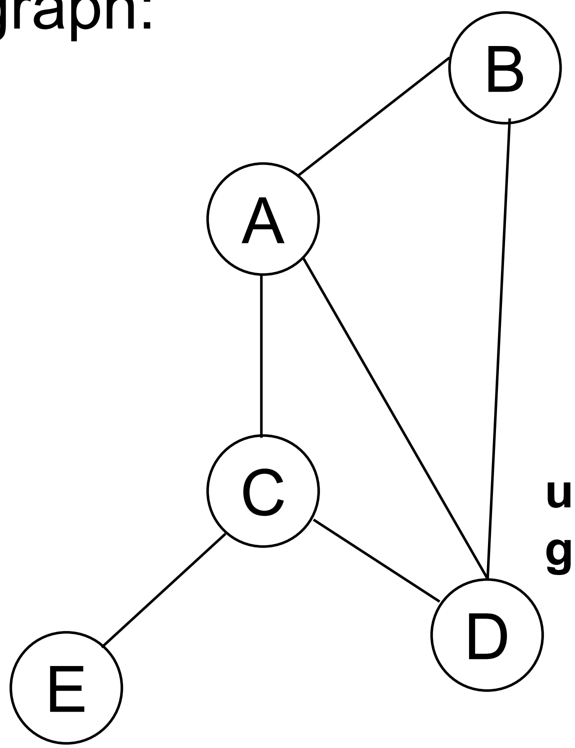
# Single Source Shortest Path

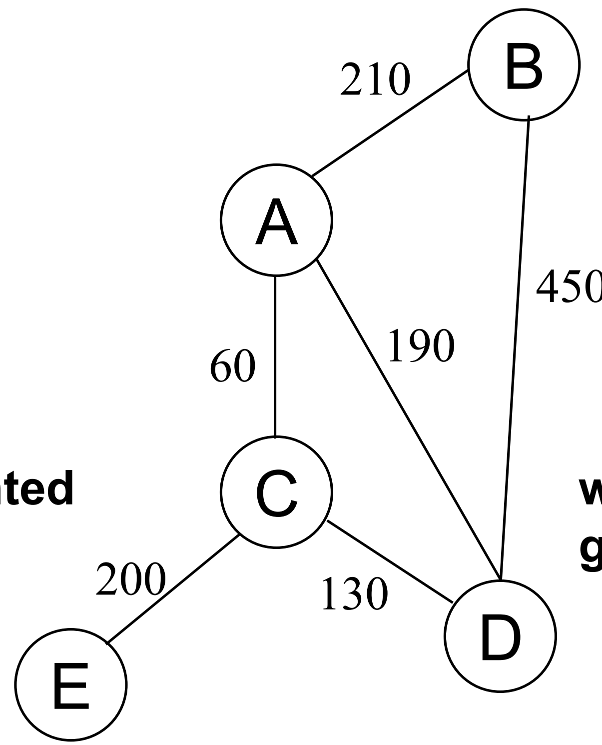
# (Dijkstra’s Algorithm)

# Shortest Path Problems

- **What is shortest path ?**
  - shortest length between two vertices for an unweighted graph:
  - smallest cost between two vertices *for* a weighted graph:




**unweighted graph**




**weighted graph**

# Shortest Path Problems

- How can we find the shortest route between two points on a map?
- Model the problem as a graph problem:
  - Road map is a weighted graph:

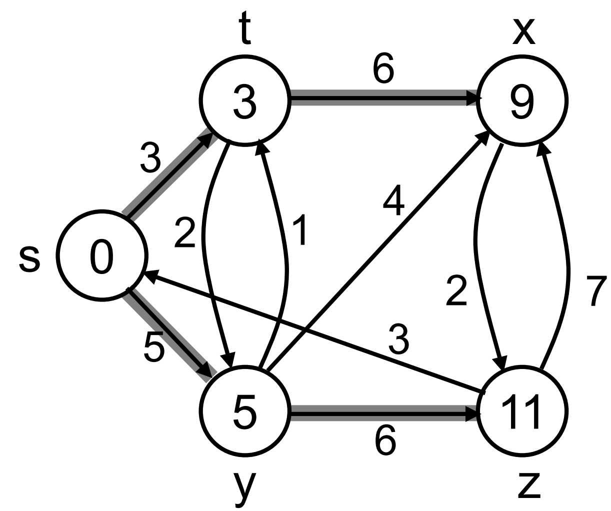
    vertices = cities

    edges = road segments between cities

    edge weights = road distances
  - Goal: find a shortest path between two vertices (cities)

