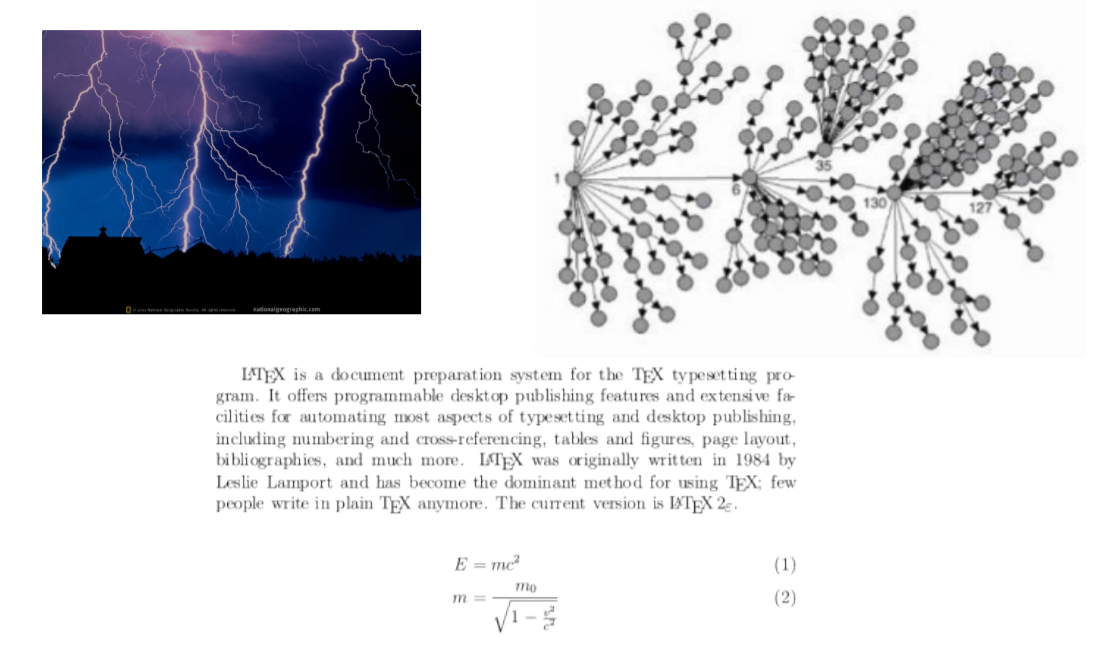
**Dijkstra’s Algorithm**

**Applications of Dijkstra’s Algorithm:**

Dijkstra’s Shortest Path Algorithm in Java. Dijkstra’s Algorithms describes how to find the shortest path from one node to another node in a directed weighted graph. This article presents a Java implementation of this algorithm. This means that the algorithm checks for all descendants that the following condition holds:

**A. The shortest path problem**

[**1. Shortest path**](http://www.vogella.com/tutorials/JavaAlgorithmsDijkstra/article.html#shortestpath_problem)**:** Finding the shortest path in a network is a commonly encountered problem. For example you want to reach a target in the real world via the shortest path or in a computer network a network package should be efficiently routed through the network.

[**2. Graph**](http://www.vogella.com/tutorials/JavaAlgorithmsDijkstra/article.html#shortestpath_graph)A graph is made out of *nodes* and directed *edges* which define a connection from one node to another node. A node (or vertex) is a discrete position in a graph. Edges can be directed an undirected. Edges have an associated distance (also called costs or weight). The distance between two nodes a and b is labeled as [a,b]. The mathematical description for graphs is G= {V,E}, meaning that a graph is defined by a set of vertexes (V) and a collection of edges. The *order* of a graph is the number of nodes. The *size* of a graph is the number of edges

**3. Typical graph problems**

Typical graph problems are described in the following list.

* finding the shortest path from a specific node to another node
* finding the maximum possible flow through a network where each edges has a pre-defined maximum capacity (maximum-flow minimum-cut problem)

The following will focus on finding the shortest path from one node to another node in a graph.

**B. Dijkstra’s algorithm**

[**1. High level description**](http://www.vogella.com/tutorials/JavaAlgorithmsDijkstra/article.html#dijkstra_overview)**:** The *Dijkstra Algorithm* finds the shortest path from a source to all destinations in a directed graph (single source shortest path problem). During this process it will also determine a spanning tree for the graph.

[**2. Algorithms Description**](http://www.vogella.com/tutorials/JavaAlgorithmsDijkstra/article.html#dijkstra_algorithms)**:** Dijkstra partitions all nodes into two distinct sets: unsettled and settled. Initially all nodes are in the unsettled sets, e.g. they must be still evaluated. A node is moved to the settled set if a shortest path from the source to this node has been found.

Initially the distance of each node to the source is set to a very high value.

First only the source is in the set of unsettledNodes. The algorithms runs until the unsettledNodes are empty. In each iteration it selects the node with the lowest distance from the source out of the unsettled nodes. It reads all edges which are outgoing from the source and evaluates for each destination node, in the edges which are not yet settled, if the known distance from the source to this node can be reduced while using the selected edge. If this can be done then the distance is updated and the node is added to the nodes which need evaluation.

