The Shortest Path Problem

Lecture 11

The Shortest Path Problem

- Given a directed graph $G=(V,A)$ with a source vertex $s \in V$.
- Let each arc $a \in A$ have a non-negative length ℓ_a
- The length of a path P in the graph is:

$$\ell(P) = \sum_{a \in P} \ell_a$$

- This induces the following problem:

The Shortest Path Problem. Find the shortest length paths from s to every other vertex $v \in V$.

Dijkstra's Shortest Path Algorithm

Dijkstra's Shortest Path Algorithm [Dijkstra 1957]

Set $S = \emptyset$

Set $d(s) = 0$ and $d(v) = \infty, \quad \forall v \in V \setminus \{s\}$

While $S \neq V$

If $v = \underset{u \in V \setminus S}{\operatorname{argmin}} \; d(u)$ then set $S \leftarrow S \cup \{v\}$

For each arc (v, w):

If $d(w) > d(v) + \ell_{vw}$ and $w \in V \setminus S$

then set $d(w) = d(v) + \ell_{vw}$

Dijkstra's Shortest Path Algorithm

Set $\mathcal{S} = \emptyset$

Set $d(s) = 0$ and $d(v) = \infty$. $\forall v \in V \setminus \{s\}$

While $\mathcal{S} \neq V$

If $v = \underset{u \in V \setminus \mathcal{S}}{\operatorname{argmin}} \; d(u)$ then set $\mathcal{S} \leftarrow \mathcal{S} \cup \{v\}$

For each arc (v, w):

If $d(w) > d(v) + \ell_{vw}$ and $w \in V \setminus \mathcal{S}$

then set $d(w) = d(v) + \ell_{vw}$

- This is a **greedy algorithm** as at each step it myopically selects the vertex with the smallest label.
- As described, the algorithm finds just the shortest path **distances**.
- It is easy to enhance the algorithm to keep track of the **shortest paths** themselves as well...

Dijkstra's Shortest Path Algorithm

Dijkstra's Shortest Path Algorithm [Dijkstra 1957]

Set $\mathcal{S} = \emptyset$

Set $\mathcal{T} = \emptyset$, $\text{pred}(s) \leftarrow$ " $-$ " , and $\text{pred}(v) \leftarrow \square$

Set $d(s) = 0$ and $d(v) = \infty, \quad \forall v \in V \setminus \{s\}$

While $\mathcal{S} \neq V$

If $v = \underset{u \in V \setminus \mathcal{S}}{\text{argmin}} \; d(u)$

then set $\mathcal{S} \leftarrow \mathcal{S} \cup \{v\}$ and $\mathcal{T} \leftarrow \mathcal{T} \cup (\text{pred}(v), v)$

For each arc (v, w):

If $d(w) > d(v) + \ell_{vw}$ and $w \in V \setminus \mathcal{S}$

then set $d(w) = d(v) + \ell_{vw}$

$\text{pred}(w) \leftarrow v$

Dijkstra's Shortest Path Algorithm

Set $S = \emptyset$

Set $d(s) = 0$ and $d(v) = \infty$. $\forall v \in V \setminus \{s\}$

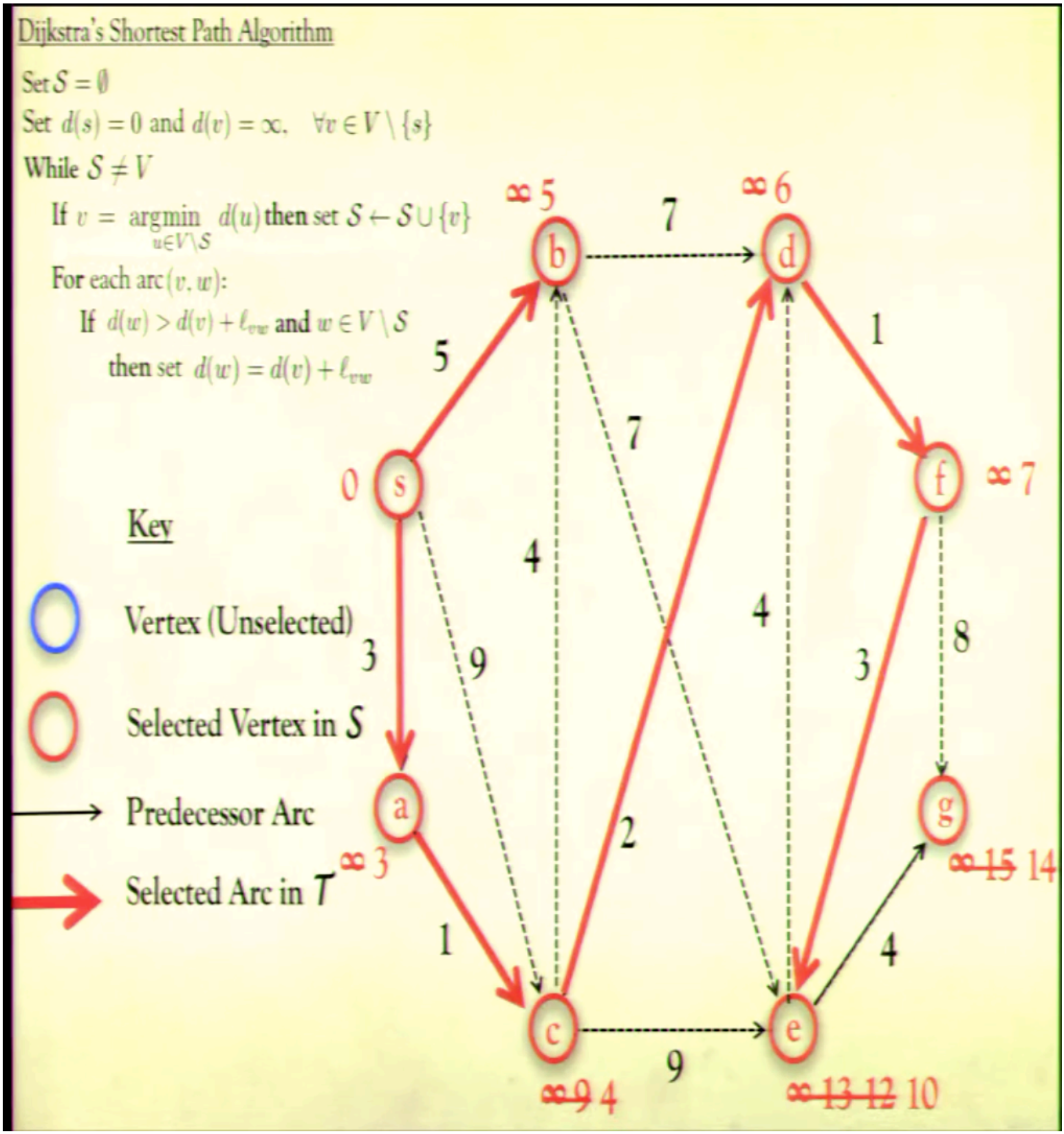
While $S \neq V$

If $v = \underset{u \in V \setminus S}{\operatorname{argmin}}\ d(u)$ then set $S \leftarrow S \cup \{v\}$

