15.053/8 April 11, 2013

Introduction to Networks

Quotes for today

"A journey of a thousand miles begins with a single step."

-- Confucius

“You cannot travel the path until you have become the path itself”

-- Buddha

Network Models

- Optimization models
- Can be solved much faster than other LPs
- Applications to industrial logistics, supply chain management, and a variety of systems
- Today's lecture: introductory material, Eulerian tours, the Shortest Path Problem
- Application of Network Models:

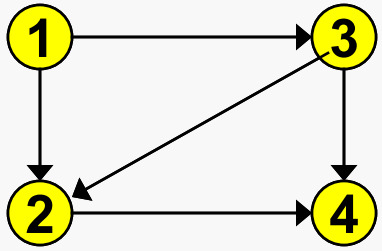
http://jorlin.scripts.mit.edu/docs/publications/52-applications%20of%20network.pdf

Notation and Terminology

Note: Network terminology is not (and never will be) standardized. The same concept may be denoted in many different ways.

Called:

- NETWORK
- directed graph
- digraph
- graph



Class Handouts (Ahuja, Magnanti, Orlin)

Network G = (N, A)

Node set N = {1, 2, 3, 4}

Arc Set {(1,2), (1,3), (3,2), (3,4), (2,4)}

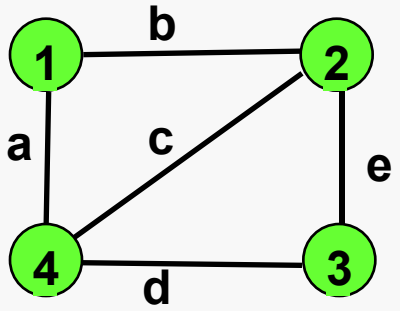
Also Seen

Graph G = (V, E)

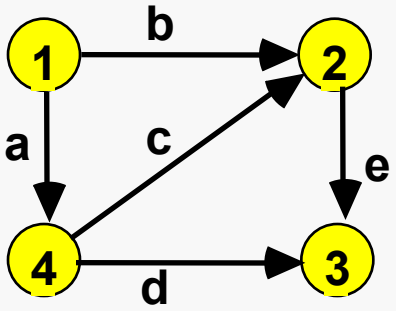
Vertex set V

Edge set E

Directed and Undirected Networks



An Undirected Graph



A Directed Graph

- Networks are used to transport commodities
  - physical goods (products, liquids)
  - communication
  - electricity, etc.
- The field of Network Optimization concerns optimization problems on networks

Networks are Everywhere

- Physical Networks
  - Road Networks
  - Railway Networks
  - Airline traffic Networks
  - Electrical networks, e.g., the power grid
  - Communication networks

- Social networks
  - Organizational charts
  - friendship networks
  - interaction networks (e.g., cell calls)

Overview:

- Most O.R. models have networks or graphs as a major aspect
- Next two lectures: focus on network optimization problems.
- Next: representations of networks
  - Pictorial
  - Computer representations

LOST: An Illustrative Example

Image removed due to copyright restrictions.

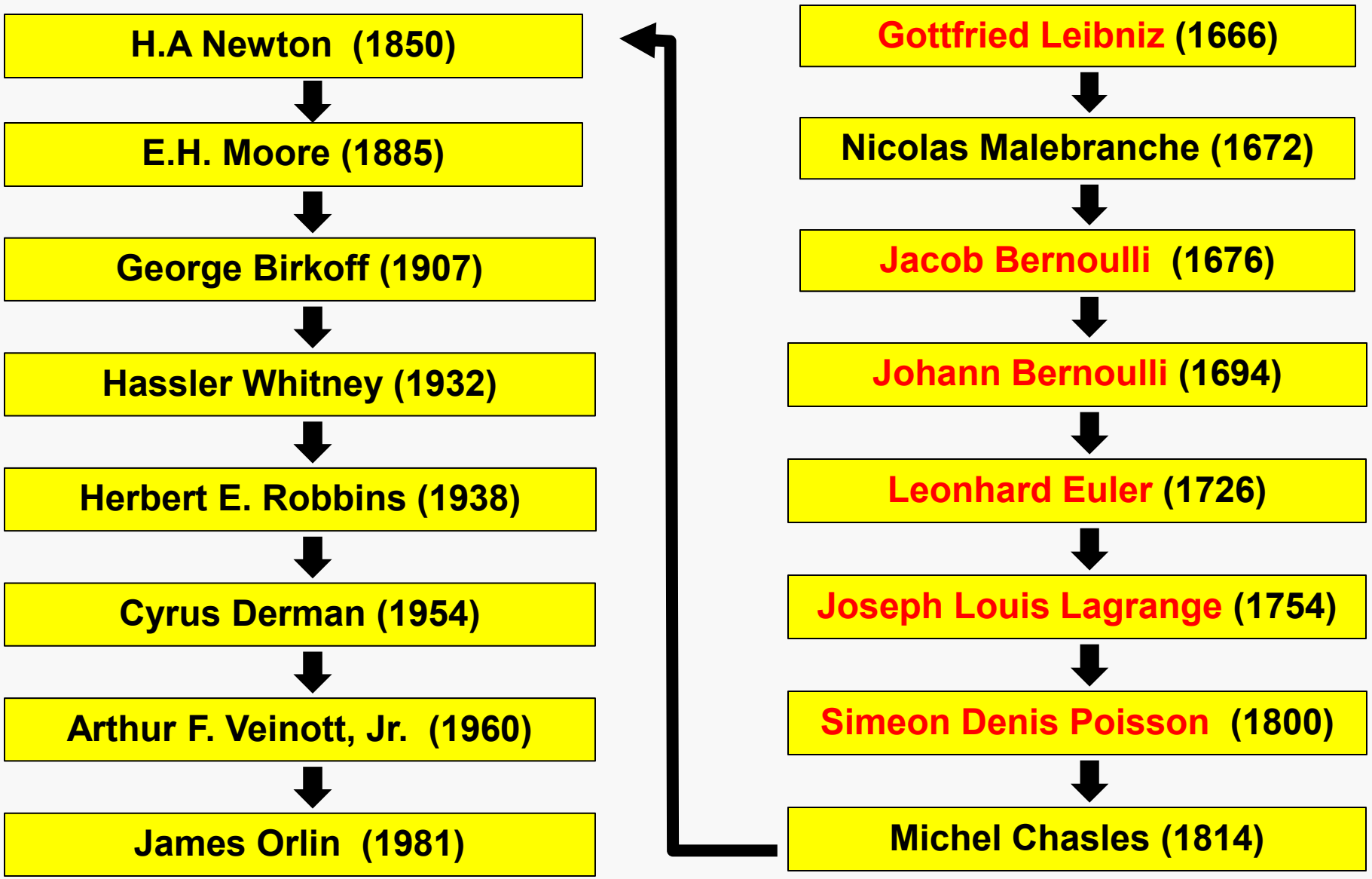
See interactive graphic "The Web of Intrigue" in "As Lost Ends, Creators Explain How They Did It, What's Going On." *Wired Magazine*, April 19, 2010.

***Wired* has a handy character chart (yes, there might be spoilers!), created by bioinformics scientist Martin Krzywinski using Circos software (even that sentence is confusing) that shows how all of the characters are related. For example, if you're wondering how many of the characters are related via "romance," click on Romance and the chart will change to show you that. Same with Chance, Family, Occupational, Touched By Jacob, Undisclosed.**

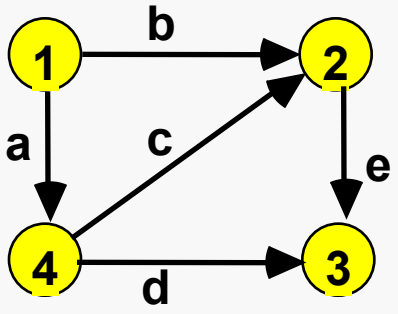
An elegant representation of arcs

http://flare.prefuse.org/apps/dependency_graph

Graph of Ph.D. advisors



The Adjacency Matrix (for directed graphs)



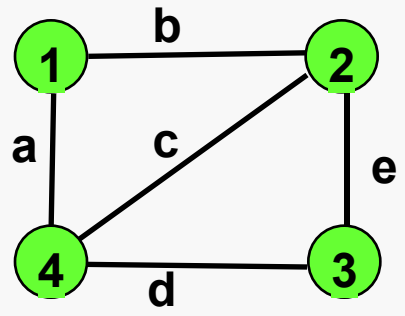
A Directed Graph

	1	2	3	4
1	0	1	0	1
2	0	0	1	0
3	0	0	0	0
4	0	1	1	0

- Have a row for each node
- Have a column for each node
- Put a 1 in row i- column j if (i, j) is an arc

What would happen if (4, 2) became (2, 4)?

The Adjacency Matrix (for undirected graphs)



An Undirected Graph

	1	2	3	4	degree
1	0	1	0	1	2
2	1	0	1	1	3
3	0	1	0	1	2
4	1	1	1	0	3

- Have a row for each node
- Have a column for each node
- Put a 1 in row i- column j if (i, j) is an arc

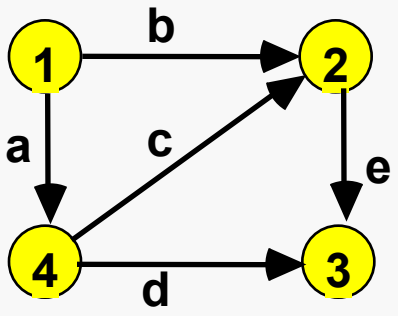
The ***degree*** of a node is the number of incident arcs

Note: each arc shows up twice in the adjacency matrix.

Question. Is it possible that the number of nodes of odd degree is odd?

1. Yes
2. No

Arc list representations



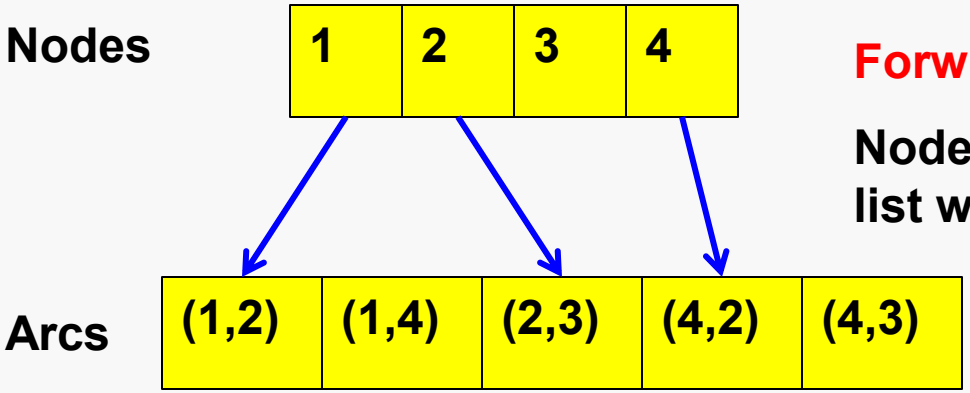
A Directed Graph

1: (1,2), (1,4)

2: (2,3)

3: ø

4: (4,2), (4,3)



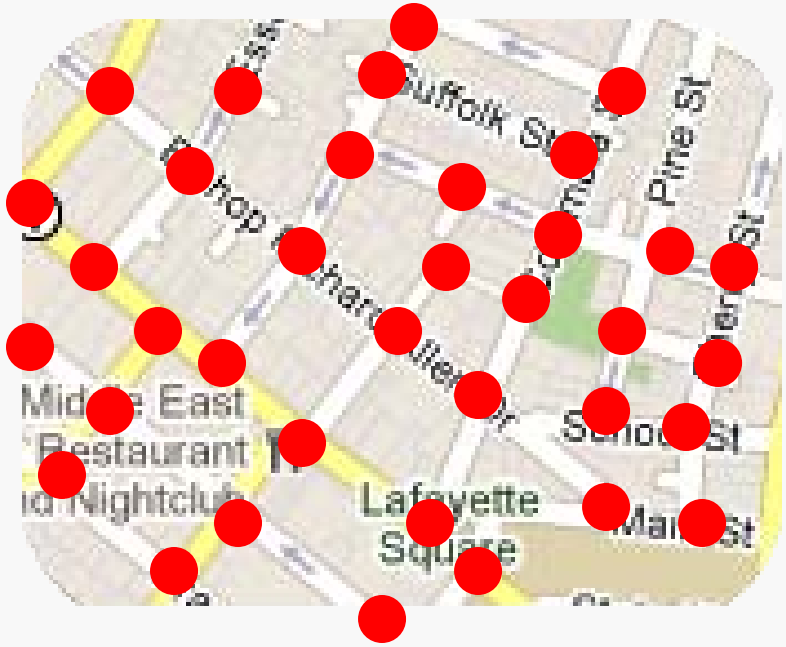
Forward Star Representation.

Node i points to first arc on arc list whose head is node i.

