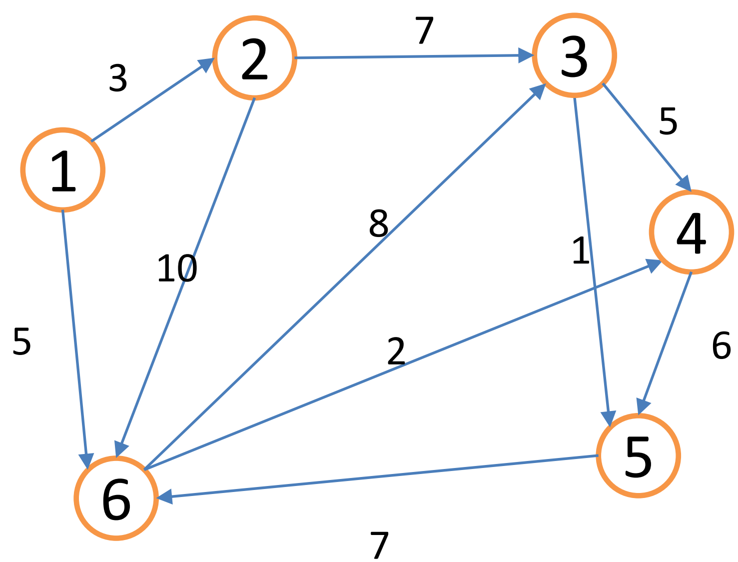
Shortest Path Algorithms

Dijkstra’s Algorithm

Example Graph and Adjacency Matrix

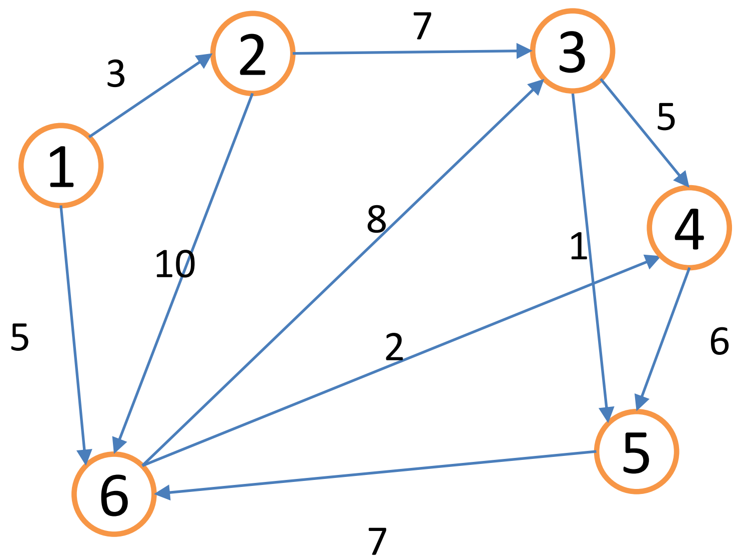
Graph



Adjacency Matrix (distance)

	1	**2**	**3**	**4**	**5**	**6**
1	0	3	∞	∞	∞	5
2	∞	0	7	∞	∞	10
3	∞	∞	0	5	1	∞
4	∞	∞	∞	0	6	∞
5	∞	∞	∞	∞	0	7
6	∞	∞	8	2	∞	0

Various Paths From Node 1 to 5

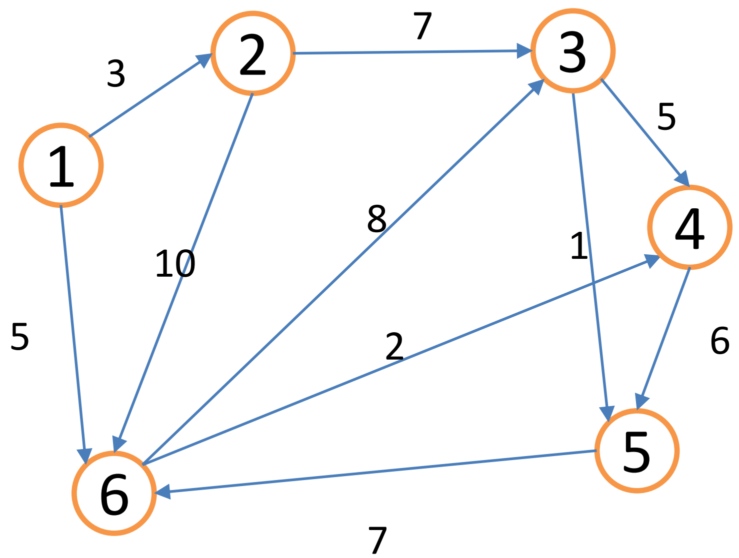


Paths to node 5

Nodes	Cost
1 2 3 4 5	
1 2 3 5	
1 2 6 3 5	
1 2 6 3 4 5	
1 2 6 4 5	
1 6 3 5	
1 6 3 4 5	
1 6 4 5	

We can see that there are many paths from **1** to **5**. How do we find the shortest?

Various Paths From Node 1 to 5



Paths to node 5

Nodes	Cost
1 2 3 4 5	21
1 2 3 5	11
1 2 6 3 5	22
1 2 6 3 4 5	32
1 2 6 4 5	21
1 6 3 5	14
1 6 3 4 5	24
1 6 4 5	13

We can see that there are many paths from **1** to **5**. How do we find the shortest?

Dijkstra's Algorithm

Edsger Wybe Dijkstra
1930 – 2002

Main contributions:
- Dijkstra's algorithm
- Semaphore

Recipient of Turing Award (1972)

