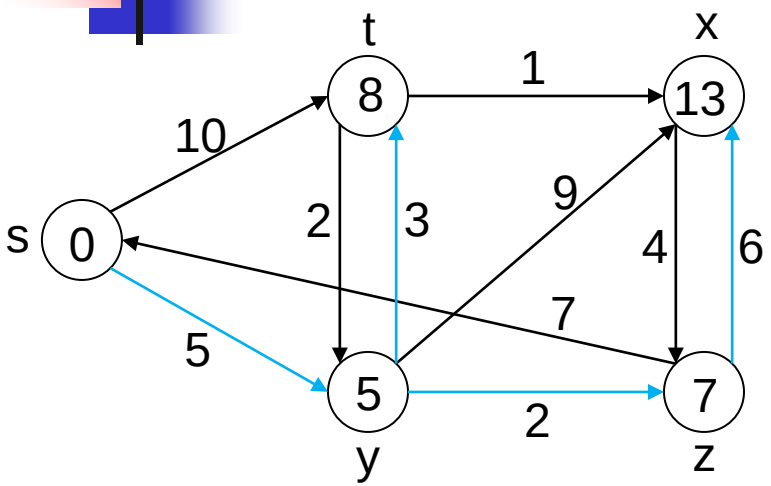
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, ~~t~~, ~~y~~, x, ~~z~~]

u = t

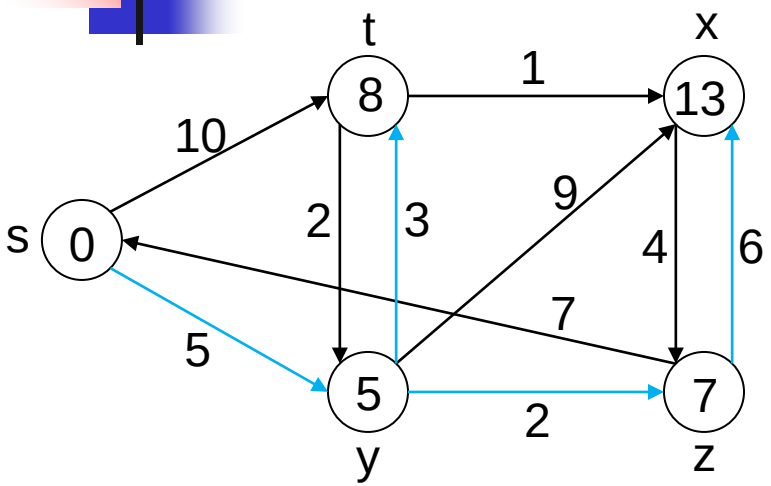
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, ~~t~~, ~~y~~, x, ~~z~~]

u = t

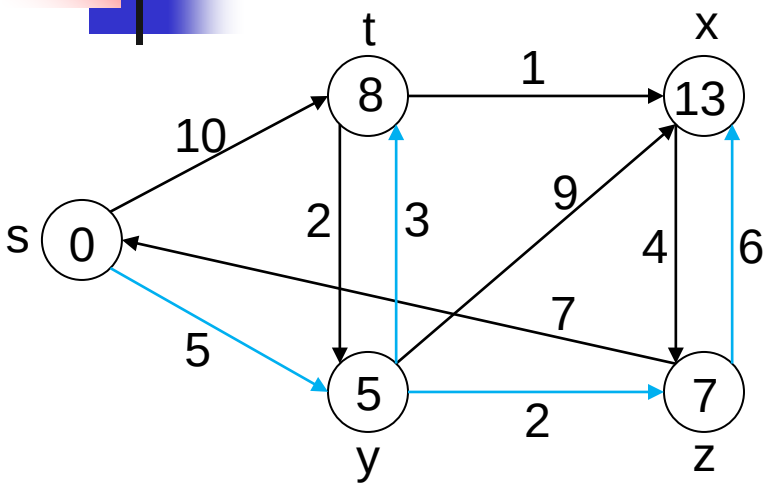
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, ~~t~~, ~~y~~, x, ~~z~~]

