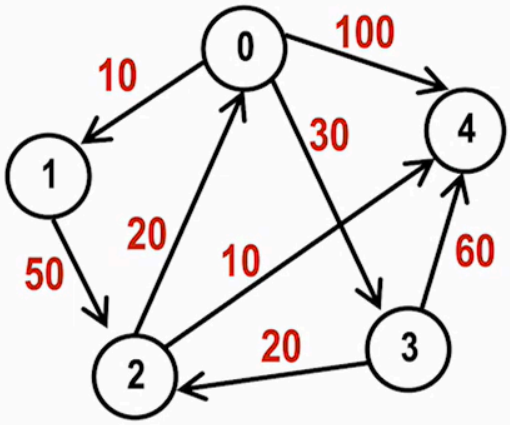
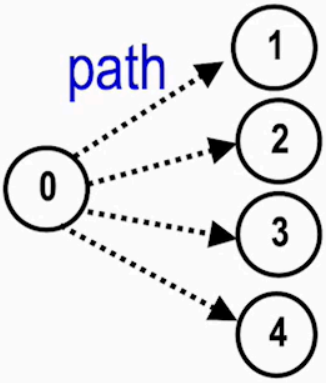
The Single-Source Shortest Paths Problem

The Single-Source Shortest Paths Problem

- **Given:**
  - Directed graph G = (V, E) in which each arc has a nonnegative label
- **Problem:**
  - Find the cost of the shortest path from a given SOURCE vertex, say **Vertex 0**, to the other vertices 1, 2, 3, 4
  - Cost of the path may represent something different like time.





The Single-Source Shortest Paths Problem

- **Application:**
  - Let G be a map of airline flights,
    - in which each vertex represents a city and
    - each arc **v** → **w** an airline route from city **v** to city **w**.
  - The label on arc **v** → **w** is the **time** to fly from city v to city w.
  - Solving the single-source shortest paths problem for this directed graph would determine the **minimum travel time** from a given city to every other city on the map

The Single-Source Shortest Paths Problem

- **Solution:**
  - use a “greedy” technique, often called as ***Dijkstra’s algorithm***

function Dijkstra

```
function Dijkstra
/* Dijkstra computes the cost of the shortest paths from vertex 0 to every
   vertex of a directed graph */
{
   S  = { 0 };

   for ( i = 1; i < n; i++)
       D[i] = C[0, i];          /* initializes D */

   for  (x = 1; x  < n; x++)
   {
      a) choose a vertex w in V-S such that
            D[w] is a minimum;   /* except the source vertex */
      b) add w to S;
      c) for each vertex v in V-S do
           D[v] = min( D[v], D[w] + C[w,v];
   }
}
```

Determine the data structures needed in this function.

```
function Dijkstra
{
  S = { 0 };

  for ( i = 1; i < n; i++)
     D[i] = C[0, i];     /* initializes D */

  for  (x = 1; x  < n; x++)
  {
    a) choose a vertex w in V-S
       such that D[w] is a minimum;
    b) add w to S;
    c) for each vertex v in V-S do
            D[v] = min( D[v], D[w] + C[w,v];
  }
}
```

Simulation
(Initializing D)

