# Review



## Problems we have covered so far:

- Shortest path problem
- Production and inventory control (Wagner–Whitin)
- Resource allocation (e.g. knapsack)
- Travelling Salesperson Problem
- Pattern recognition (e.g. DNA comparison)

## Today

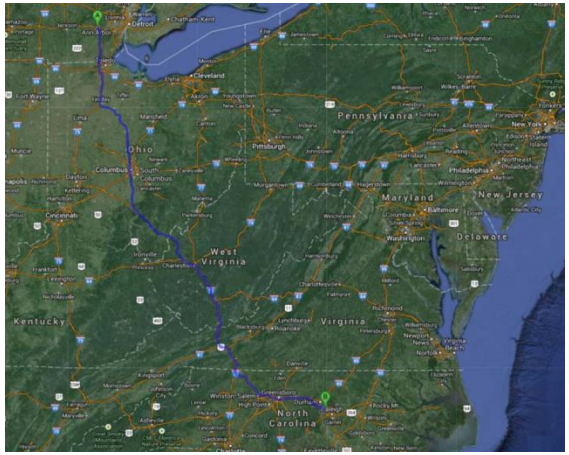
- Shortest path problems on acyclic and general networks

# Review

DPs can be used for finding the shortest path that joins two points in a network

- This is a fundamental problem to which DP applies (e.g. google maps)

- Many problems can be formulated as a shortest path problem




**Ann Arbor, MI**

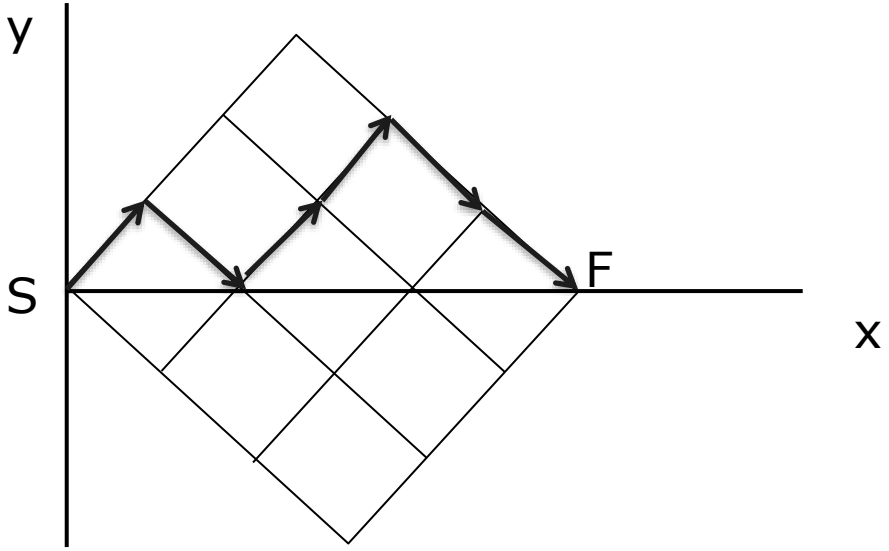
1. Head **south** toward **E Washington St**
2. Turn left onto **E Washington St**
3. Take the 1st left onto **S State St**
4. Take the 1st right onto **E Huron St**
5. Continue onto **Washtenaw Ave**
6. Slight right to merge onto **US-23 S** toward **Interstate 94/Toledo/Detroit**
Entering Ohio
7. Merge onto **I-475 S/US-23 S**
8. Take the exit onto **I-75 S** toward **Dayton**
9. Take exit **156** to merge onto **OH-15 E/US-68 S** toward **US-23/Kenton/Columbus**
10. Continue straight onto **OH-15 E**
11. Continue onto **US-23 S**
12. Slight left to stay on **US-23 S** (signs for **Columbus**)
13. Merge onto **I-270 W** via the ramp to **Indianapolis**
14. Take exit **22** to merge onto **OH-315 S** toward **Columbus**
15. Keep left at the fork, follow signs for **Interstate 71 S/Cincinnati** and merge onto **I-71 S**

# Review

General formulation for shortest path problems on a directed acyclic network:

- The state is denoted by coordinates (x,y)
- Admissible paths move to the right, guaranteeing no cycles

Example:



# Review

Dynamic Programming Formulation:

- Let $c^+(x,y)$ be the cost of moving from state (x,y) to (x+1, y+1) and let $c^-(x,y)$ be the cost of moving from state (x,y) to (x+1, y-1)

- Let $(x^F,y^F)$ denote the terminal (destination) state

- The optimality equations are:

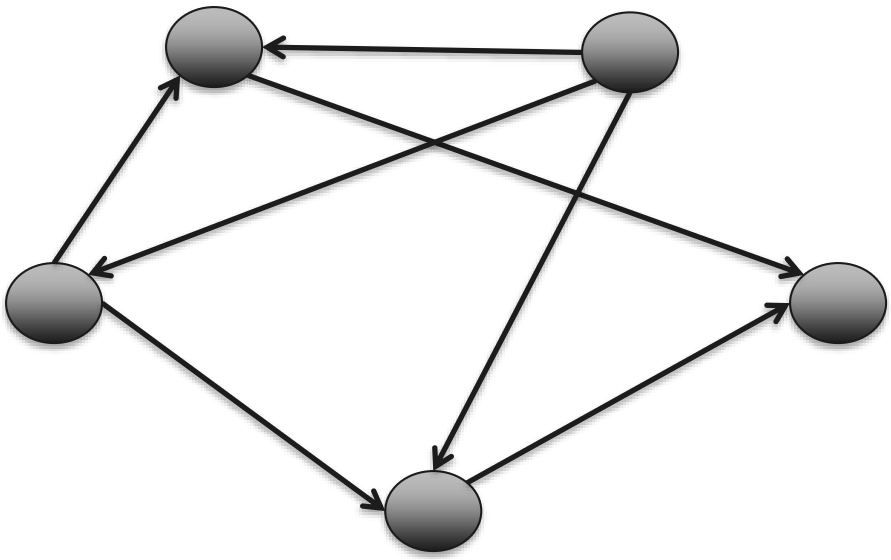
$$v(x,y) = min \begin{cases} c^+(x,y) + v(x+1, y+1) \\ c^-(x,y) + v(x+1, y-1) \end{cases}$$

$$v(x^F, y^F) = 0$$

# Acyclic Network

Things get a little more complicated for general networks because there is no obvious node ordering. A **cycle** in a network is a path that begins and ends at the same node. An **acyclic network** has no cycles.

Consider the following network:

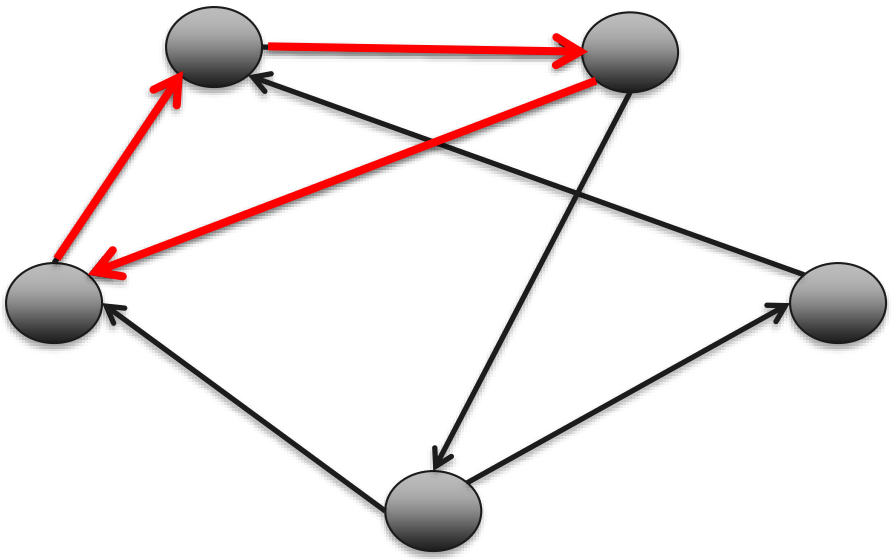


Is this an acyclic graph? Yes

# Acyclic Network

A **cycle** in a network is a path that begins and ends at the same node. An **acyclic network** has no cycles.

Consider the following network:



Is this an acyclic graph? No

# Acyclic Network



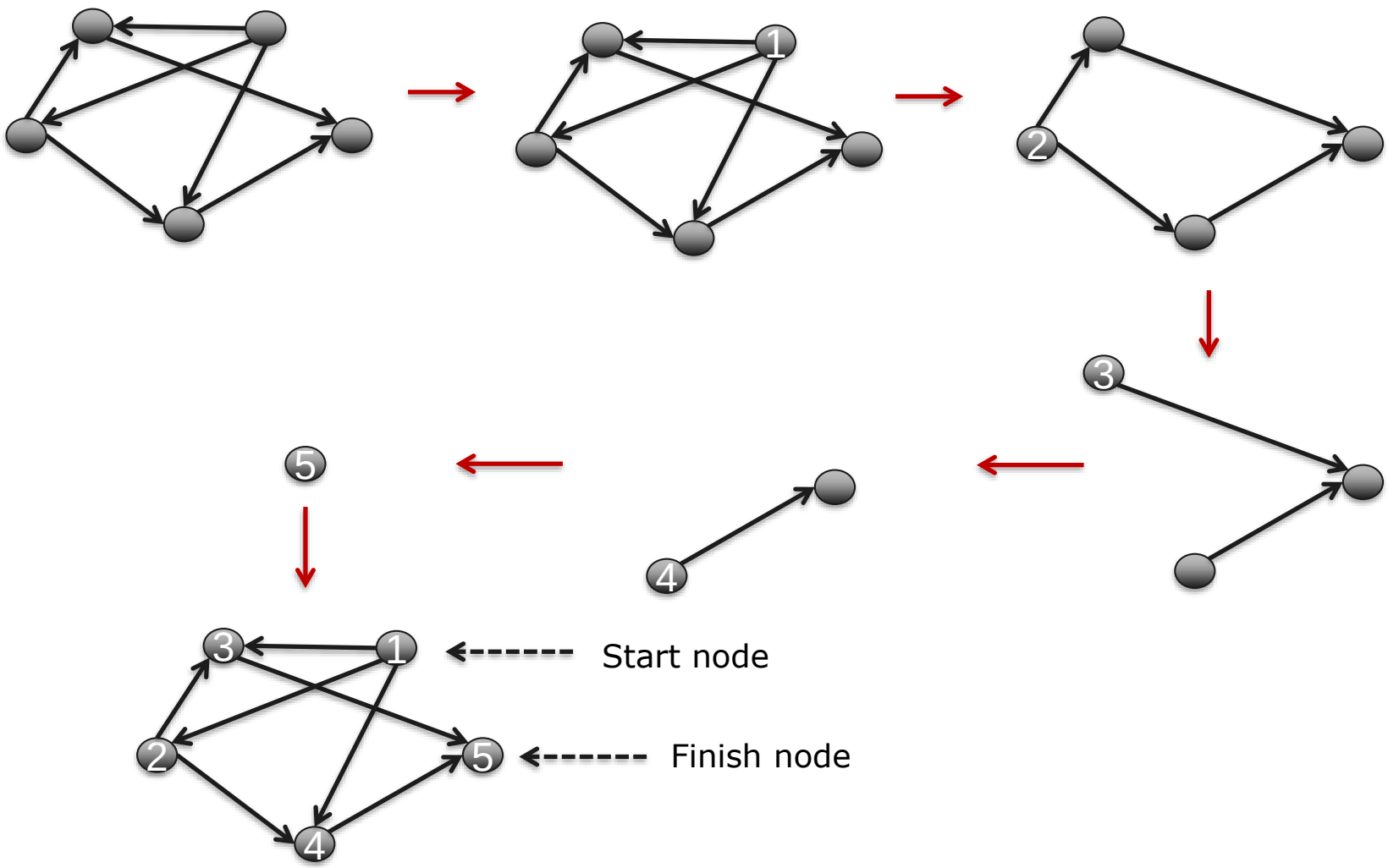
**Property:** In any acyclic network you can label nodes in such a way that **if** there is an arc from node *i* to node *j* **then** i< j.

<u>Node Labeling Algorithm:</u>

1. Choose any node that has only outward-pointing arcs
2. Assign this node label *L*. Remove this node and its arcs from the network. If there are no more nodes STOP. Otherwise, set $L = L+1$ and return to Step 1.

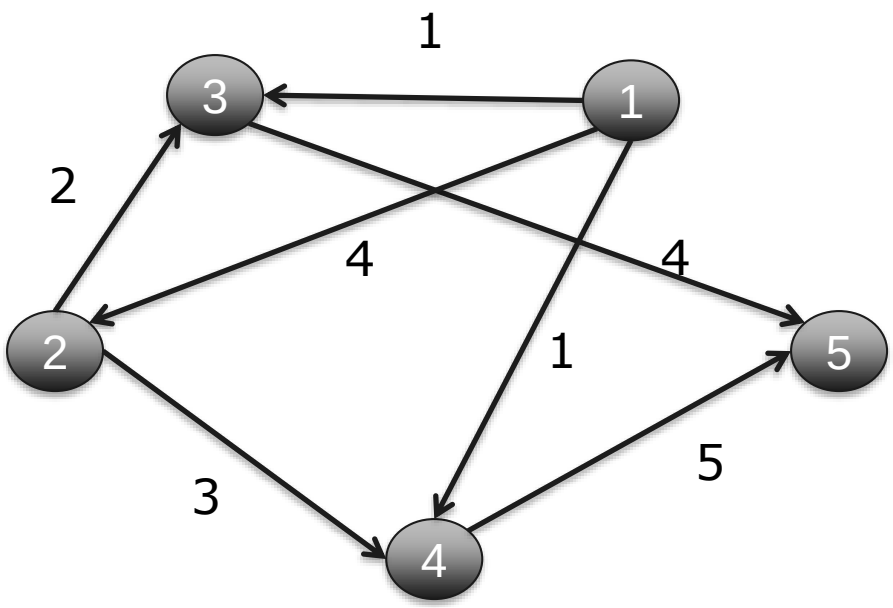
# Acyclic Network

Example: Applying the node labeling algorithm:



# Example

Find the shortest path from <u>every node to node 5</u>:




Solve the shortest path DP based on ordering of nodes according to the assigned labels

# DP Formulation

Following is a DP formulation for the shortest path problem on a labeled acyclic network

Let $v_i$ denote the length of the shortest path from *node i* to *node N*

Optimality Equations:

$$v_i = \min_{j>i}\{v_j + d_{ij}\} \; for \; i = 1,2,\ldots,N-1$$

$$v_N = \; 0 \; (boundary \; condition)$$

$$d_{ij} = \infty \; if \; no \; arc \; between \; nodes \; i \; and \; j.$$

# Forward DP Formulation

The shortest path problem can be solved via an alternative **forward dynamic programming** formulation

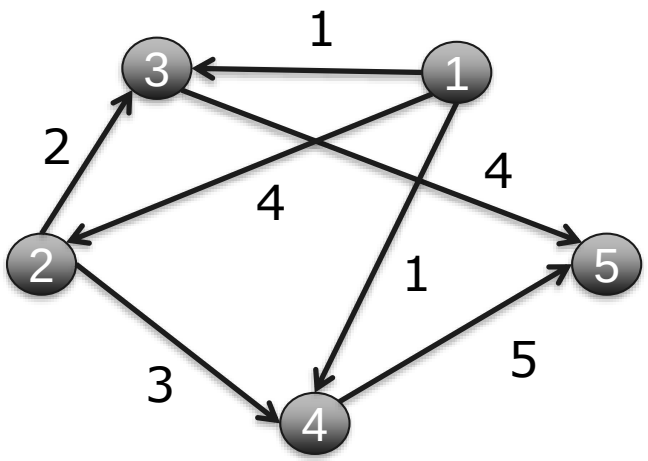
Let $v_i$ denote the length of the shortest path from *node 1* to *node i*

Optimality Equations:

$$v_i = \min_{j<i}\{v_j + d_{ji}\} \ for \ \ i = 2,3,\dots,N$$

$$v_1 = \ 0 \ (boundary\ condition)$$

$$d_{ij} = \infty \ \ if\ no\ arc\ between\ nodes\ i\ and\ j.$$




*For many problems there is a natural forward dynamic programming formulation

# General Networks

General networks may contain ***cycles***. In such cases there is no longer a consistent ordering of the nodes. Forward dynamic programming works well for these problems.

A forward dynamic programming algorithm for the General Shortest Path problem can be defined using the following:

- Let $N_i$ be the set of nodes composed of the *i* closest nodes (by shortest path) to node 1
- $v_i(j) =$ length of the shortest path from node 1 to $j$ when restricted to use nodes in $N_i$

# General Networks

**Algorithm for the General Shortest Path Problem:**

Boundary Condition: For stage 1 set $N_1 = \{1\}, v_1(j) = d_{1j}$. Note that $d_{1j} = \infty$ if there is no arc from node 1 to j. Set i=2.

1. Find $k_i = \underset{j \notin N_{i-1}}{\operatorname{argmin}}\, v_{i-1}(j)$.

2. Set $N_i = N_{i-1} \cup \{k_i\}$.

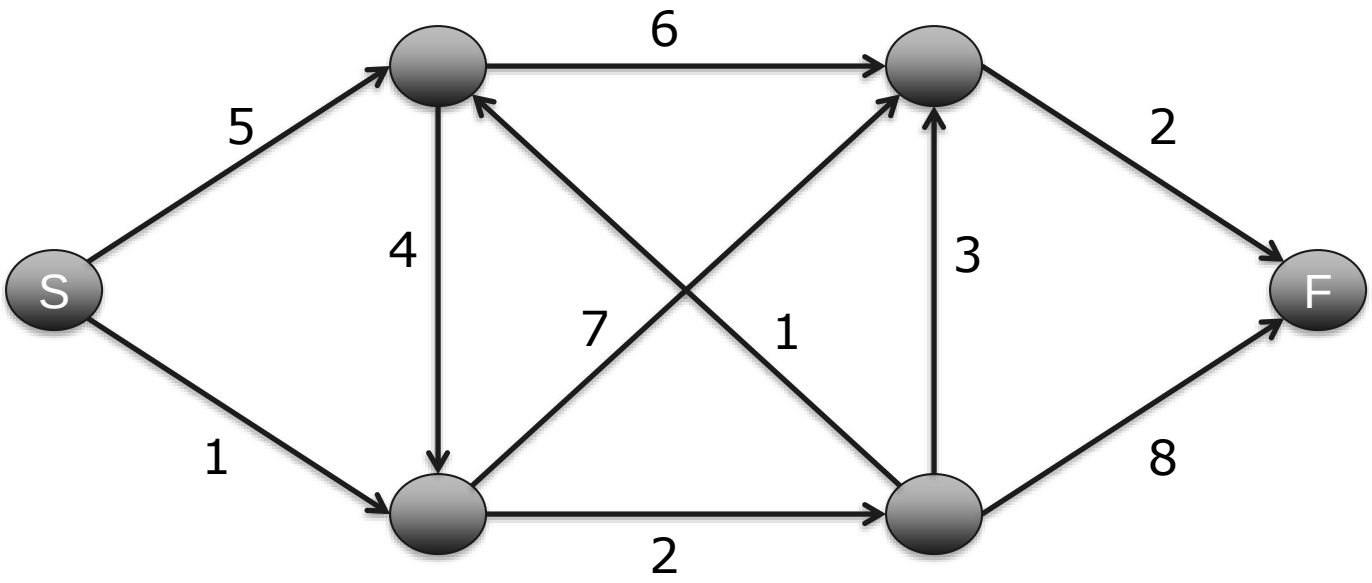
3. Compute $v_i(j)$ as:

$$v_i(j) = \begin{cases} v_{i-1}(j) & if \quad j \in N_i \\ \min\{v_{i-1}(j), v_{i-1}(k_i) + d_{k_i,j}\} & if\ j \notin N_i \end{cases}$$

Set i = i+1 and return to step 1.

# Example: General Networks

Find the shortest path between nodes S and F in the following network:




For a general cyclic network there is no consistent node labeling. Instead, the algorithm controls the order of nodes.

# General Networks

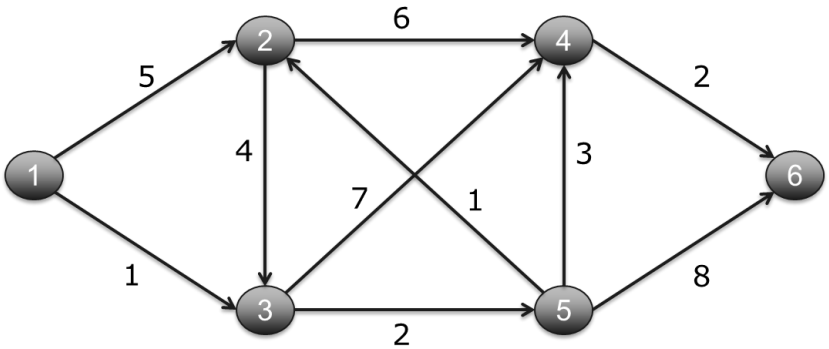
**Algorithm for the General Shortest Path Problem:**

Boundary Condition: For stage 1 set $N_1 = \{1\}, v_1(j) = d_{1j}$. Note that $d_{1j} = \infty$ if there is no arc from node 1 to j. Set i=2.

1. Find $k_i = \underset{j \notin N_{i-1}}{\operatorname{argmin}}\, v_{i-1}(j)$.

2. Set $N_{i-1} \cup \{k_i\}$.

3. Compute $v_i(j)$ as:




$$v_i(j) = \begin{cases} v_{i-1}(j) & if \quad j \in N_i \\ \min\{v_{i-1}(j), v_{i-1}(k_i) + d_{k_i,j}\} & if\ j \notin N_i \end{cases}$$

Set i = i+1 and return to step 1.

# Dijkstra's Algorithm

**Dijkstra's algorithm** attempts to simplify the forward DP solution process by using node labels to keep track of estimates in the forward DP solution process.

Step 1: Start with node 1. Give it a permanent label of 0 and label each connected node with a temporary label equal to the distance from node 1. All other nodes have temporary label ∞.

Step 2: Choose the node with the smallest temporary label and make its label permanent (call this node i). For each connected node j that has a temporary label replace the temporary label with:

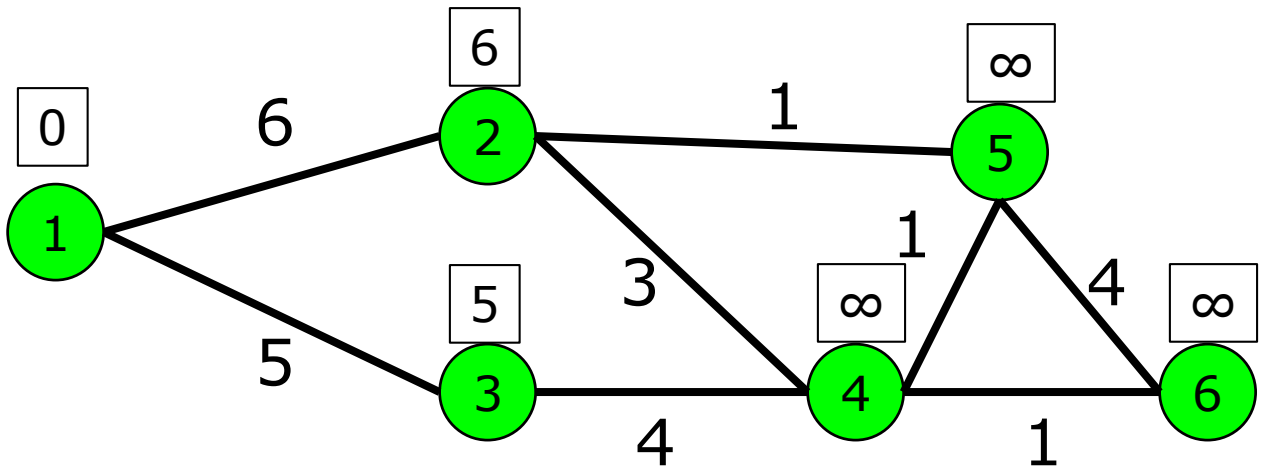
$$\text{Min}\{\text{current label, node } i\text{'s permanent label} + d_{ij}\}$$