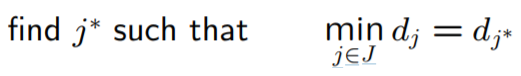
In pseudocode the algorithm can be described as follows. Please note that Dijkstra also determines the pre-successor of each node on its way to the source. I leave that out of the pseudo code to simplify it.

**Algorithm Steps:**

**Step 1.** Initialization • Assign the zero distance value to node s, and label it as Permanent. [The state of node s is (0, p).] • Assign to every node a distance value of ∞ and label them as Temporary. [The state of every other node is (∞, t).] • Designate the node s as the current node

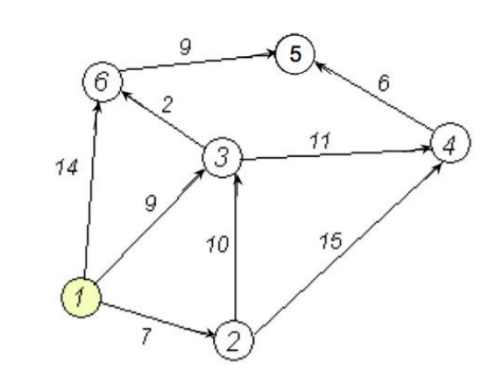
**Step 2.** Distance Value Update and Current Node Designation Update Let i be the index of the current node.

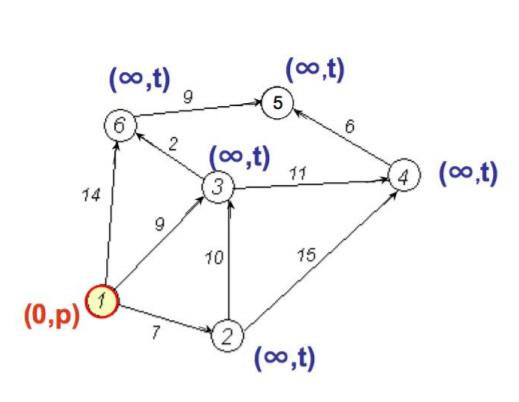
(1) Find the set J of nodes with temporary labels that can be reached from the current node i by a link (i, j). Update the distance values of these nodes. • For each j ∈ J, the distance value dj of node j is updated as follows new

where cij is the cost of link (i, j), as given in the network problem.

(2) Determine a node j that has the smallest distance value dj among all nodes j ∈ J,

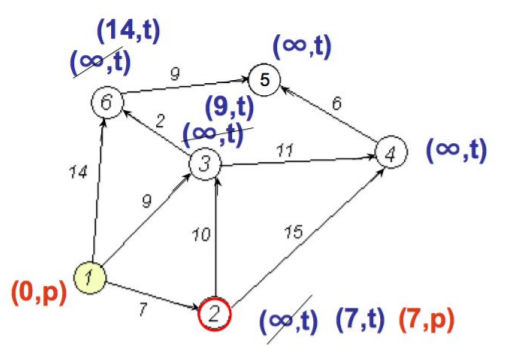
(3) Change the label of node j ∗ to permanent and designate this node as the current node

If all nodes that can be reached from node s have been permanently labeled, then stop - we are done. If we cannot reach any temporary labeled node from the current node, then all the temporary labels become permanent - we are done. Otherwise, go to Step 2.

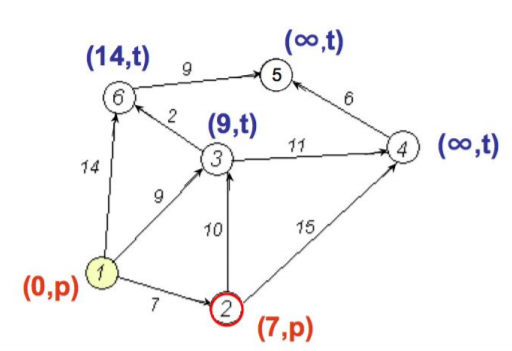
**Example:** We want to find the shortest path from node 1 to all other nodes using Dijkstra’s algorithm.

**Step 1 - Initialization:**

* Node 1 is designated as the current node •
* The state of node 1 is (0, p)
* Every other node has state (∞, t)

