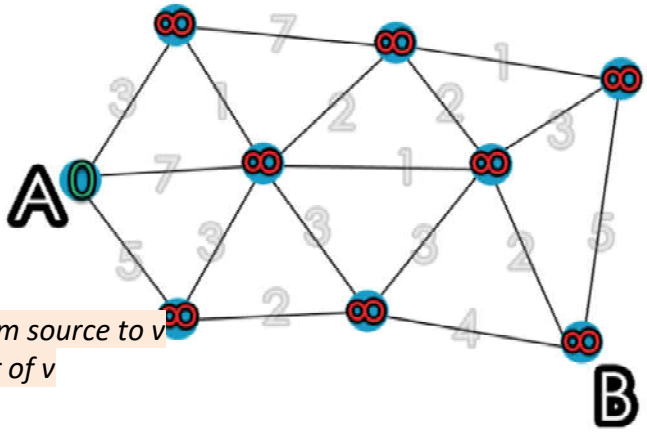
Dijkstra + Fibonacci Heap Pseudo Code :

Algorithm Pseudo Code




```
1. function Dijkstra(Graph, source):
2.     dist[source] ← 0                          // Initialization
3.
4.     create vertex priority queue Q

5.     for each vertex v in Graph:
6.         if v ≠ source
7.             dist[v] ← INFINITY                // Unknown distance from source to v
8.         prev[v] ← UNDEFINED                   // Predecessor of v

9.         Q.add_with_priority(v, dist[v])

10.Q := the set of all nodes in Graph  // all nodes in the graph are unoptimized(thus are in Priority Queue Q)

11.    while Q is not empty:                     // The main loop
12.        u ← Q.extract_min()                   // Remove and return best vertex
13.        for each neighbor v of u:             // only v that are still in Q :  has not yet been removed from Q.
14.            alt ← dist[u] + dist_between(u, v)
15.            if alt < dist[v]                  // Relax (u,v)
16.                dist[v] ← alt
17.                prev[v] ← u
18.                Q.decrease_priority(v, alt)

19.    return dist, prev
```

From <https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm>
From <http://www.gitta.info/Accessibiliti/en/html/Dijkstra_learningObject1.html>

Reading the shortest path from source to target by reverse iteration (Using Stack):

```
1  S ← empty sequence
2  u ← target
3  if prev[u] is defined or u = source:        // Do something only if the vertex is reachable
4      while u is defined:                     // Construct the shortest path with a stack S
5          insert u at the beginning of S      // Push the vertex onto the stack
6          u ← prev[u]                         // Traverse from target to source
```

From <https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm>