Shortest Path Problem (4A)

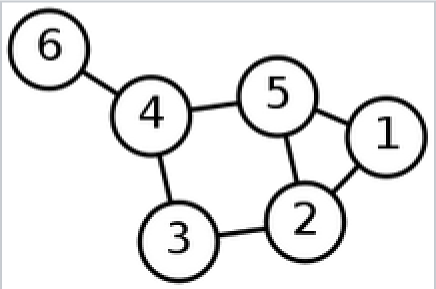
Young Won Lim
5/11/18

Please send corrections (or suggestions) to youngwlim@hotmail.com.

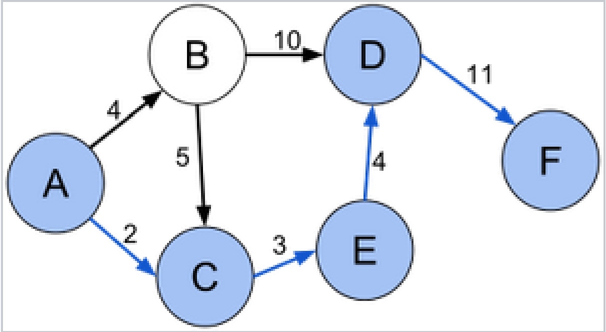
This document was produced by using LibreOffice and Octave.

Shortest Path Problem

the shortest path problem is the problem of finding a path between two vertices (or nodes) in a graph such that the sum of the weights of its constituent edges is minimized.



(6, 4, 5, 1) and (6, 4, 3, 2, 1) are both paths between vertices 6 and 1



Shortest path (A, C, E, D, F) between vertices A and F in the weighted directed graph

https://en.wikipedia.org/wiki/Shortest_path_problem

Types of Shortest Path Problems

The **single-pair shortest path problem:**
to find shortest paths from a **source** vertex v to a **destination** vertex w in a graph

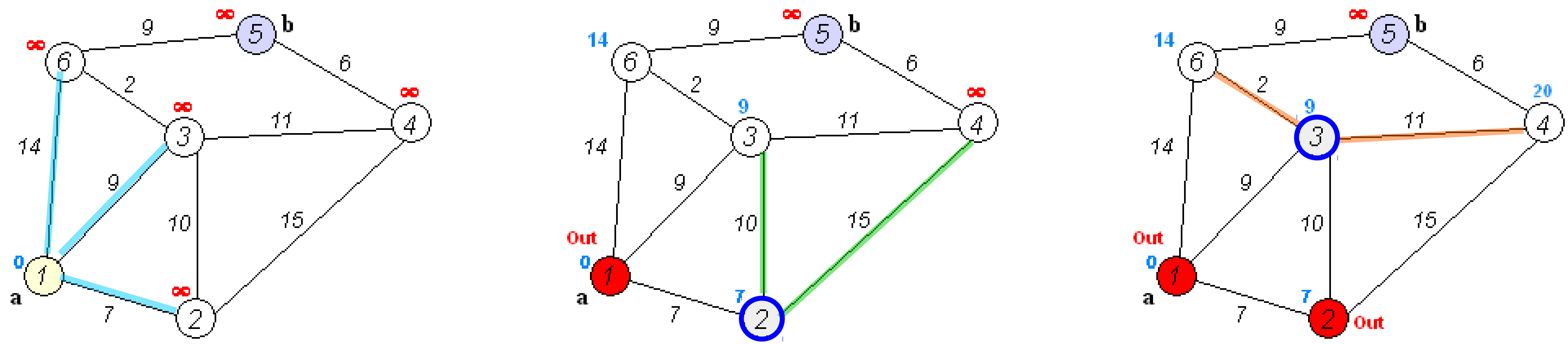
The **single-<u>source</u> shortest path problem:**
to find shortest paths from a **source** vertex v to **all** other vertices in the graph.

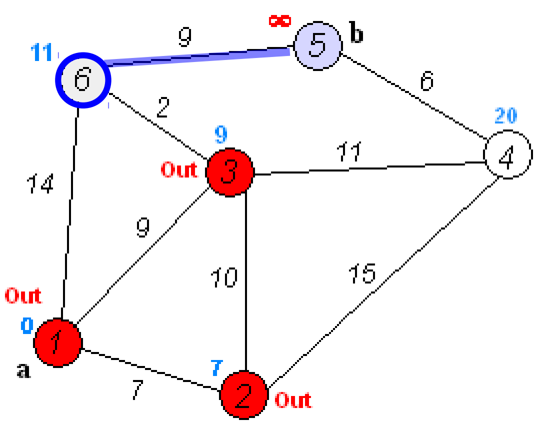
The **single-<u>destination</u> shortest path problem:**
to find shortest paths from **all** vertices in the directed graph to a single **destination** vertex v. This can be reduced to the single-source shortest path problem by reversing the arcs in the directed graph.

The **all-pairs shortest path problem:**
to find shortest paths between every **pair** of vertices v, v' in the graph.

https://en.wikipedia.org/wiki/Shortest_path_problem

Dijkstra’s Algorithm Example Summary

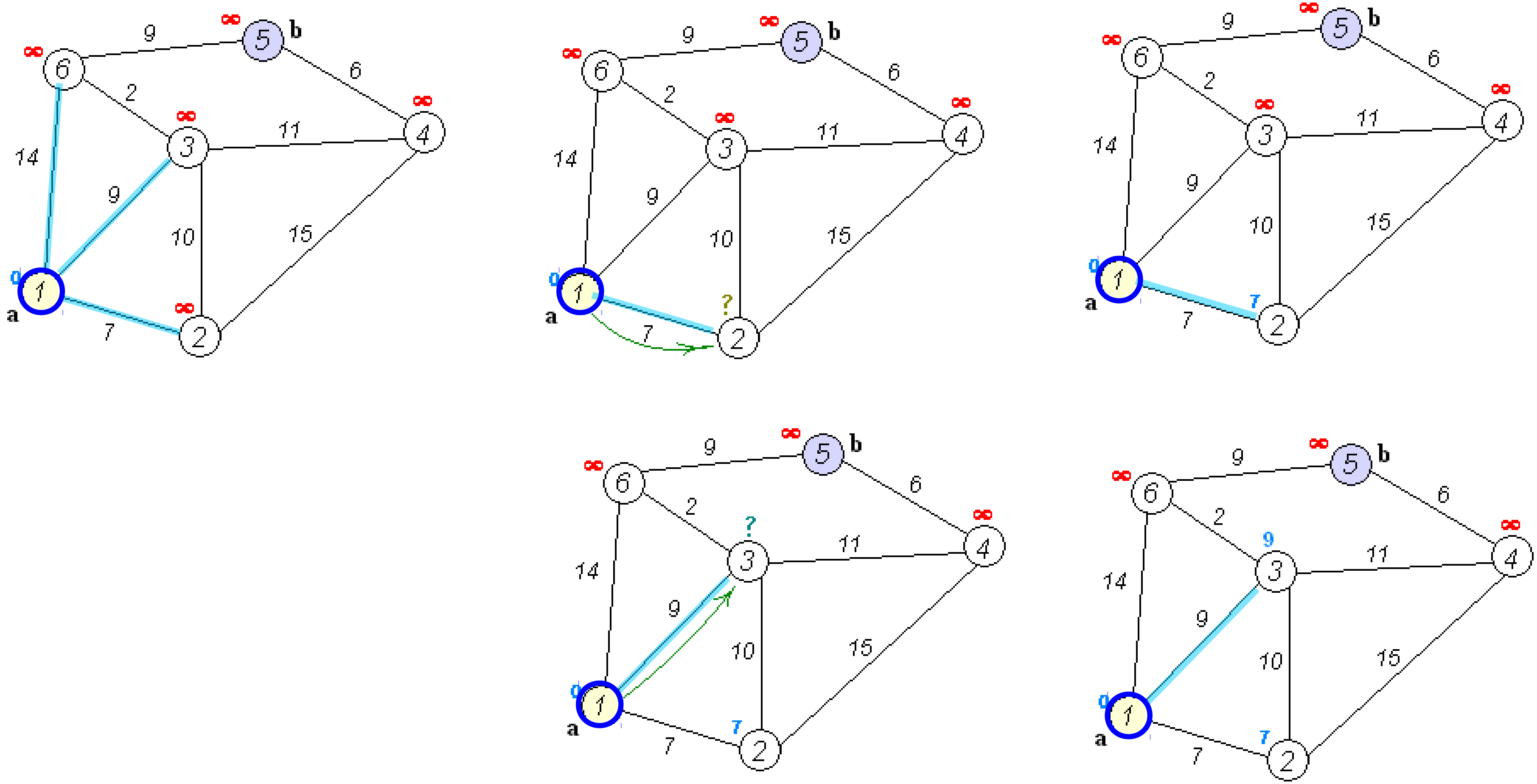




- the initial node
- the current node
- the visited nodes

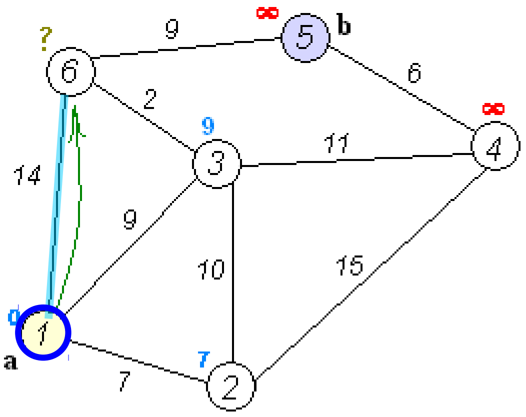
https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#/media/File:Dijkstra_Animation.gif

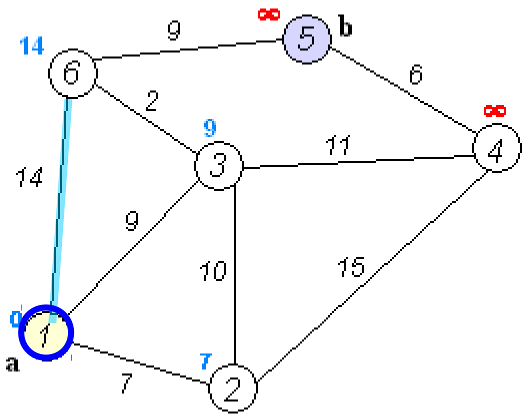
Dijkstra’s Algorithm Example (1)

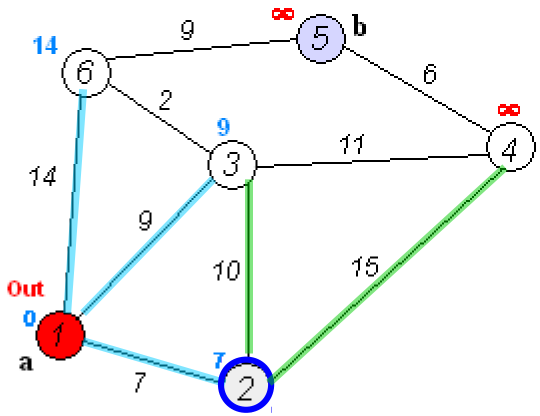


https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#/media/File:Dijkstra_Animation.gif

Dijkstra’s Algorithm Example (2)