Make the new smallest temporary label a permanent label. Continue through the network repeating Step 2.

# Example: Iteration 1—Step 1



Step 1: Start with node 1. Give it a permanent label of 0 and label each connected node with a temporary label equal to the distance from node 1. All other nodes have temporary label ∞.



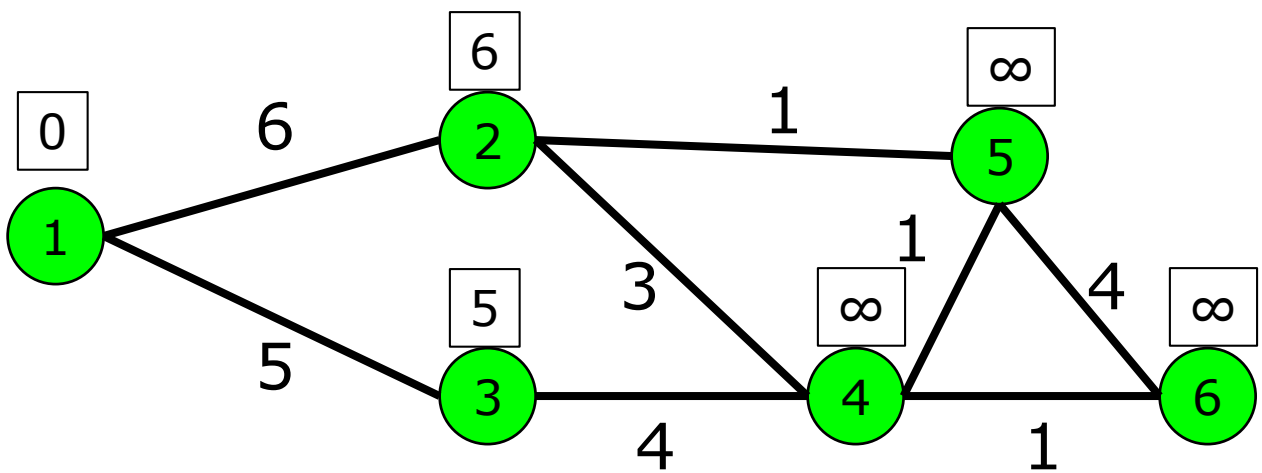
# Example: Iteration 2—Step 2



Step 2:

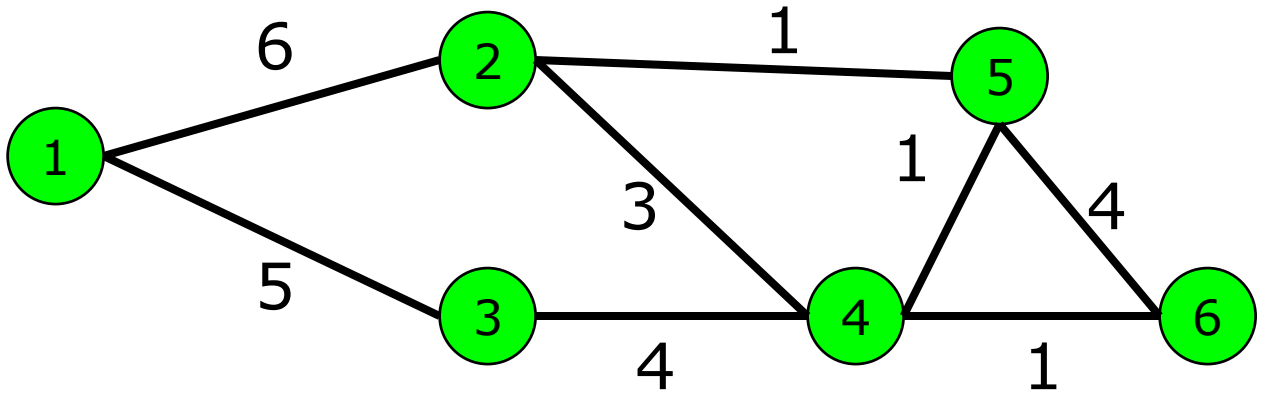
- Choose the node with the smallest temporary label and make its label permanent (call this node i).
- For each connected node j that has a temporary label replace the temporary label with:

$$\min\{current\ label,\ node\ i\ label + d_{ij}\}$$




Iteration 1 Node Labels: $\{0^*, 6, 5^*, 9, \infty, \infty\}$

# Optimal Solution




Iteration 1 Node Labels: $\{0^*, 6, 5^*, 9, \infty, \infty\}$

Iteration 2 Node Labels: $\{0^*, 6^*, 5^*, 9, 7, \infty\}$

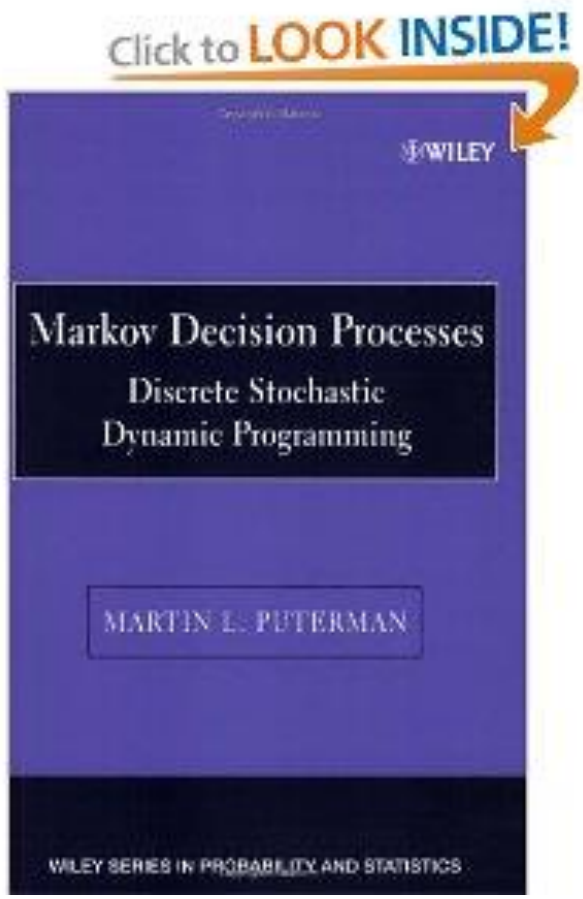
Iteration 3 Node Labels: $\{0^*, 6^*, 5^*, 8, 7^*, 11\}$

Iteration 4 Node Labels: $\{0^*, 6^*, 5^*, 8^*, 7^*, 9\}$

Optimal solution: 1 → 2 → 5 → 4 → 6, Path Length = 9

# Reading

We will start getting into stochastic dynamic programming next class:




A this point you should have covered Chapters 1 – 3 of Dreyfus and Law (Sections 1.11 and 3.5-3.10 are not "required")

Chapters 2 – 5 of Puterman are on Ctools

Start reading Chapter 2 of Puterman to stay ahead of class