u = t

v = y

alt = 8 + 2 = 10

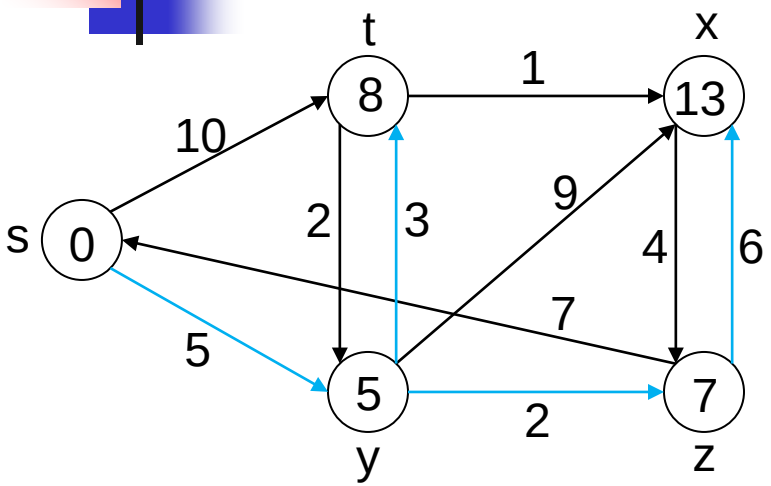
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, ~~t~~, ~~y~~, x, ~~z~~]

u = t

v = y

alt = 8 + 2 = 10 > 5

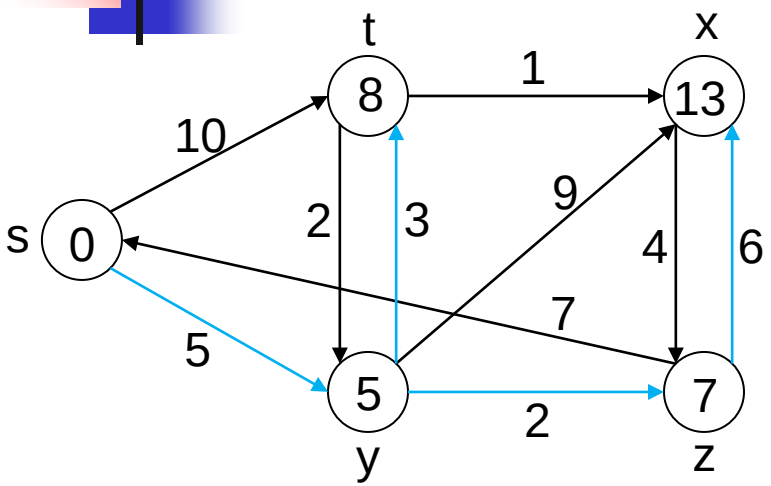
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, ~~t~~, ~~y~~, x, ~~z~~]

u = t

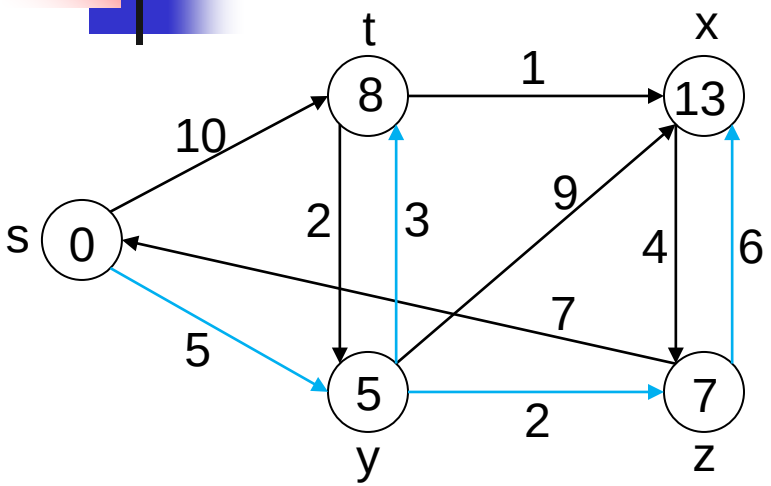
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, ~~t~~, ~~y~~, x, ~~z~~]