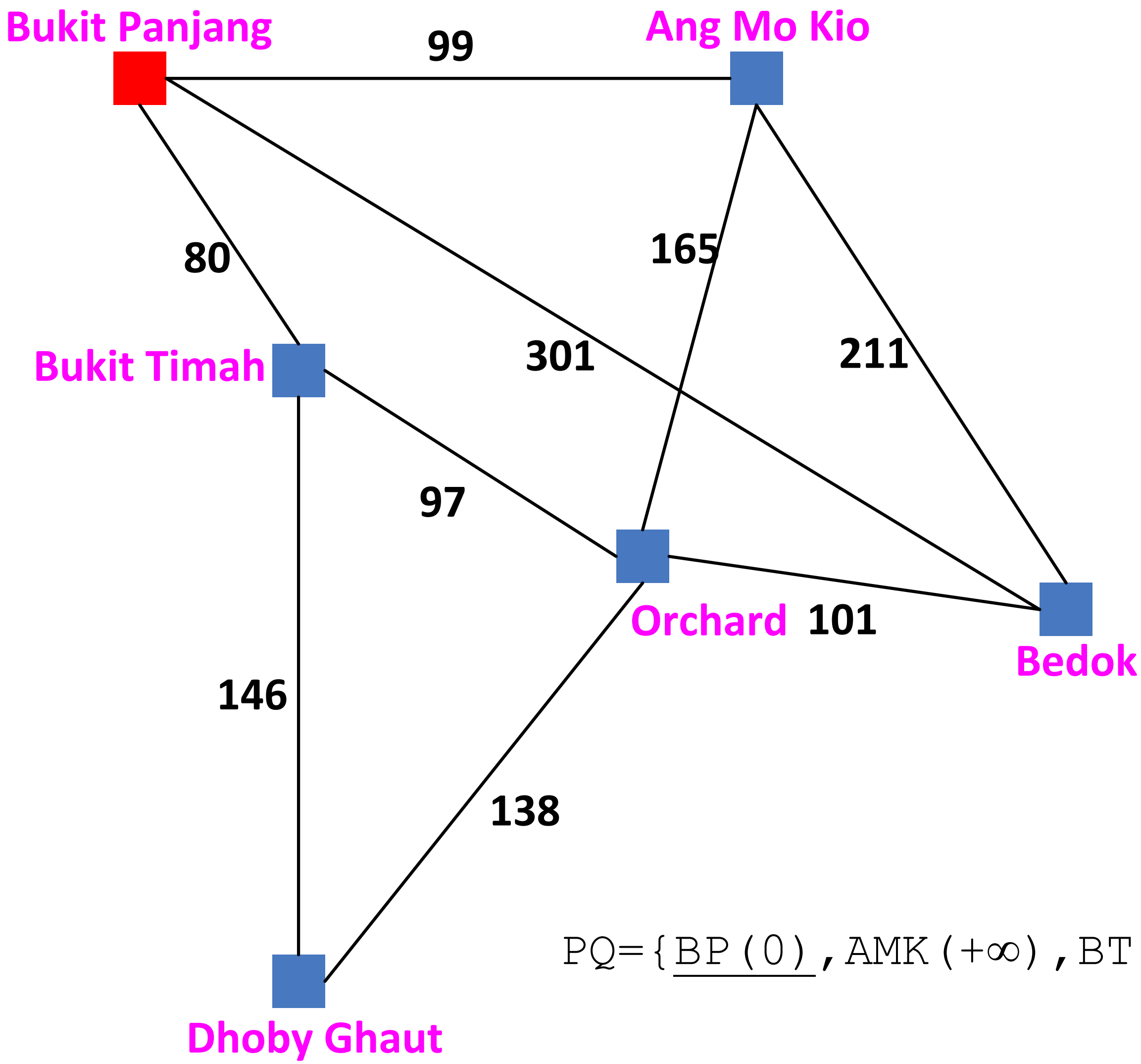
Dijkstra's Algorithm

- Finds the shortest path from the source node to **all** other nodes in the graph with **non-negative** edge costs.
- Greedy algorithm

Dijkstra’s Algorithm

```
Dijkstra(Graph, source){
   dist[source] = 0;
   for (each vertex v in Graph){
        if(v ≠ source)
            dist[v] = +∞;
        previous[v] = undefined;
        PQ.add_with_priority(v,dist[v]);
   }
   while (PQ is not empty){
      u = PQ.extract_min();
      for (each neighbor v of u)
         if (dist[u] + cost(u, v) < dist[v]){
             dist[v] = dist[u] + cost(u, v);
             previous[v] = u;
             PQ.decrease_priority(v,dist[v]);
         }
   }
}
```

Dijkstra's Algorithm

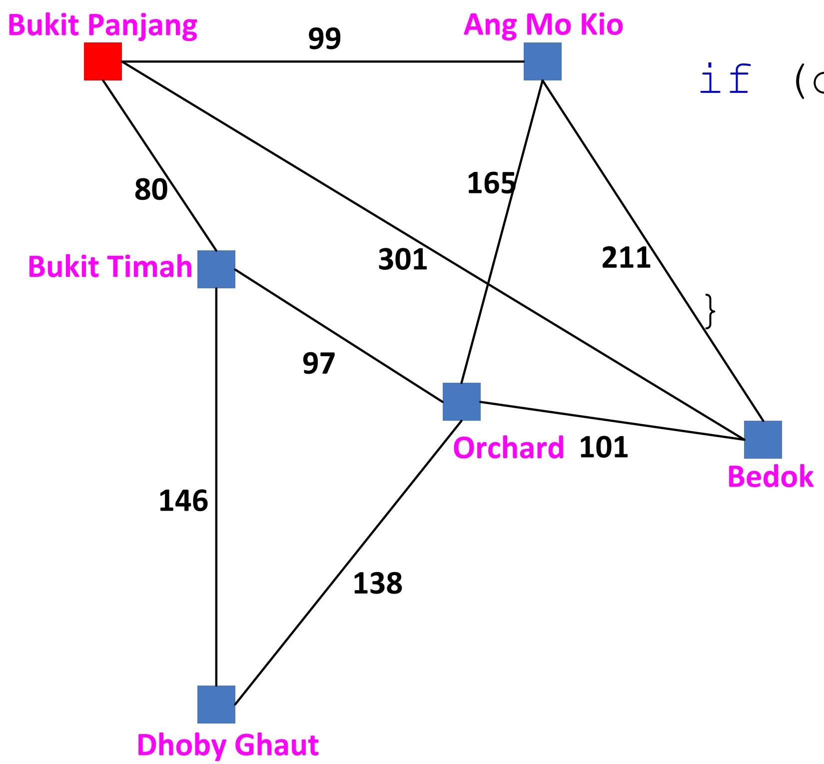



```
previous[BP]  = NA
previous[AMK] = NA
previous[BT]  = NA
previous[O]   = NA
previous[B]   = NA
previous[DG]  = NA
```

```
dist[BP]  = 0
dist[AMK] = +∞
dist[BT]  = +∞
dist[O]   = +∞
dist[B]   = +∞
dist[DG]  = +∞
```