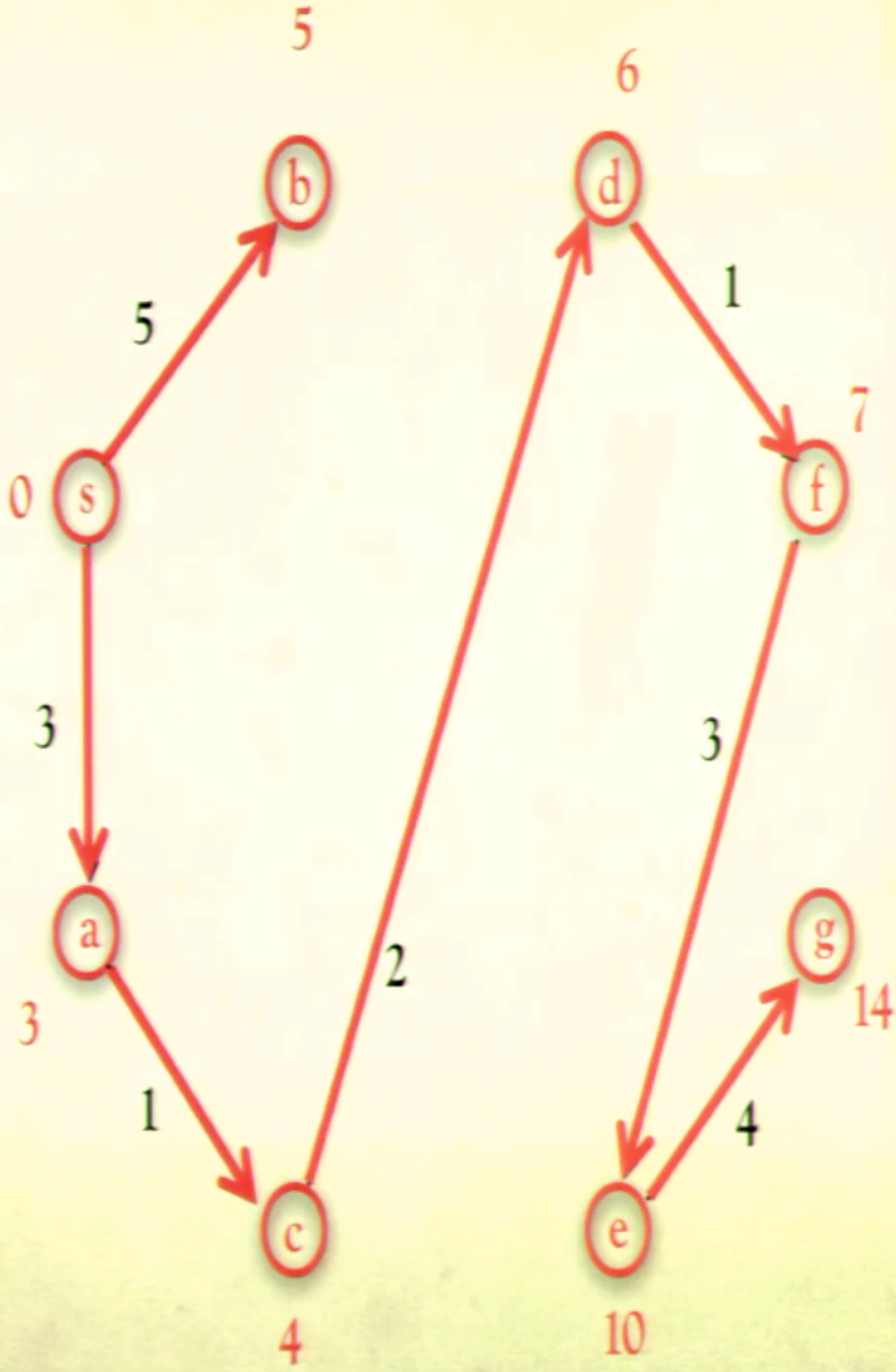
For each arc (v, w):