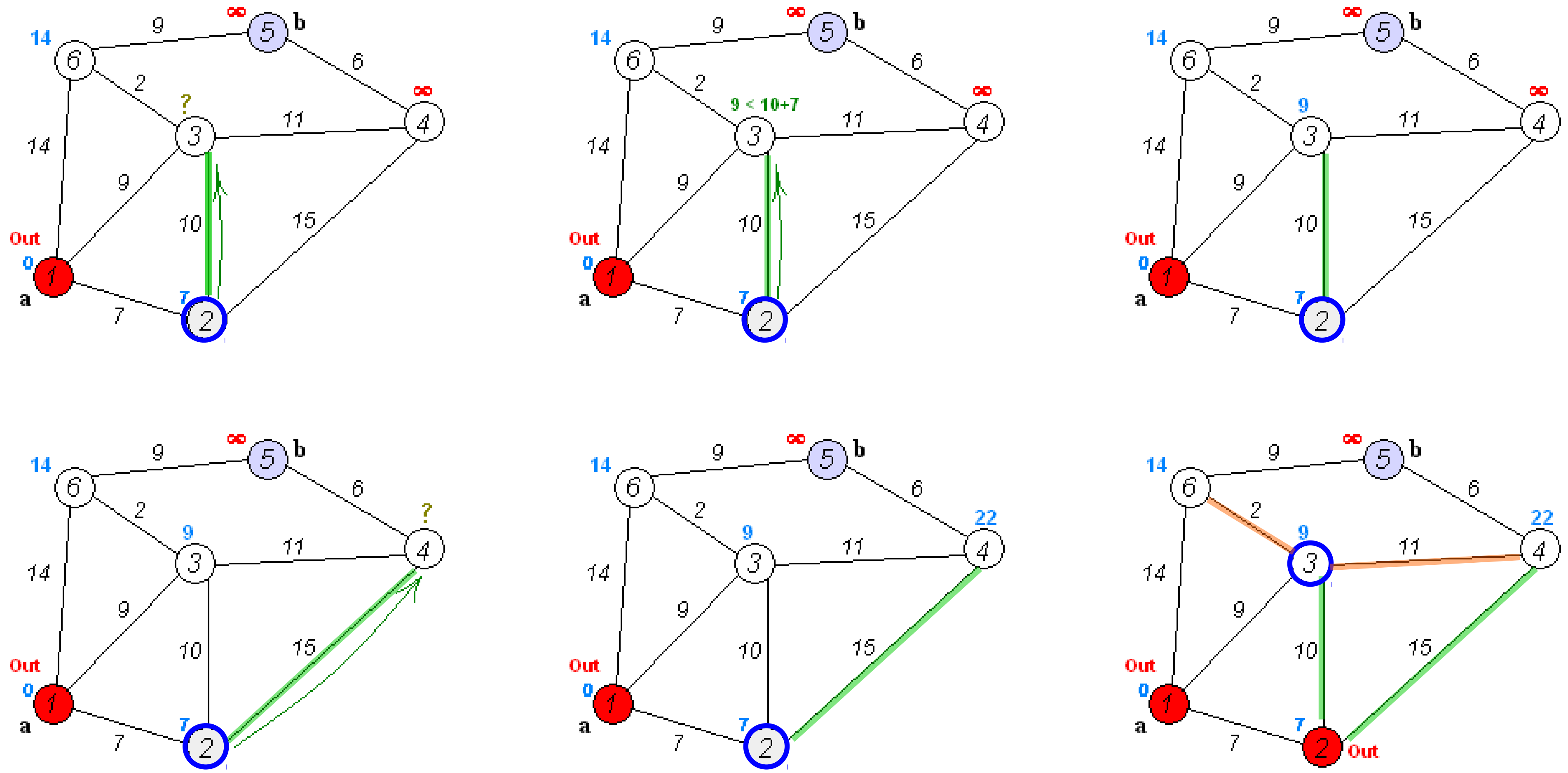






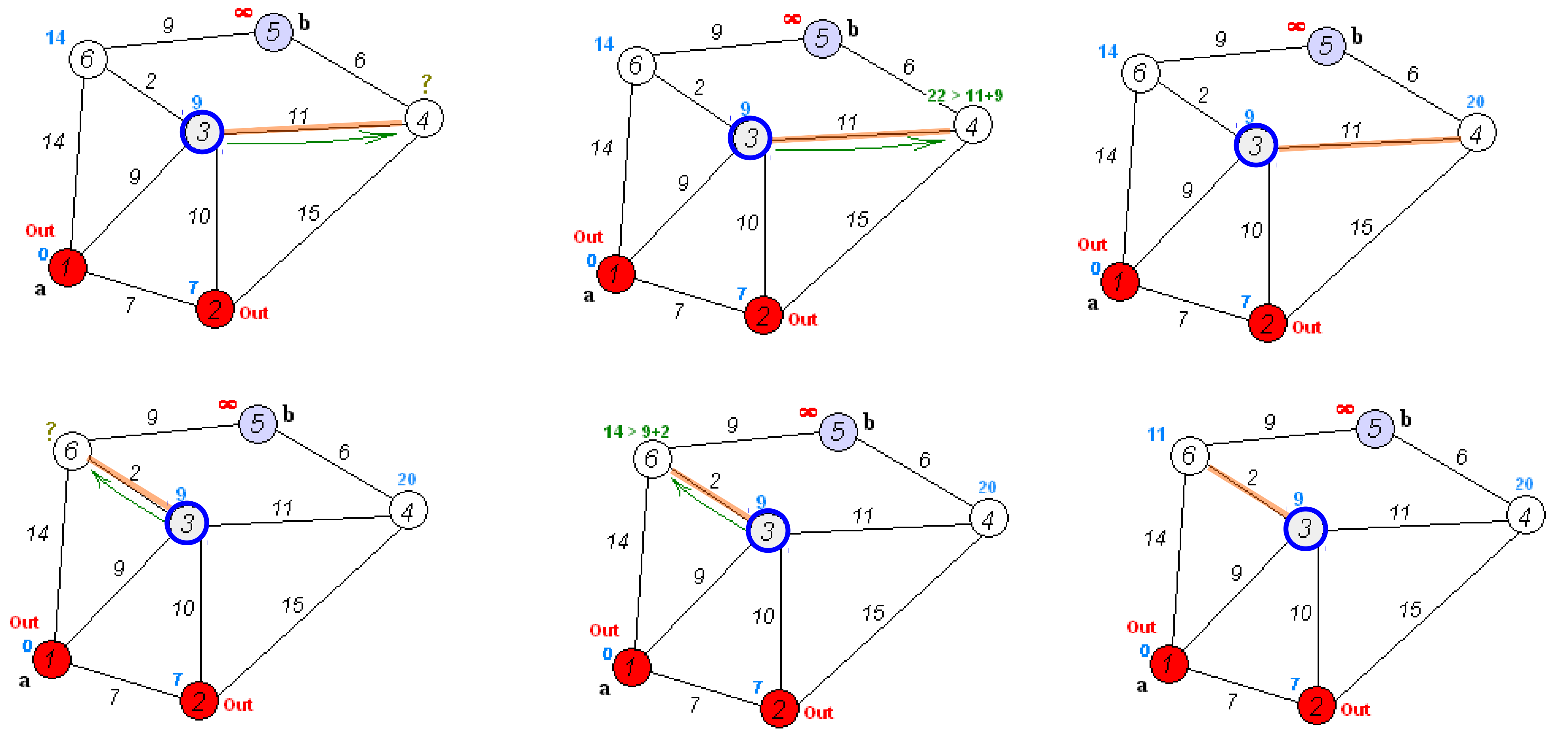
https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#/media/File:Dijkstra_Animation.gif