# Shortest Path Problems

- **Input:**
  - Directed graph G = (V, E)
  - Weight function w : E → **R**




- **Weight of path** p = $\langle v_0, v_1, \ldots, v_k \rangle$

$$w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

- **Shortest-path weight** from u to v:

$$\delta(u, v) = \begin{cases} \min\{w(p) : u \overset{p}{\rightsquigarrow} v\} & \text{if there exists a path from } u \text{ to } v \\ \infty & \text{otherwise} \end{cases}$$

- Shortest path u to v is any path p such that w(p) = δ(u, v)

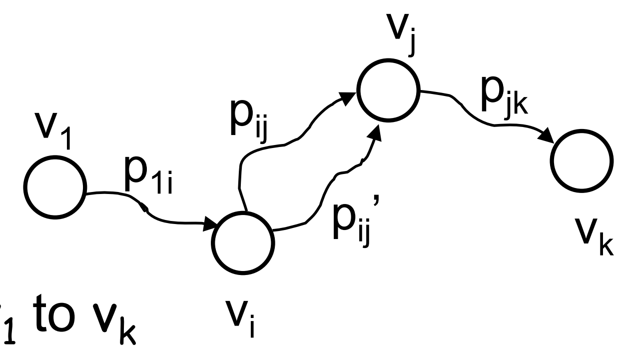
# Variants of Shortest Paths

- **Single-source shortest path**
  - $G = (V, E) \Rightarrow$ find a shortest path from a given source vertex $s$ to each vertex $v \in V$
- **Single-destination shortest path**
  - Find a shortest path to a given destination vertex $t$ from each vertex $v$
  - Reverse the direction of each edge $\Rightarrow$ single-source
- **Single-pair shortest path**
  - Find a shortest path from $u$ to $v$ for given vertices $u$ and $v$
  - Solve the single-source problem
- **All-pairs shortest-paths**
  - Find a shortest path from $u$ to $v$ for every pair of vertices $u$ and $v$

# Optimal Substructure of Shortest Paths

Given:

- A weighted, directed graph $G = (V, E)$
- A weight function $w: E \rightarrow \mathbf{R}$,
- A shortest path $p = \langle v_1, v_2, \ldots, v_k \rangle$ from $v_1$ to $v_k$
- A subpath of p: $p_{ij} = \langle v_i, v_{i+1}, \ldots, v_j \rangle$, with $1 \leq i \leq j \leq k$




Then: $p_{ij}$ is a shortest path from $v_i$ to $v_j$

**Proof**: $p = v_1 \overset{p_{1i}}{\rightsquigarrow} v_i \overset{p_{ij}}{\rightsquigarrow} v_j \overset{p_{jk}}{\rightsquigarrow} v_k$

$$w(p) = w(p_{1i}) + w(p_{ij}) + w(p_{jk})$$

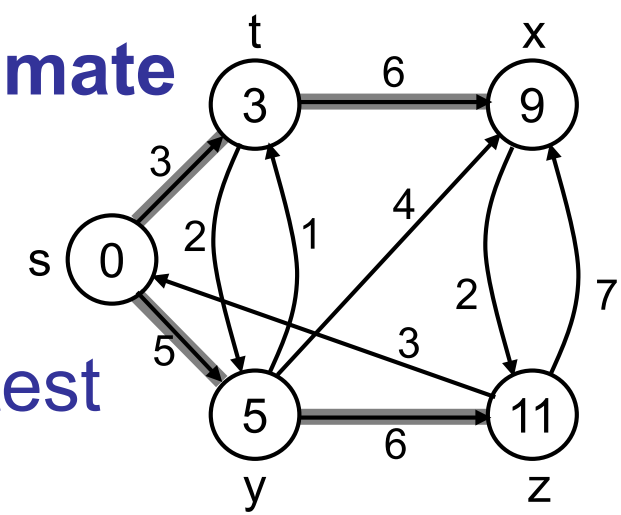
Assume $\exists\ p_{ij}'$ from $v_i$ to $v_j$ with $w(p_{ij}') < w(p_{ij})$

$\Rightarrow w(p') = w(p_{1i}) + w(p_{ij}') + w(p_{jk}) < w(p)$ contradiction!