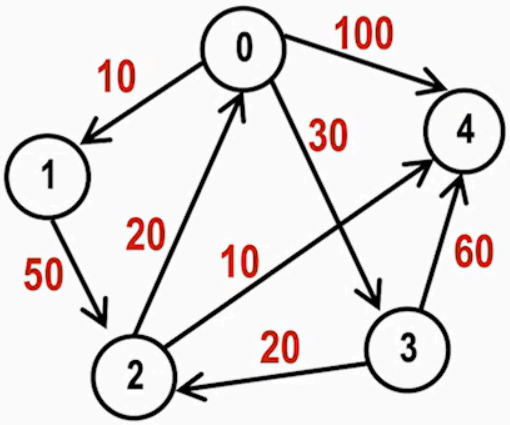
V = { 0, 1, 2, 3, 4 }

S = { 0 }

D

0	
1	10
2	∞
3	30
4	100

Graph G



Labeled Adj. matrix C

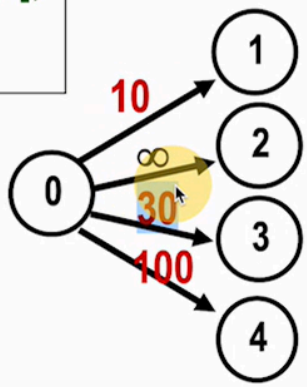
	0	1	2	3	4
0	∞	10	∞	30	100
1	∞	∞	50	∞	∞
2	20	∞	∞	∞	10
3	∞	∞	20	∞	60
4	∞	∞	∞	∞	∞

```
function Dijkstra
{
  S = { 0 };

  for ( i = 1; i < n; i++)
     D[i] = C[0, i];     /* initializes D */

  for  (x = 1; x  < n; x++)
  {
    a) choose a vertex w in V-S
       such that D[w] is a minimum;
    b) add w to S;
    c) for each vertex v in V-S do
            D[v] = min( D[v], D[w] + C[w,v];
  }
}
```

Simulation
(Illustrating D)

V = { 0, 1, 2, 3, 4 }

S = { 0 }



D

0	
1	10
2	∞
3	30
4	100

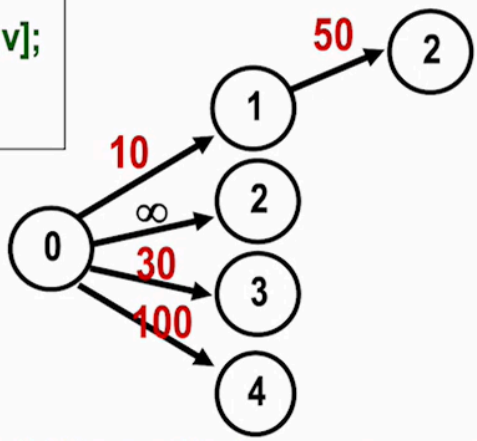
```
function Dijkstra
{
  S = { 0 };

  for ( i = 1; i < n; i++)
     D[i] = C[0, i];     /* initializes D */

  for  (x = 1; x  < n; x++)
  {
    a) choose a vertex w in V-S
       such that D[w] is a minimum;
    b) add w to S;
    c) for each vertex v in V-S do
           D[v] = min( D[v], D[w] + C[w,v];
  }
}
```

Simulation

(x = 1)

V = { 0, 1, 2, 3, 4 }

S = { 0 , 1 }



D

0	
1	10
2	60
3	30
4	100

x = 1

w = 1

v = 2: D[2] = min(∞, 10 + 50) = 60

v = 3: D[3] = min(30, 10 + ∞) = 30

v = 4: D[4] = min(100,10 + ∞) = 100

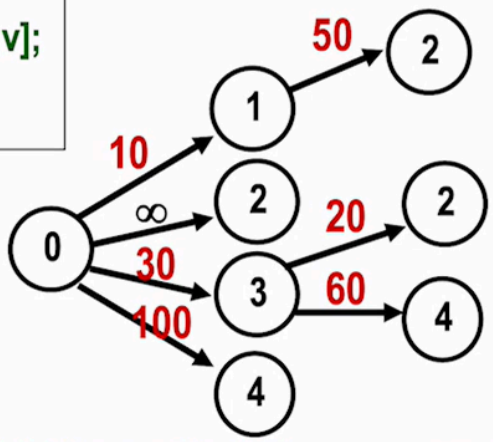
```
function Dijkstra
{
  S = { 0 };

  for ( i = 1; i < n; i++)
     D[i] = C[0, i];     /* initializes D */

  for  (x = 1; x  < n; x++)
  {
    a) choose a vertex w in V-S
       such that D[w] is a minimum;
    b) add w to S;
    c) for each vertex v in V-S do
            D[v] = min( D[v], D[w] + C[w,v];
  }
}
```