Which uses computer space more efficiently for large road networks: the adjacency matrix or adjacency lists?

e.g. consider a road network with 10,000 nodes, and with 40,000 arcs

1. **Adjacency matrix**
2. **Adjacency lists**



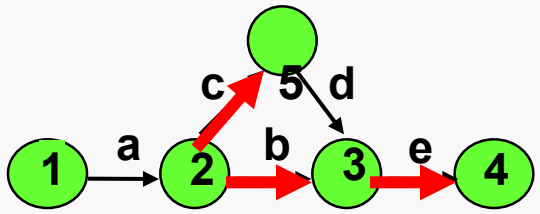
On network representations

- Each representation has its advantages
  - Major purpose of a representation
    - efficiency in algorithms
    - ease of use

- Next: definitions for networks

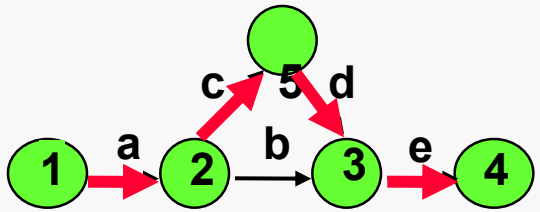
Path: **Example: 5, 2, 3, 4.**
(or 5, c, 2, b, 3, e, 4)

- **No node is repeated.**
- **Directions are ignored.**



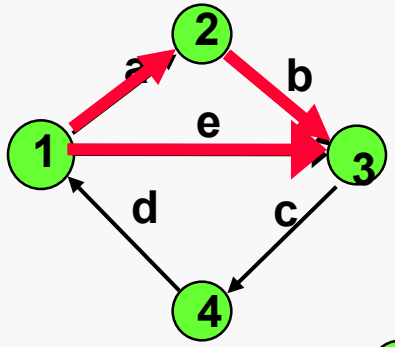
Directed Path **. Example: 1, 2, 5, 3, 4**
(or 1, a, 2, c, 5, d, 3, e, 4)

- **No node is repeated.**
- **Directions are important.**



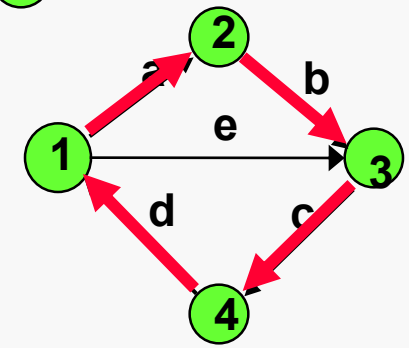
Cycle (or circuit or loop)
1, 2, 3, 1. (or 1, a, 2, b, 3, e)

- **A path with 2 or more nodes, except that the first node is the last node.**
- **Directions are ignored.**

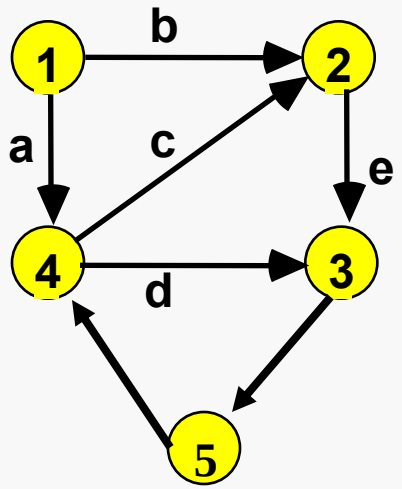


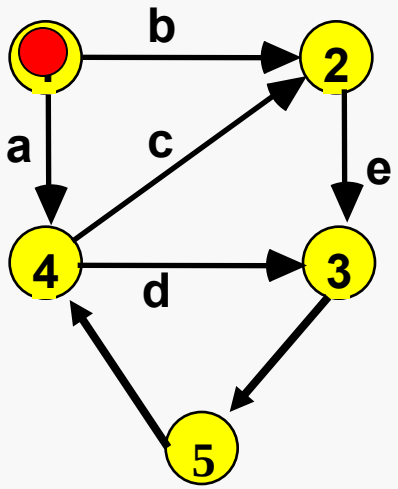
Directed Cycle: (1, 2, 3, 4, 1) or
1, a, 2, b, 3, c, 4, d, 1

- **No node is repeated, except that the first node is the last node.**
- **Directions are important.**



Walks





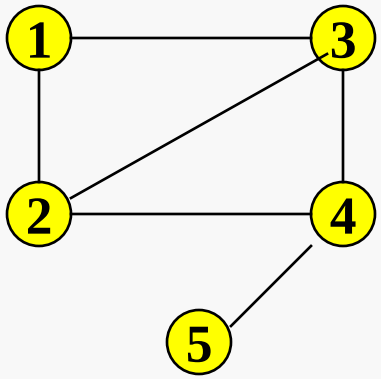
Walks **are paths that can repeat nodes and arcs**

Example of a *directed walk*: 1-2-3-5-4-2-3-5

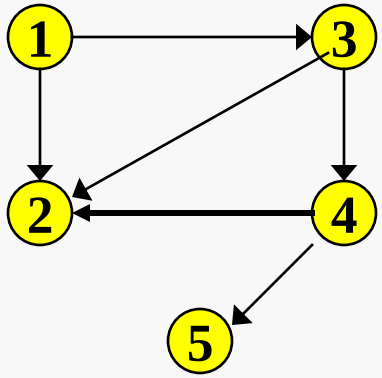
A walk is *closed* if its first and last nodes are the

A closed walk is a cycle except that it can repeat nodes and arcs.

More terminology



An undirected network is *connected* if every node can be reached from every other node by a path

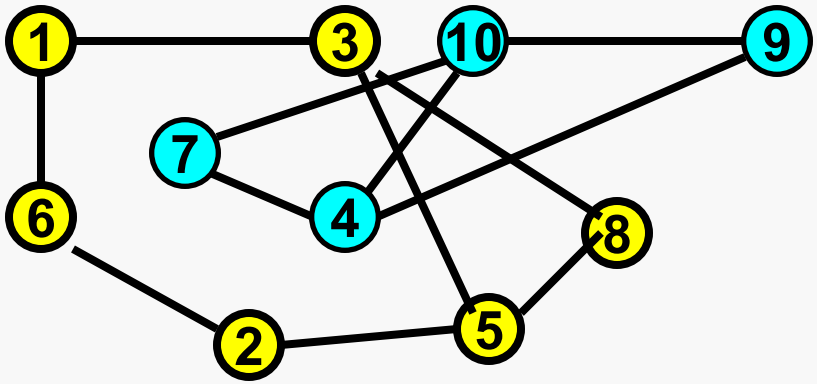


A directed network is *connected* if it's undirected version is connected.

This directed graph is connected, even though there is no directed path between 2 and 5.

On connectivity

There are simple efficient procedures for determining if a graph is connected.



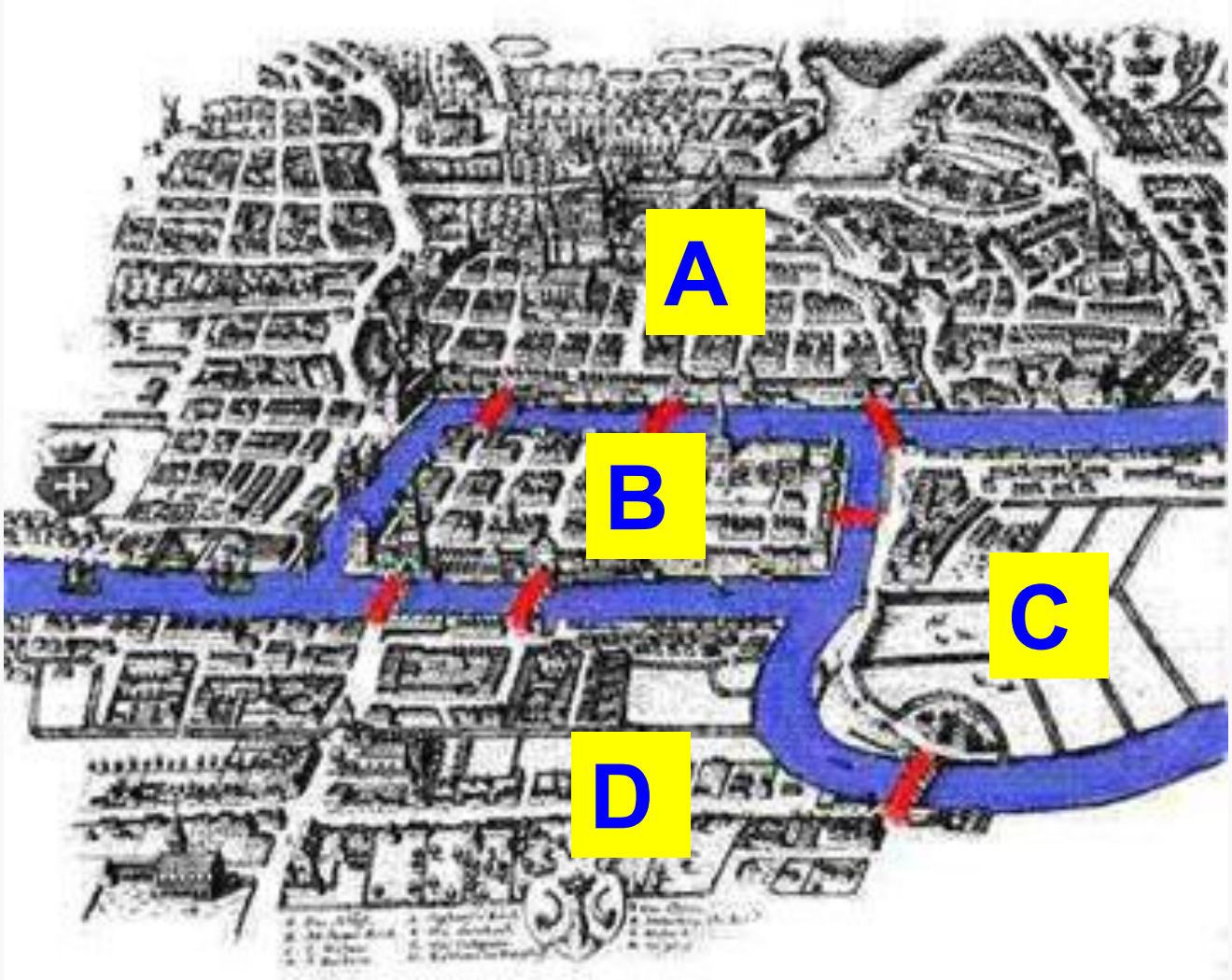
Here is a graph with two *components*, that is maximally connected subgraphs.

We will not describe these algorithms, but will do a more general algorithm later in this lecture

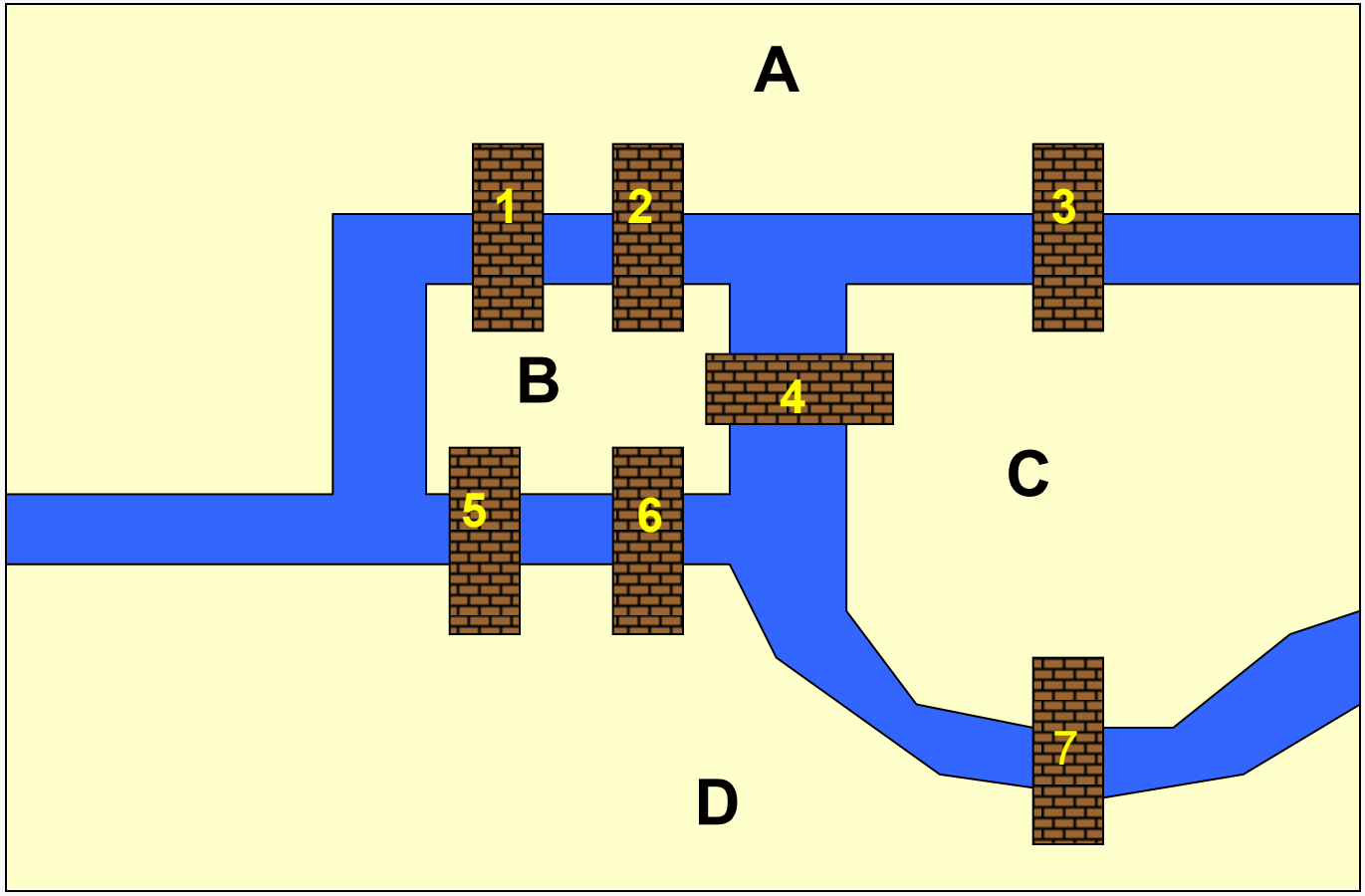
The Bridges of Koenigsberg: Euler 1736

- "Graph Theory" began in 1736
- Leonard Euler
  - Visited Koenigsberg
  - People wondered whether it is possible to take a walk, end up where you started from, and cross each bridge in Koenigsberg exactly once
  - Generally it was believed to be impossible