# Shortest-Path Representation

For each vertex v ∈ V:

- d[v] = δ(s, v): a **shortest-path estimate**
  - Initially, d[v]=∞
  - Reduces as algorithms progress
- π[v] = **predecessor** of v on a shortest path from *s*
  - If no predecessor, π[v] = NIL
  - π induces a tree—**shortest-path tree**
- Shortest paths & shortest path trees are not unique



# Initialization

*Alg.:* INITIALIZE-SINGLE-SOURCE(V, s)

1. **for** each $v \in V$
2. **do** $d[v] \leftarrow \infty$
3. $\pi[v] \leftarrow$ NIL
4. $d[s] \leftarrow 0$

- All the shortest-paths algorithms start with INITIALIZE-SINGLE-SOURCE

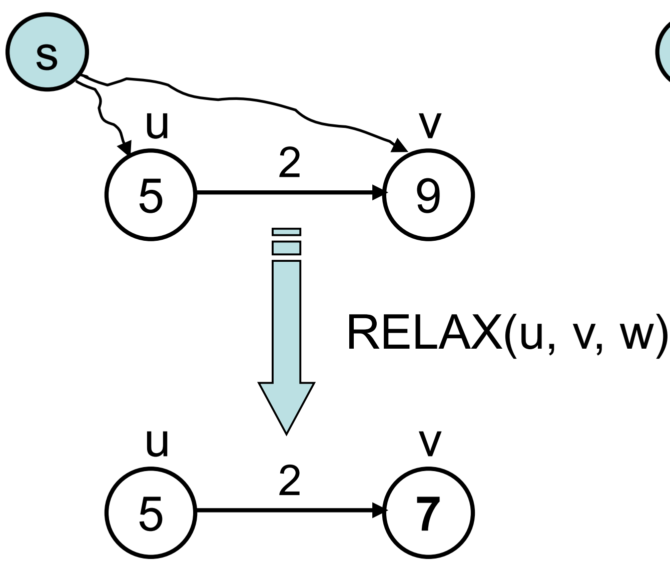
# Relaxation

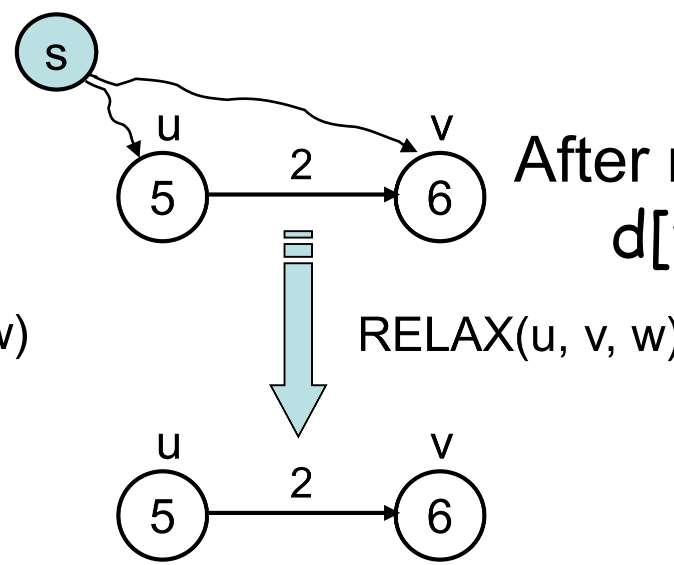
- **Relaxing** an edge (u, v) = testing whether we can improve the shortest path to v found so far by going through u

  If $d[v] > d[u] + w(u, v)$

  we can improve the shortest path to v

  $\Rightarrow$ update $d[v]$ and $\pi[v]$




After relaxation:

$d[v] \leq d[u] + w(u, v)$

# RELAX(u, v, w)

1. **if** $d[v] > d[u] + w(u, v)$
2. **then** $d[v] \leftarrow d[u] + w(u, v)$
3. $\pi[v] \leftarrow u$

