Minimum weight spanning tree:

Put a non-negative weight on each edge.

Weight of a tree: sum of weights of its edges.

Problem: find a spanning tree of graph G with minimum weight.

Kruskal's algorithm: sort edges by weight.

Then for each edge: add it to the tree if its endpoints are in different components.

One application:

A cable company must install cable to a new neighbourhood. The cables are constrained to be buried along certain paths. The cost varies for different paths. A minimum weight spanning tree gives the cheapest way to connect everyone to cable.

Water distribution network

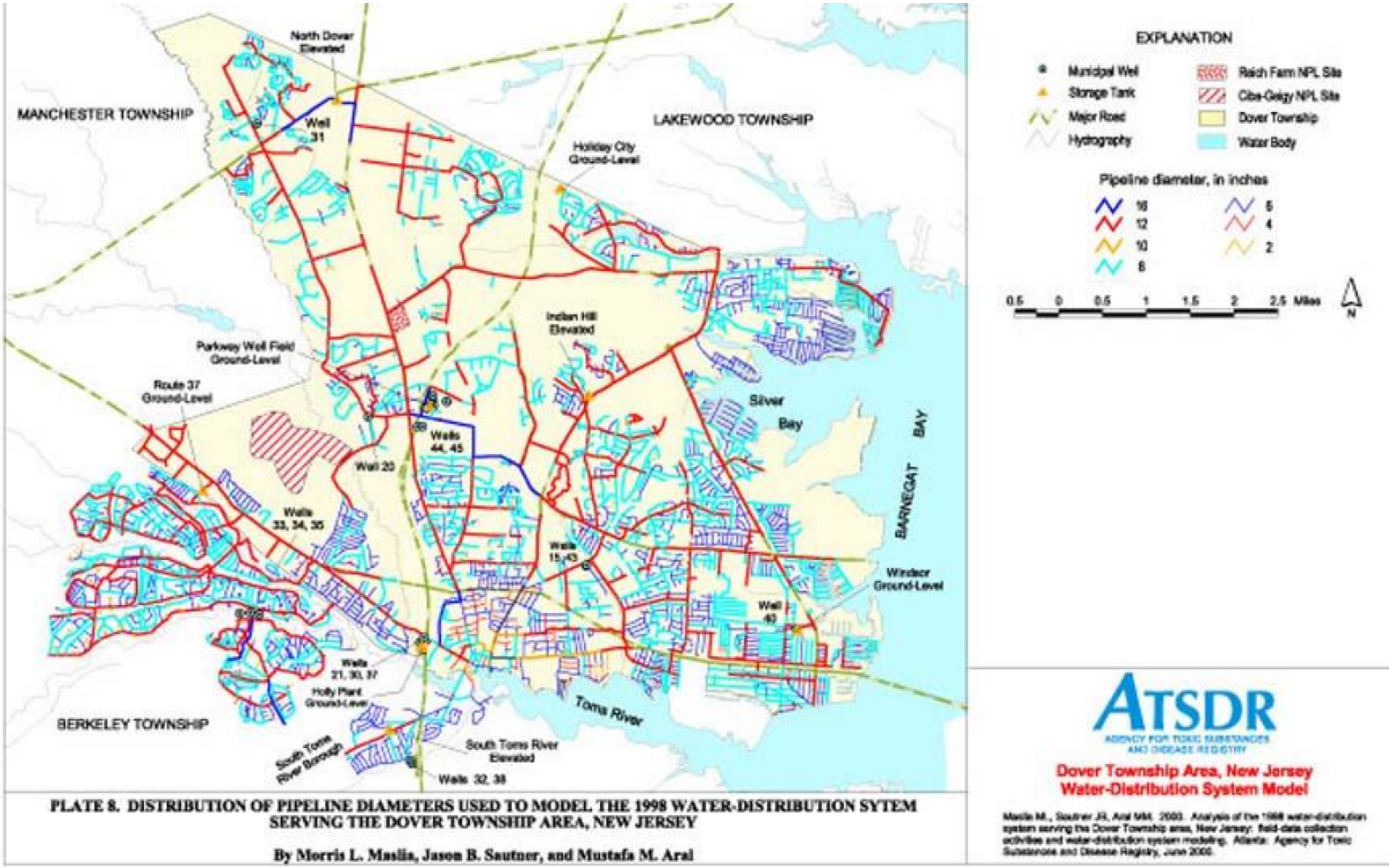


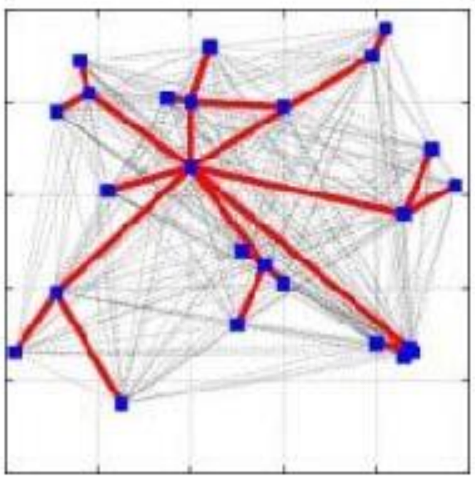
PLATE 8. DISTRIBUTION OF PIPELINE DIAMETERS USED TO MODEL THE 1998 WATER-DISTRIBUTION SYTEM SERVING THE DOVER TOWNSHIP AREA, NEW JERSEY

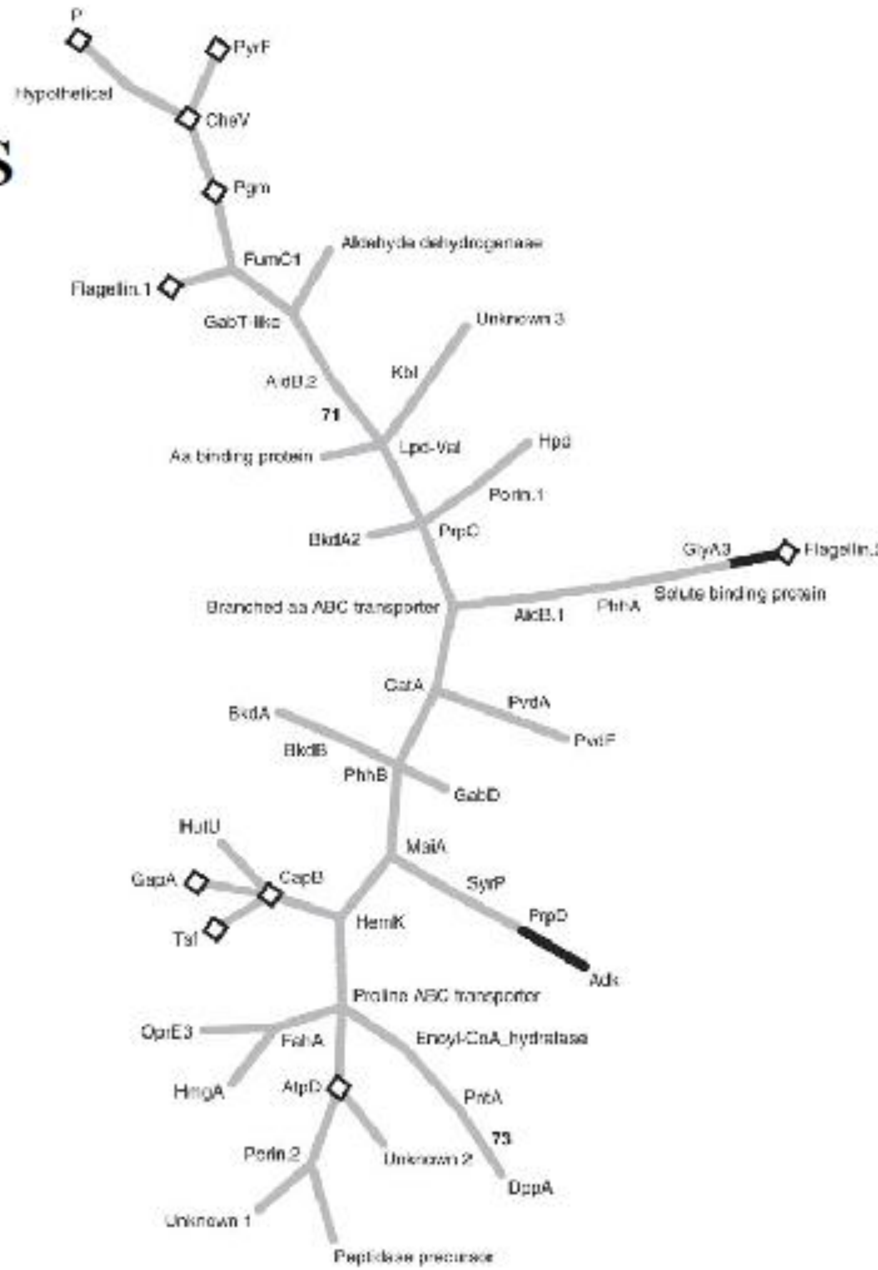
By Morris L. Maslia, Jason B. Sautner, and Mustafa M. Aral