PQ={BP(0),AMK(+∞),BT(+∞),O(+∞),B(+∞),DB(+∞)}

Dijkstra's Algorithm



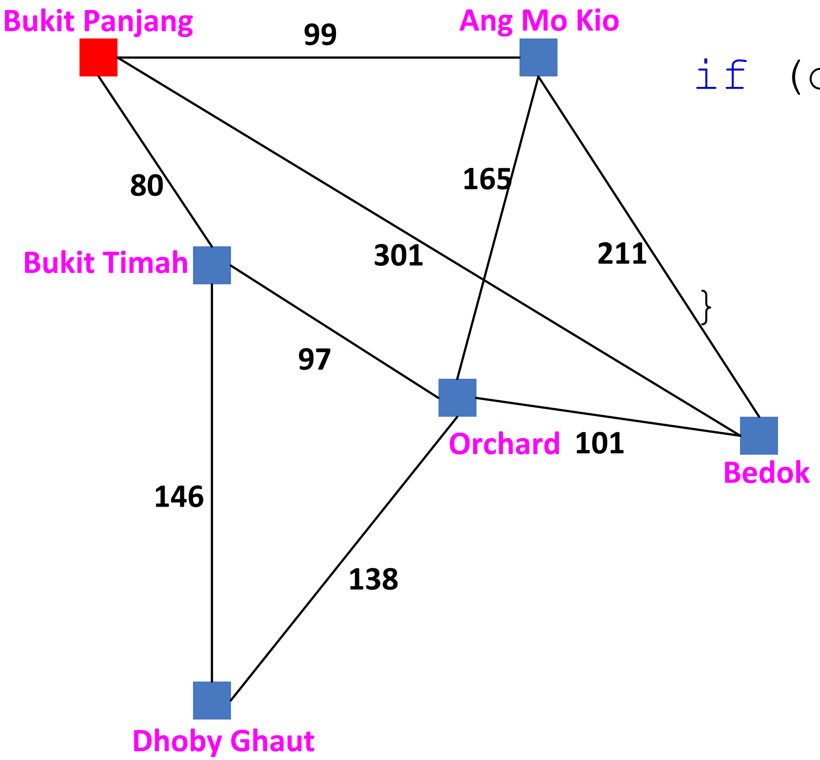
BP

BT

```
if (dist[u] + cost(u, v) < dist[v]){
        dist[v] = dist[u] + cost(u, v);
        previous[v] = u;
        PQ.decrease_priority(v,dist[v]);
}
```

PQ={BP(0),AMK(+∞),BT(+∞),O(+∞),B(+∞),DB(+∞)}

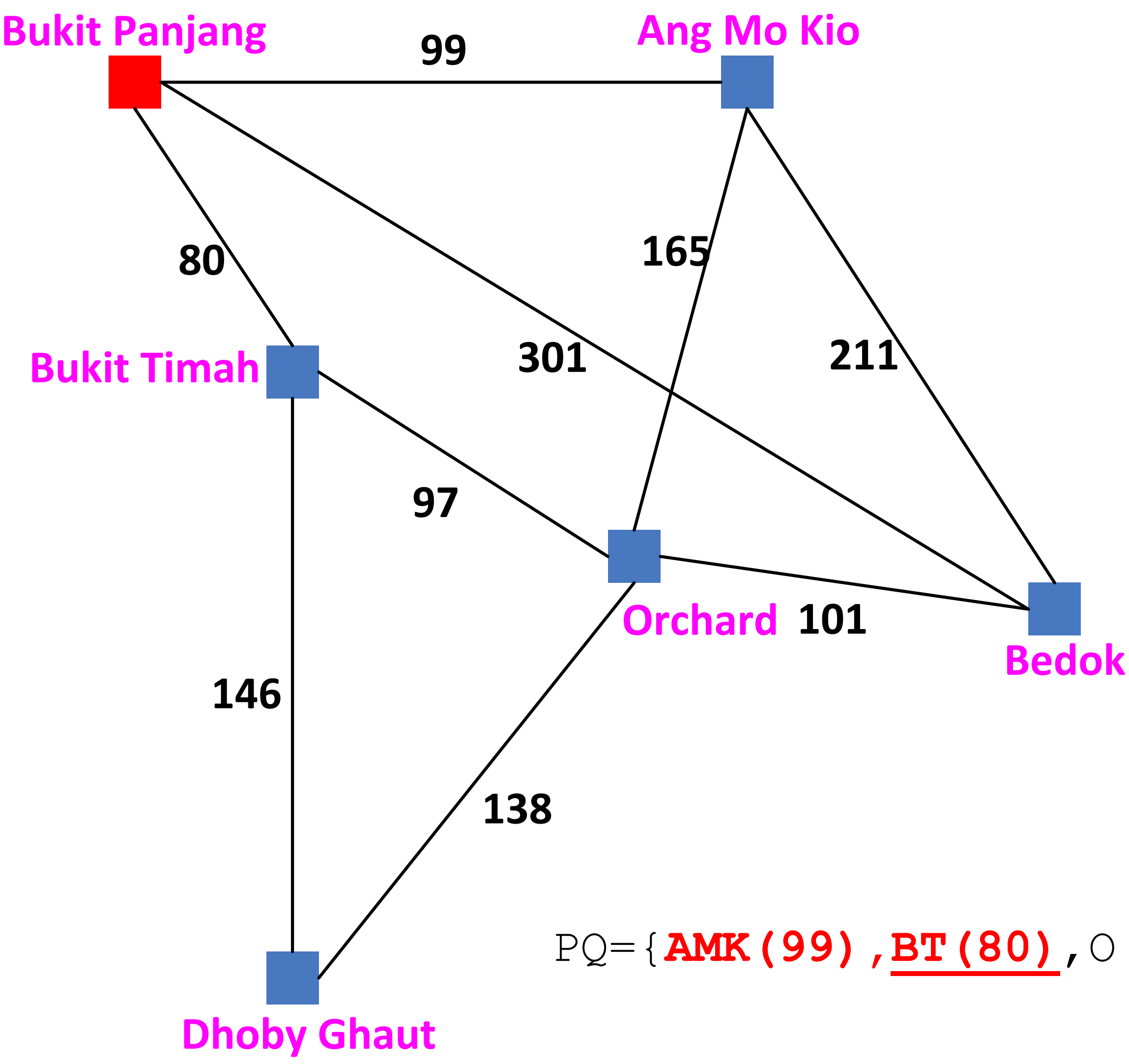
Dijkstra's Algorithm




```
if (dist[BP] + cost(BP, BT) < dist[BT]) {
        dist[v] = 0 + 80;
        previous[BT] = BP;
        PQ.decrease_priority(BT,80);
}
```

PQ={BP(0),AMK(+∞),BT(+∞),O(+∞),B(+∞),DB(+∞)}

Dijkstra's Algorithm

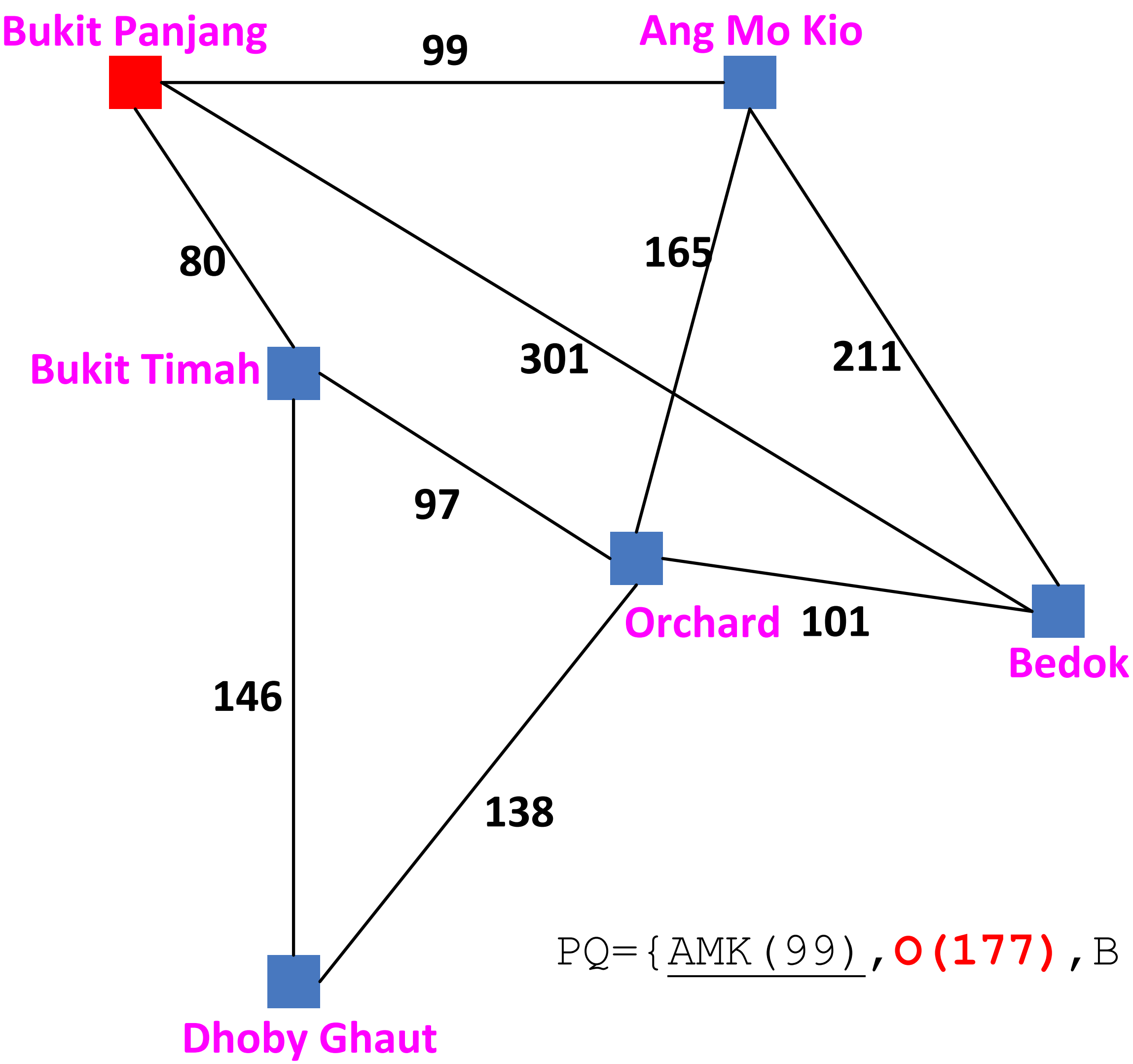



```
previous[BP]  = NA
previous[AMK] = BP
previous[BT]  = BP
previous[O]   = NA
previous[B]   = BP
previous[DG]  = NA
```

```
dist[BP]  = 0
dist[AMK] = 99
dist[BT]  = 80
dist[O]   = +∞
dist[B]   = 301
dist[DG]  = +∞
```

PQ={**AMK(99)**,**<u>BT(80)</u>**,O(+∞),**B(301)**,DB(+∞)}

Dijkstra's Algorithm

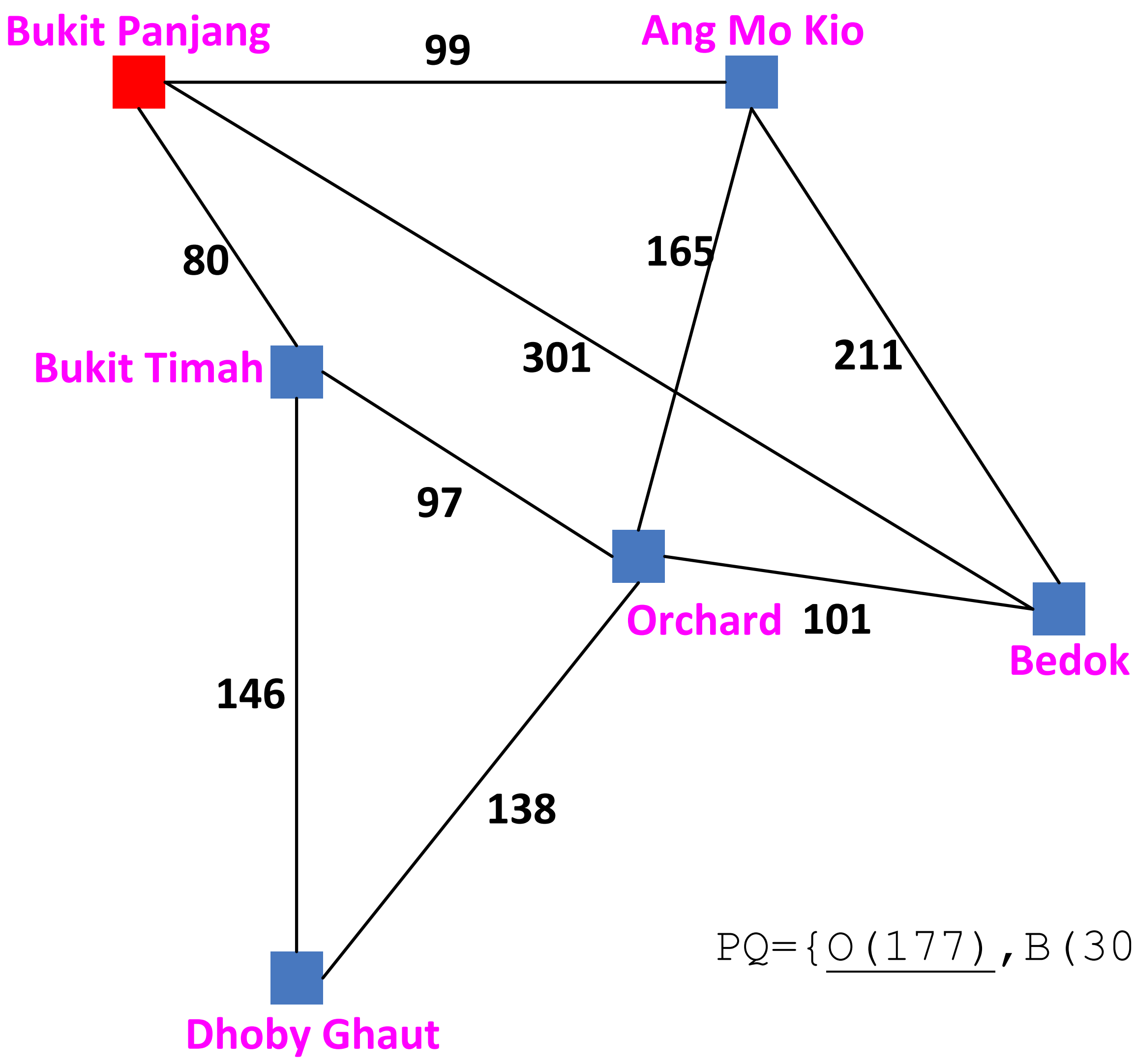



```
previous[BP]   = NA
previous[AMK]  = BP
previous[BT]   = BP
previous[O]    = BT
previous[B]    = BP
previous[DG]   = BT
```

```
dist[BP]   = 0
dist[AMK]  = 99
dist[BT]   = 80
dist[O]    = 177
dist[B]    = 301
dist[DG]   = 226
```