u = t

v = x

alt = 8 + 1 = 9

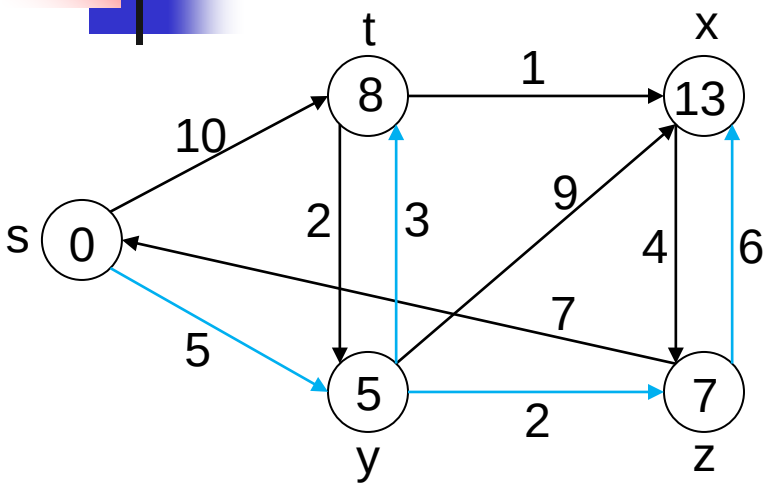
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, ~~t~~, ~~y~~, x, ~~z~~]

u = t

v = x

alt = 8 + 1 = 9 < 13

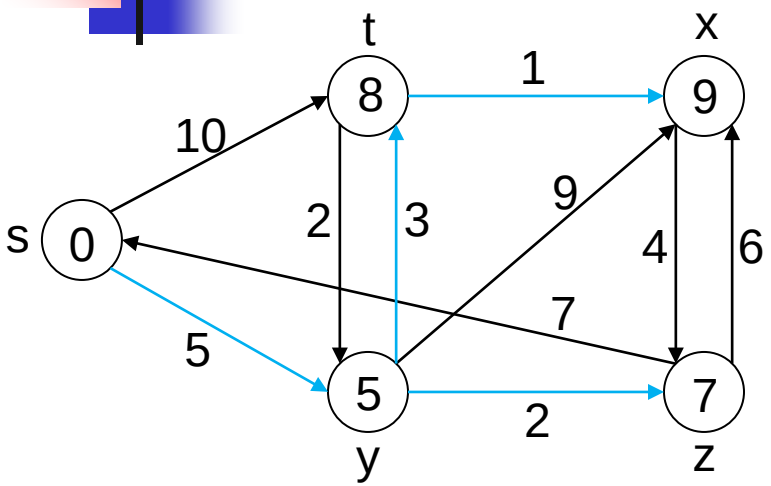
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, ~~t~~, ~~y~~, x, ~~z~~]

u = t

v = x

alt = 8 + 1 = 9 < 13

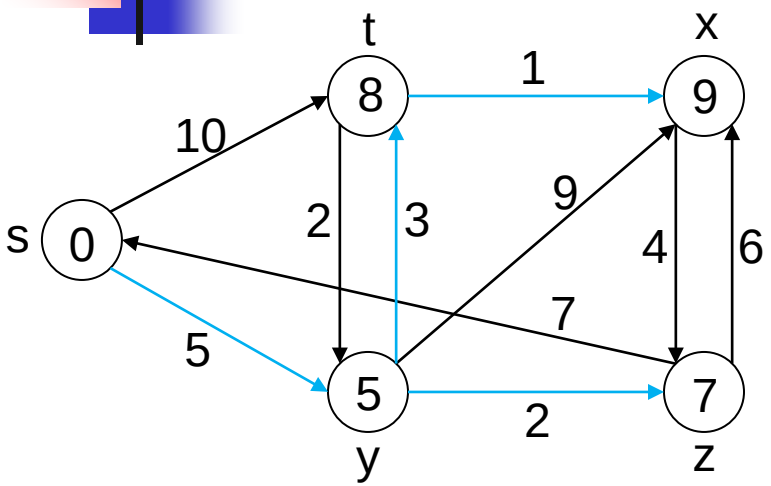
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, ~~t~~, ~~y~~, ~~x~~, ~~z~~]