Dijkstra's Algorithm Example (3)



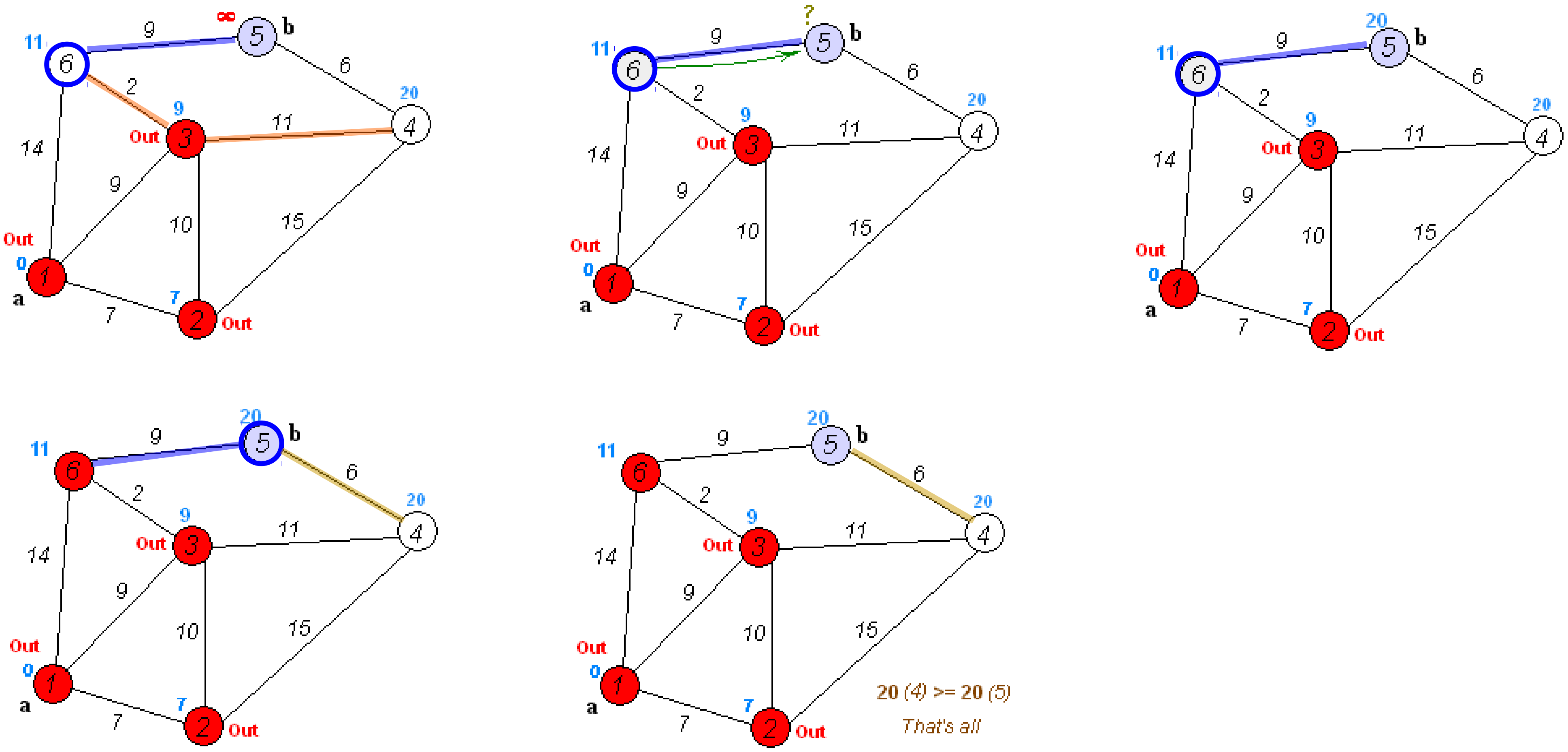
https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#/media/File:Dijkstra_Animation.gif

Dijkstra’s Algorithm Example (4)



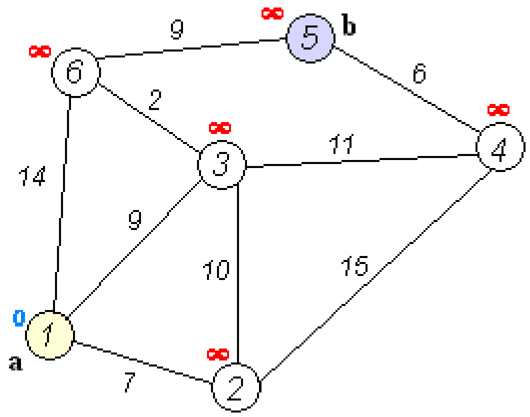
https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#/media/File:Dijkstra_Animation.gif

Dijkstra’s Algorithm Example (5)



https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#/media/File:Dijkstra_Animation.gif

Hamiltonian Cycles



https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#/media/File:Dijkstra_Animation.gif

Dijkstra’s Algorithm (1)

Let the node at which we are starting
be called the **initial node**.
Let the **distance** of node Y be
the **distance** from the **initial node** to Y.
Dijkstra's algorithm will assign some **initial distance values** and will try to improve them step by step.