PQ={AMK(99),**O(177)**,B(301),**DB(226)**}

Dijkstra's Algorithm

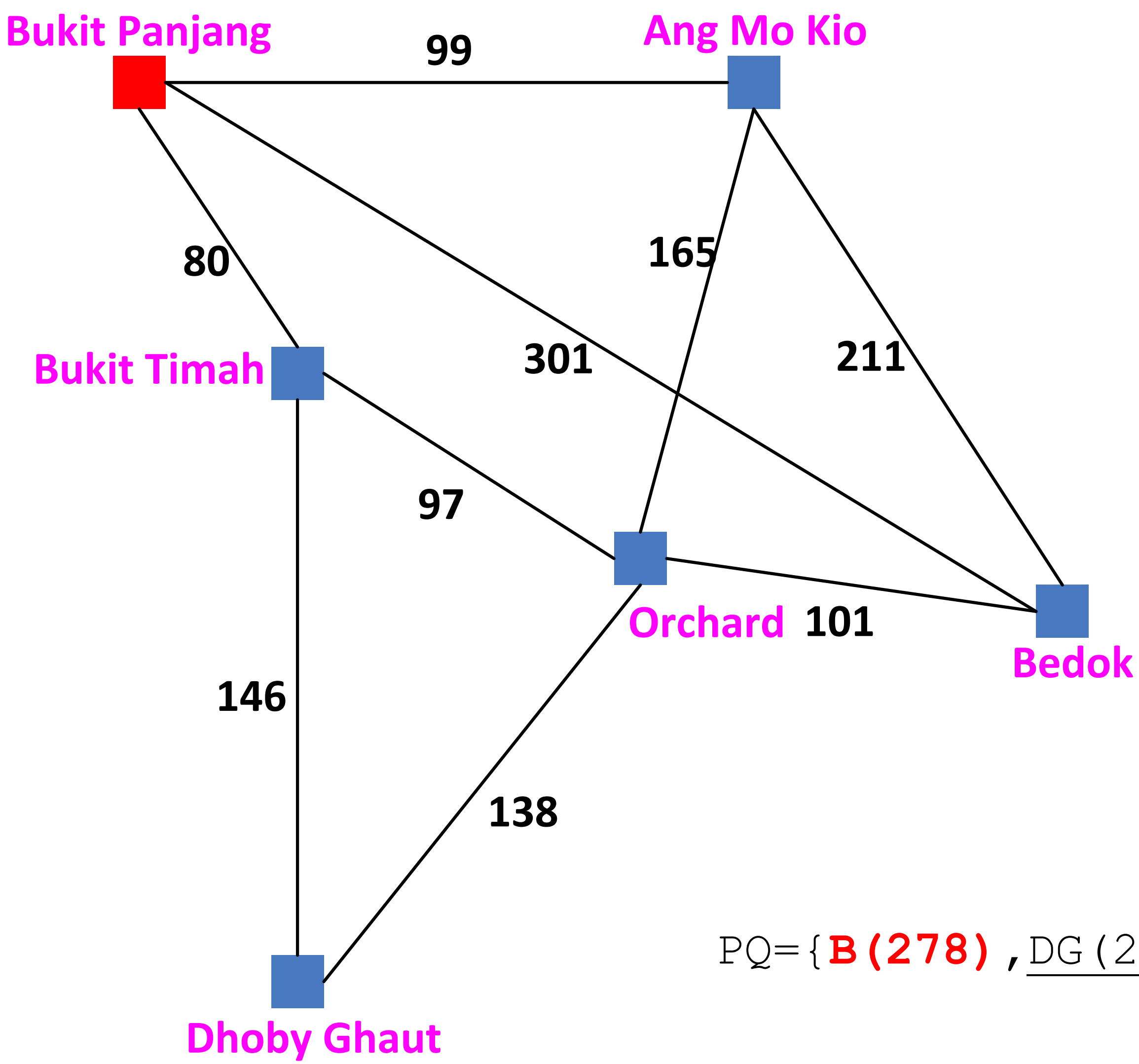



```
previous[BP]  = NA
previous[AMK] = BP
previous[BT]  = BP
previous[O]   = BT
previous[B]   = BP
previous[DG]  = BT
```

```
dist[BP]  = 0
dist[AMK] = 99
dist[BT]  = 80
dist[O]   = 177
dist[B]   = 301
dist[DG]  = 226
```