If $d(w) > d(v) + \ell_{vw}$ and $w \in V \setminus S$

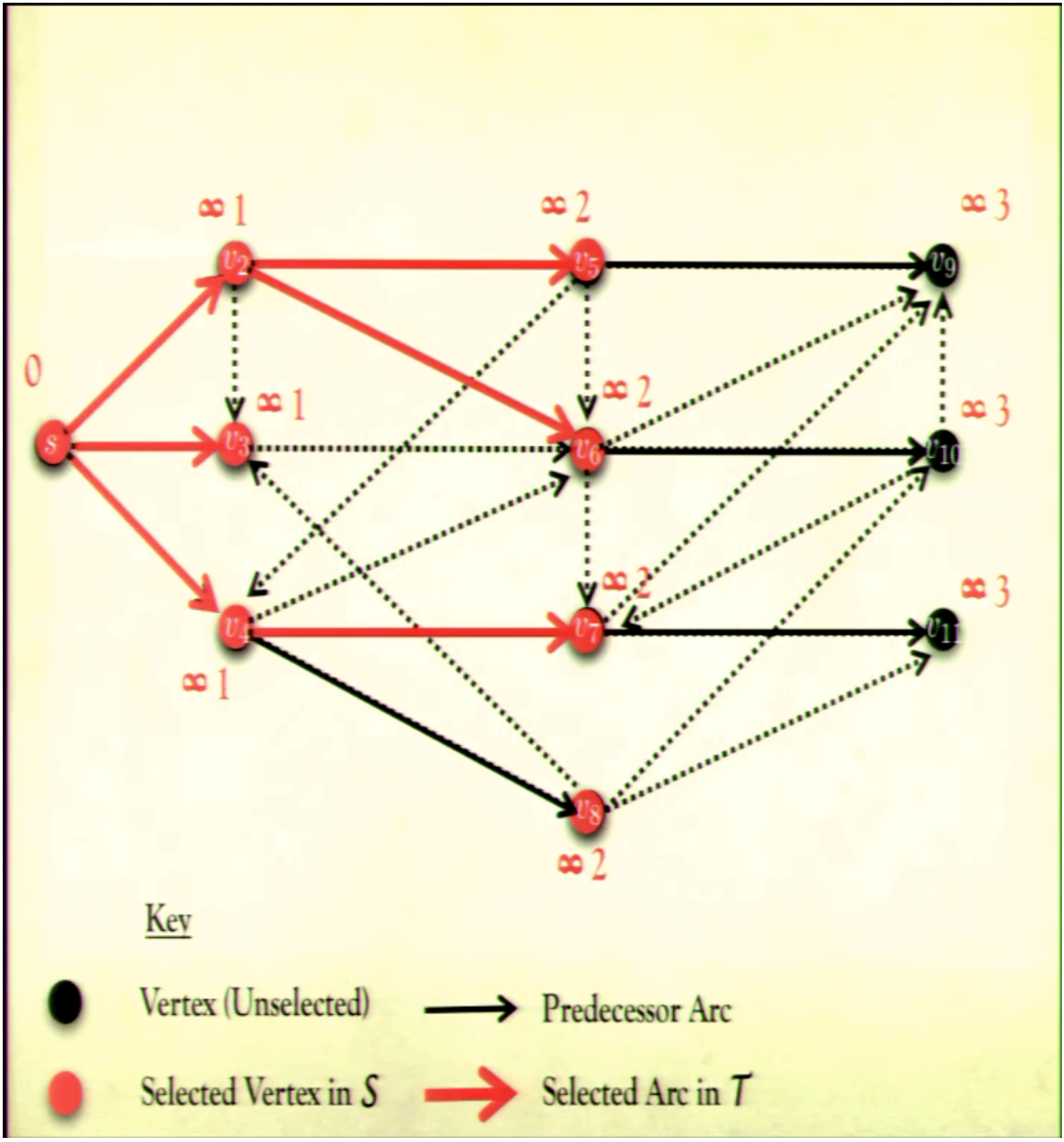
then set $d(w) = d(v) + \ell_{vw}$

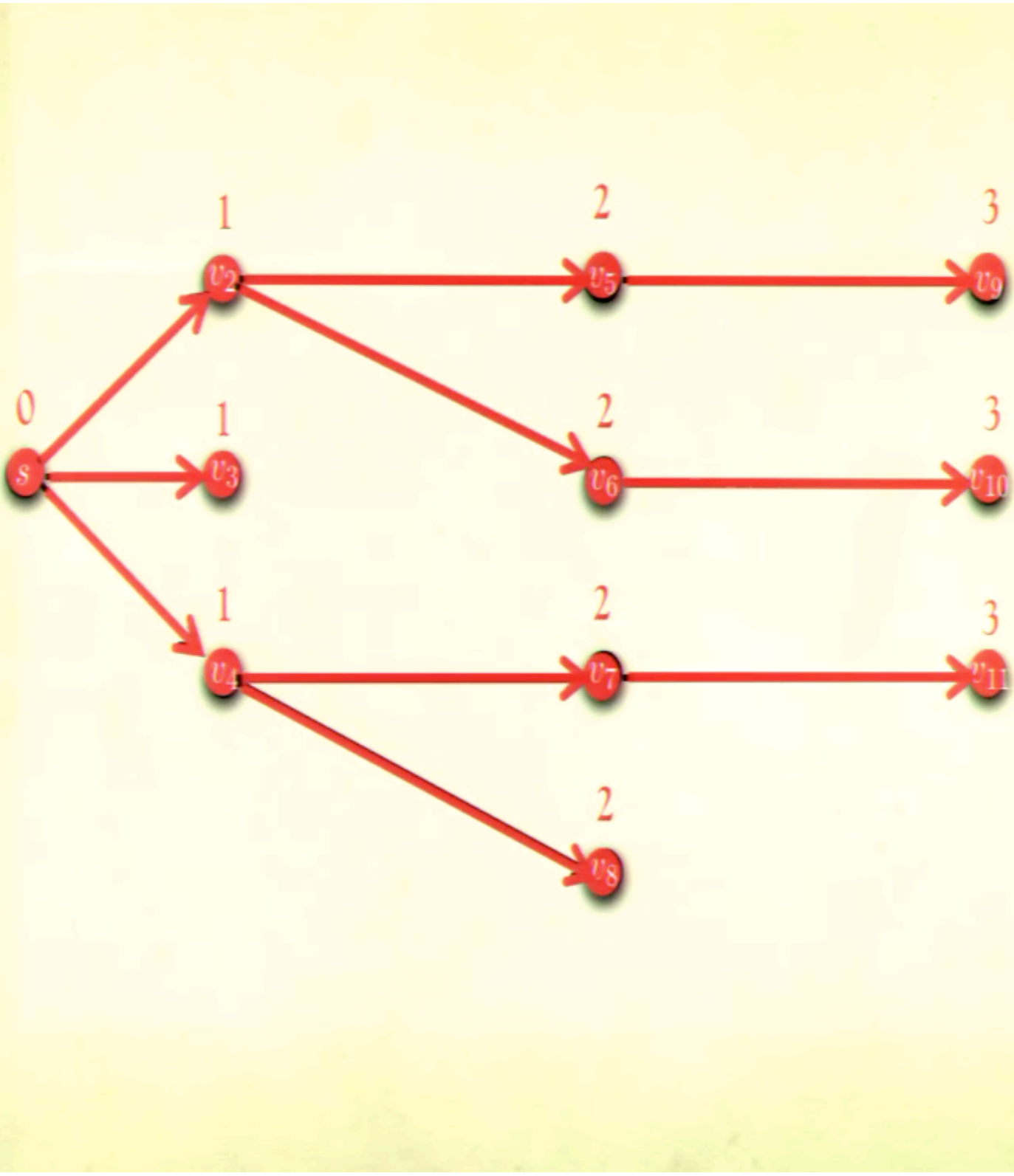