1. Mark all nodes **unvisited**.
Create a set of all the unvisited nodes called the **unvisited set**.

2. Assign to every node a **tentative distance value**:
set it to **zero** for our initial node and
to **infinity** for all other nodes.
Set the **initial node** as **current**.

https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#/media/File:Dijkstra_Animation.gif

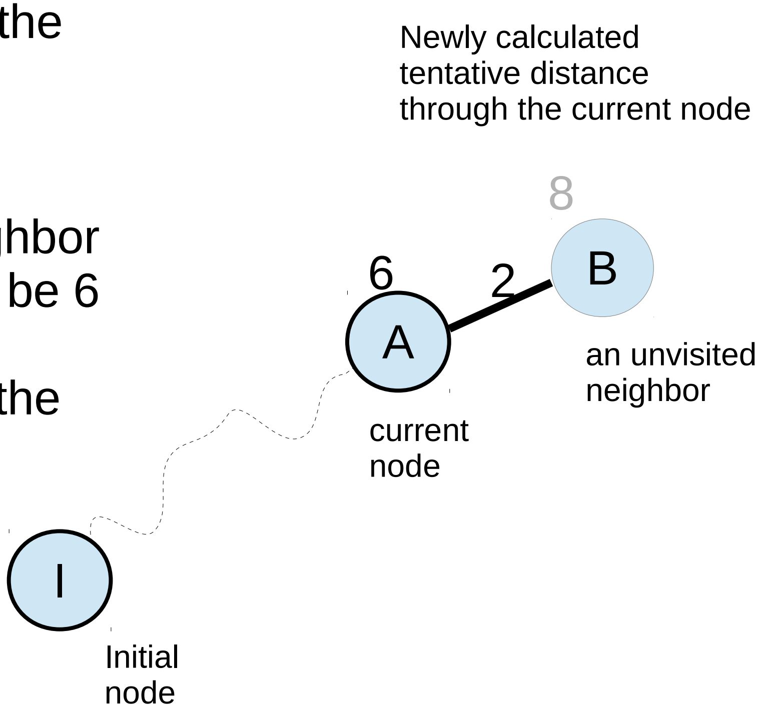
Dijkstra's Algorithm (2)

3. Remove the **current node** from the **unvisited set**

For all the **unvisited neighbors** of the **current node**, calculate their **tentative distances** through the **current** node.

Compare the newly calculated tentative distance to the current assigned value and assign the smaller one.

For example, if the current node A is marked with a distance of 6, and the edge connecting it with a neighbor B has length 2, then the distance to B through A will be 6 + 2 = 8. If B was previously marked with a distance greater than 8 then change it to 8. Otherwise, keep the current value.



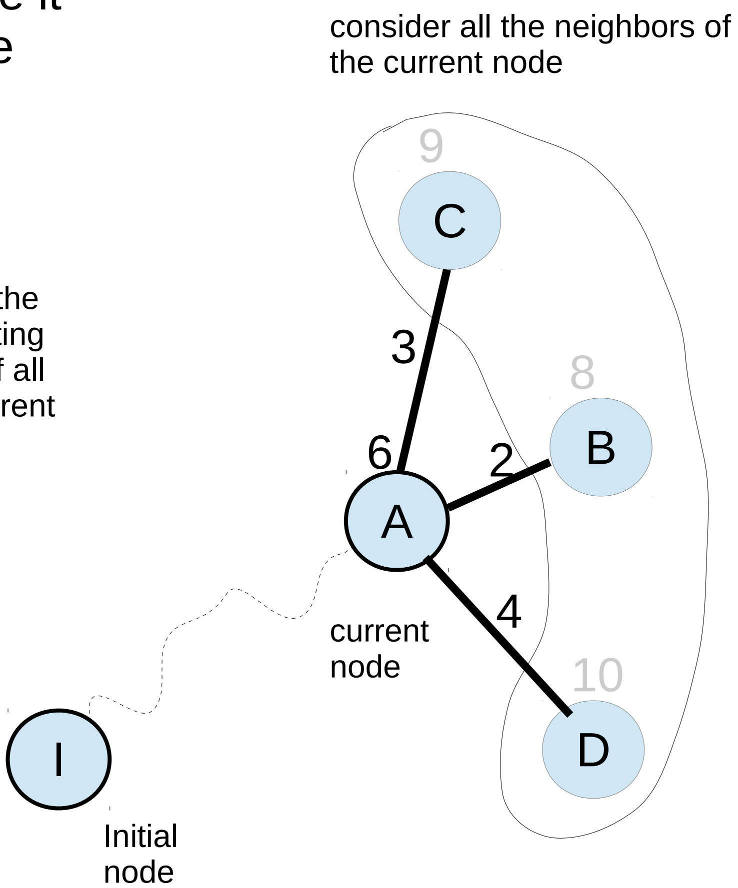
https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#/media/File:Dijkstra_Animation.gif

Dijkstra's Algorithm (3)

4. After considering <u>all</u> of the **neighbors** of the **current node**, mark the **current** node as **visited** and remove it from the **unvisited set**. A **visited node** will never be checked again.

current node : chosen node with the <u>smallest</u> tentative distance from the **unvisited set**

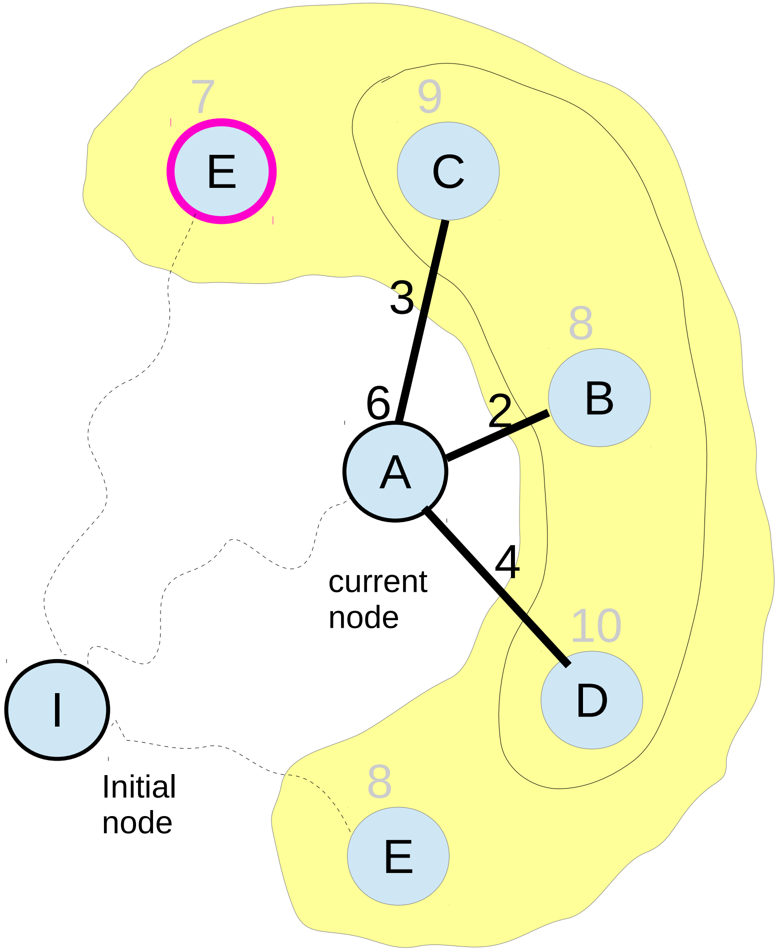
current node : move to the **visited** set, after calculating the tentative distances of all the **neighbors** of the current node



https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#/media/File:Dijkstra_Animation.gif