Maslia M., Sautner JB, Aral MM. 2000. Analysis of the 1998 water-distribution system serving the Dover Township area, New Jersey: field-data collection activities and water-distribution system modeling. Atlanta: Agency for Toxic Substances and Disease Registry, June 2000.

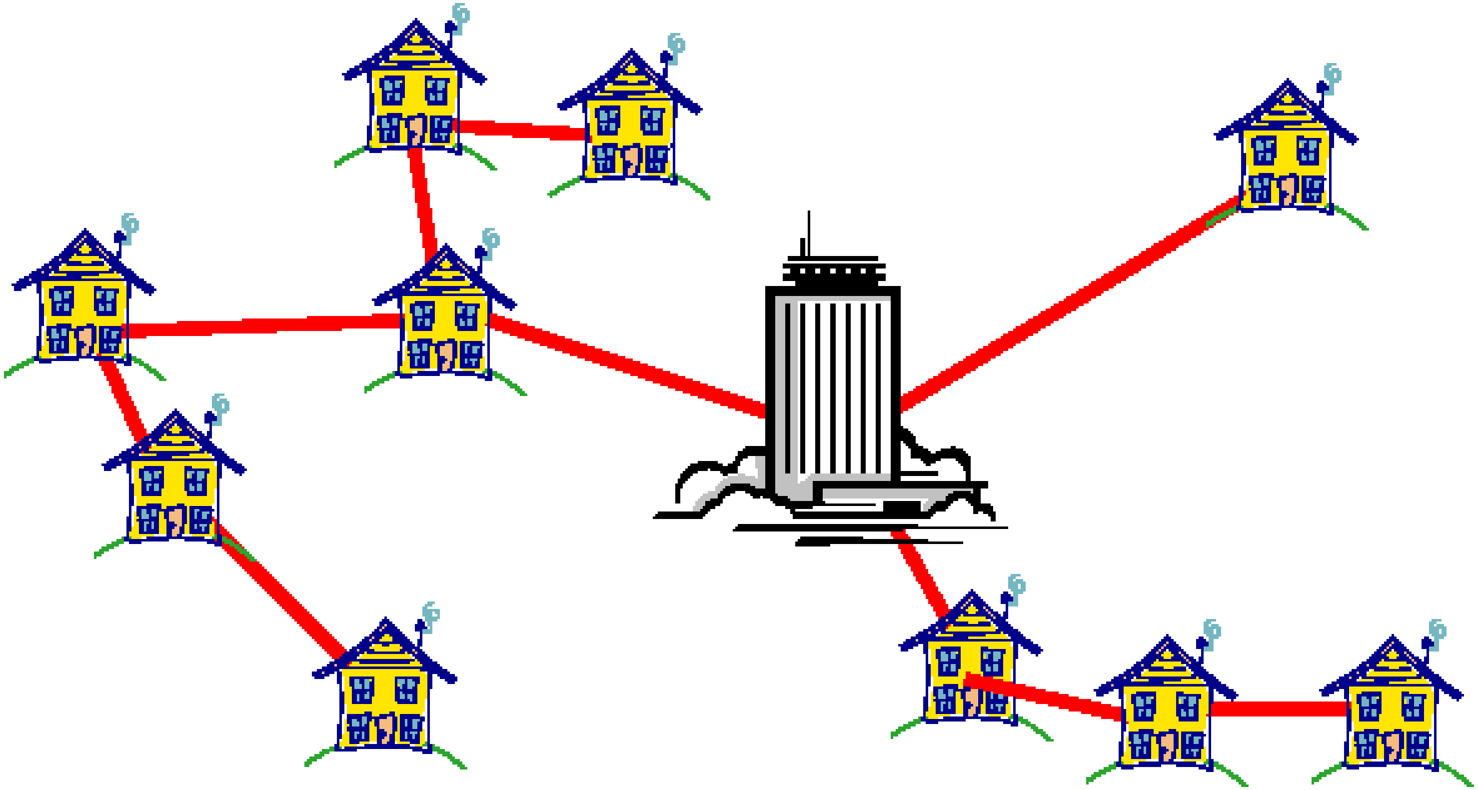
MST Applications

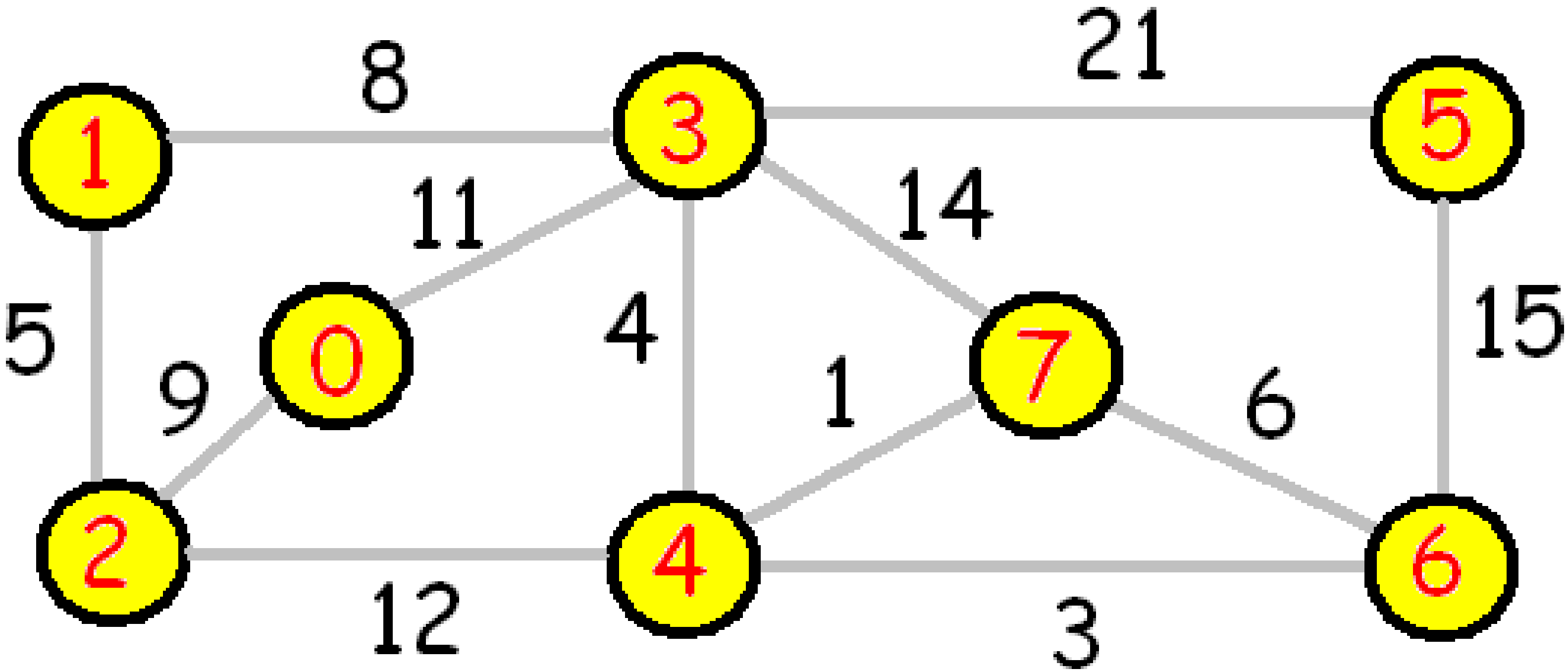
- Gene mutations
- MST of a complete graph