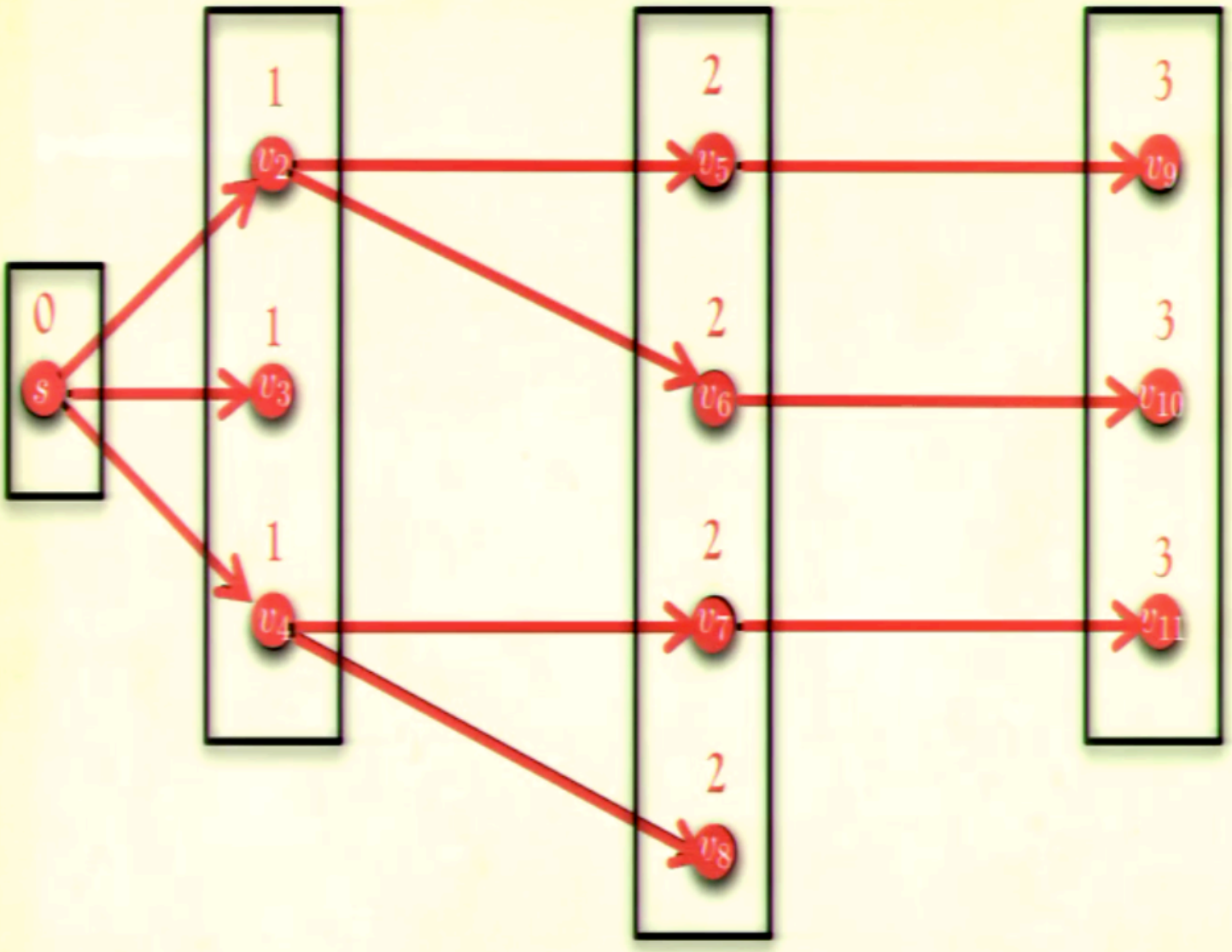
The Shortest Paths



What happens in the special case where each arc length is exactly one?