Dijkstra's Algorithm (4)

5. Move to the **next unvisited node** with the smallest tentative distances and repeat the above steps which check neighbors and mark visited.



https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#/media/File:Dijkstra_Animation.gif

Dijkstra's Algorithm (5)

5-a. If the **destination** node has been marked **visited**
(when planning a route between two specific nodes)

or if the smallest tentative distance among the nodes in the unvisited set is **infinity** (when planning a complete traversal; occurs when there is no connection between the initial node and remaining unvisited nodes),

then stop. The algorithm has finished.

5-b. Otherwise, select the **unvisited** node that is marked with the <u>smallest</u> tentative distance,
set it as the new **current node**, and go back to step 3.

https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#/media/File:Dijkstra_Animation.gif

Dijkstra's Algorithm – Pseudocode 1

```
function Dijkstra(Graph, source):

    create vertex set Q

    for each vertex v in Graph:            // Initialization
        dist[v] ← INFINITY                 // Unknown distance from source to v
        prev[v] ← UNDEFINED                // Previous node in optimal path from source
        add v to Q                         // All nodes initially in Q (unvisited nodes)

    dist[source] ← 0                       // Distance from source to source

    while Q is not empty:
        u ← vertex in Q with min dist[u]   // Node with the least distance
                                           // will be selected first
        remove u from Q

        for each neighbor v of u:          // where v is still in Q.      for each v in Q:
            alt ← dist[u] + length(u, v)
            if alt < dist[v]:              // A shorter path to v has been found
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```

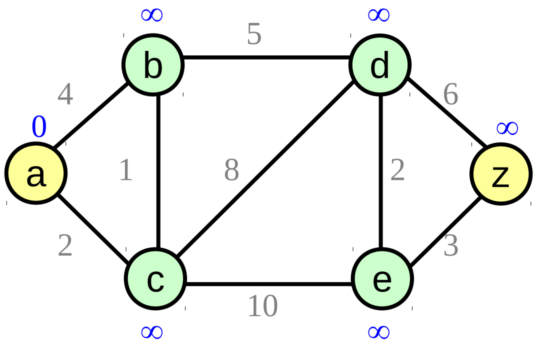
