Lecture 10

Instructor: Karteek Sreenivasaiah

18th September 2018

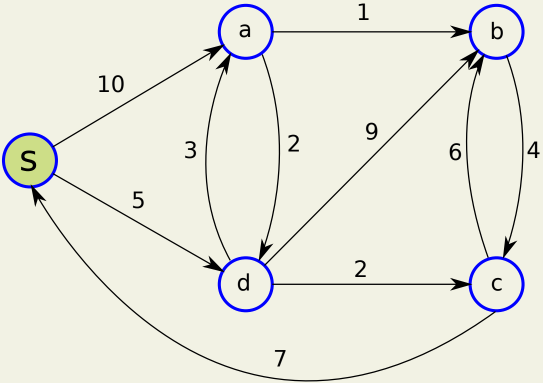
Recap: Shortest paths

Input:

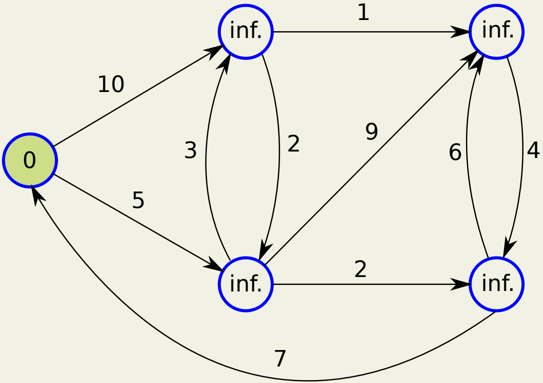
- Graph $G = (V, E)$
- Weight function $w : E \to \mathbb{Z}^+$
- Source vertex $s \in V$

Goal: Compute the shortest path from s to all reachable vertices.

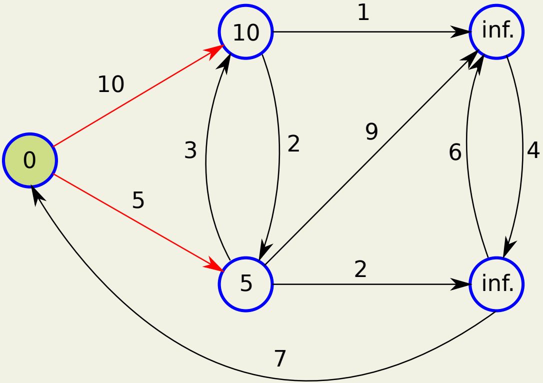
Example graph



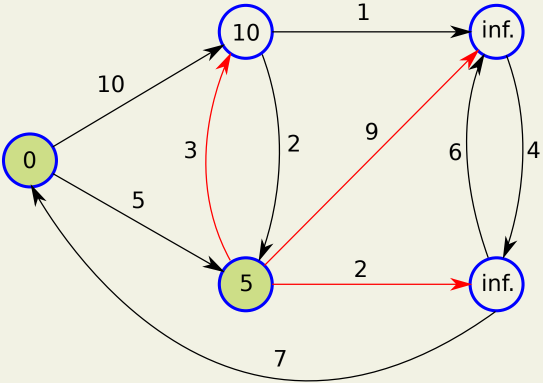
Example graph



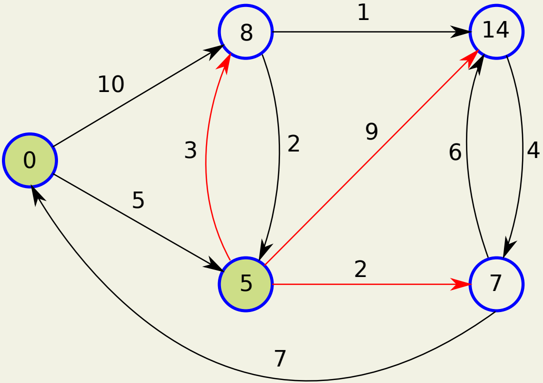
Example graph



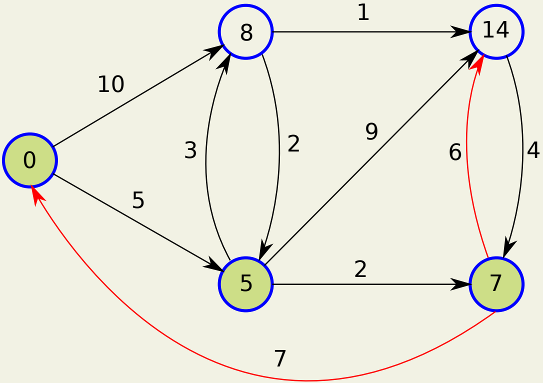
Example graph



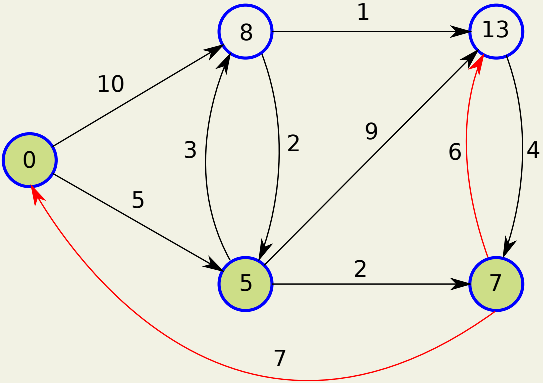
Example graph



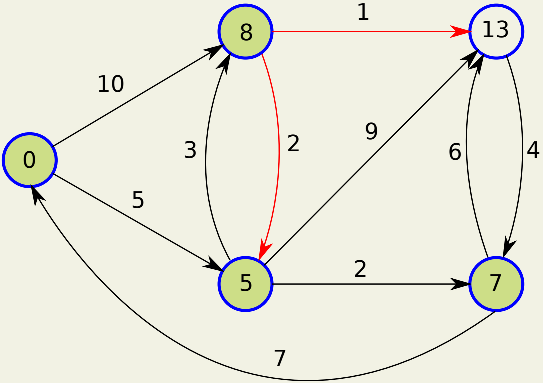
Example graph



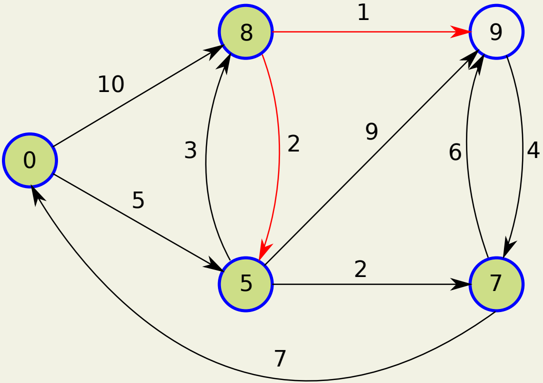
Example graph



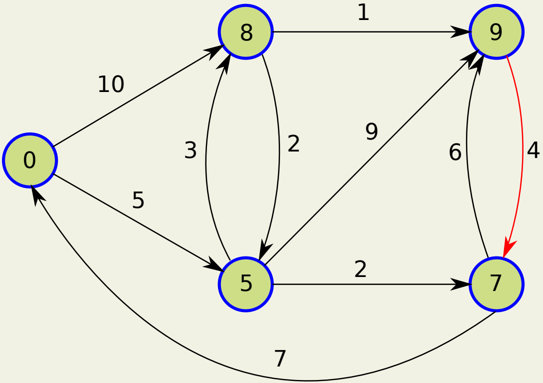
Example graph



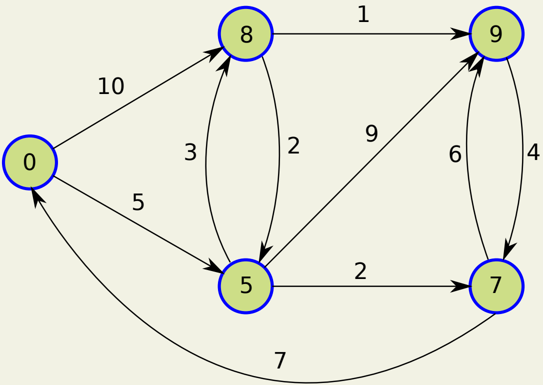
Example graph



Example graph



Example graph



Dijkstra's algorithm

"It is the algorithm for the shortest path, which I designed in about twenty minutes. One morning I was shopping in Amsterdam with my young fiancée, and tired, we sat down on the café terrace to drink a cup of coffee and I was just thinking about whether I could do this, and I then designed the algorithm for the shortest path. As I said, it was a twenty-minute invention."

-Edsger Dijkstra

Source: Wikipedia and "An Interview with Edsger W. Dijkstra". Communications of the ACM

Dijkstra's Algorithm Pseudocode

Algorithm 1 Dijkstra's algorithm

For all $u \in V$, $d[u] \leftarrow \infty$, $\pi[u] \leftarrow$ NIL
$d[s] \leftarrow 0$
Initialize min-priority queue $Q \leftarrow V$
$S \leftarrow \emptyset$
while $Q \neq \emptyset$ **do**
    $u \leftarrow \textsc{Extract-Min}(Q)$
    $S \leftarrow S \cup \{u\}$
    for each $v \in \mathcal{N}(u)$ **do**
        if $d[u] + w(u, v) < d[v]$ **then**
            $d[v] \leftarrow d[u] + w(u, v)$
            DECREASE-KEY$(v, d[v])$.
            $\pi[v] \leftarrow u$
        end if
    end for
end while

Proof of Correctness

Theorem

At the end of Dijkstra's algorithm, we have:

$$\forall u \in U, d[u] = \delta(s, u)$$

Proof

Loop Invariant:
At the start of each iteration, we have $\forall v \in S, d[v] = \delta(v)$.
Init: At the start of the first iteration, $S = \emptyset$.
Maintenance: Look at the start of the iteration in which a vertex $u \in V$ was added to S.
If u is added to S, then it must be reachable.
If $u = s$, then the claim holds. So assume $u \neq s$.

Proof of Correctness

Claim 1: $d[u] \geq \delta(s, u)$

Take a shortest path σ from s to u.

Let y be the first vertex on σ that is outside S.

Let $x \in S$ be the vertex on σ just before y.

So the path σ looks like:

$$s \overset{\sigma_1}{\leadsto} x \to y \overset{\sigma_2}{\leadsto} u$$

Claim 2: $d[y] = \delta(s, y)$.

Proof of Correctness

$$\sigma = s \overset{\sigma_1}{\rightsquigarrow} x \to y \overset{\sigma_2}{\rightsquigarrow} u$$

Claim 1: $d[u] \geq \delta(s, u)$.
Claim 2: $d[y] = \delta(s, y)$.
Since y appears before u in σ, we have $\delta(s, y) \leq \delta(s, u)$.
Hence:

$$d[y] = \delta(s, y) \leq \delta(s, u) \leq d[u]$$

Although y and u were in $V \setminus S$, Extract-Min returned u.
This means $d[u] \leq d[y]$. Hence:
$d[y] = \delta(s, y) = \delta(s, u) = d[u]$

$\square$

Proof of Correctness

Claim 2

$$\sigma = s \overset{\sigma_1}{\rightsquigarrow} x \to y \overset{\sigma_2}{\rightsquigarrow} u$$

We have $d[y] = \delta(s, y)$

Proof

From loop invariant, for all vertices that were added to S before u, we computed the correct shortest distance.
So $d[x] = \delta(s, x)$.
We updated $d[y]$ when we added x to S.
Now we note a *convergence* property:
If y is on a shortest path σ from s to u.
Then, the path formed by σ from s to y is a shortest path from s to y.

$\square$

Proof of Correctness

Claim 1

$d[u] \geq \delta(s, u)$

Proof

Induction on number of times d is updated after initialization.

Base case: Immediately after init, $\forall v$, $d[v] = \infty$ except $d[s] = 0$. So the claim holds.

Step: Assume claim for up to k many updates on d.

The value of $d[u]$ is updated when:

- We visit a vertex v and there exists edge (v, u).
- $d[u] > d[v] + w((v, u))$.

The new $d[u]$ is $d[v] + w((v, u))$.

The hypothesis holds for vertex v: $d[v] \geq \delta(s, v)$. So:

$$d[u] = d[v] + w((u, v)) \geq \delta(s, v) + w((u, v)) \geq \delta(s, u)$$

Spanning Tree

Definition: An undirected graph G is *connected* if every vertex is reachable from every other vertex.

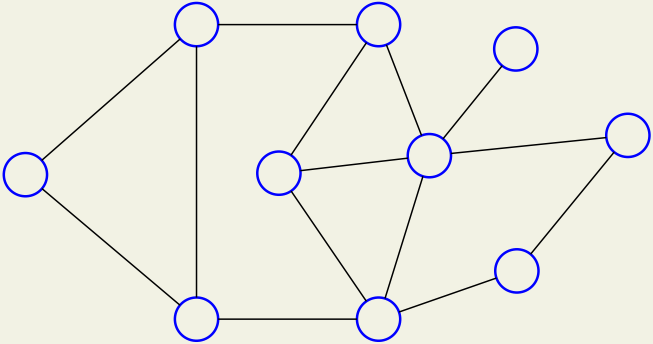
A graph $T = (V, E')$ is a spanning tree of an undirected connected graph $G = (V, E)$ if:

- $E' \subseteq E$.
- T is a *tree*. i.e., there are no cycles in T.
- T is connected.