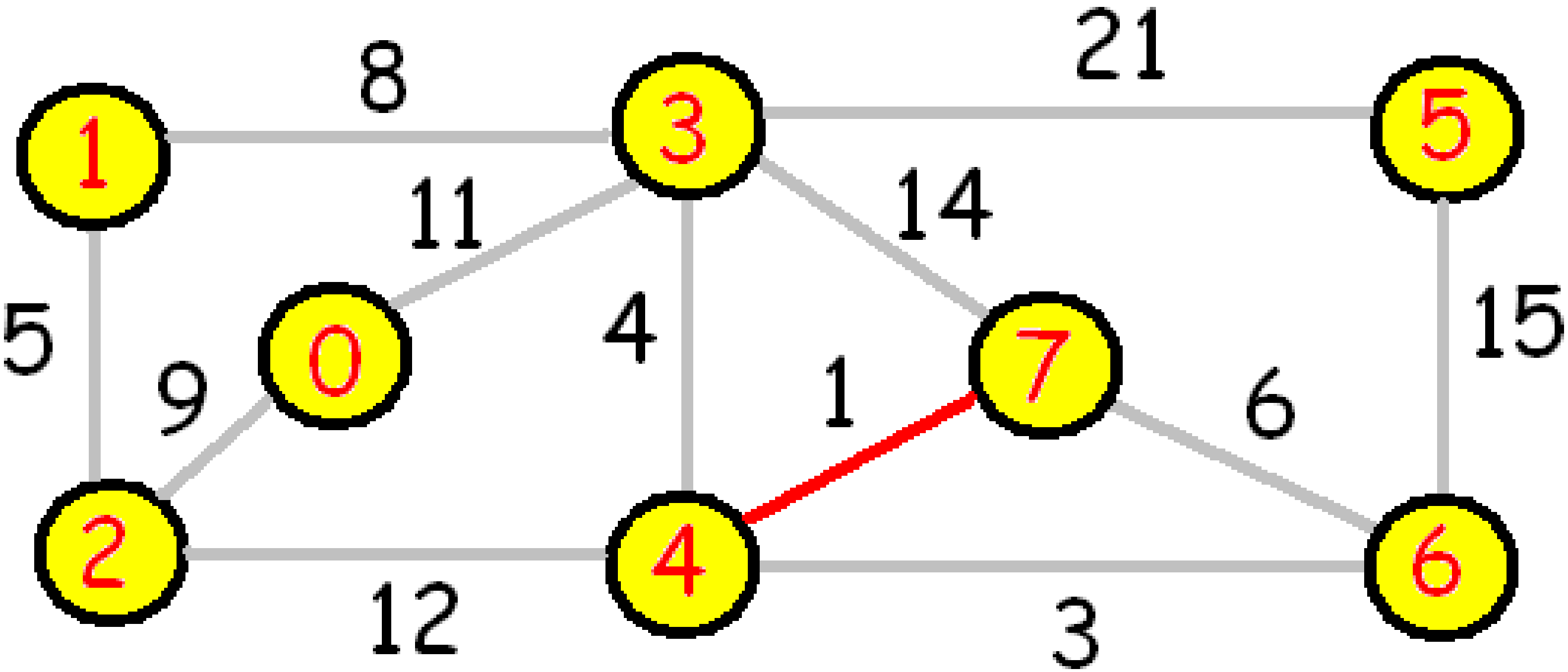
Minimize Wiring





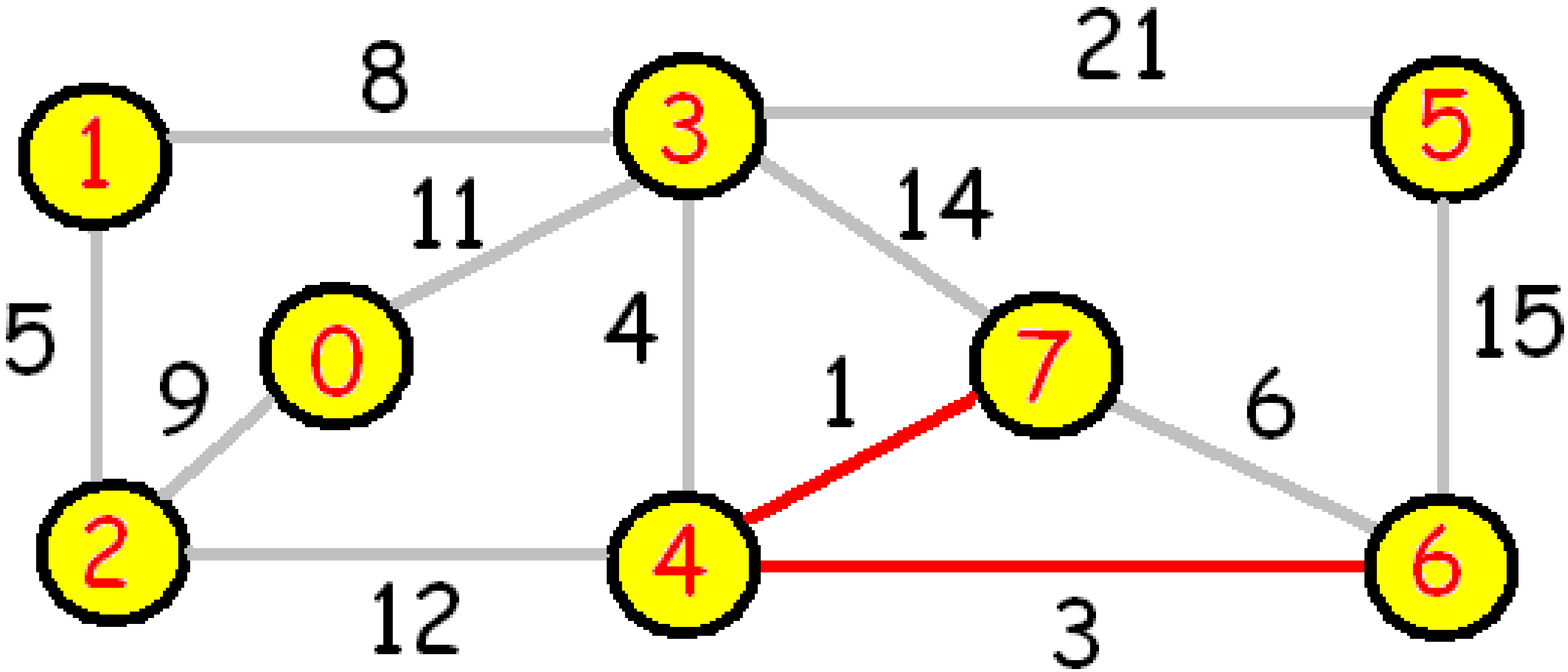
Edge weights:

1, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 21



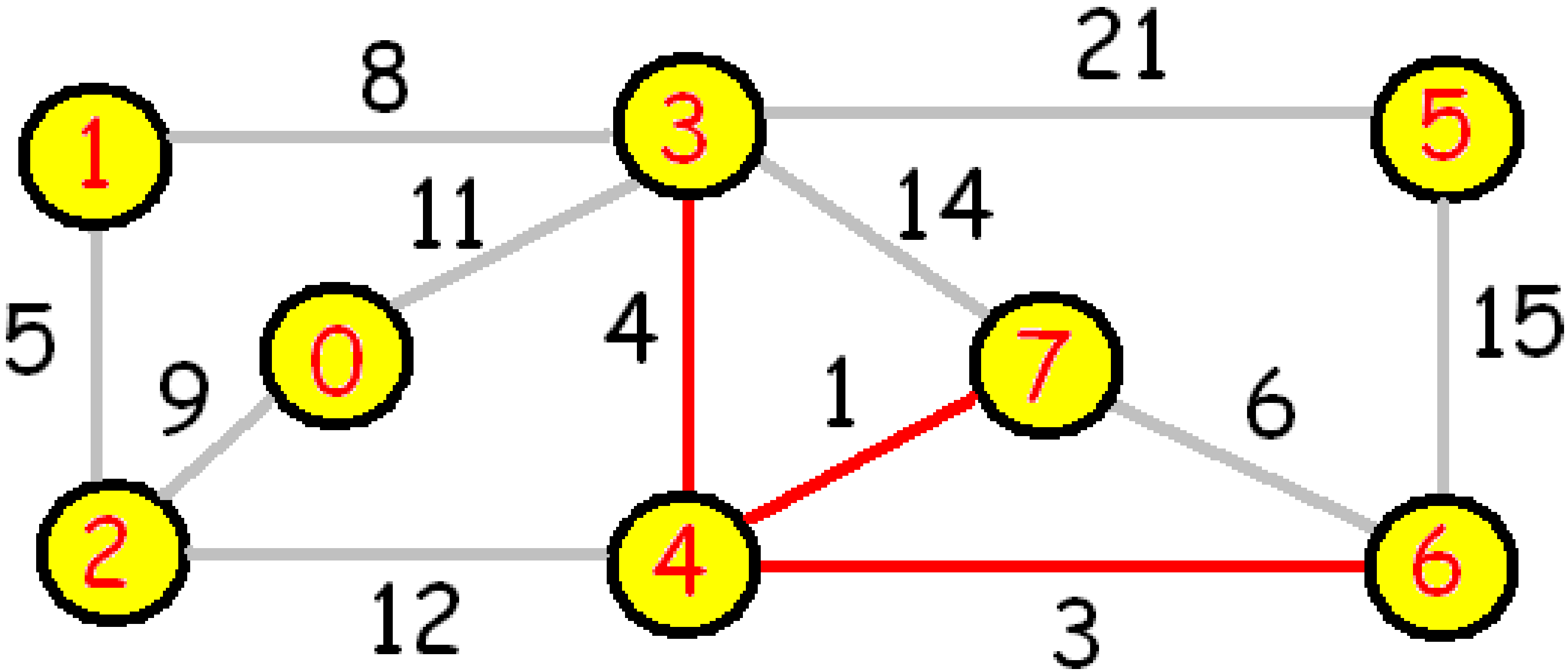
Edge weights:

1, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 21



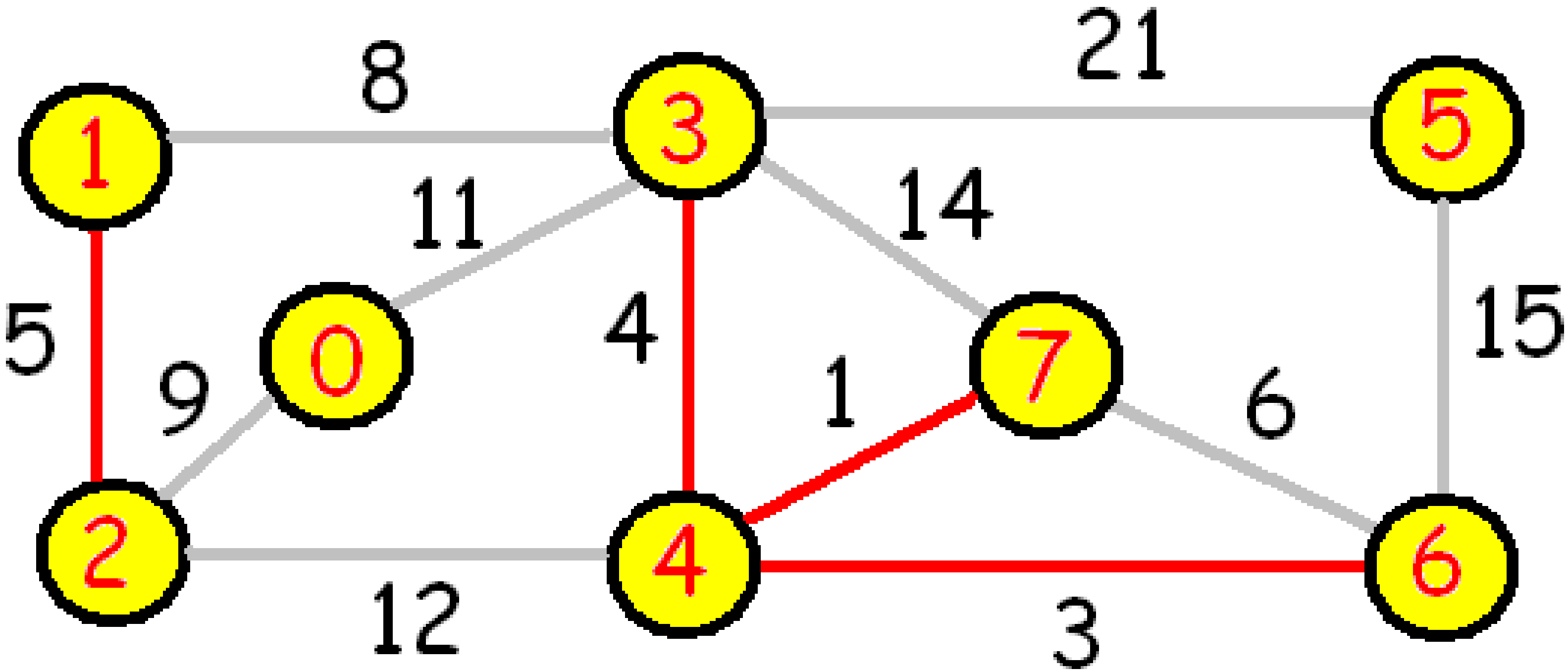
Edge weights:

1, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 21



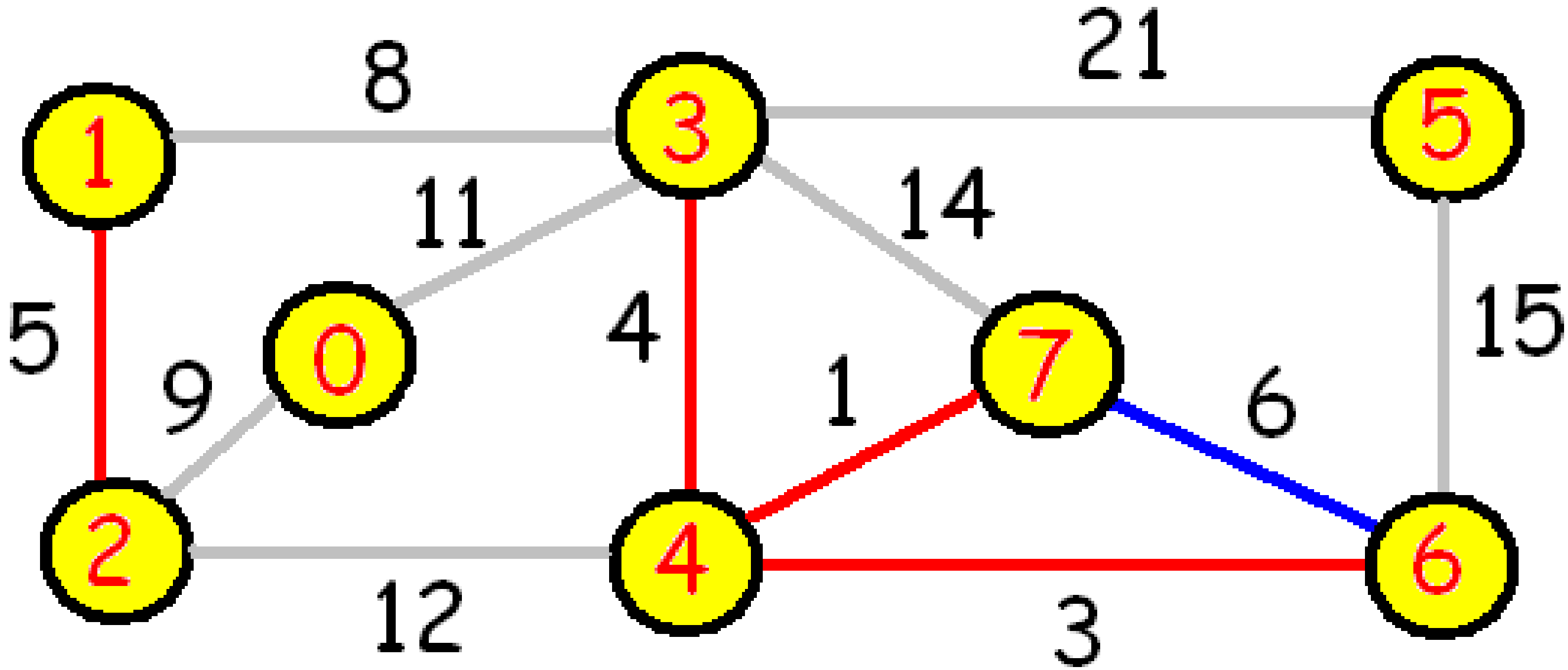
Edge weights:

1, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 21



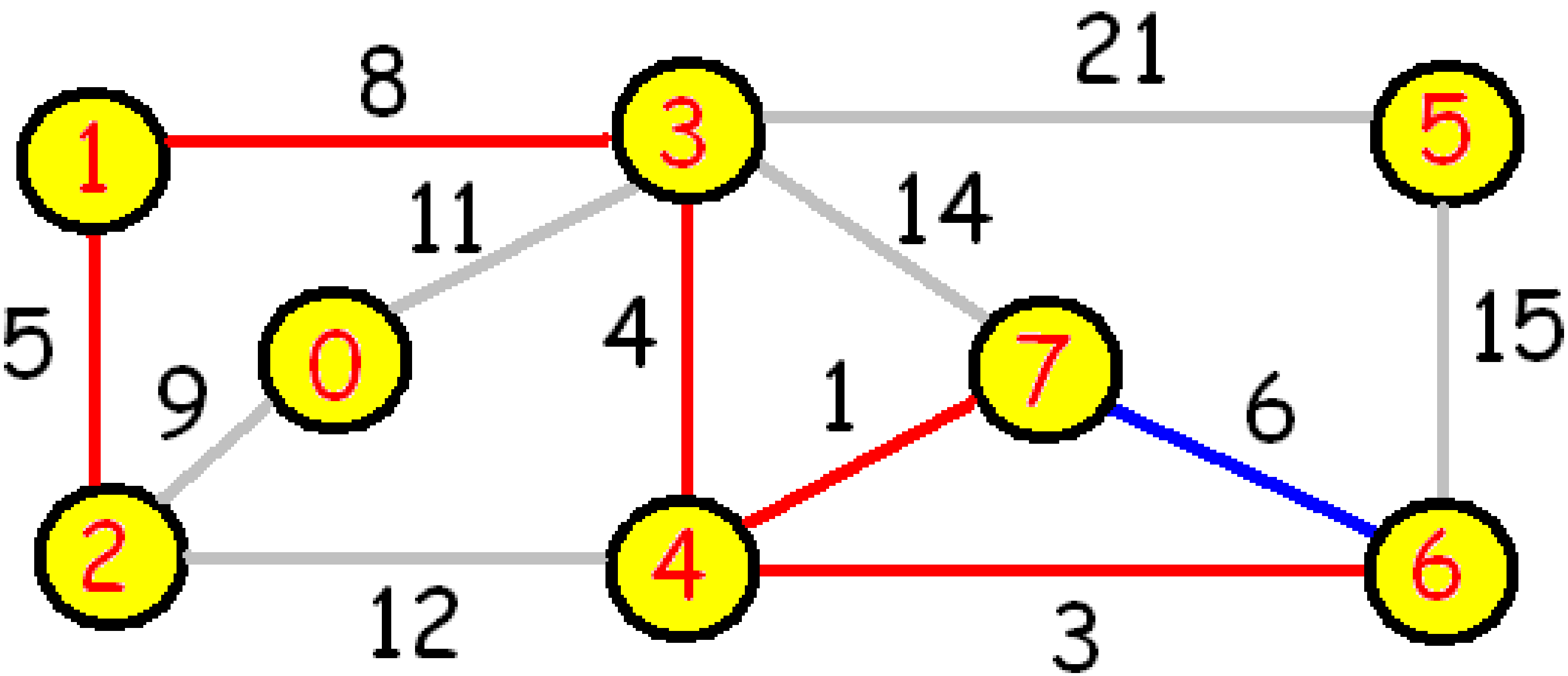
Edge weights:

1, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 21



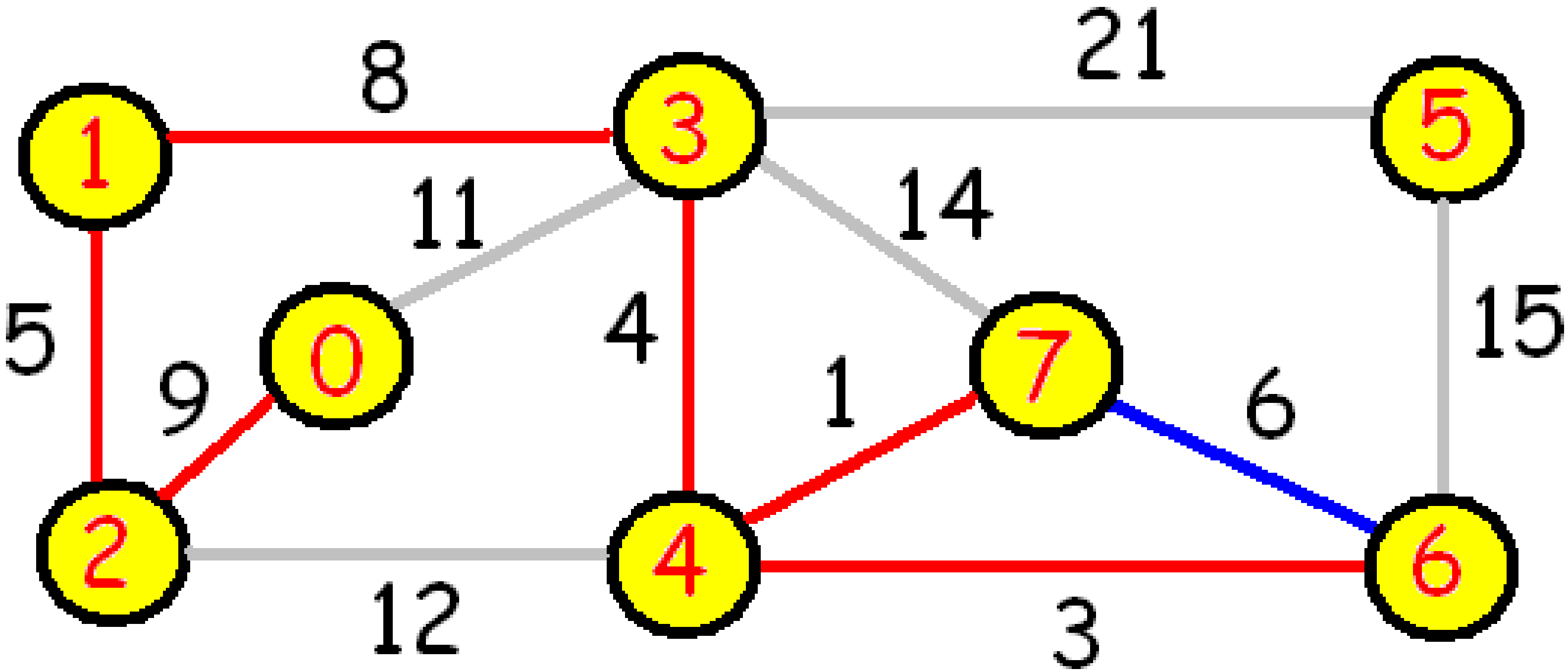
Edge weights:

1, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 21



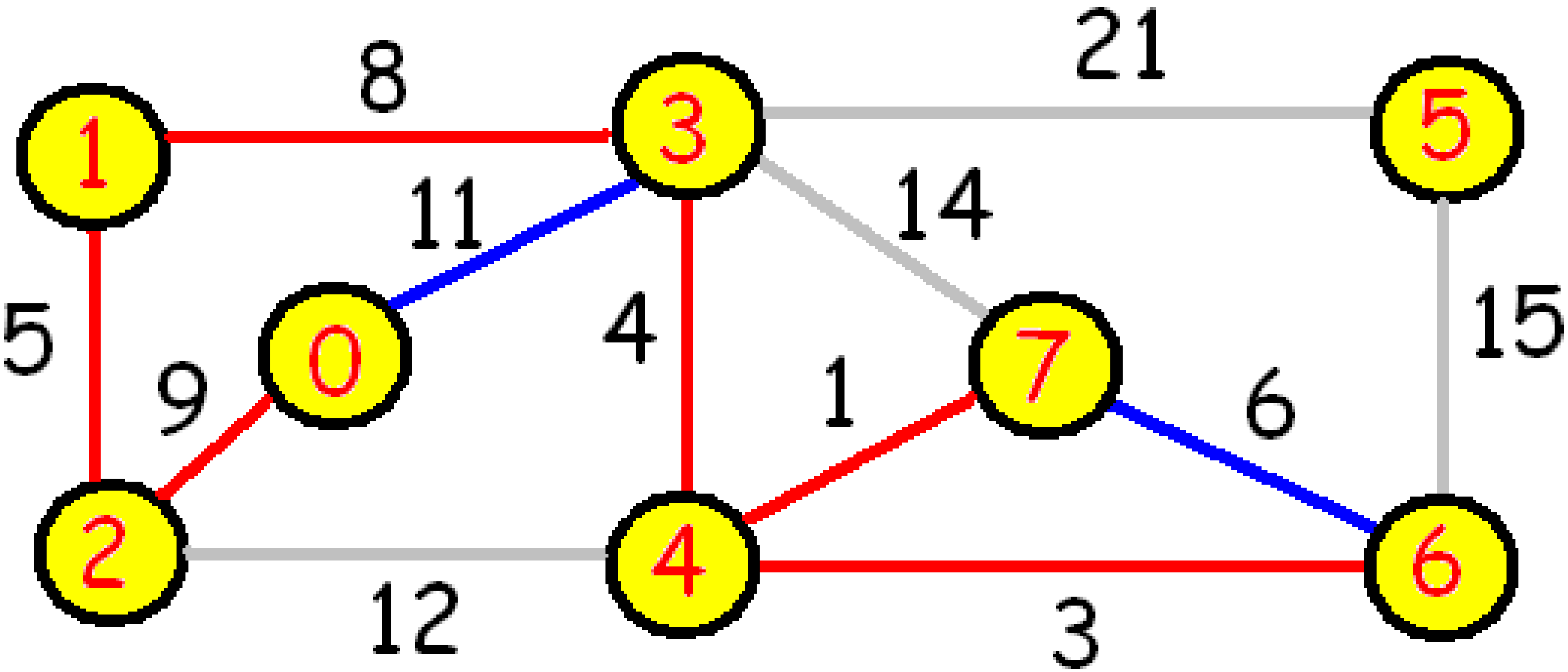
Edge weights:

1, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 21



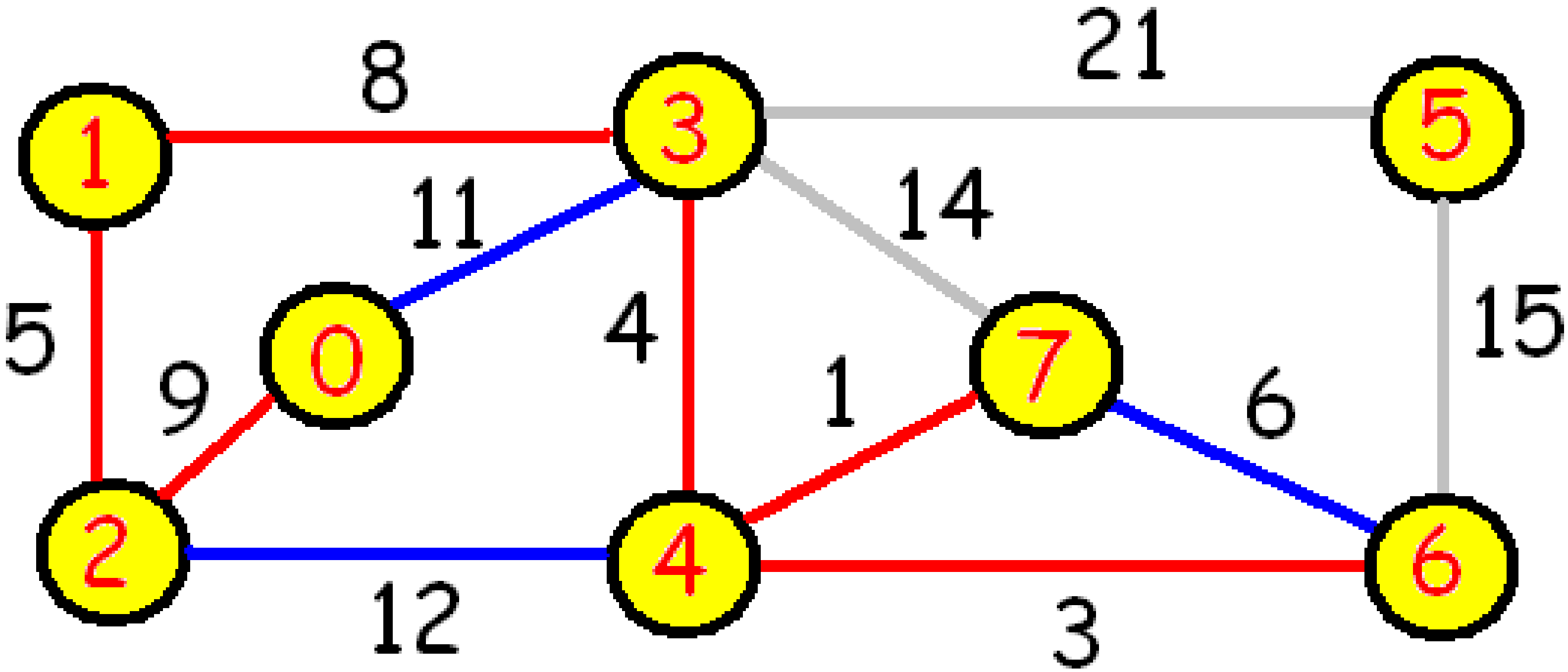
Edge weights:

1, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 21



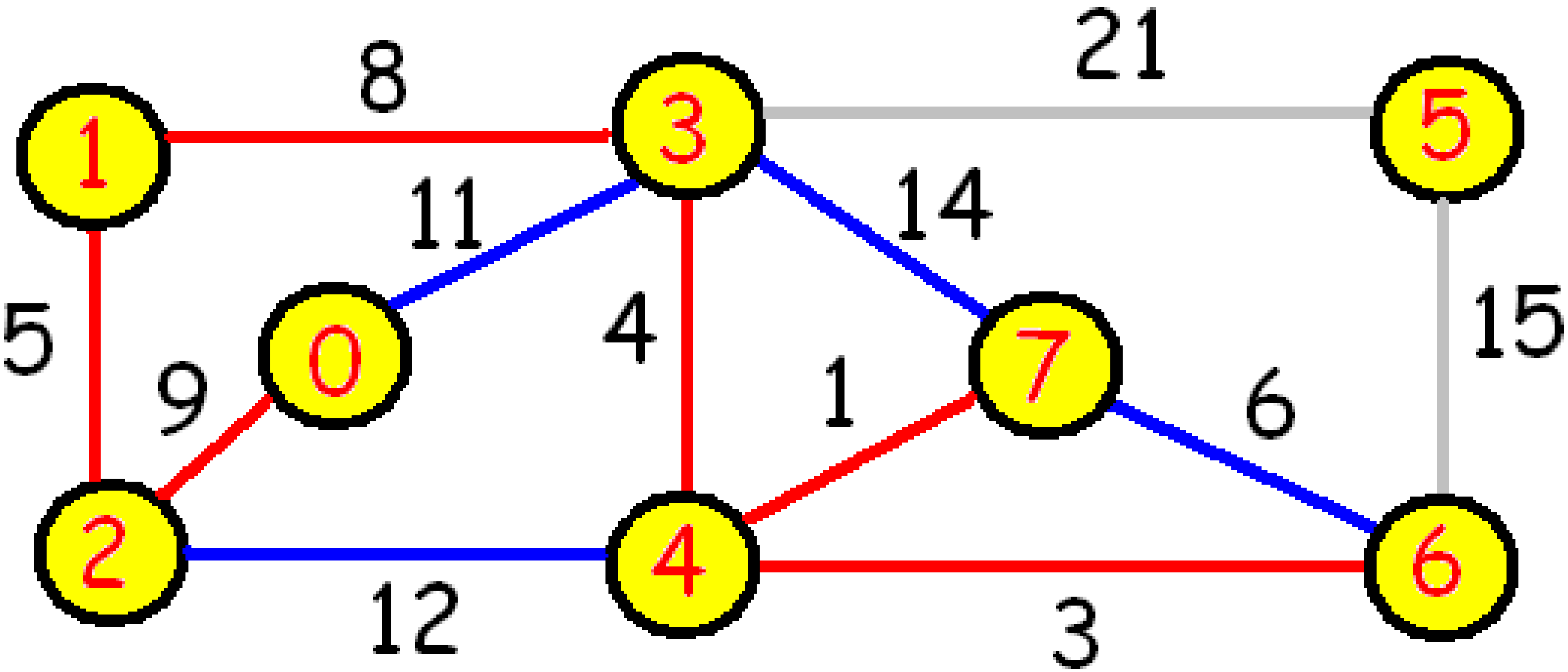
Edge weights:

1, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 21



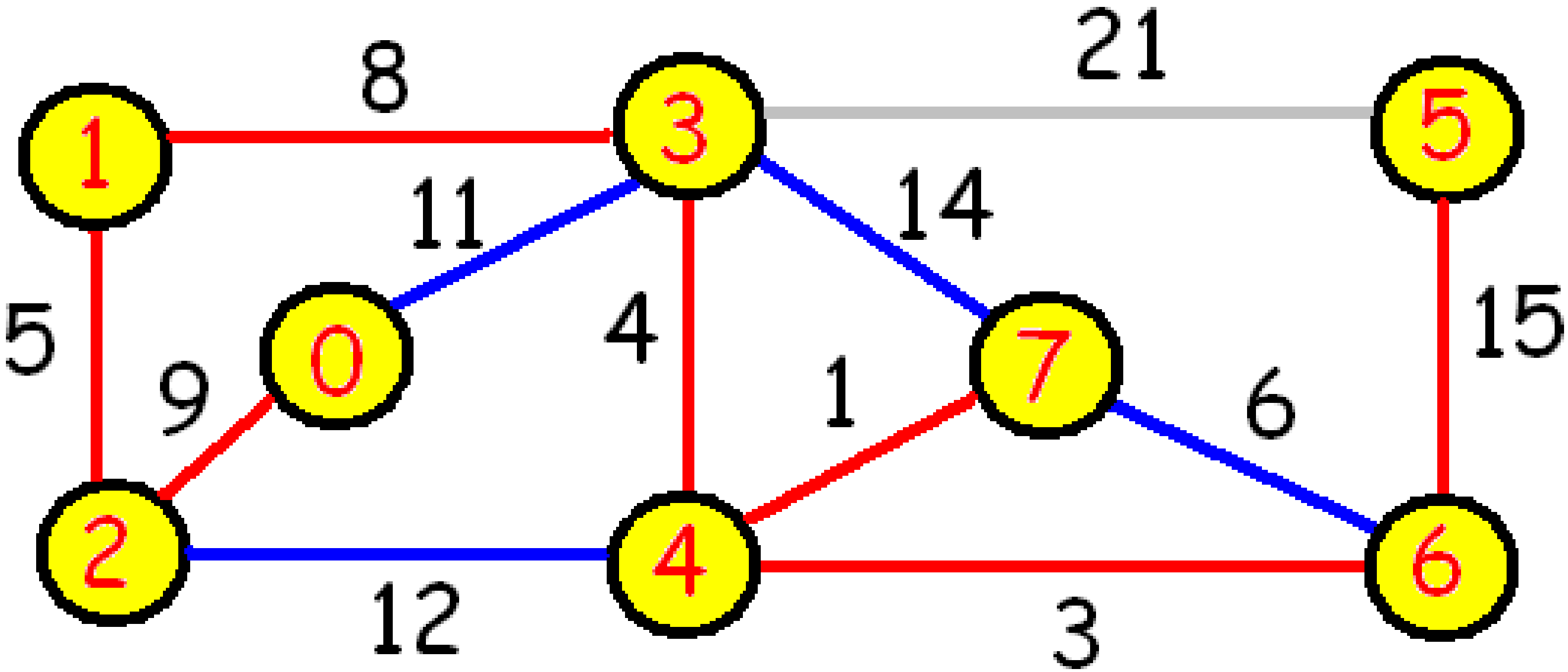
Edge weights:

1, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 21



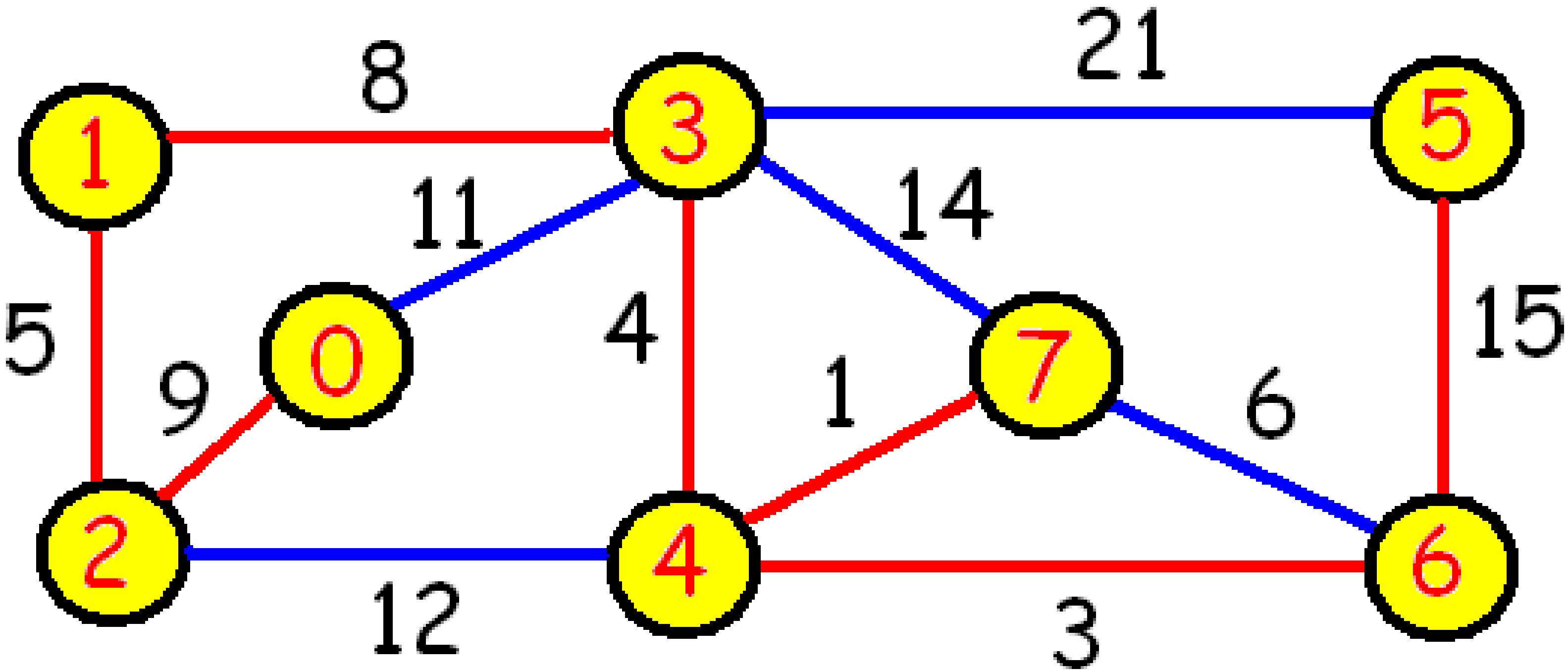
Edge weights:

1, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 21



Edge weights:

1, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 21

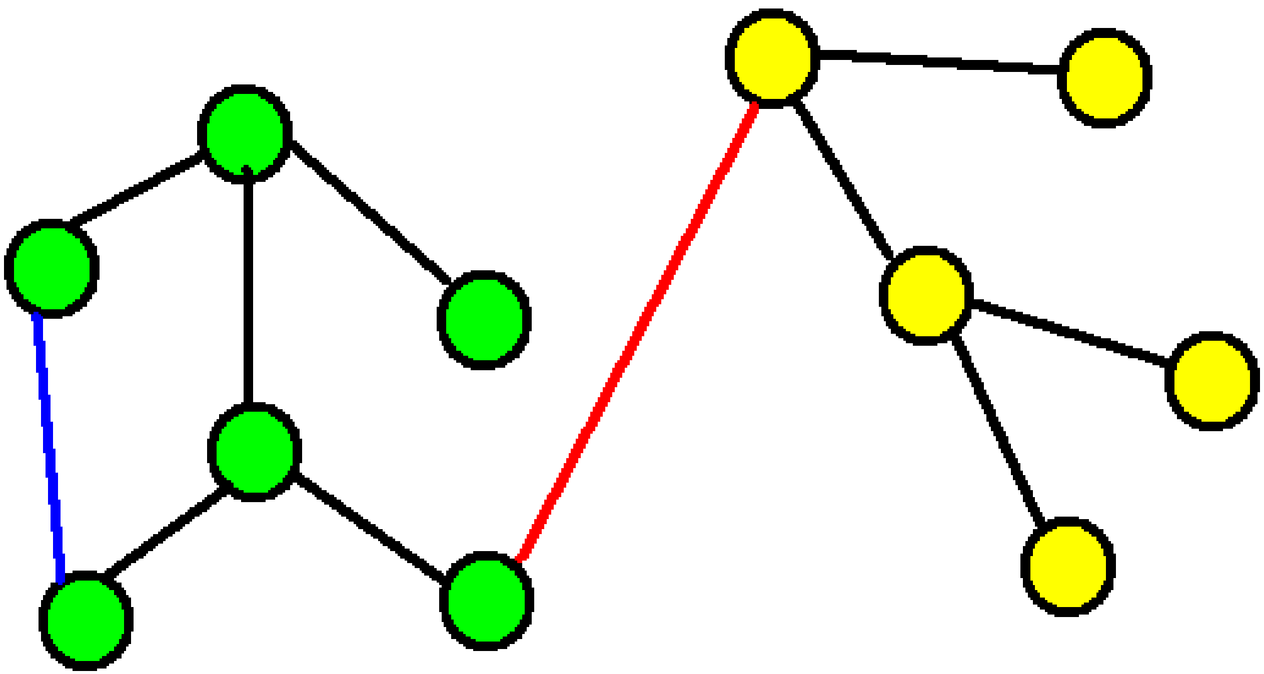


Edge weights:

1, 3, 4, 5, 6, 8, 9, 11, 12, 14, 15, 21

How can we determine quickly at each step whether adding a new edge creates a cycle?

Or equivalently, given an edge (u,v) are u and v in the same component?



The UNION/FIND data structure is a dynamic data structure for graphs used to keep track of the connected components.

It has 2 routines:

FIND(u): returns the name of the component containing vertex u

UNION(u, v): unions together the components containing u and v (corresponding to an addition of edge (u,v) to the graph).