The town of Koenigsberg

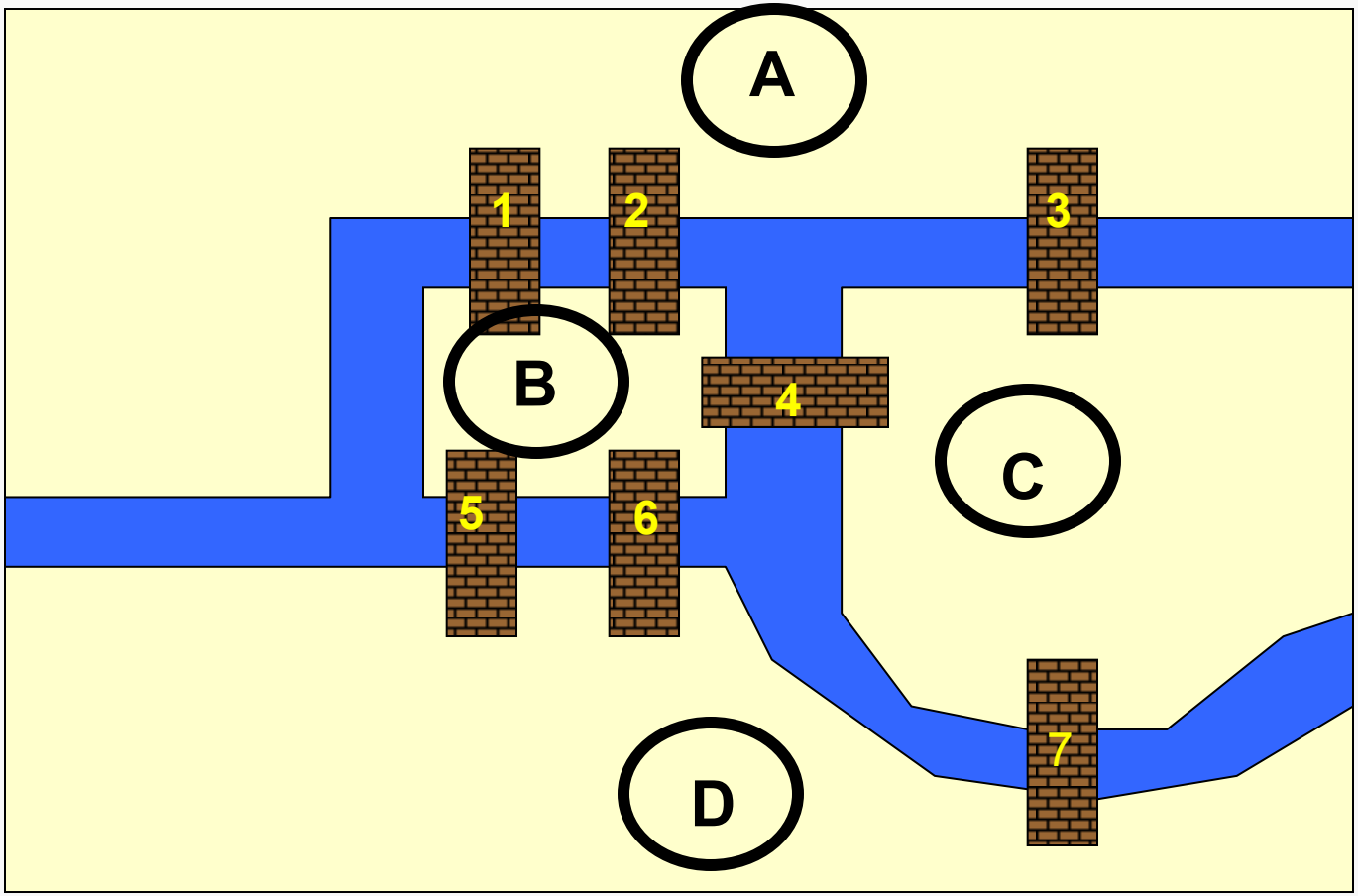


The Bridges of Koenigsberg: Euler 1736



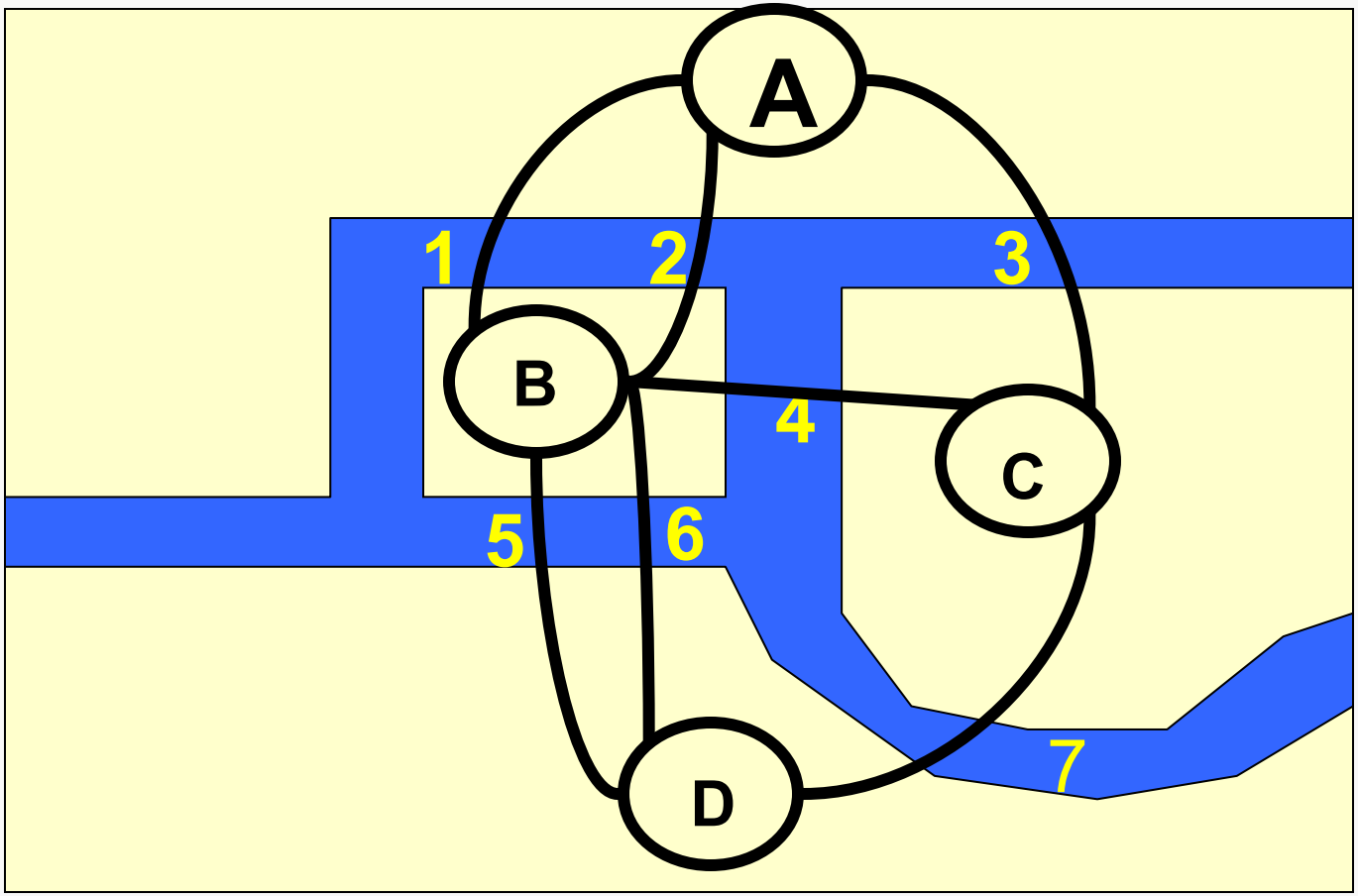
Is it possible to start in A, cross over each bridge exactly once, and end up back in A?

The Bridges of Koenigsberg: Euler 1736



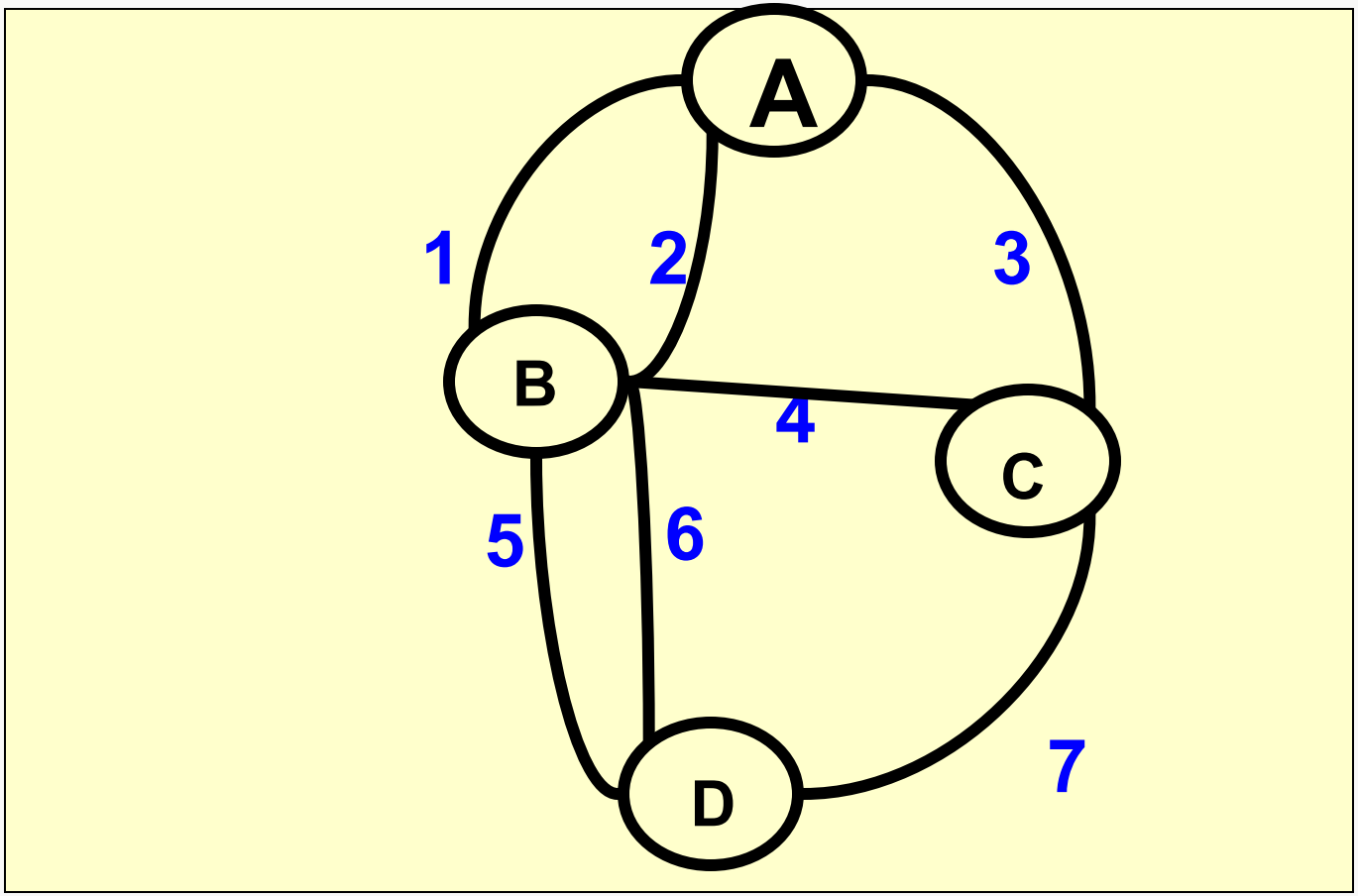
Conceptualization: Land masses are nodes

The Bridges of Koenigsberg: Euler 1736



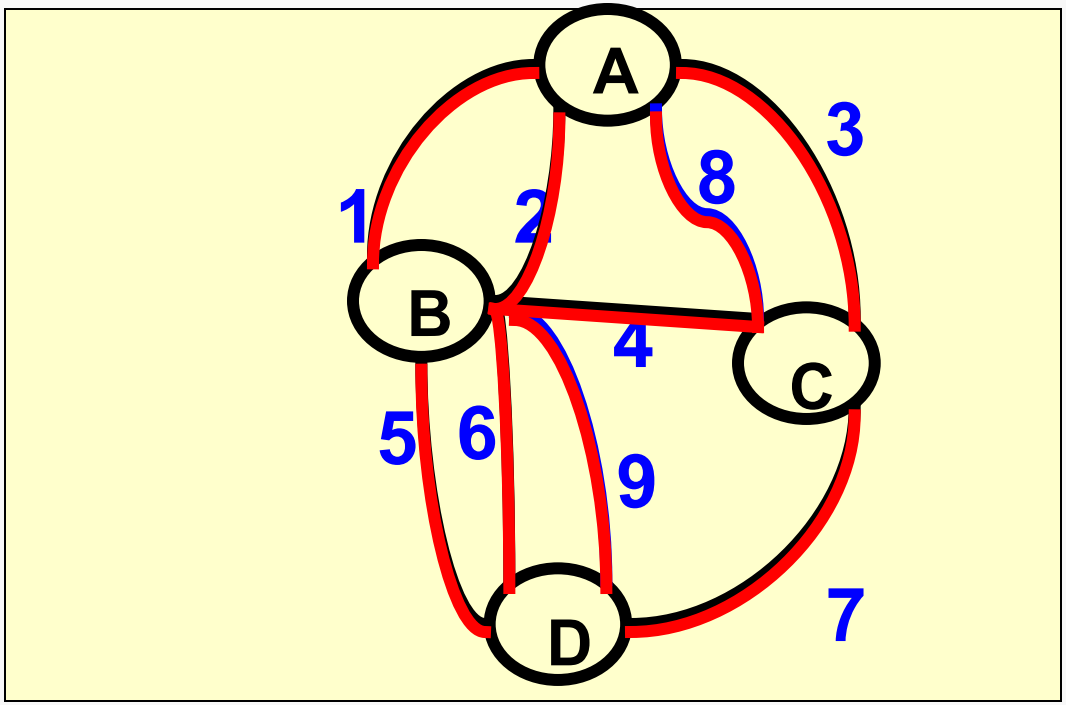
Conceptualization: Bridges are arcs

The Bridges of Koenigsberg: Euler 1736



Translation to graphs or networks: Is there a walk starting at A and ending at A and passing through each arc exactly once? Why isn't there such a walk?

Adding two bridges creates such a walk



Here is the walk.

A, 1, B, 5, D, 6, B, 4, C, 8, A, 3, C, 7, D, 9, B, 2, A

Note: the number of arcs incident to B is twice the number of times that B appears on the walk.

Eulerian cycle: a closed walk that passes through each arc exactly once

- *Degree* of a node = number of arcs incident to the node
- Necessary condition: each node has an even degree.
- Why necessary? The degree of a node j is twice the number of times j appears on the walk (except for the initial and final node of the walk.)