Simulation

(x = 2)

V = { 0, 1, 2, 3, 4 }

S = { 0 , 1 , 3 }



D

0	
1	10
2	50
3	30
4	90

x = 2

w = 3

v = 2: D[2] = min(60, 30 + 20) = 50

v = 4: D[4] = min(100,30 + 60) = 90

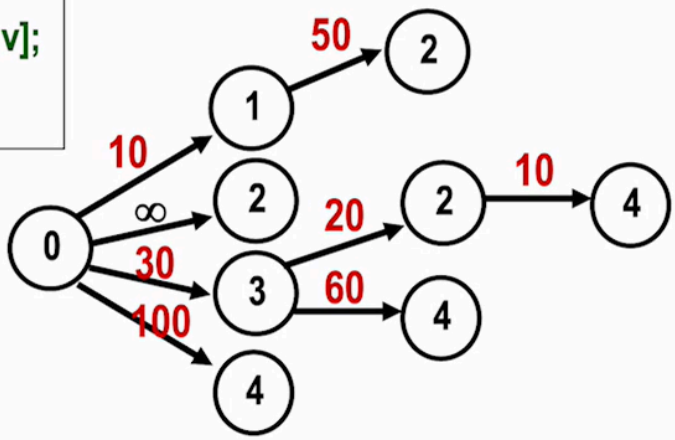
```
function Dijkstra
{
  S = { 0 };

  for ( i = 1; i < n; i++)
     D[i] = C[0, i];     /* initializes D */

  for  (x = 1; x  < n; x++)
  {
    a) choose a vertex w in V-S
       such that D[w] is a minimum;
    b) add w to S;
    c) for each vertex v in V-S do
            D[v] = min( D[v], D[w] + C[w,v];
  }
}
```

Simulation

(x = 3)

V = { 0, 1, 2, 3, 4 }

S = { 0 , 1 , 3 , 2 }



D

0	
1	10
2	50
3	30
4	60

x = 3

w = 2

v = 4: D[4] = min(90, 50 + 10) = 60

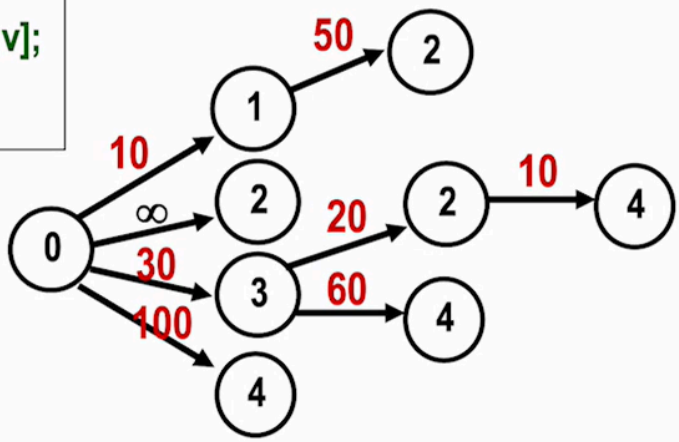
```
function Dijkstra
{
  S = { 0 };

  for ( i = 1; i < n; i++)
     D[i] = C[0, i];     /* initializes D */

  for  (x = 1; x  < n; x++)
  {
    a) choose a vertex w in V-S
       such that D[w] is a minimum;
    b) add w to S;
    c) for each vertex v in V-S do
            D[v] = min( D[v], D[w] + C[w,v];
  }
}
```

Simulation

(x = 4)

V = { 0, 1, 2, 3, 4 }

S = { 0 , 1 , 3 , 2 , 4 }



	D
0	
1	10
2	50
3	30
4	60

x = 4

w = 4