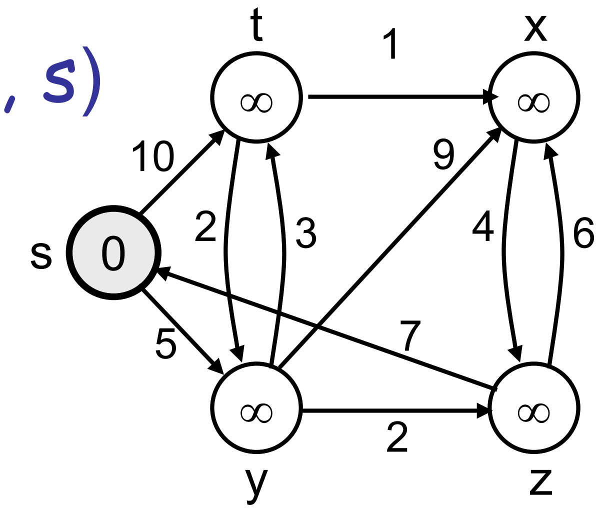
- All the single-source shortest-paths algorithms
  - start by calling INIT-SINGLE-SOURCE
  - then relax edges
- The algorithms differ in the order and how many times they relax each edge

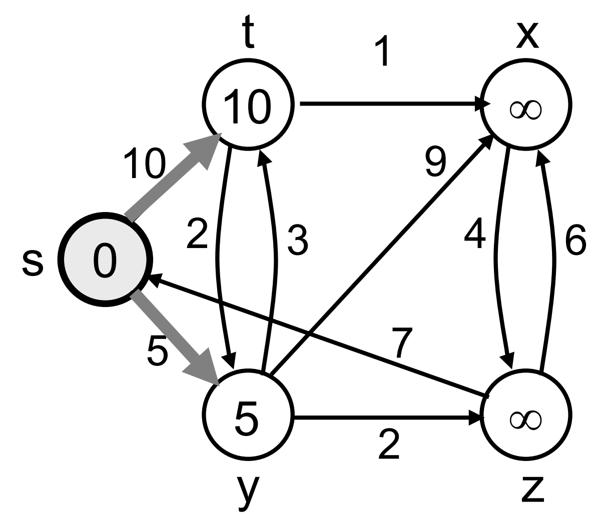
# Dijkstra's Algorithm

- Single-source shortest path problem:
  - No negative-weight edges: $w(u, v) > 0 \ \forall \ (u, v) \in E$
- Maintains two sets of vertices:
  - S = vertices whose final shortest-path weights have already been determined
  - Q = vertices in V – S: min-priority queue
    - Keys in Q are estimates of shortest-path weights (d[v])
- Repeatedly select a vertex $u \in V - S$, with the minimum shortest-path estimate d[v]