Theorem. *A graph has an eulerian cycle if and only if the graph is connected and every node has even degree.*

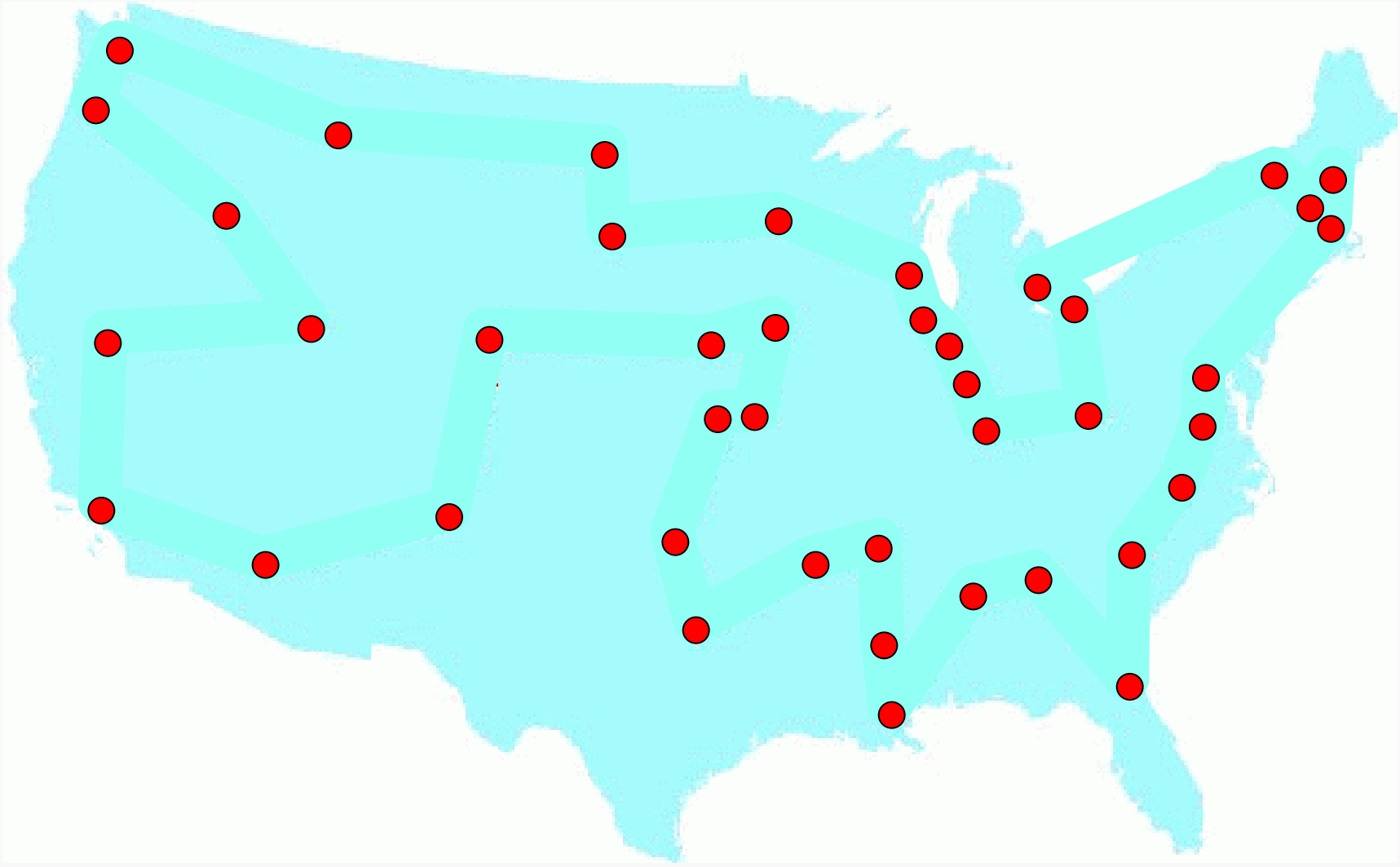
Eulerian path: **a walk that is not closed and passes through each arc exactly once**

Theorem. A graph has an Eulerian path if and only if exactly two nodes have odd degree and the graph is connected.

Eulerian cycles

- Eulerian cycles and extensions are used in practice
- Mail Carrier routes:
  - visit each city block at least once
  - minimize travel time
  - other constraints in practice?
- Trash pickup routes
  - visit each city block at least once
  - minimize travel time
  - other constraints in practice?

Traveling Salesman Problem
The 48 city problem.

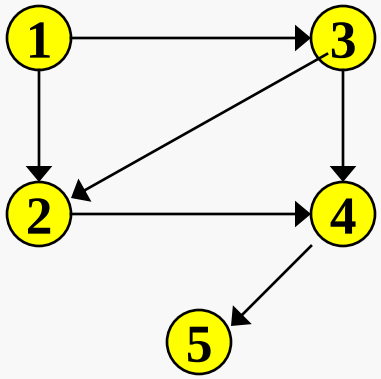


George Dantzig, Ray Fulkerson, and Selmer Johnson (1954)

Mental Break

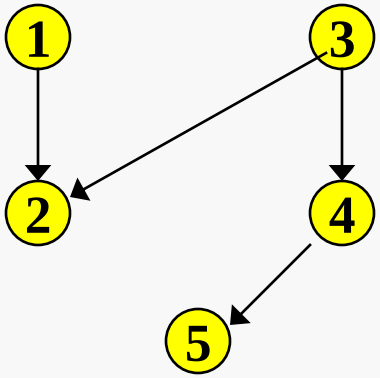
Doodling in Math Class: Snakes + Graphs

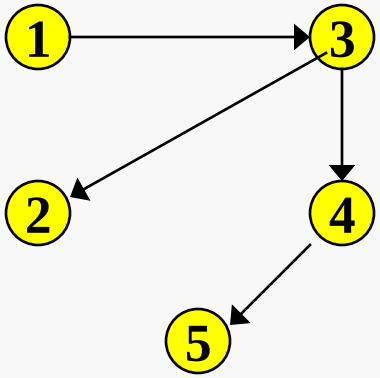
More Definitions

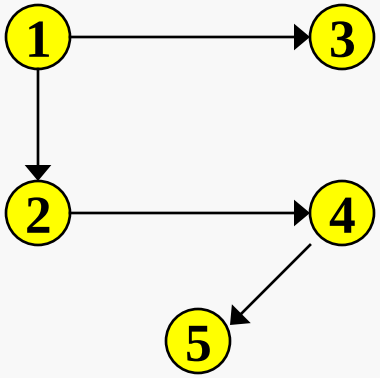


A network is ***connected*** if every node can be reached from every other node by a path

A *spanning tree* is a connected subset of a network including all nodes, but containing no cycles.

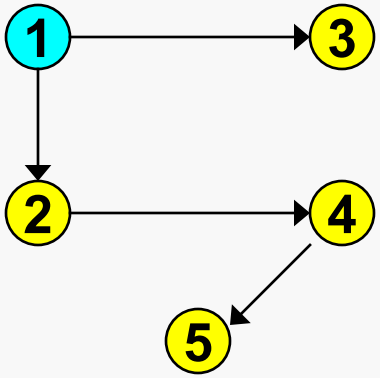


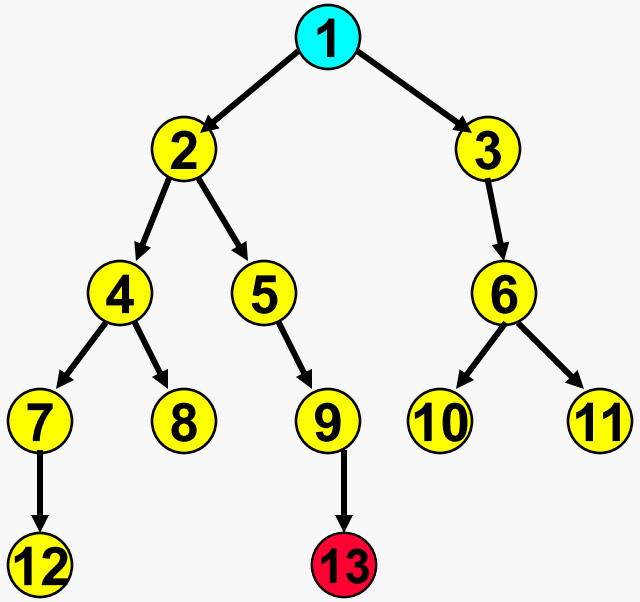




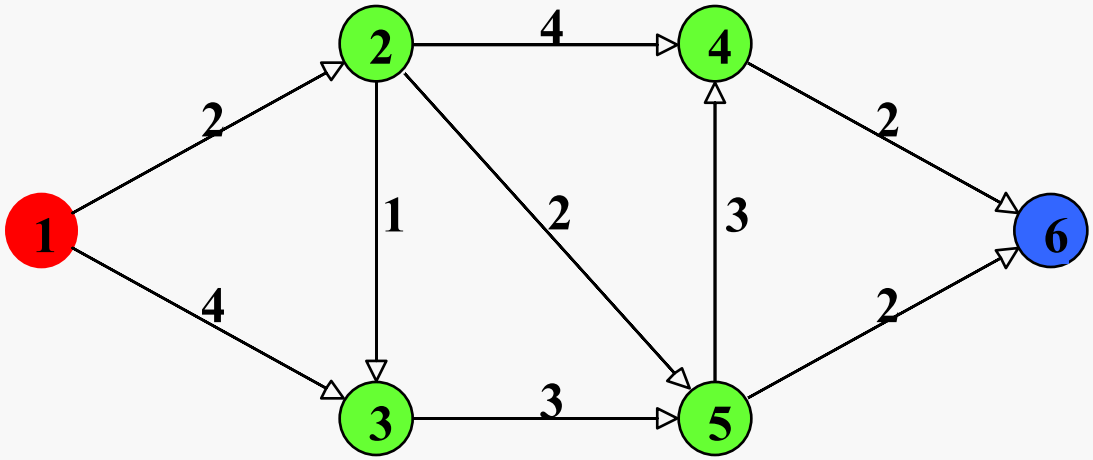
More on Trees

- An out-tree is a spanning tree in which every node has exactly one incoming arc except for the root.
- Theorem. In an out-tree, there is a directed path from the root to all other nodes. (All paths come out of the root).
- One can find the path by starting at the end and working backwards.





The Shortest Path Problem



What is the shortest path from a source node (often denoted as s) to a sink node, (often denoted as t)?

What is the shortest path from node 1 to node 6?

Assumptions for this lecture:

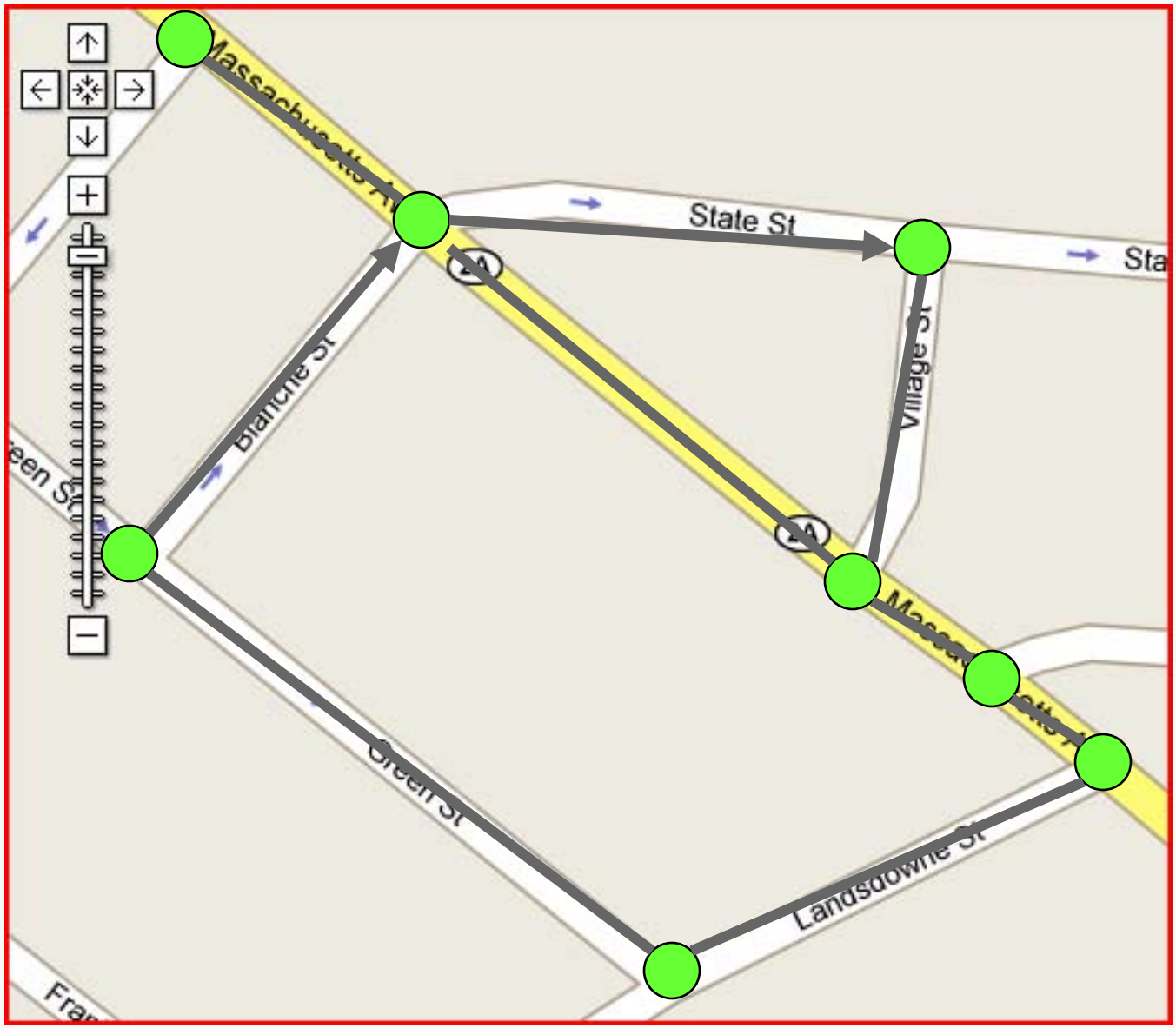
1. **There is a path from the source to all other nodes.**
2. **All arc lengths are non-negative**

Shortest Path Problem

- Where does it arise in practice?
  - Common applications
    - shortest paths in a vehicle (Navigator)
    - shortest paths in internet routing
    - shortest paths around MIT
  - and less obvious applications, as in the course readings (see URL on slide 3 of this lecture).

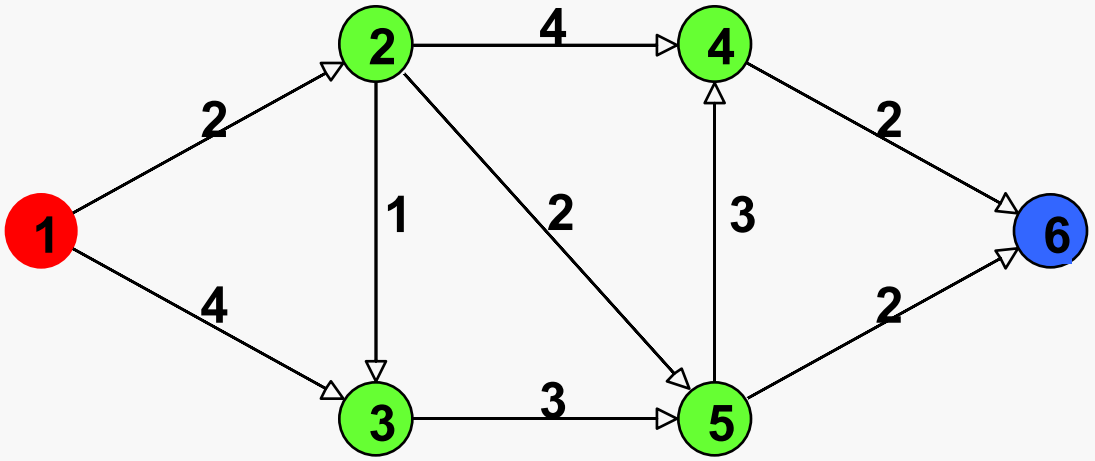
- How will we solve the shortest path problem?
  - Dijkstra's algorithm

Application 1: Shortest paths in a Transportation Network



Add a node for every "intersection".
Add arcs for roads.

Dijkstra's Algorithm



Exercise:
Find the shortest paths by inspection.