PQ={<u>O(177)</u>,B(301),DG(226)}

Dijkstra's Algorithm

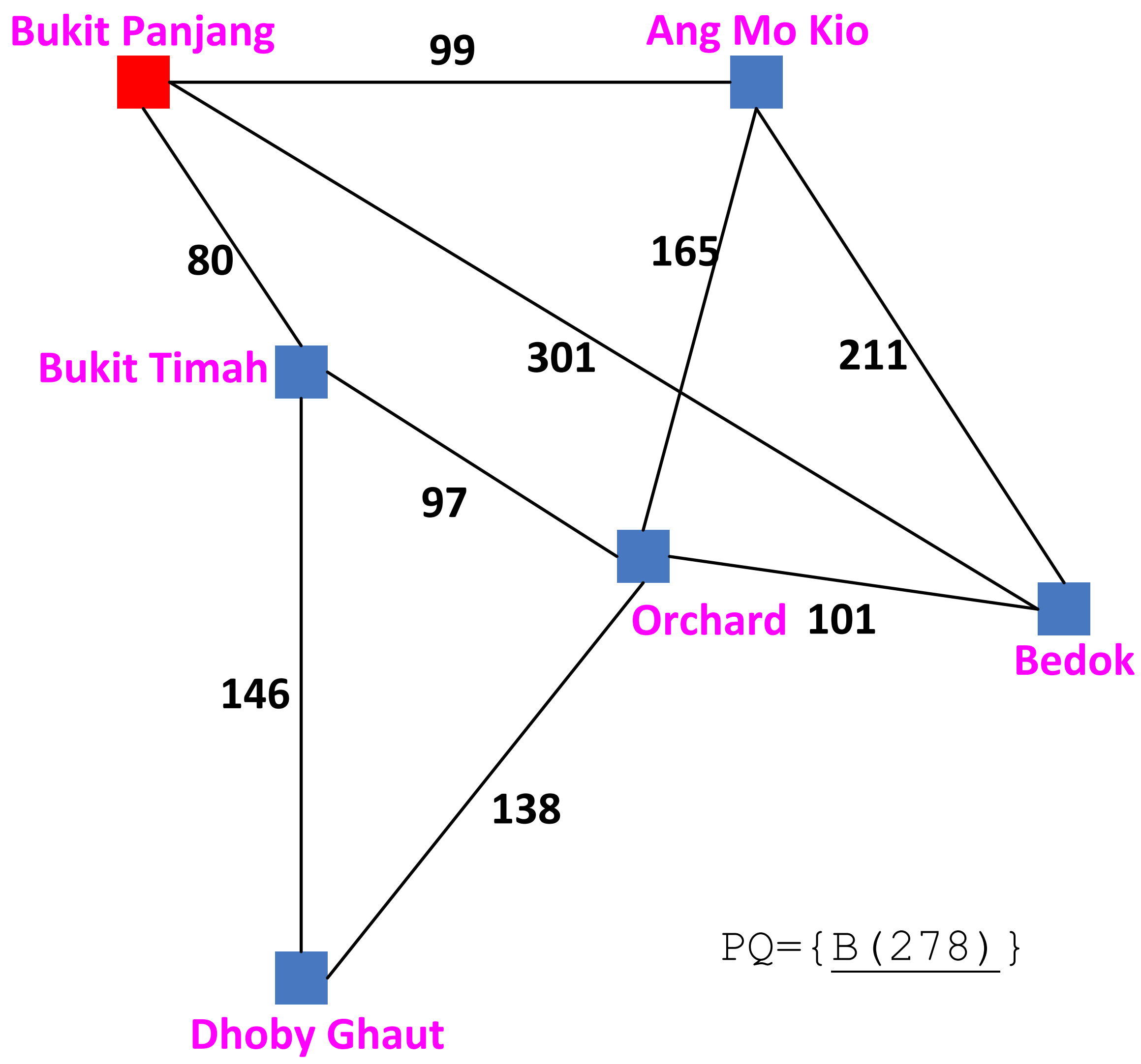



```
previous[BP]  = NA
previous[AMK] = BP
previous[BT]  = BP
previous[O]   = BT
previous[B]   = O
previous[DG]  = BT
```

```
dist[BP]  = 0
dist[AMK] = 99
dist[BT]  = 80
dist[O]   = 177
dist[B]   = 278
dist[DG]  = 226
```

PQ={**B(278)**,DG(226)}

Dijkstra's Algorithm

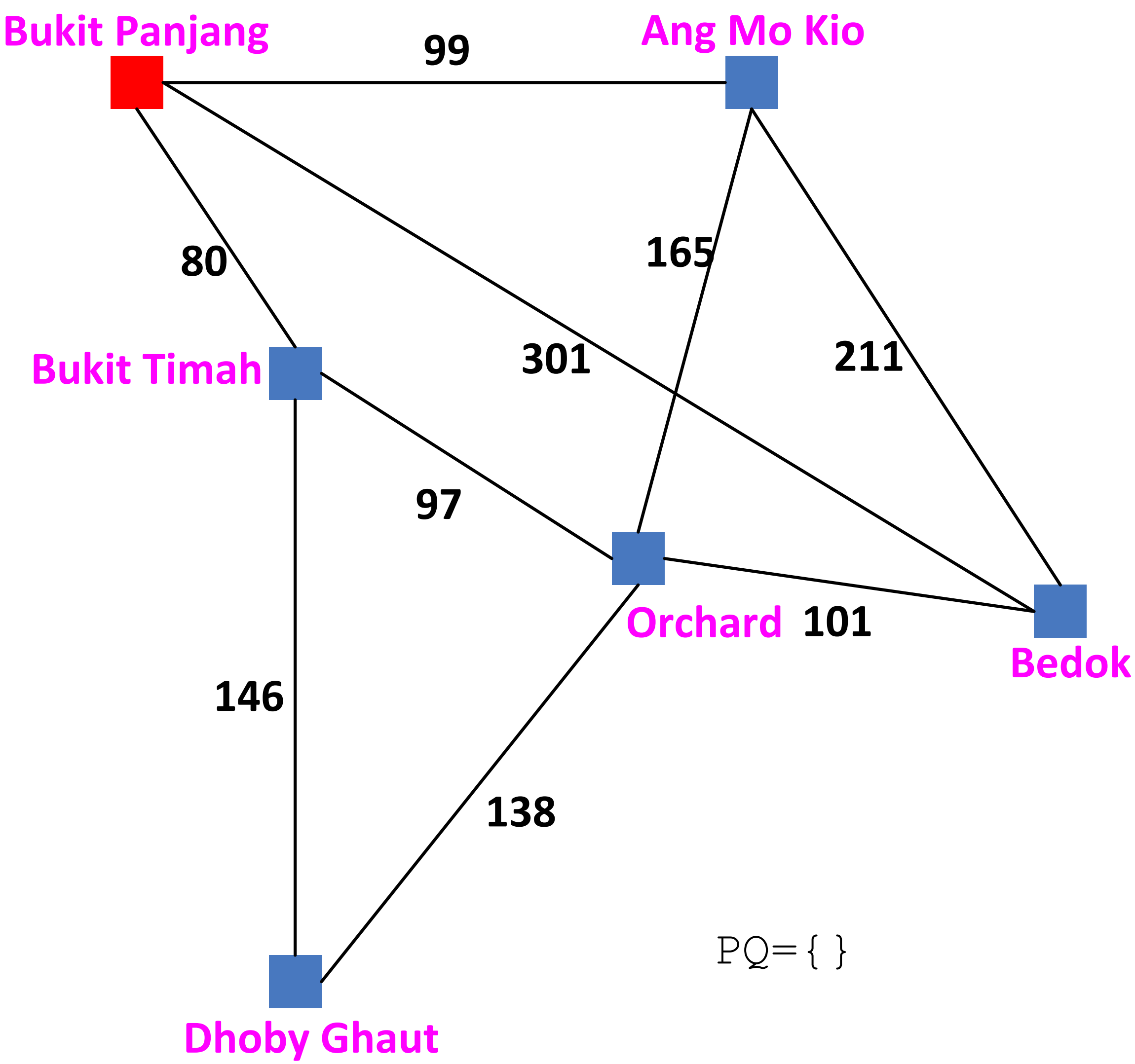



```
previous[BP]  = NA
previous[AMK] = BP
previous[BT]  = BP
previous[O]   = BT
previous[B]   = O
previous[DG]  = BT
```

```
dist[BP]  = 0
dist[AMK] = 99
dist[BT]  = 80
dist[O]   = 177
dist[B]   = 278
dist[DG]  = 226
```