u = x

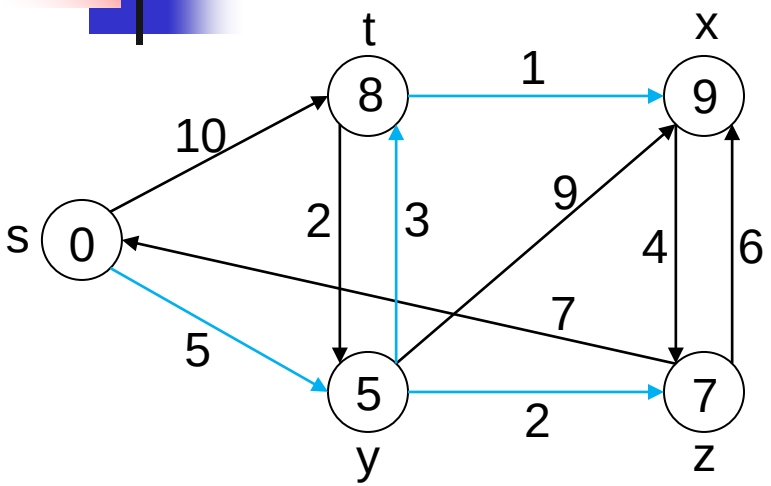
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith



Q:[~~s~~, ~~t~~, ~~y~~, ~~x~~, ~~z~~]

u = x

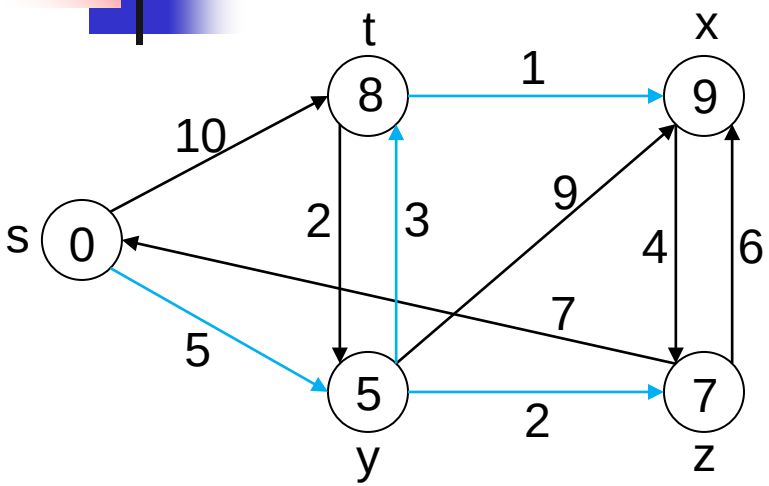
```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```

dist[v]: shortest distance from ***s*** to ***v***

previous[v]: previous vertex in shortest path to v

Q: set of vertices

Dijkstra Algorith




```
Dijkstra (G, s)
    for each vertex v in G
        dist[v] = ∞
        previous[v] = Undefined
    dist[s] = 0
    Q = G.V
    while Q is not empty
        u = node in Q with smallest dist[]
        remove u from Q
        for each neighbor v of u
            alt = dist[u] + dist_between(u, v)
            if alt < dist[v]
                dist[v] = alt
                previous[v] = u
    return previous[]
```