Exercise: **find the shortest path from node 1 to all other nodes. Keep track of distances using labels, d(i) and each node's immediate predecessor, pred(i).**

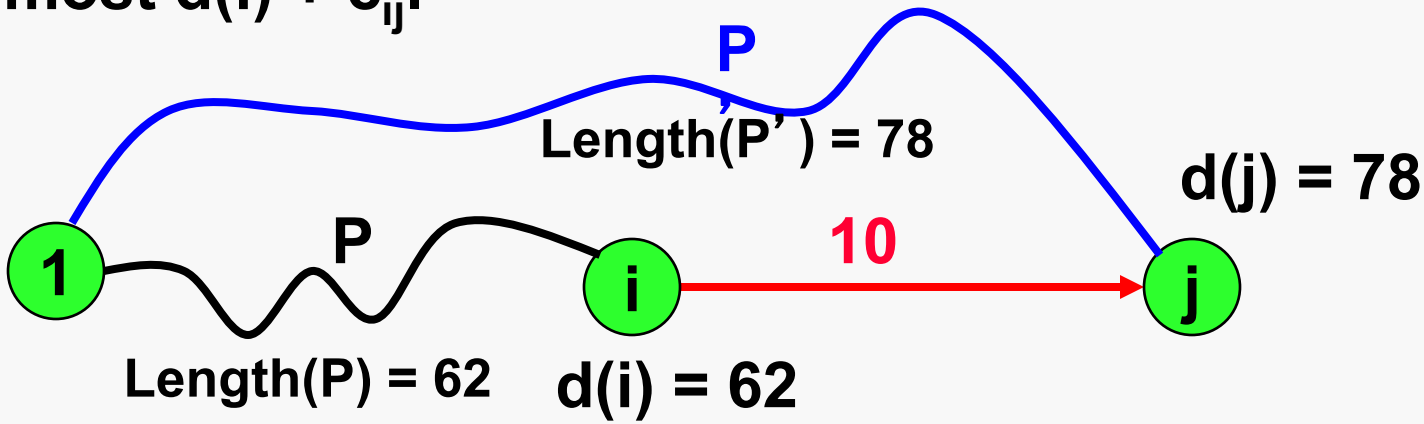
d(1)= 0, pred(1)=0;

d(2) = 2, pred(2)=1

Find the other distances, in order of increasing distance from node 1.

Key observations

- Suppose that d(i) is the length of some path from node 1 to node i.
- Suppose that there is an arc (i, j) of length c_{ij}.
- Then there is a path from node 1 to node j of length at most d(i) + c_{ij}.



In this case, there is a path from 1 to j of length 72. We can reduce d(j) to 72.

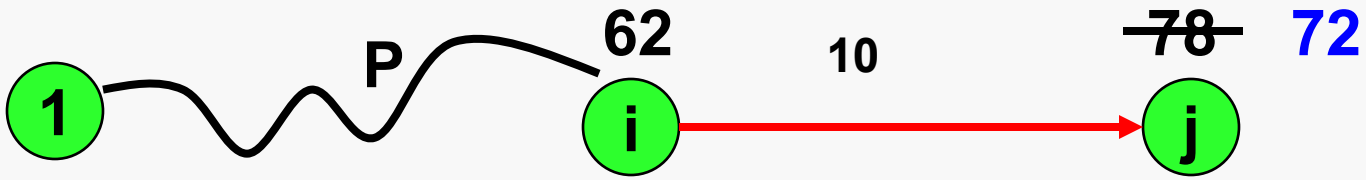
A Key Procedure in Shortest Path Algorithms

- At each iteration d(j) is the length of some path from node 1 to node j. (If no path is known, then d(j) = ∅)

Procedure Update(i)

for each (i,j) ∈ A(i) do

if d(j) > d(i) + c_{ij} then d(j) : = d(i) + c_{ij} and pred(j) : = i;



Up to this point, the best path from 1 to j had length 78

But P, (i,j) is a path from 1 to j of length 72.

Dijkstra's Algorithm

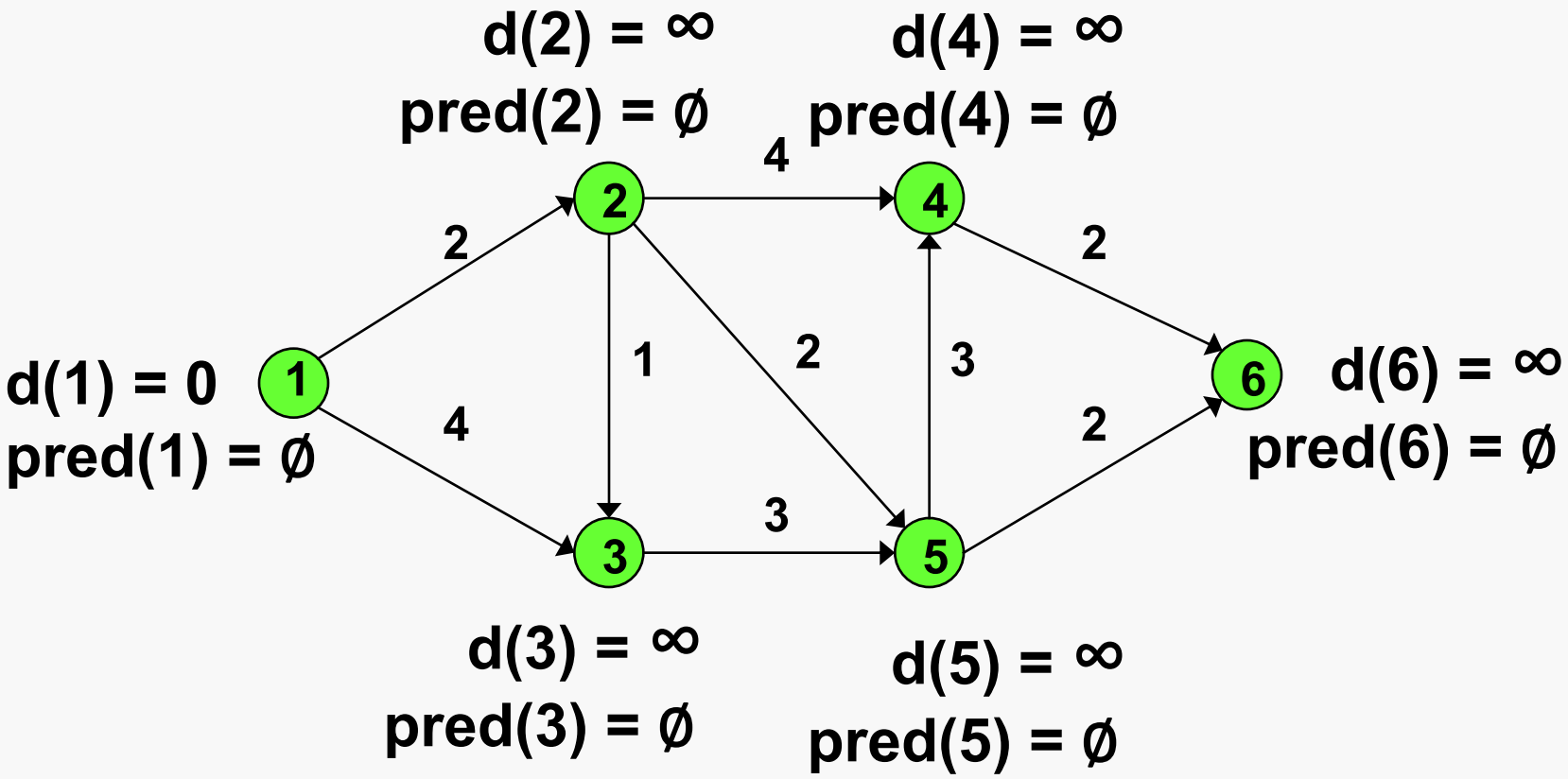
```
begin
    d(s) : = 0 and pred(s) : = 0;
    d(j) : = ∅ for each j ∈ N - {s};
    LIST : = {s};
    while  LIST  ≠ ∅ do
    begin
          let  d(i)  : =  min {d(j) : j ∈ LIST};
          remove node i from LIST;
          update(i)
          if d(j) decreases from ∞,
             place j inLIST
    end
end
```

Initialize distances.

LIST = set of temporary nodes

Select the node i on LIST with minimum distance label, and then update(i)

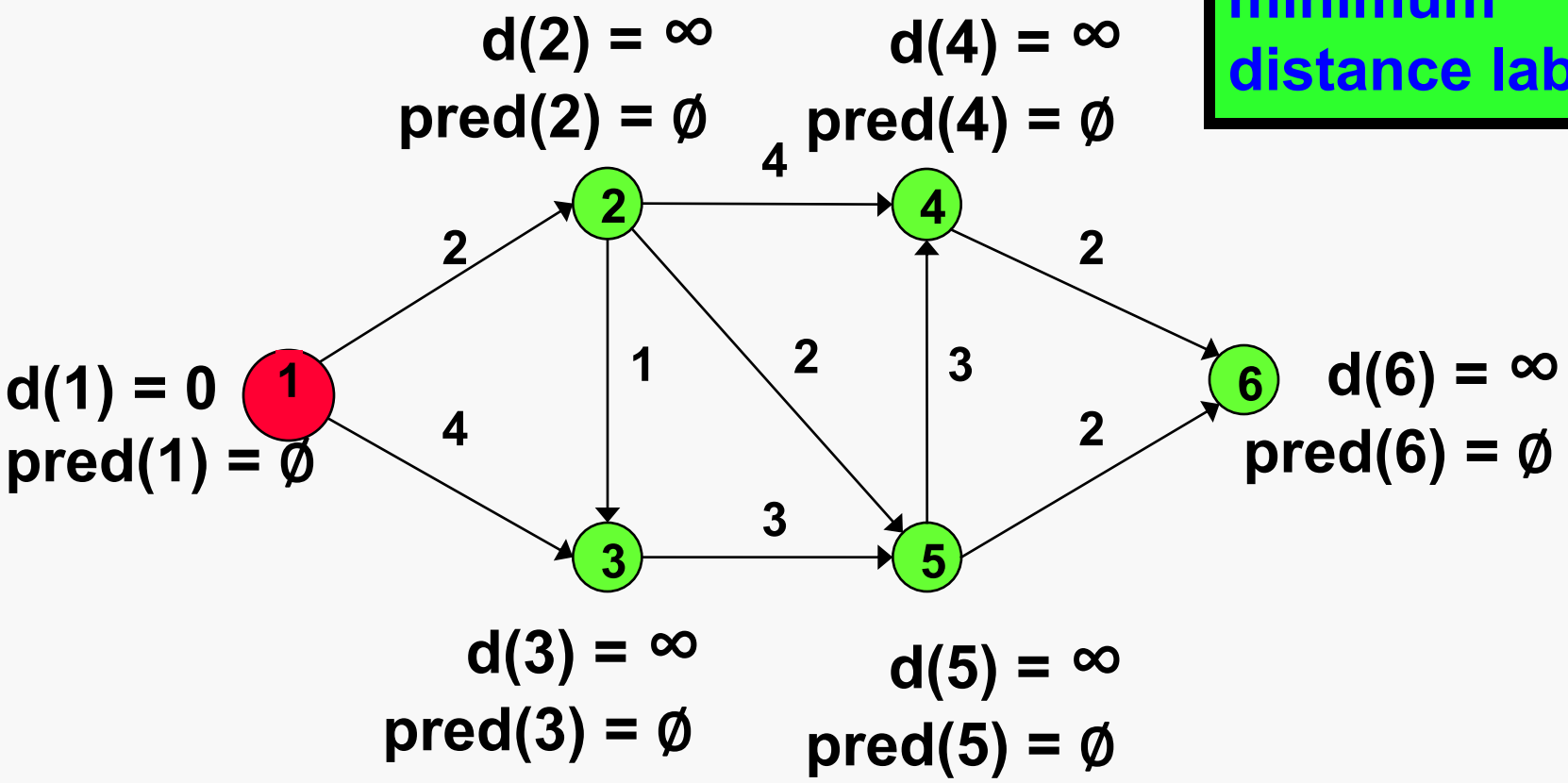
Initialize



LIST = {1,

d( ) = {0,

Find the node i on LIST with minimum distance label.

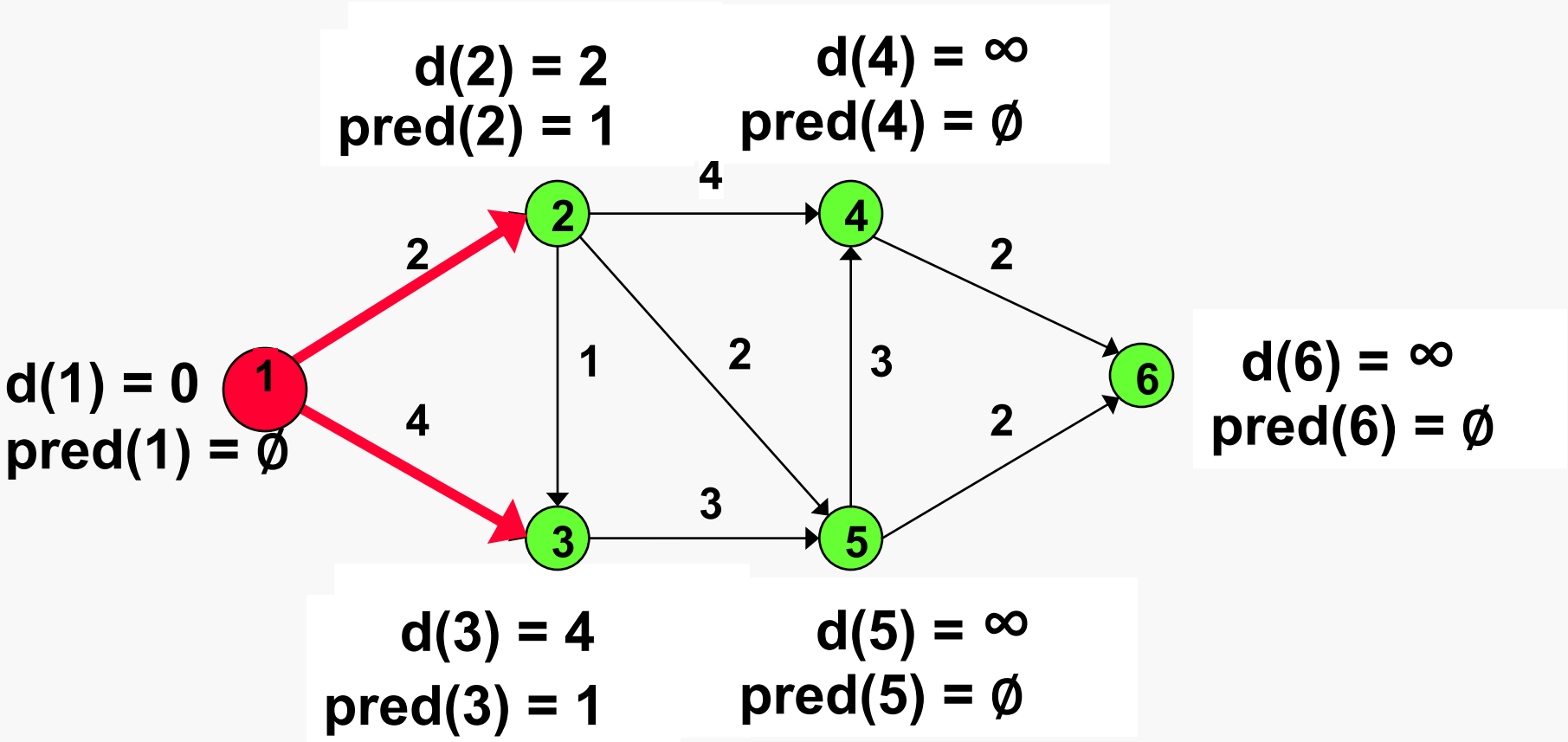


LIST = {

d( ) = {

Remove i from LIST. Make i permanent.