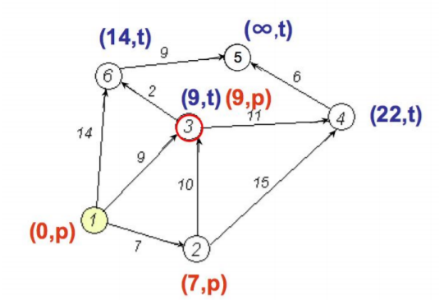


**Step 2:**

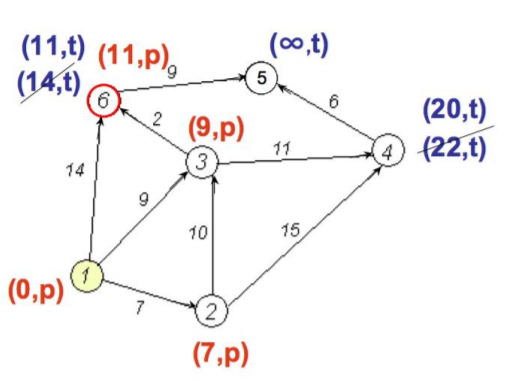
* Nodes 2, 3,and 6 can be reached from the current node 1
* Update distance values for these nodes
  + d2 = min{∞, 0 + 7} = 7
  + d3 = min{∞, 0 + 9} = 9
  + d6 = min{∞, 0 + 14} = 14
* Now, among the nodes 2, 3, and 6, node 2 has the smallest distance value
* The status label of node 2 changes to permanent, so its state is (7, p), while the status of 3 and 6 remains temporary
* Node 2 becomes the current node

**Step 3:**

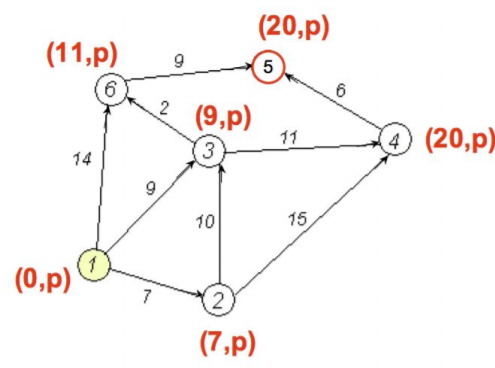
We are not done, not all nodes have been reached from node 1, so we perform another iteration (back to Step 2)



**Another Implementation of Step 2:**

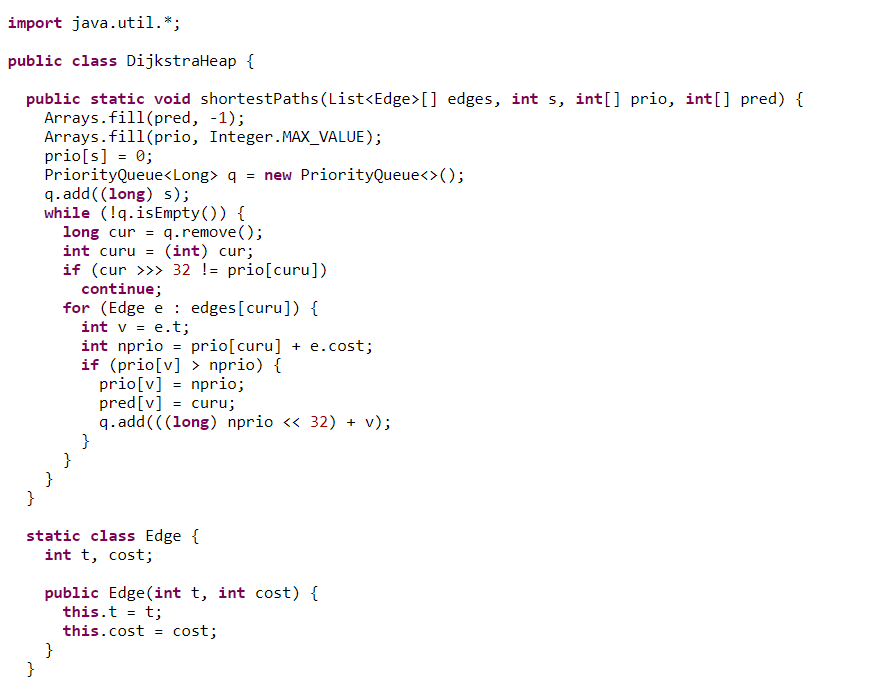
* Nodes 3 and 4 can be reached from the current node 2
* Update distance values for these nodes
  + d3 = min{9, 7 + 10} = 9
  + d6 = min{∞, 7 + 15} = 22
* Now, between the nodes 3 and 4 node 3 has the smallest distance value
* The status label of node 3 changes to permanent, while the status of 6 remains temporary
* Node 3 becomes the current node We are not done (Step 3 fails), so we perform another Step 2

**Another Step 2:**

* Nodes 6 and 4 can be reached from the current node 3
* Update distance values for them
  + d4 = min{22, 9 + 11} = 20
  + d6 = min{14, 9 + 2} = 11
* Now, between the nodes 6 and 4 node 6 has the smallest distance value
* The status label of node 6 changes to permanent, while the status of 4 remains temporary
* Node 6 becomes the current node We are not done (Step 3 fails), so we perform another Step 2

**Another Step 2:**

* Node 5 can be reached from the current node 6
* Update distance value for node 5
  + d5 = min{∞, 11 + 9} = 20
* Now, node 5 is the only candidate, so its status changes to permanent
* Node 5 becomes the current node From node 5 we cannot reach any other node. Hence, node 4 gets permanently labeled and we are done.

**Shortest paths. Dijkstra's algorithm in O(E \* logV):**