PQ={B(278)}

Dijkstra's Algorithm




```
previous[BP]  = NA
previous[AMK] = BP
previous[BT]  = BP
previous[O]   = BT
previous[B]   = O
previous[DG]  = BT
```

```
dist[BP]  = 0
dist[AMK] = 99
dist[BT]  = 80
dist[O]   = 177
dist[B]   = 278
dist[DG]  = 226
```

PQ={}

Dijkstra's Algorithm

- Time Complexity in terms of number of vertices: n and number of edges: m
  - $O(n) \times T_{Extract\text{-}Min} + O(m) \times T_{Decrease\text{-}Key}$

Data Structure	$T_{Extract\text{-}Min}$	$T_{Decrease\text{-}Key}$	Total
Array			
Binary Heap			

Dijkstra's Algorithm

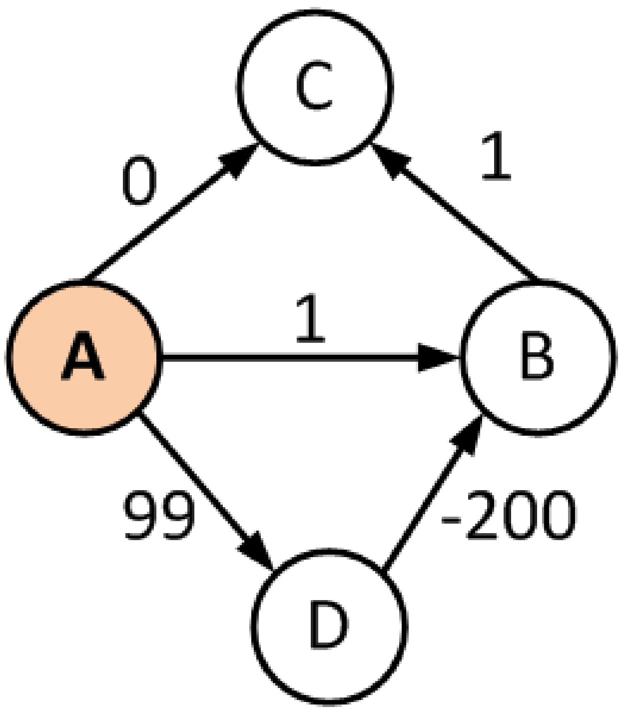
- Time Complexity in terms of number of vertices: n and number of edges: m
  - $O(n) \times T_{Extract\_Min} + O(m) \times T_{Decrease\_Key}$

Data Structure	$T_{Extract\text{-}Min}$	$T_{Decrease\text{-}Key}$	Total
Array	$O(n)$	$O(1)$	$O(n^2)$
Binary Heap	$O(\log(n))$	$O(\log(n))$	$O(m \log(n))$

- Space complexity: $O(n)$

Dijkstra's Algorithm

- Note: it **may** not work on graph with negative cost values.



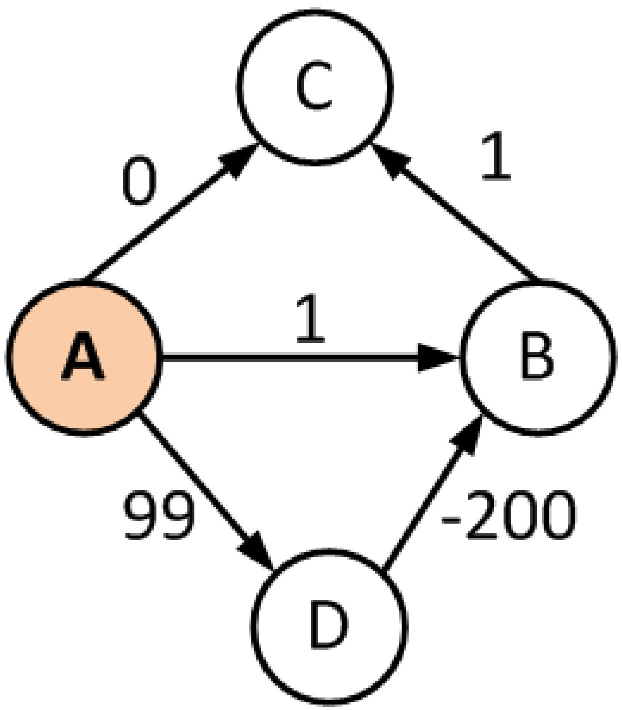

```
dist[A] = 0
dist[B] = +∞
dist[C] = +∞
dist[D] = +∞

previous[A] = NA
previous[B] = NA
previous[C] = NA
previous[D] = NA

PQ={A(0),B(+∞),C(+∞),D(+∞)}
```

Dijkstra's Algorithm

- Note: it **may** not work on graph with negative cost values.