update(1)



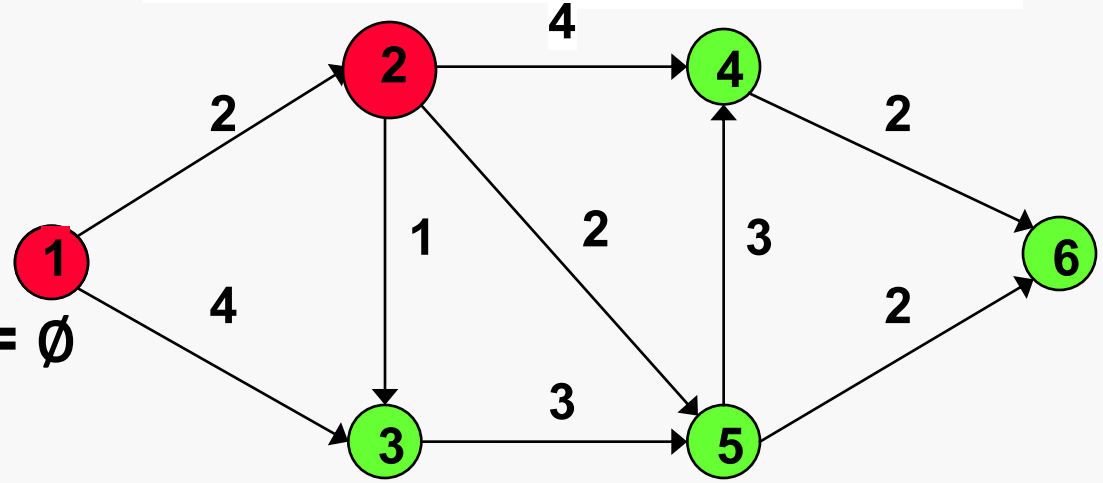
LIST = { 2, 3

d( ) = { 2, 4

Find the node i on LIST with minimum distance label.

d(2) = 2
pred(2) = 1

d(4) = ∞
pred(4) = ∅



d(1) = 0
pred(1) = ∅

d(6) = ∞
pred(6) = ∅

d(3) = 4
pred(3) = 1

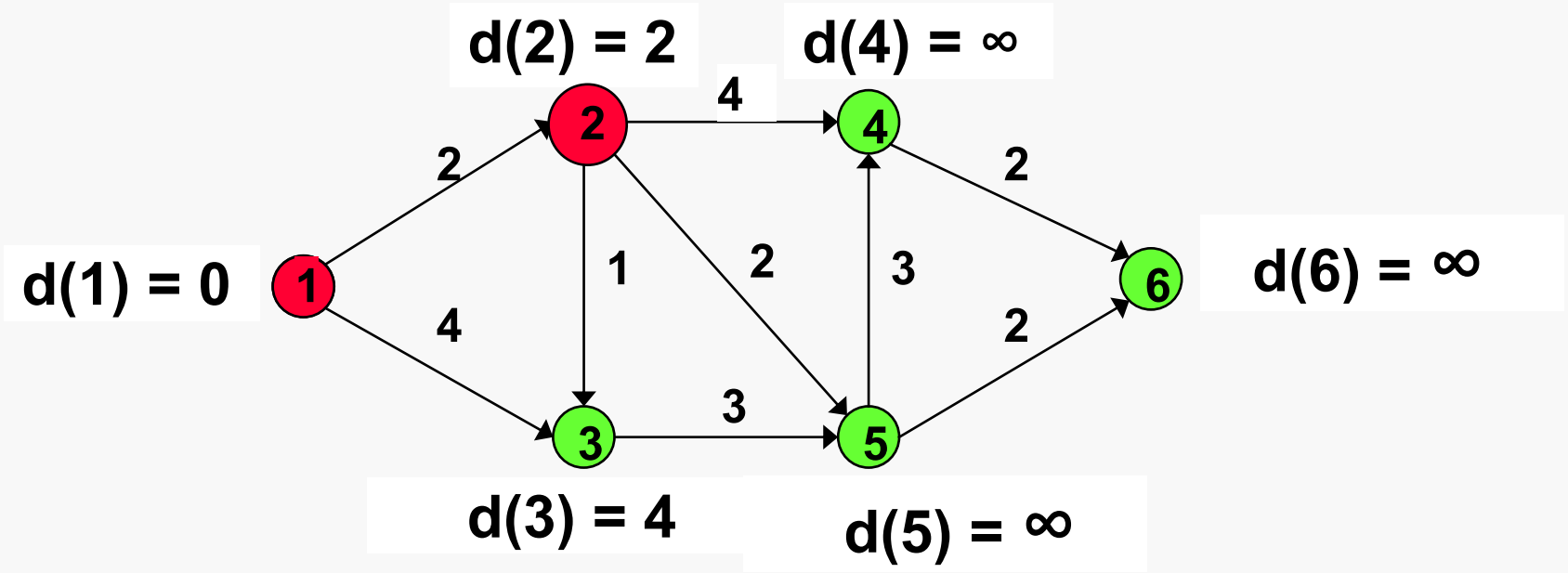
d(5) = ∞
pred(5) = ∅

LIST = { 3

d( ) = { 4

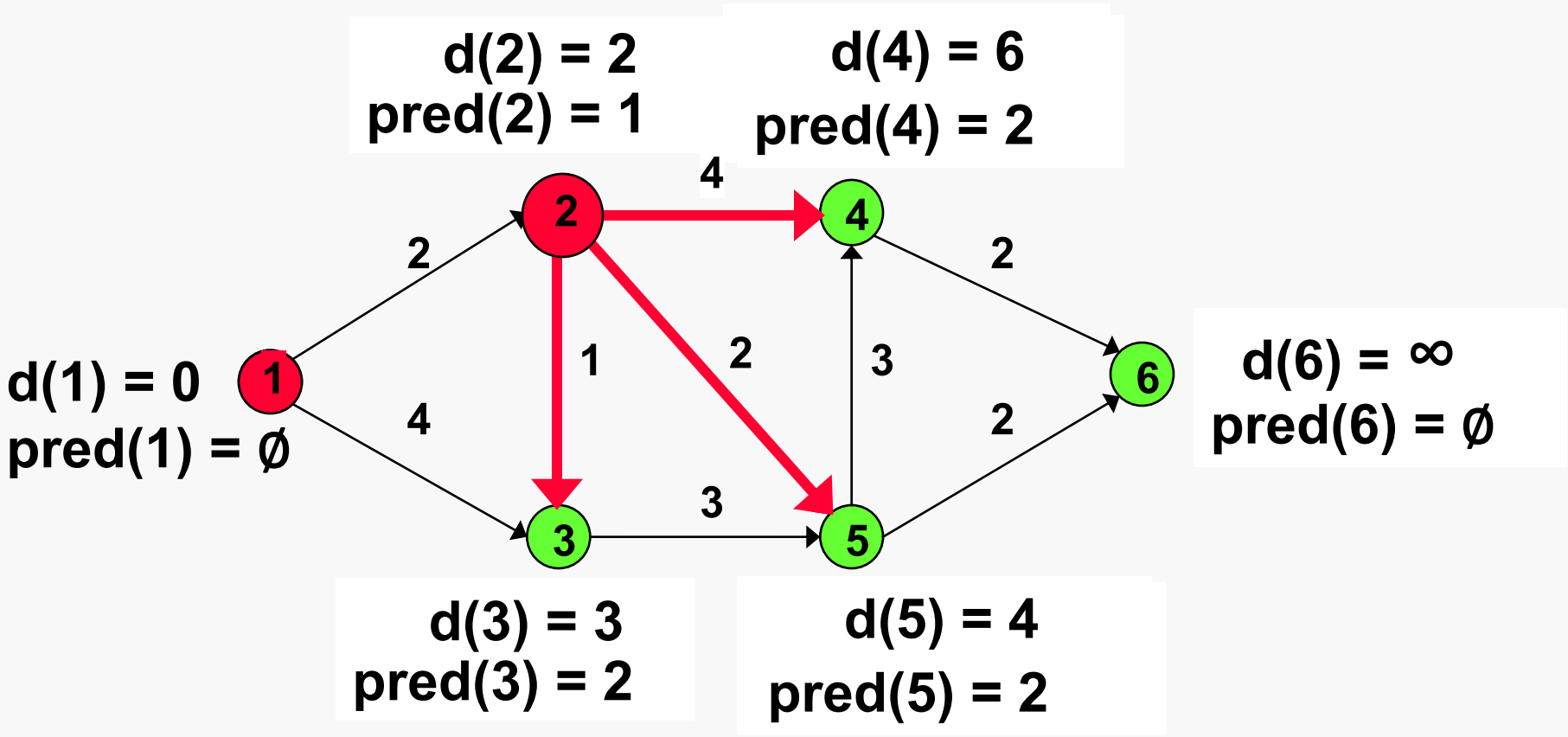
Remove i from LIST. Make i permanent.

Arcs (2, 3), (2, 4) and (2,5) will be scanned next. Which nodes will have their distance label changed?



1. 2, 3, 4 and 5
2. 3, 4, and 5
3. 4 and 5
4. none of the above.

Update(2)



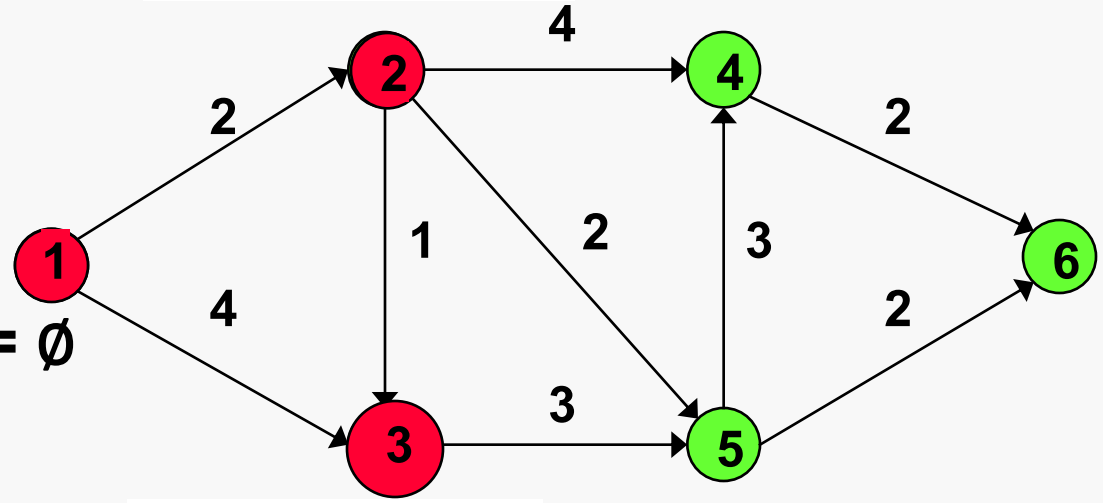
LIST = { 3, 4, 5

d( ) = { 3, 6, 4

Find the node i on LIST with minimum distance label.

d(2) = 2
pred(2) = 1

d(4) = 6
pred(4) = 2



d(1) = 0
pred(1) = ∅

d(6) = ∞
pred(6) = ∅

d(3) = 3
pred(3) = 2

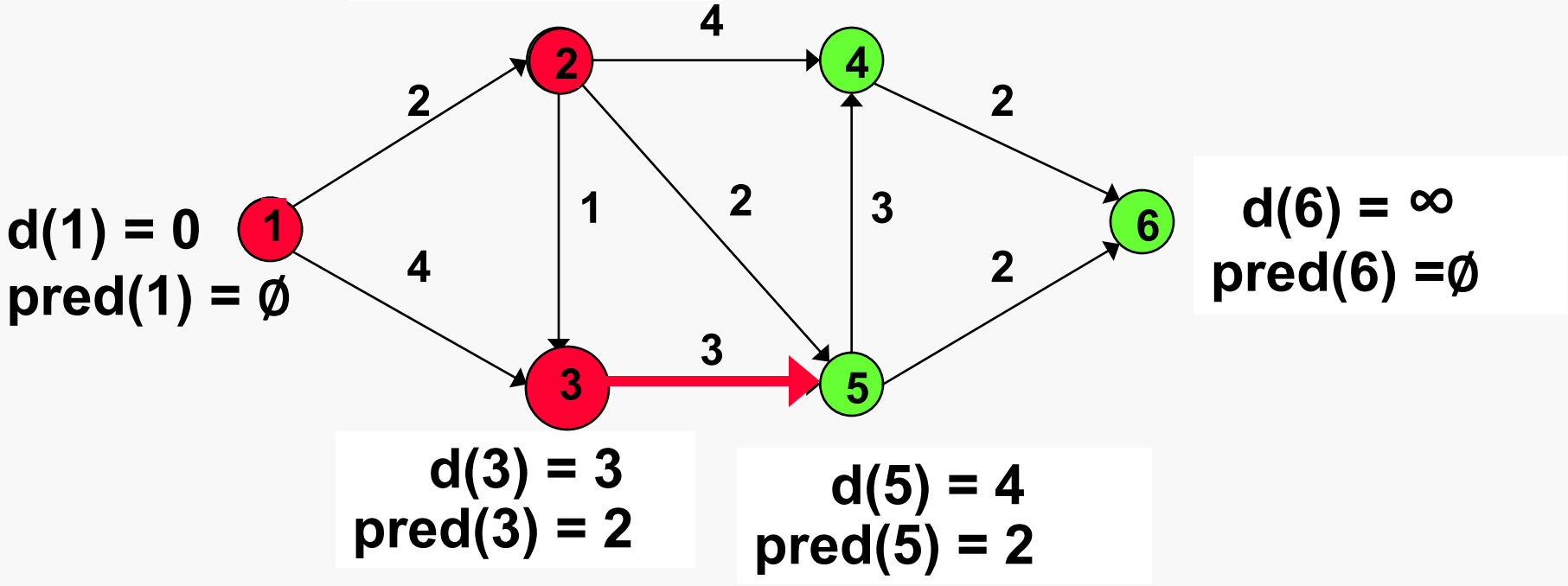
d(5) = 4
pred(5) = 2

LIST = {4, 5

d( ) = {6, 4

Remove i from LIST. Make i permanent.