https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#/media/File:Dijkstra_Animation.gif

Dijkstra's Algorithm – Pseudocode 2

Procedure Dijkstra(**G**: weighted connected simple graph, with all positive weights)
{**G** has vertices $a = v_0, v_1, \ldots, v_n = z$ and length $w(v_i, v_j)$
where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in **G**}
for i := 1 to n
    L(v_i) := ∞
L(a) := 0
S := { }
{the labels are now initialized so that the label of a is 0 and
All other labels are ∞, and S is the empty set}
while z ∉ S
    u := a vertex not in S with L(u) minimal
    S := S ∪ {u}
    for all vertices v not in S
        if L(u) +w(u,v) < L(u) then L(v) := L(u) + w(u,v)
        {this adds a vertex to S with minimal label and
        updates the labels of vertices not in S}
return L(z) {L(z) = length of a shortest path from a to z}

Discrete Mathematics and It's Applications, K. H. Rosen

Dijkstra Algorithm Pseudocode 2 Example (0)

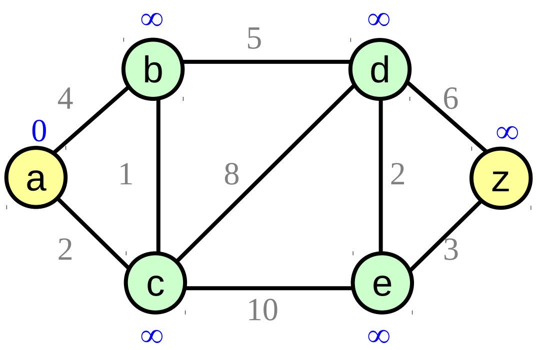


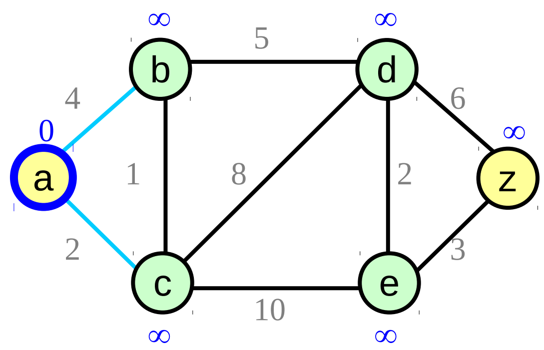
$$
\begin{array}{c c}
 & \begin{array}{cccccc} a & b & c & d & e & z \end{array} \\
\begin{array}{c} a \\ b \\ c \\ d \\ e \\ z \end{array} &
\begin{bmatrix}
\infty & 4 & 2 & \infty & \infty & \infty \\
4 & \infty & 1 & 5 & \infty & \infty \\
2 & 1 & \infty & 8 & 10 & \infty \\
\infty & 5 & 8 & \infty & 2 & 6 \\
\infty & \infty & 10 & 8 & \infty & 3 \\
\infty & \infty & \infty & 6 & 3 & \infty
\end{bmatrix}
\end{array}
$$

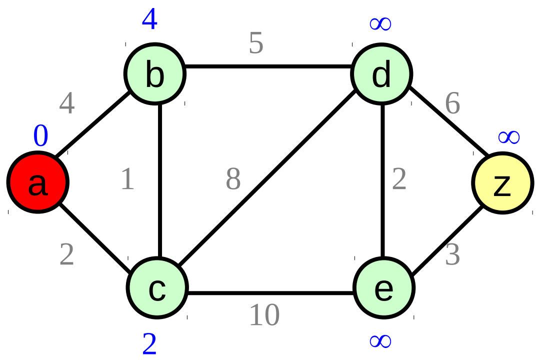
$$w(u_i, u_j)$$

Discrete Mathematics and It's Applications, K. H. Rosen

Dijkstra Algorithm Pseudocode 2 Example (1)







$S = \{a\}$

$L(a) + w(a, b) = 0 + 4 \; < \; L(b) = \infty$

$L(a) + w(a, c) = 0 + 2 \; < \; L(c) = \infty$

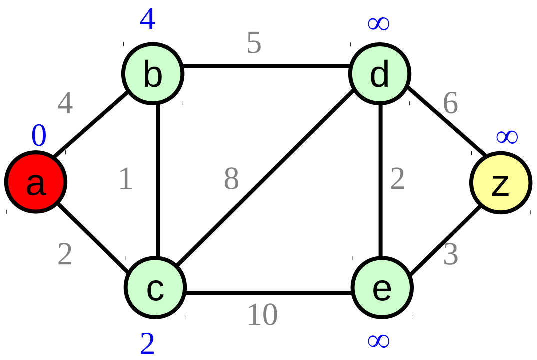
$L(a) + w(a, d) = 0 + \infty \; = \; L(d) = \infty$

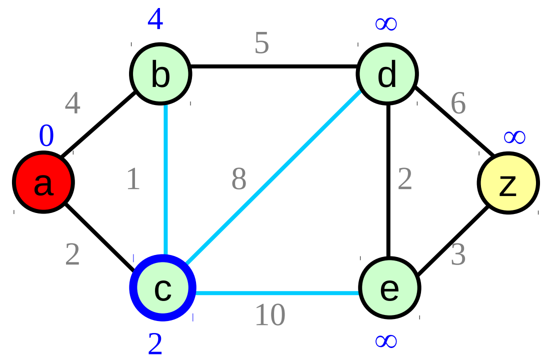
$L(a) + w(a, e) = 0 + \infty \; = \; L(e) = \infty$