∞ 1
∞ 2
∞ 3
v_2
v_5
v_9
0
s
∞ 1
∞ 2
∞ 3
v_3
v_6
v_{10}
∞ 2
∞ 3
v_4
v_7
v_{11}
∞ 1
v_8
∞ 2
Key
Vertex (Unselected)
Predecessor Arc
Selected Vertex in S
Selected Arc in T


0
1
2
3
s
u_2
u_5
u_9
1
2
3
u_3
u_6
u_{10}
1
2
3
u_4
u_7
u_{11}
2
u_8



- This is the **breadth-first search algorithm!**

The Shortest Path Graph

- To show the algorithm does work, first observe that S is a set of **vertices** and T is a set of **arcs**.
- The following *notation* will then be helpful:
  - Let S^k be the set of vertices in S at the end of the kth iteration.
  - Let T^k be the set of arcs in T at the end of the kth iteration.
- Furthermore, observe the arcs in T^k are all between vertices in S^k.

$$\Longrightarrow \quad G^k = (S^k, T^k) \quad \text{is a } \textbf{directed graph}.$$

- Finally, observe that $G = G^n$ is the final output of Dijkstra's algorithm.
- So let's investigate what G looks like...

The Shortest Path Tree

- It turns out that $G = G^n$ is a **directed tree** (*arborescence*) rooted at s.
- To show this we prove:

Theorem. The graph G^k is a directed tree (arborescence) rooted at s.

Proof.

Base Case: $k=1$

- Observe S^1 contains only the vertex s.
- But T^1 contains no arcs as $pred(s) = \text{“}-\text{”}$
- Thus $G^1 = (S^1, T^1)$ is trivially an *arborescence* rooted at s.

Induction Hypothesis: $G^{k-1} = (S^{k-1}, T^{k-1})$ is an arborescence rooted at s.

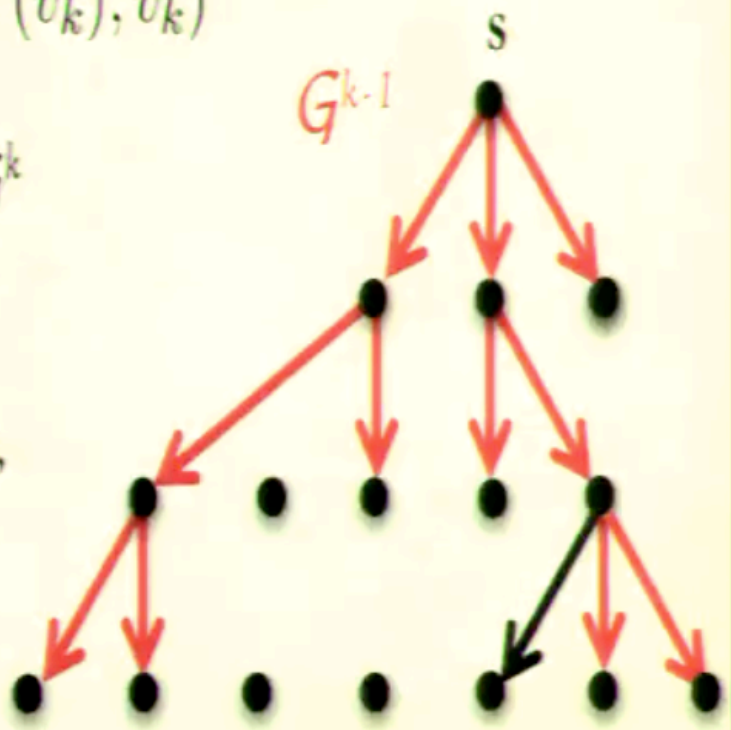
Proof [cont].

Induction Step: Consider $G^k = (S^k, T^k)$

- Let v_k be the vertex added to S in the kth iteration.

$$\Longrightarrow S^k = S^{k-1} \cup \{v_k\}$$

$$\Longrightarrow T^k = T^{k-1} \cup (\text{pred}^{k-1}(v_k), v_k)$$



- Thus v_k has in-degree exactly 1 in G^k and out-degree exactly 0.

- Moreover, by the *induction hypothesis*, G^{k-1} is an **arborescence** rooted at s.