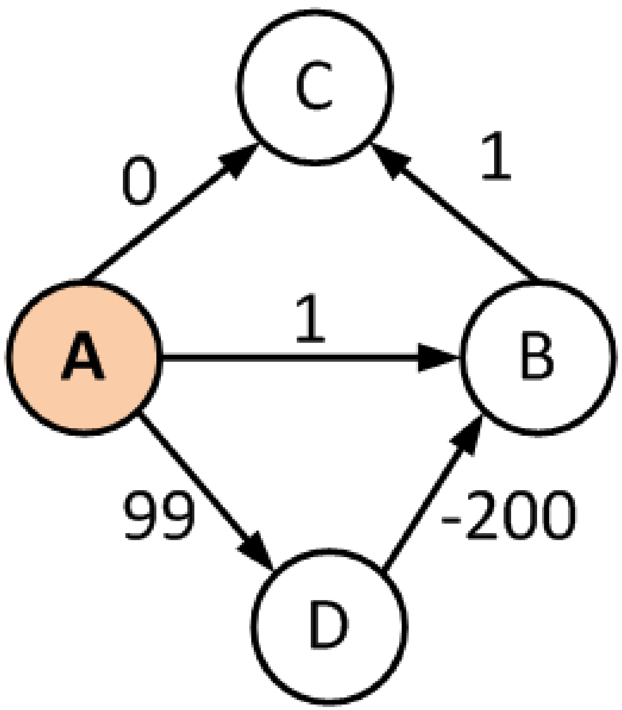

```
dist[A] = 0
dist[B] = 1
dist[C] = 0
dist[D] = 99

previous[A] = NA
previous[B] = A
previous[C] = A
previous[D] = A

PQ={B(1),C(0),D(99)}
```

Dijkstra's Algorithm

- Note: it **may** not work on graph with negative cost values.



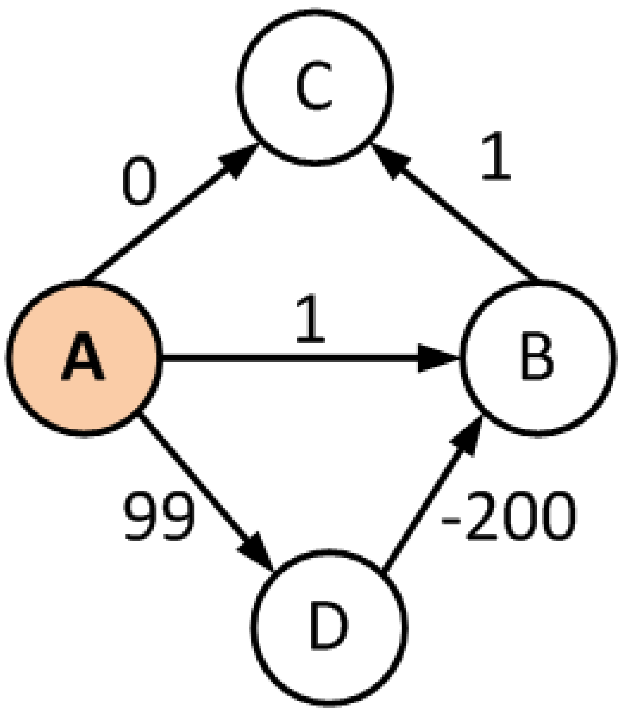

```
dist[A] = 0
dist[B] = 1
dist[C] = 0
dist[D] = 99

previous[A] = NA
previous[B] = A
previous[C] = A
previous[D] = A

PQ={B(1),D(99)}
```

Dijkstra's Algorithm

- Note: it **may** not work on graph with negative cost values.



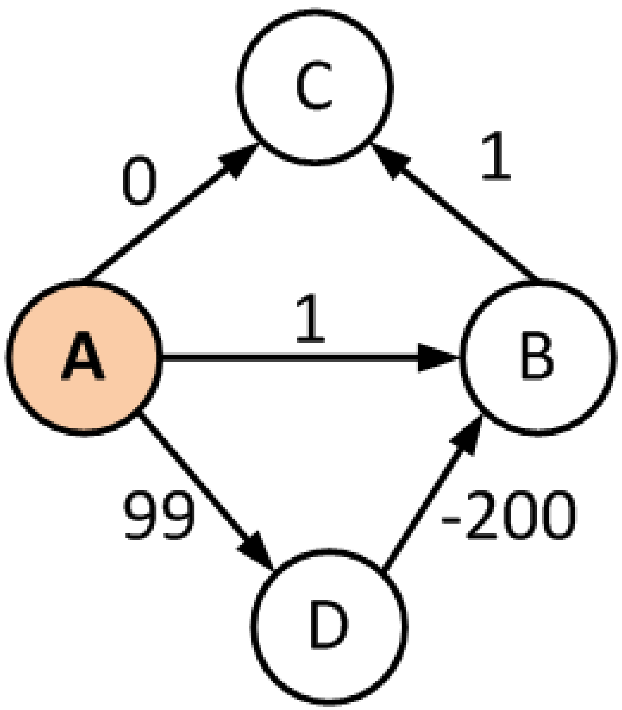

```
dist[A] = 0
dist[B] = 1
dist[C] = 0
dist[D] = 99

previous[A] = NA
previous[B] = A
previous[C] = A
previous[D] = A

PQ={D(99)}
```

Dijkstra's Algorithm

- Note: it **may** not work on graph with negative cost values.



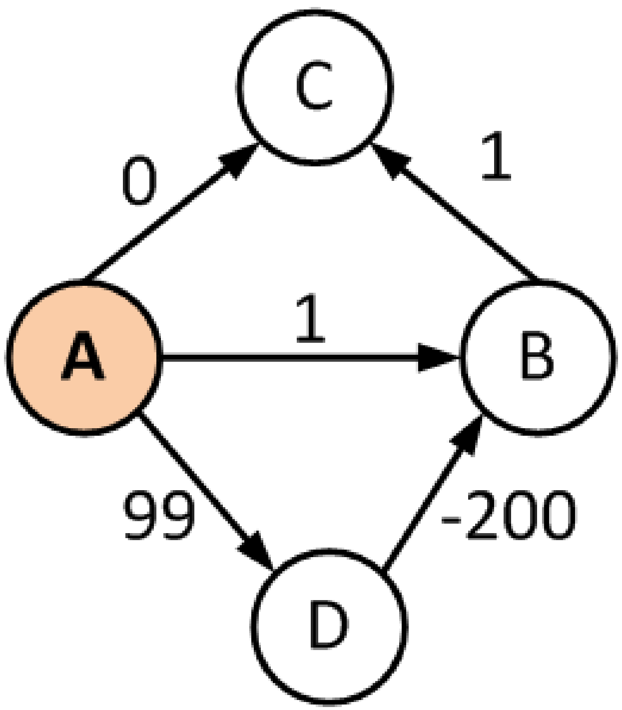

```
dist[A] = 0
dist[B] = -101
dist[C] = 0
dist[D] = 99

previous[A] = NA
previous[B] = D
previous[C] = A
previous[D] = A

PQ={}
```

Dijkstra's Algorithm

- Note: it **may** not work on graph with negative cost values.




```
dist[A] = 0
dist[B] = -101
dist[C] = 0
```
dist[C] should be -100
```
dist[D] = 99

previous[A] = NA
previous[B] = D
previous[C] = A
previous[D] = A

PQ={}
```