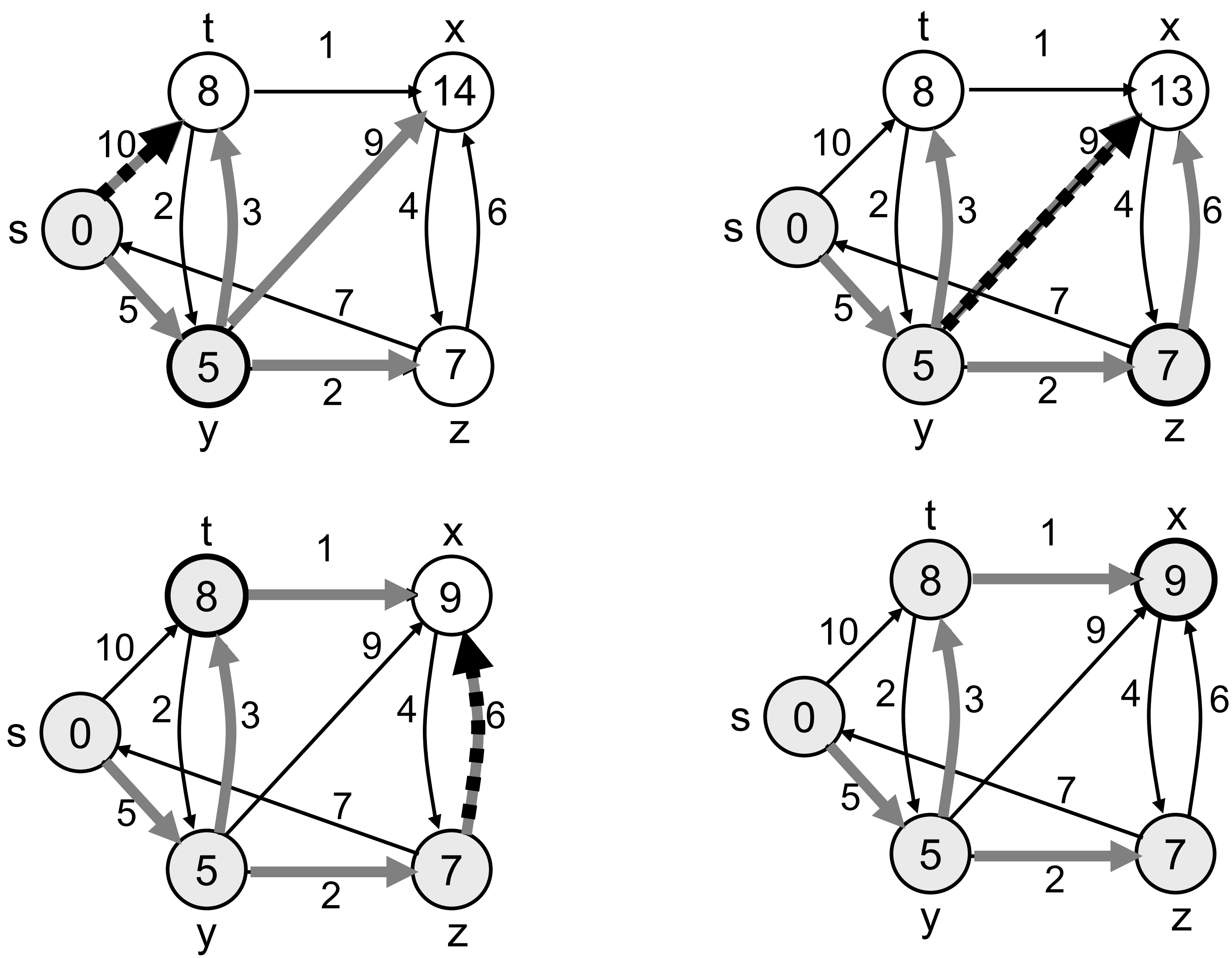
# Dijkstra (G, w, s)

```
INITIALIZE-SINGLE-SOURCE(V, s)
S ← ∅
Q ← V[G]
while Q ≠ ∅
    do u ← EXTRACT-MIN(Q)
        S ← S ∪ {u}
        for each vertex v ∈ Adj[u]
            do RELAX(u, v, w)
```



# Example



t
x
1
8
14
10
9
2
3
4
6
s
0
7
5
5
7
2
y
z
t
x
1
8
13
10
9
2
3
4
6
s
0
7
5
5
7
2
y
z
t
x
1
8
9
10
9
2
3
4
6
s
0
7
5
5
7
2
y
z
t
x
1
8
9
10
9
2
3
4
6
s
0
7
5
5
7
2
y
z

# Dijkstra's Pseudo Code

- Graph *G*, weight function *w*, root *s*

```
DIJKSTRA(G, w, s)
for each v ∈ V
        do d[v] ← ∞
d[s] ← 0
S ← ∅    ▷ Set of discovered nodes
Q ← V
while Q ≠ ∅
        do u ← EXTRACT-MIN(Q)
            S ← S ∪ {u}
            for each v ∈ Adj[u]
                do if d[v] > d[u] + w(u, v)
                    then d[v] ← d[u] + w(u, v)
```

relaxing edges

# Dijkstra (G, w, s)

1. INITIALIZE-SINGLE-SOURCE(V, s) $\longleftarrow$ $\Theta(V)$
2. $S \leftarrow \emptyset$
3. $Q \leftarrow V[G]$ $\longleftarrow$ O(V) build min-heap
4. **while** $Q \neq \emptyset$ $\longleftarrow$ Executed O(V) times
5. **do** $u \leftarrow$ EXTRACT-MIN(Q) $\longleftarrow$ O(lgV)
6. $S \leftarrow S \cup \{u\}$
7. **for** each vertex $v \in Adj[u]$
8. **do** RELAX(u, v, w) $\longleftarrow$ O(E) times; O(lgV)

Running time: $O(V\lg V + E\lg V) = O(E\lg V)$