Informally: A spanning tree for G is a tree that can be found inside G which *spans* all vertices of G.

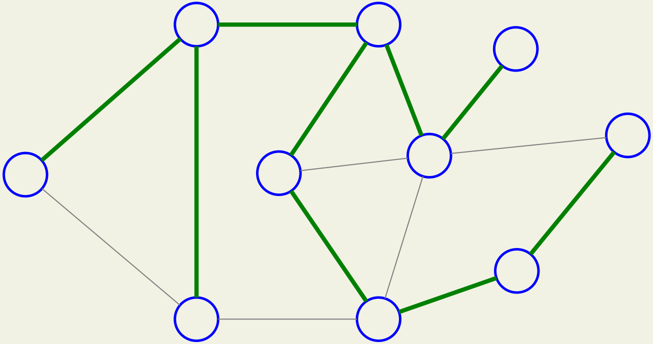
Examples

What are the possible spanning trees for this graph?



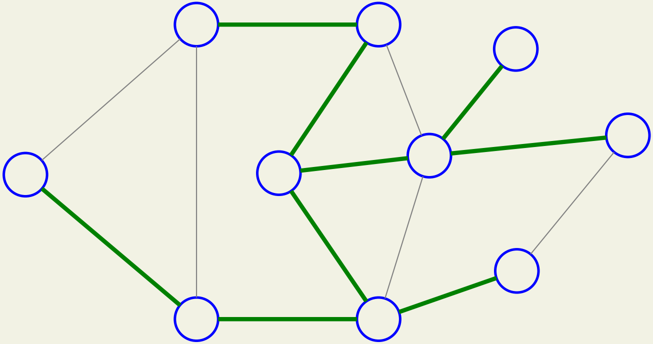
Examples

Is this a spanning tree?



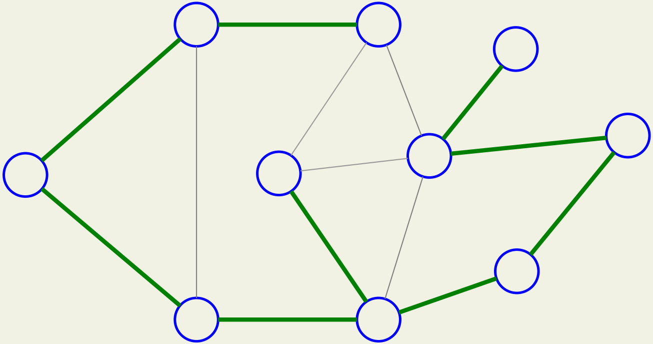
Examples

Is this a spanning tree?



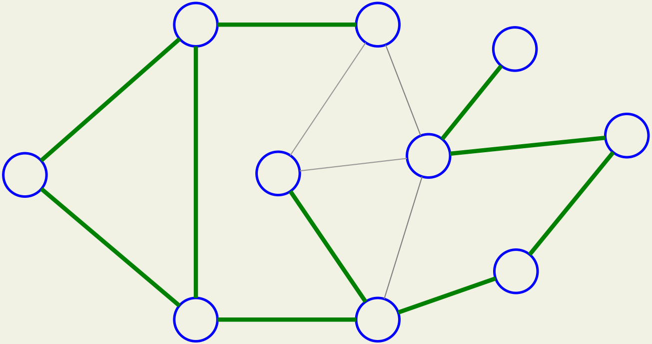
Examples

Is this a spanning tree?



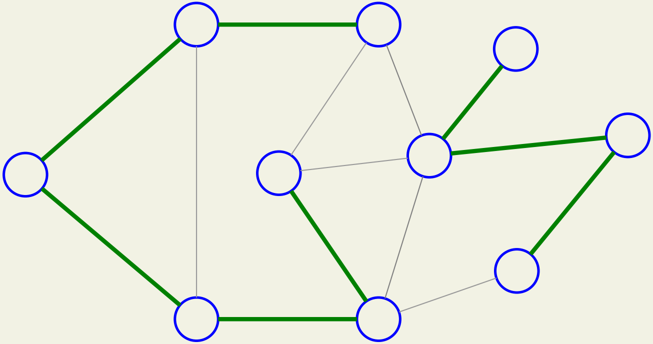
Examples

Is this a spanning tree?



Examples

Is this a spanning tree?



Minimum Spanning Tree Problem

Input

- Undirected connected graph $G = (V, E)$
- Weight function $w : E \to \mathbb{Z}^+$

Goal

Compute a spanning tree for G with minimum total weight.

Kruskal's algorthm example