Update(3)



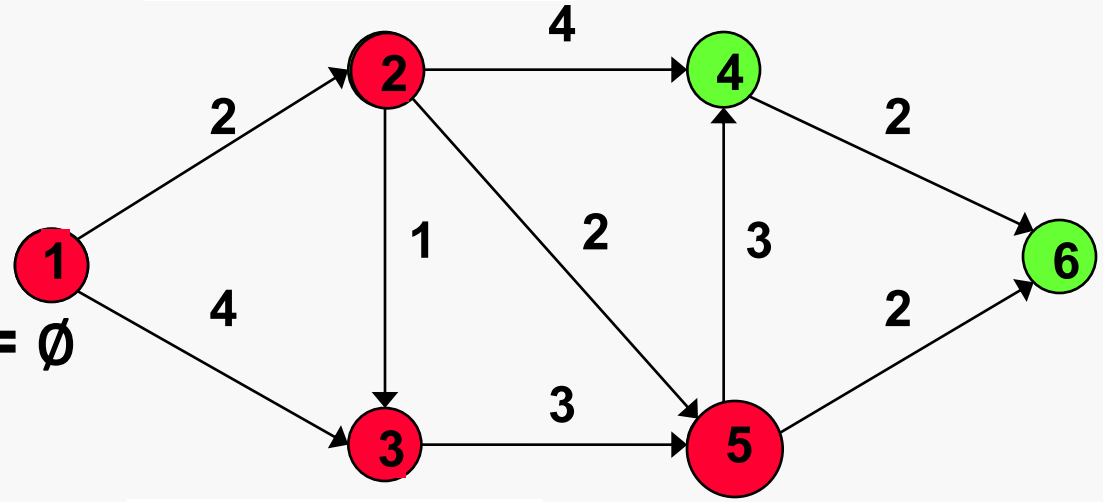
LIST = {4, 5

d( ) = {6, 4

Find the node i on LIST with minimum distance label.

d(2) = 2
pred(2) = 1

d(4) = 6
pred(4) = 2



d(1) = 0
pred(1) = ∅

d(6) = ∞
pred(6) = ∅

d(3) = 3
pred(3) = 2

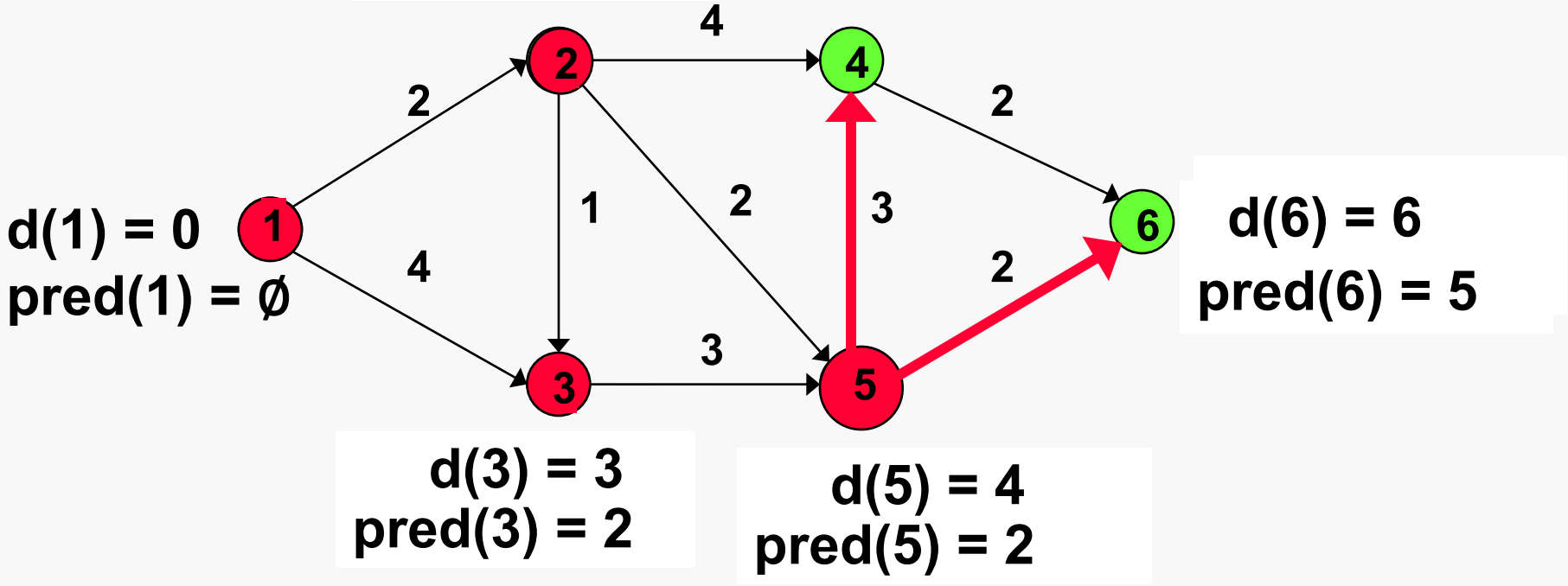
d(5) = 4
pred(5) = 2

LIST = {4

d( ) = {6

Remove i from LIST. Make i permanent.

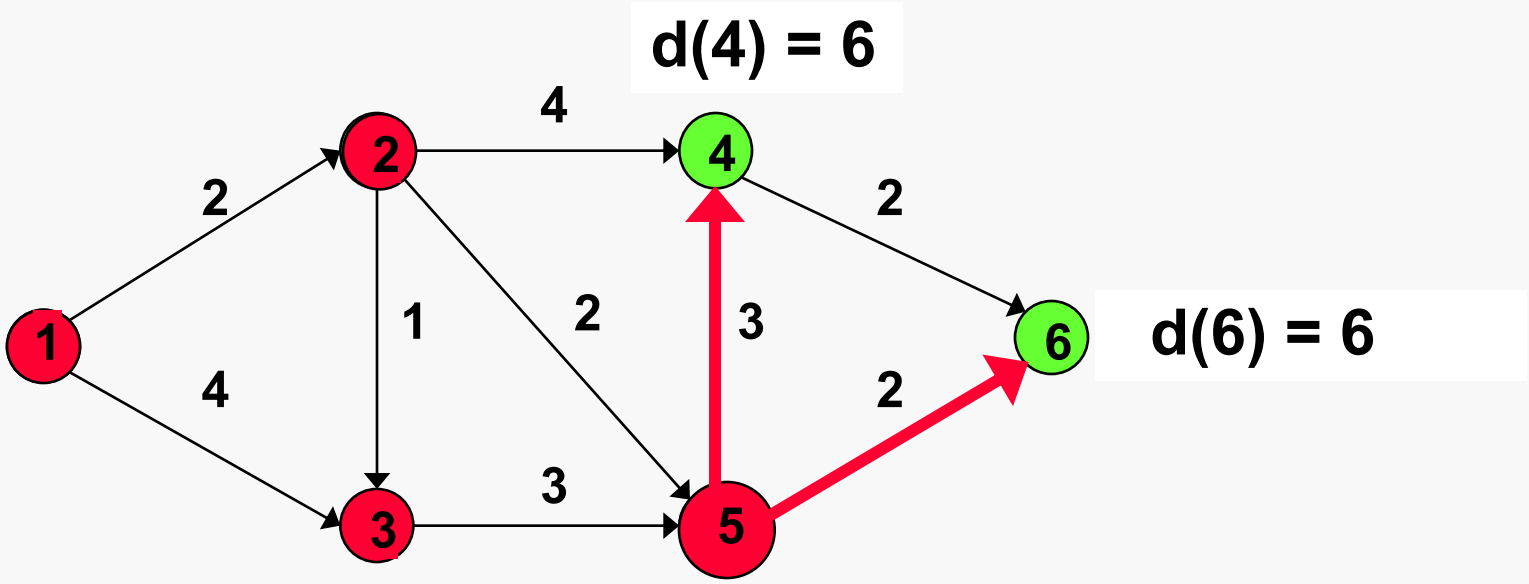
Update(5)



LIST = {4, 6

d( ) = {6, 6

Which node will be scanned next according to the usual rule?

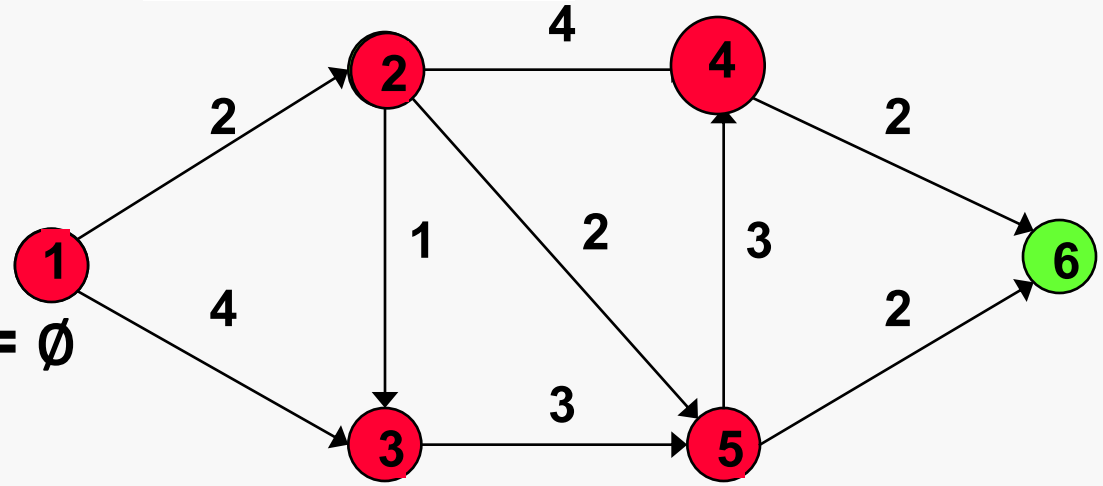


1. node 4
2. node 6
3. either node 4 or node 6; both choices are OK.

Find the node i on LIST with minimum distance label.

d(2) = 2
pred(2) = 1

d(4) = 6
pred(4) = 2



d(1) = 0
pred(1) = ∅

d(6) = 6
pred(6) = 5

d(3) = 3
pred(3) = 2

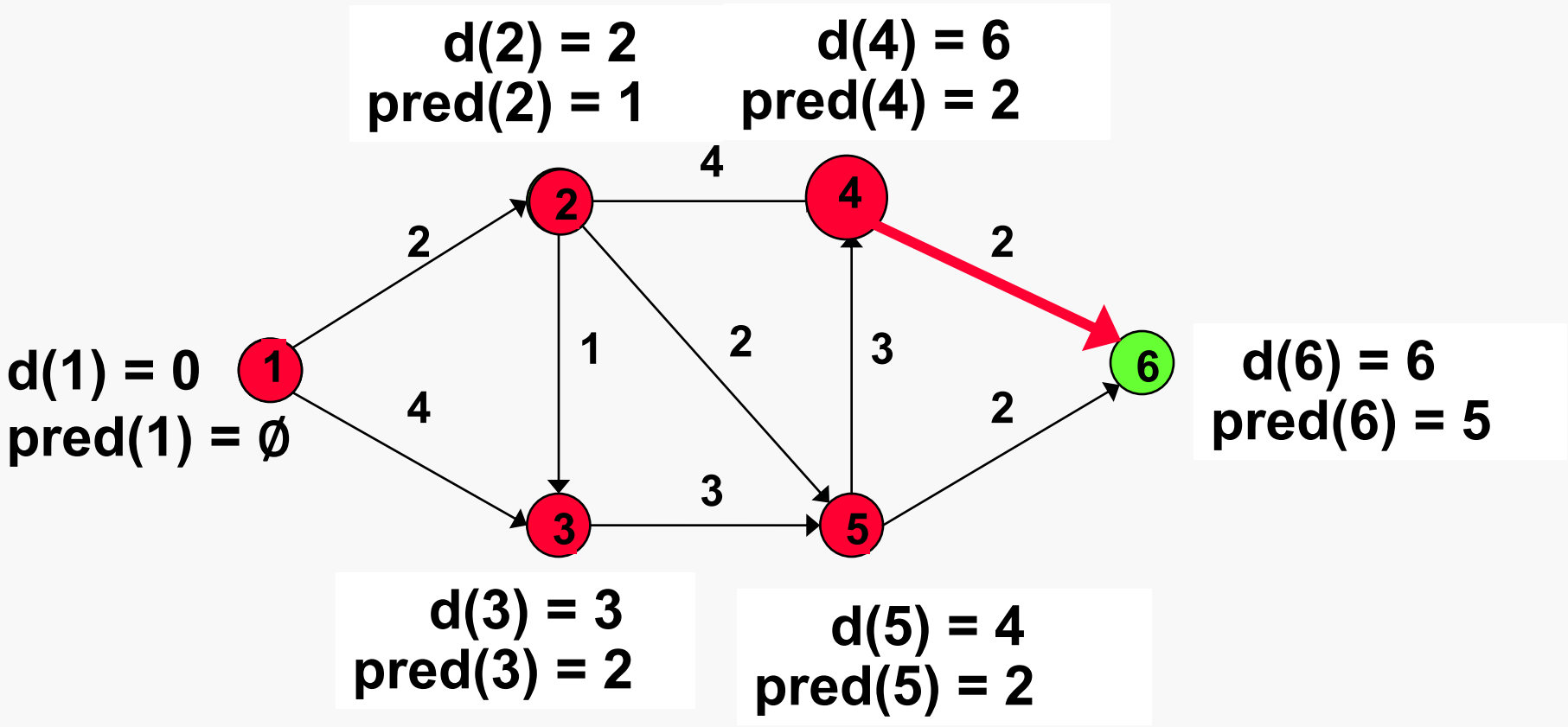
d(5) = 4
pred(5) = 2

LIST = {6

d( ) = {6

Remove i from LIST. Make i permanent.