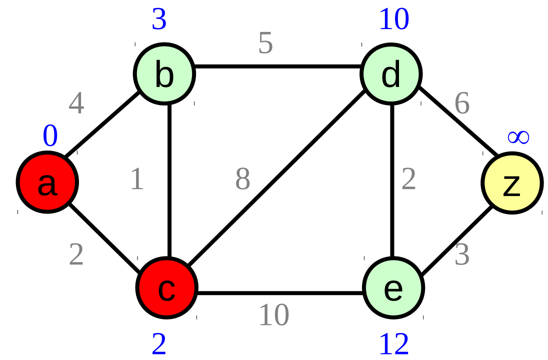
$L(a) + w(a, z) = 0 + \infty \; = \; L(z) = \infty$

Discrete Mathematics and It's Applications, K. H. Rosen

Dijkstra Algorithm Pseudocode 2 Example (2)







$S = \{a, c\}$

$L(c) + w(c, b) = 2 + 1 < L(b) = 4$ $\quad P(a, c, b) < P(a, b)$

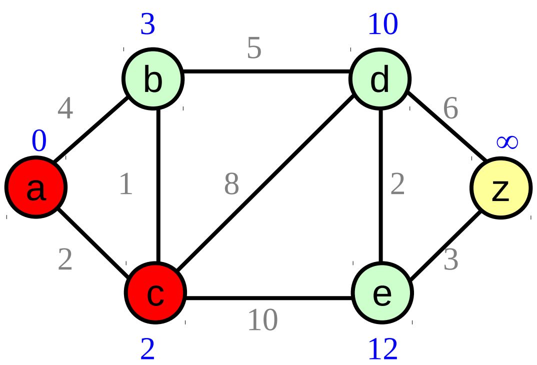
$L(c) + w(c, d) = 2 + 8 < L(d) = \infty$

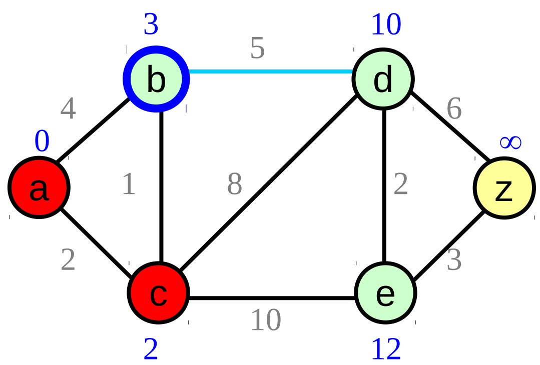
$L(c) + w(c, e) = 2 + 10 < L(e) = \infty$

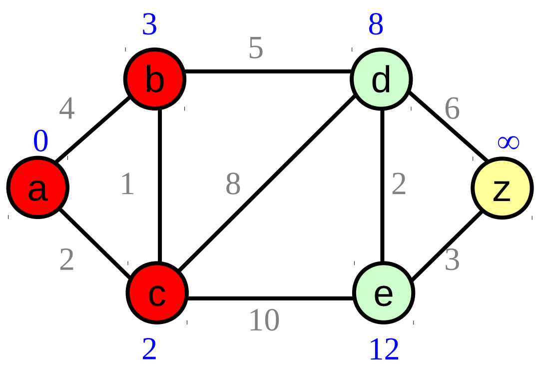
$L(c) + w(c, z) = 2 + \infty = L(z) = \infty$

Discrete Mathematics and It's Applications, K. H. Rosen

Dijkstra Algorithm Pseudocode 2 Example (3)







$S = \{a, c, b\}$

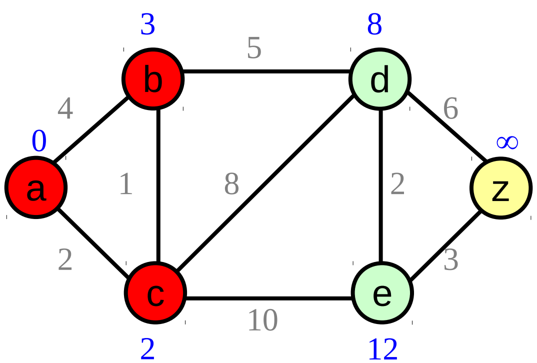
$L(b) + w(b,d) = 3 + 5 < L(d) = 10$ $\qquad$ $P(a,c,b,d) < P(a,c,d)$

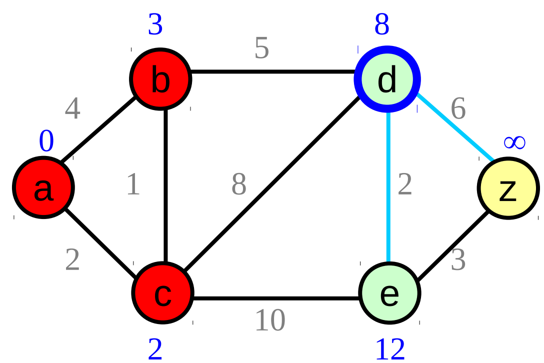
$L(b) + w(b,e) = 3 + \infty > L(e) = 12$

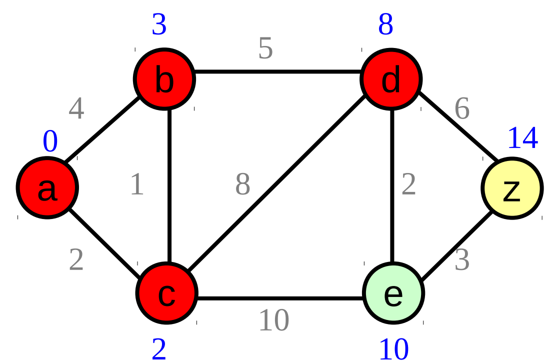
$L(b) + w(b,z) = 3 + \infty = L(z) = \infty$

Discrete Mathematics and It's Applications, K. H. Rosen

Dijkstra Algorithm Pseudocode 2 Example (4)







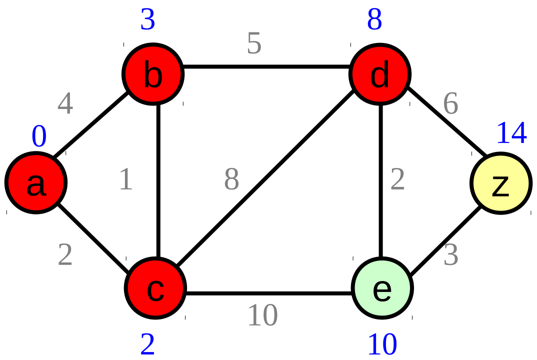
$S = \{a, c, b, d\}$

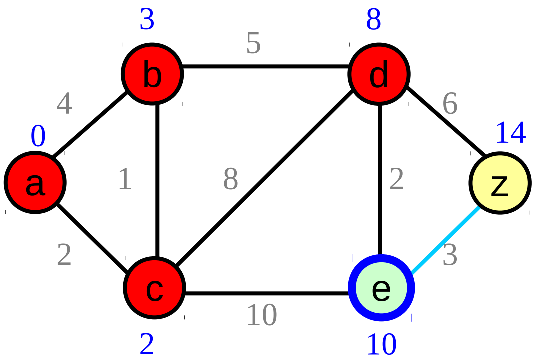
$L(d) + w(d, e) = 8 + 2 \; < \; L(e) = 12$ $\qquad P(a, c, b, d, e) \; < \; P(a, c, e)$

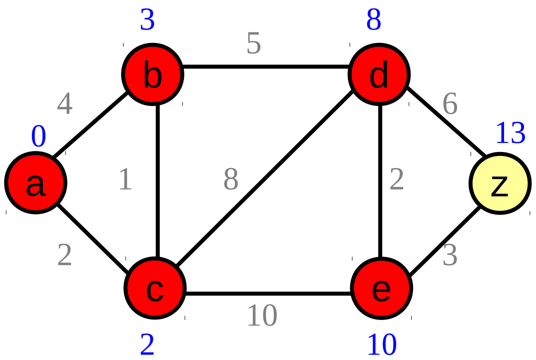
$L(d) + w(d, z) = 8 + 6 \; < \; L(z) = \infty$

Discrete Mathematics and It's Applications, K. H. Rosen

Dijkstra Algorithm Pseudocode 2 Example (5)





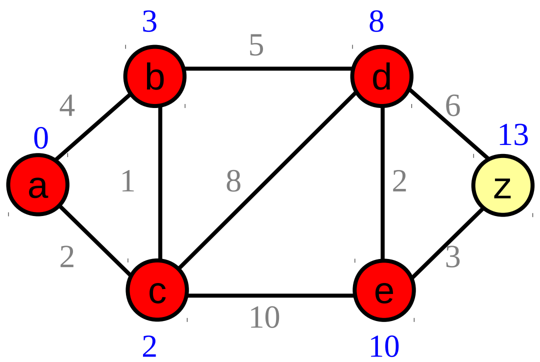


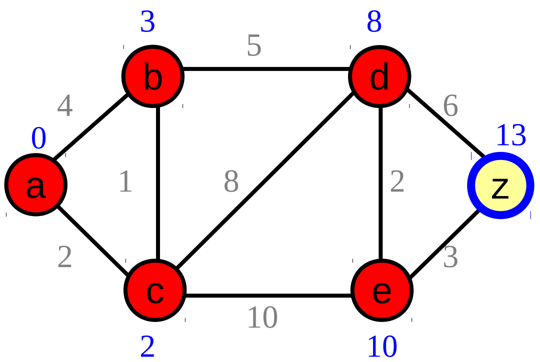
$S = \{a, c, b, d, e\}$

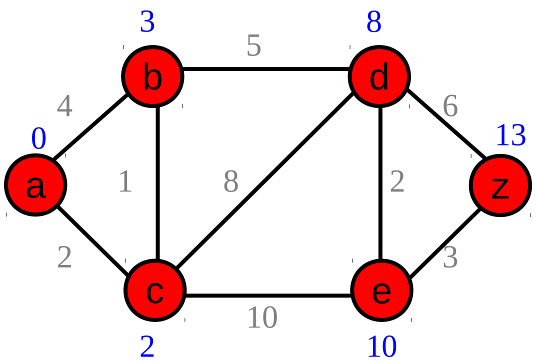
$L(e) + w(e, z) = 10 + 3 < L(z) = 14$ $\quad P(a, c, b, d, e, z) < P(a, c, b, d, z)$

Discrete Mathematics and It's Applications, K. H. Rosen

Dijkstra Algorithm Pseudocode 2 Example (6)







$$S = \{a, c, b, d, e, z\}$$

Discrete Mathematics and It's Applications, K. H. Rosen

References

[1] http://en.wikipedia.org/
[2]