- Furthermore , by definition: $\text{pred}^{k-1}(v_k) \in S^{k-1}$

$\Longrightarrow G^k = (S^k, T^k)$ is an **arborescence** rooted at s with v_k as a leaf. □

Shortest Path Distances

- So Dijkstra's algorithm outputs an **arborescence**.
- Moreover the paths induced by the arborescence are **shortest paths**...

Theorem. The graph $\mathcal{G}^k = (\mathcal{S}^k, \mathcal{T}^k)$ gives the true shortest path distances from s to every vertex in $\mathcal{S}^k$.

Proof.

Base Case: $k=1$

- Trivially true as: $d^1(s) = 0 = d^*(s)$

Induction Hypothesis:

- Assume that $\mathcal{G}^{k-1} = (\mathcal{S}^{k-1}, \mathcal{T}^{k-1})$ gives the **shortest distance** from s to *every* vertex in $\mathcal{S}^{k-1}$:

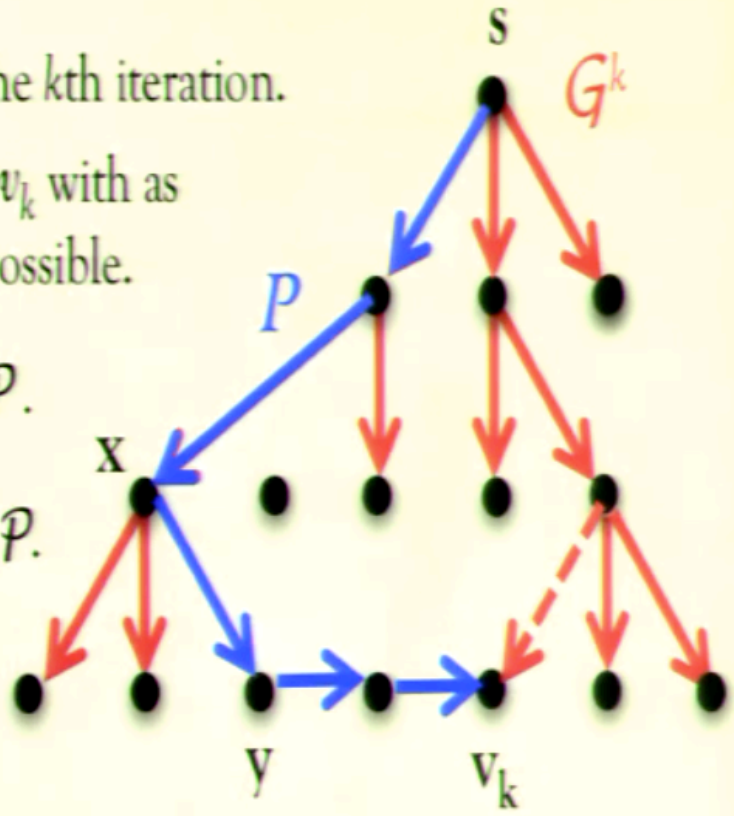
$$d^{k-1}(v) = d^*(v) \quad \forall v \in \mathcal{S}^{k-1}$$

Proof [cont].

Induction Step: Consider $G^k = (S^k, T^k)$

- Let v_k be the vertex added to S in the kth iteration.
- Take the **shortest path** P from s to v_k with as many arcs in common with G^k as possible.
- Let x be the last vertex of G^{k-1} in P.
- Let $y \notin S^k$ be the vertex after x in P.

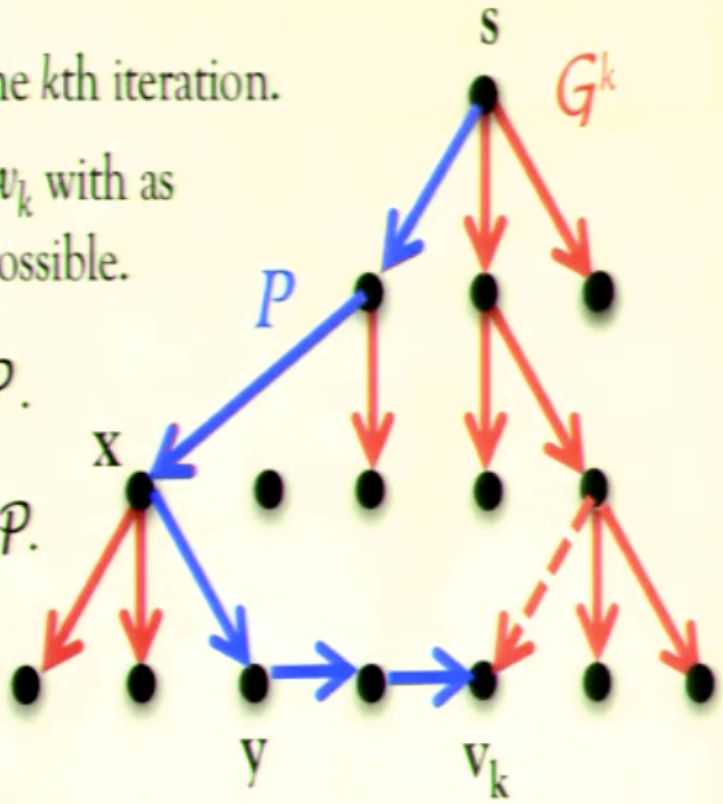
Such a y exists or P is in G^k and we are done.