Update(4)



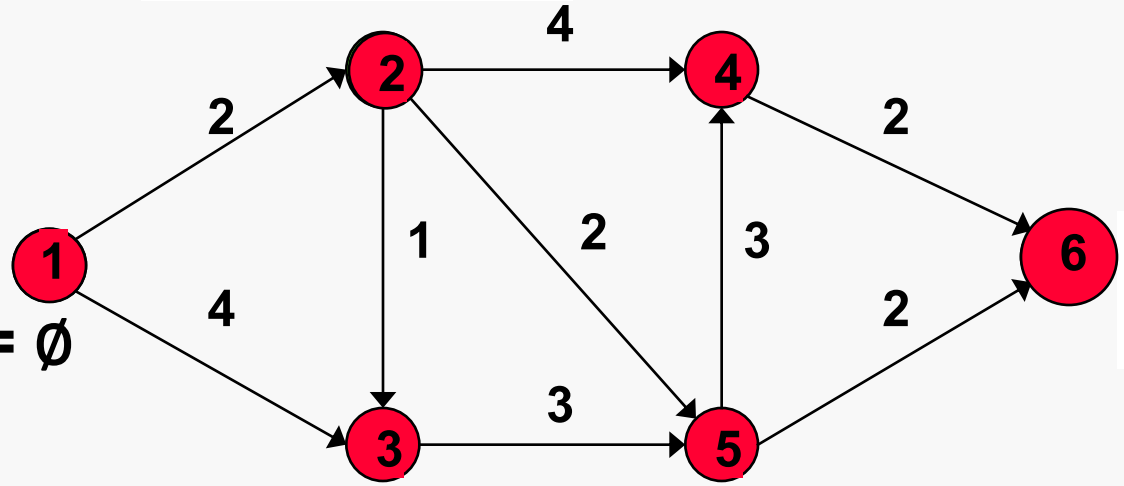
LIST = {6

d( ) = {6

Find the node i on LIST with minimum distance label.

d(2) = 2
pred(2) = 1

d(4) = 6
pred(4) = 2



d(1) = 0
pred(1) = ∅

d(6) = 6
pred(6) = 5

d(3) = 3
pred(3) = 2

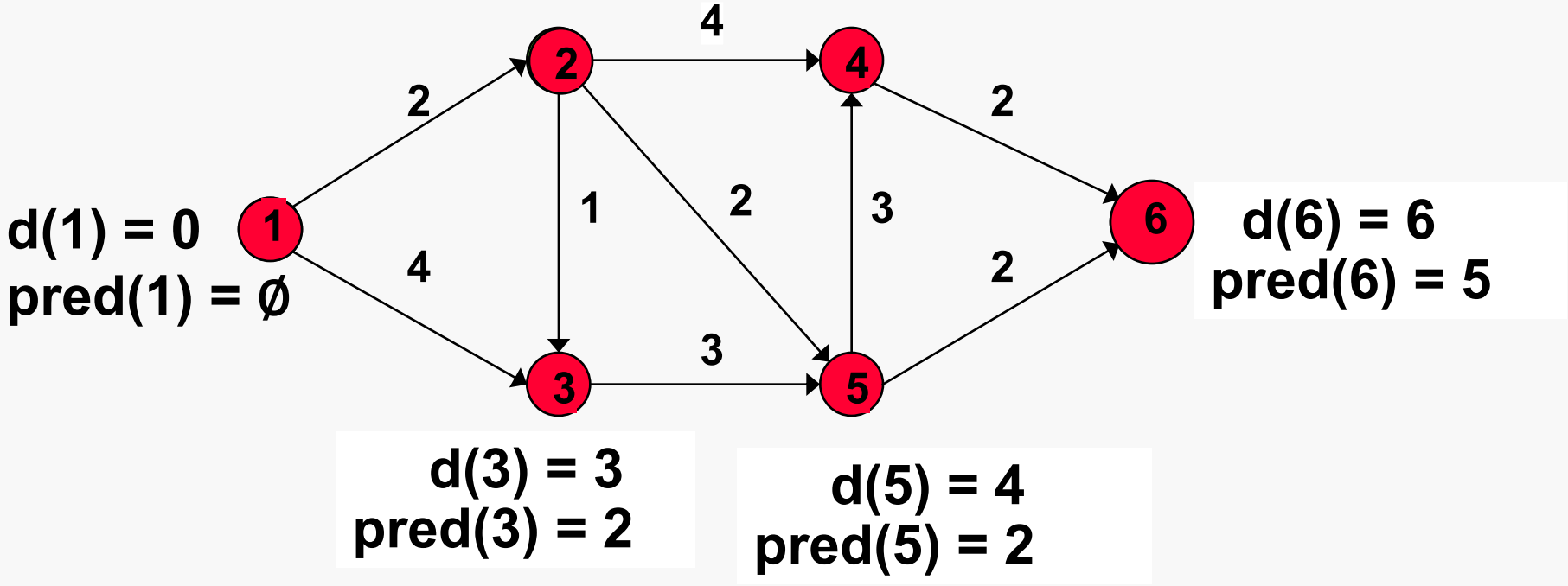
d(5) = 4
pred(5) = 2

LIST = {

d( ) = {

Remove i from LIST. Make i permanent.

Update(6)



LIST = {

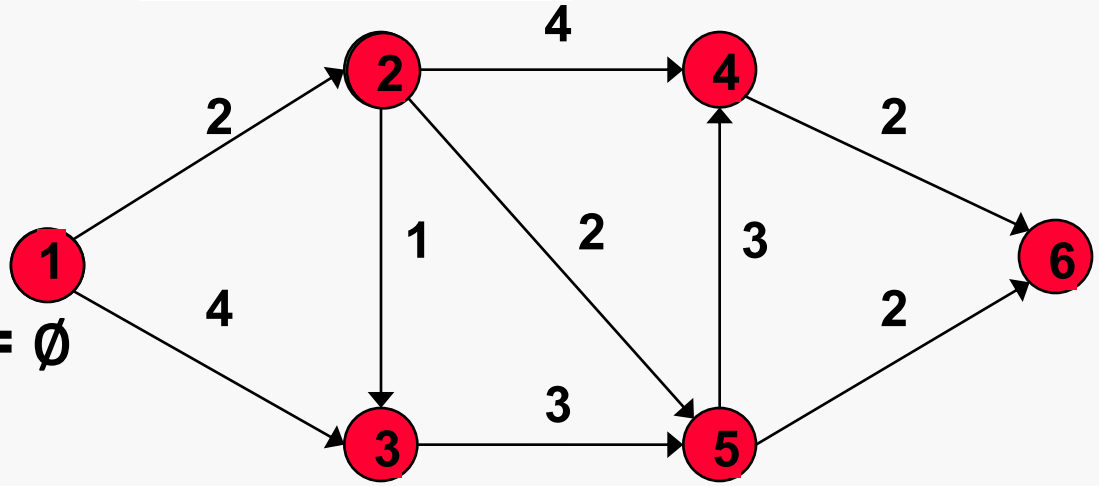
d( ) = {

Node 6 has no outgoing arcs.

Find the node i on LIST with minimum distance label.

d(2) = 2
pred(2) = 1

d(4) = 6
pred(4) = 2



d(1) = 0
pred(1) = ∅

d(6) = 6
pred(6) = 5

d(3) = 3
pred(3) = 2

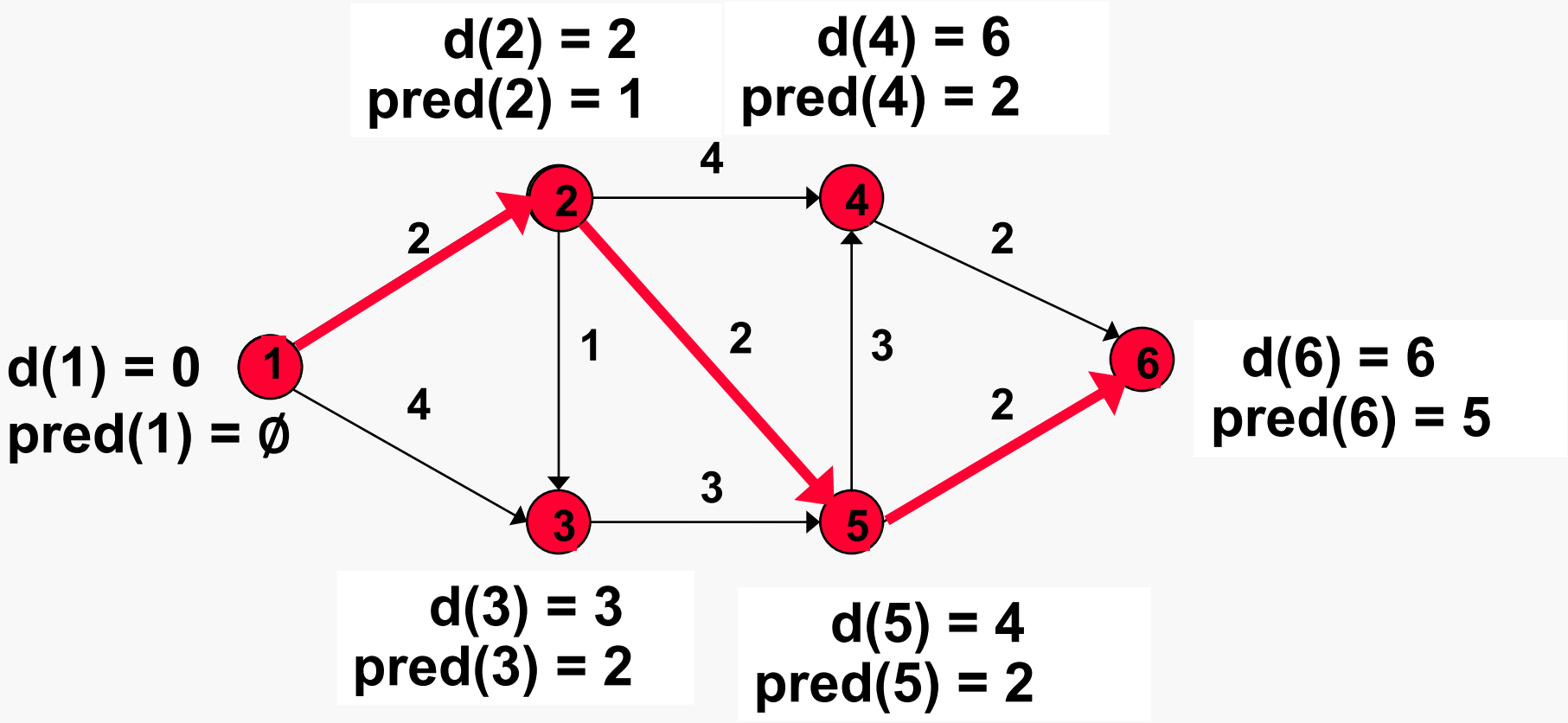
d(5) = 4
pred(5) = 2

LIST = {

d( ) = {

LIST = ∅. The algorithm ends.

The shortest path from node 1 to node 6.

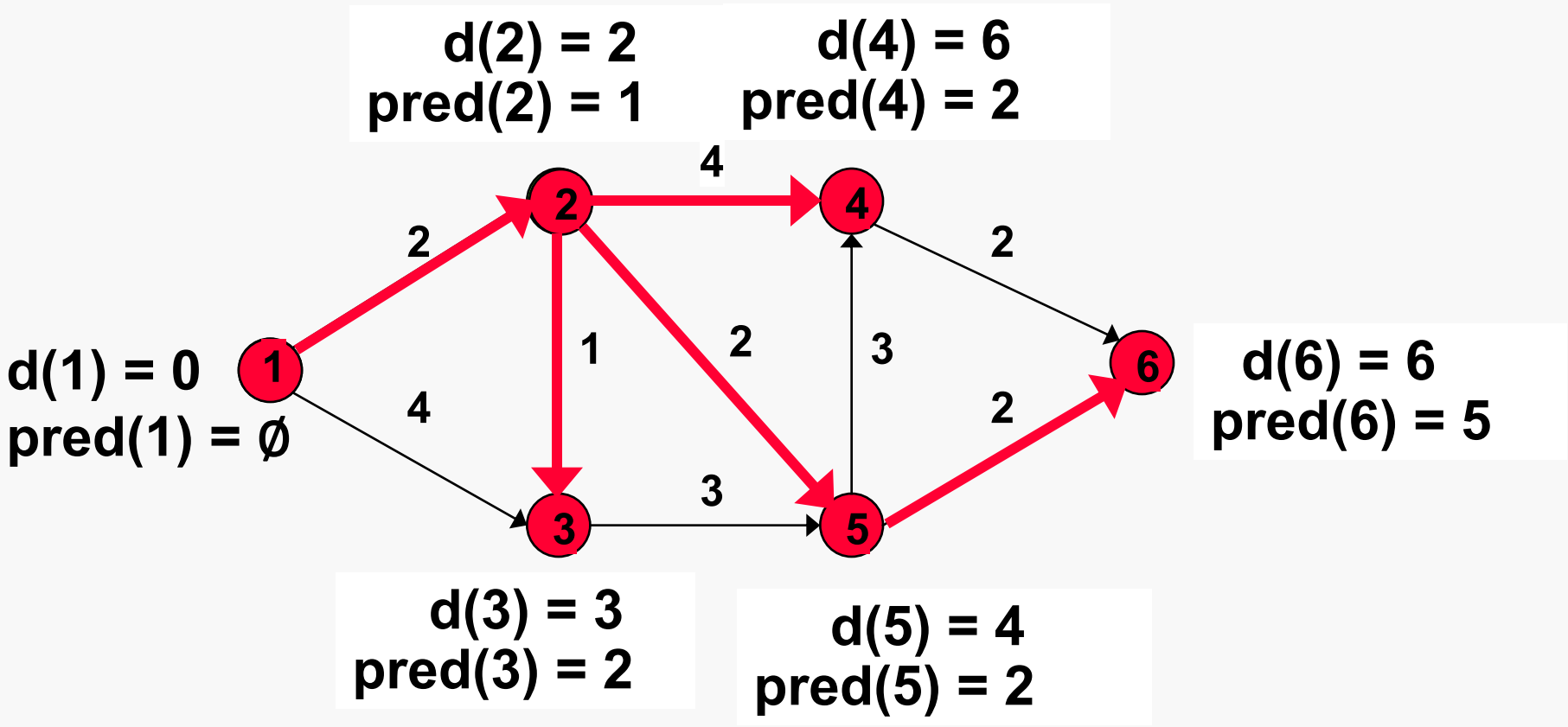


LIST = {

d( ) = {

Trace back the path from node 6 to node 1 using the predecessors.

The shortest path from node 1 to node 6.



LIST = {

d( ) = {

The "predecessor" arcs form an out-tree rooted at node 1.

Comments on Dijkstra's Algorithm

- Dijkstra's algorithm makes nodes permanent in increasing order of distance from the origin node.
- Dijkstra's algorithm is efficient in its current form. The running time grows as n^2, where n is the number of nodes
- It can be made much more efficient
- In practice it runs in time linear in the number of arcs (or almost so).

Edsger Dijkstra

1930-2002

Turing Prize 1972

- **development of Algol**
- **programming languages**
- **graph theory**

http://en.wikipedia.org/wiki/Edsger_W._Dijkstra

Summary

- The Eulerian cycle problem
- The shortest path problem
- Dijkstra's algorithm finds the shortest path from node 1 to all other nodes in increasing order of distance from the source node.
- The bottleneck operation is identifying the minimum distance label. One can speed this up, and get an incredibly efficient algorithm

15.053 Optimization Methods in Management Science

Spring 2013