Proof [cont].

Induction Step: Consider $G^k = (S^k, T^k)$

- Let v_k be the vertex added to S in the kth iteration.
- Take the **shortest path** P from s to v_k with as many arcs in common with G^k as possible.
- Let x be the last vertex of G^{k-1} in P.
- Let $y \notin S^k$ be the vertex after x in P.



- Since y is on the shortest path from s to v_k and the arc lengths are *non-negative* we have:

$$d^*(y) \leq d^*(v_k)$$

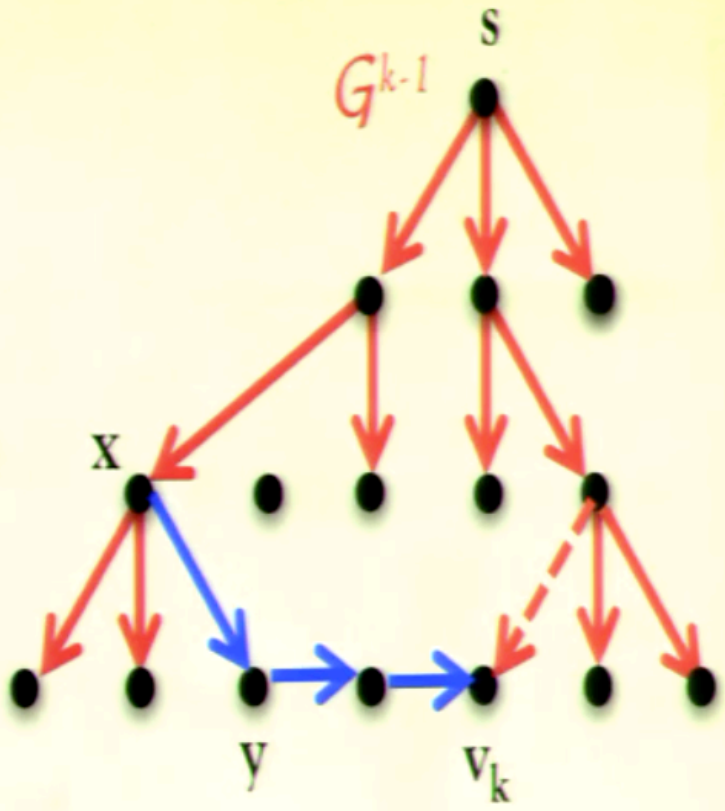
$$< d^k(v_k)$$

We have a strict inequality or we are done.

Proof [cont].

- So we have: $d^*(y) < d^k(v_k)$
- But since $x \in S^{k-1}$ we have:

$$\begin{aligned} d^k(y) &\leq d^{k-1}(x) + \ell(x,y) \\ &= d^*(x) + \ell(x,y) \\ &= d^*(y) \end{aligned}$$



- Therefore:

$$d^k(y) \leq d^*(y) < d^k(v_k)$$

- This contradicts the selection of v_k as: $d^k(y) < d^k(v_k)$ □

The Running Time

- How long does Dijkstra's algorithm take?
  - There are n iterations.
  - There are at most n distance updates in each iteration.

$$\Longrightarrow \quad \text{Runtime} = O(n^2)$$

- In fact, later in the course, we will see how to implement Dijkstra's algorithm in time $O(m \cdot \log n)$ using a data structure called a **heap**.

Applications of Dijkstra's Algorithm

- Despite being 60 years old, Dijkstra's algorithm and its variants are still widely used in practice. For example in:

  - **Open Shortest Path First** (OSPF), the most common *Interior Gateway Protocol* (IGP) for routing internet packets in an autonomous system.

  - **Route Mapping**