Lecture 12

Instructor: Subrahmanyam Kalyanasundaram

26th September 2019

Plan

- Complete the proof of correctness of Dijkstra's
- Minimum spanning trees

Weighted Graphs

A weighted graph is a graph $G = (V, E)$ with a weight function:

$$w : E \to \mathbb{Z}$$

The weight of an edge $(u, v) \in E$ is $w((u, v))$.
For this lecture, we look at directed weighted graphs with weight function $w : E \to \mathbb{Z}^+$.

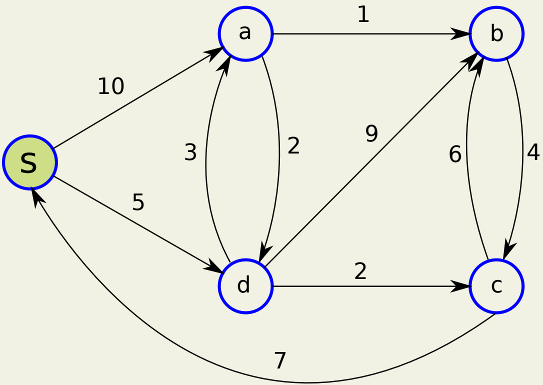
Shortest path in weighted graphs

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

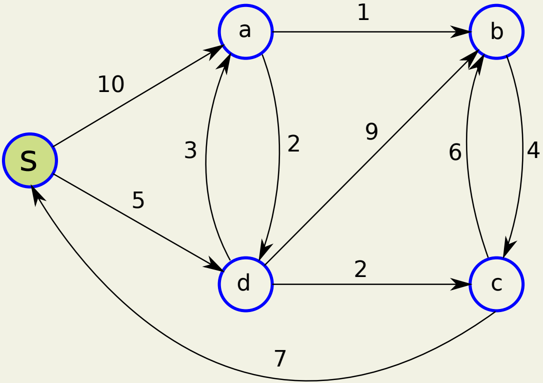


Dijkstra's Algorithm Pseudocode

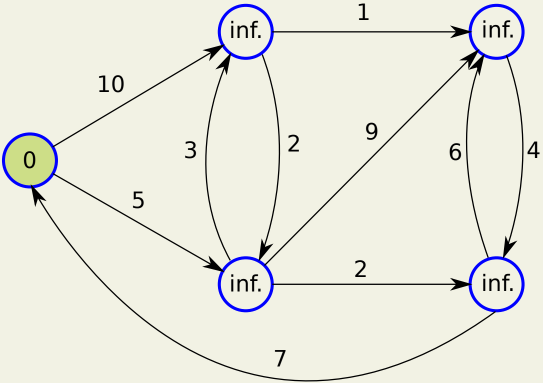
Algorithm 1 Dijkstra's algorithm

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

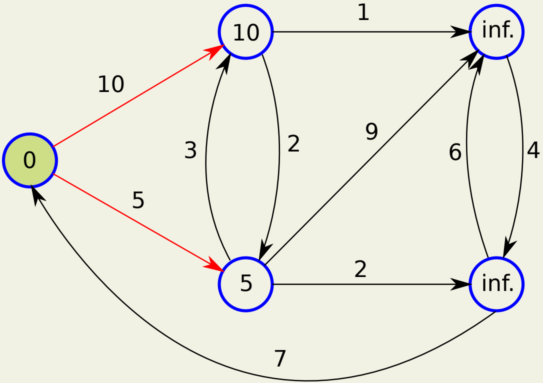
Example graph



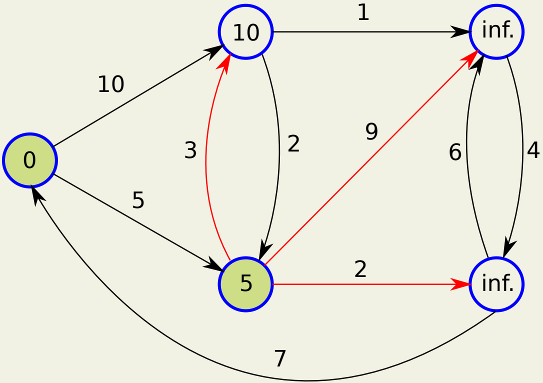
Example graph



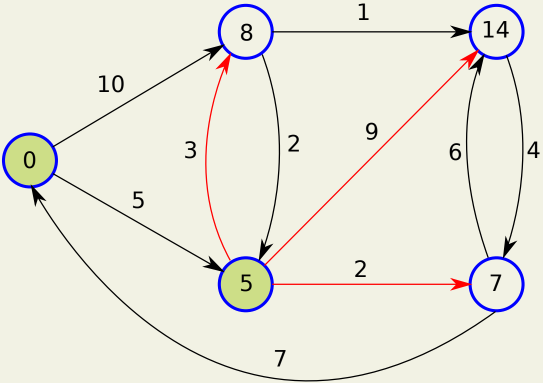
Example graph



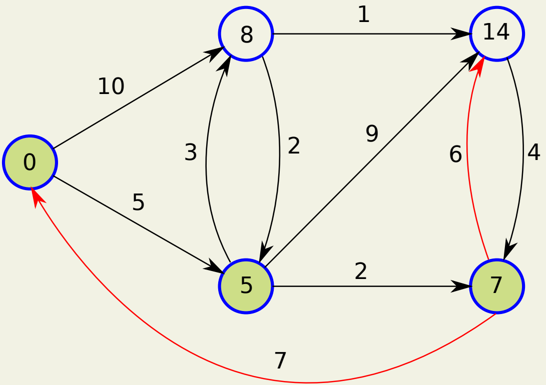
Example graph



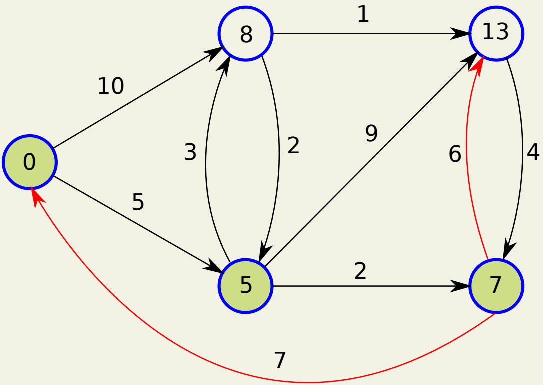
Example graph



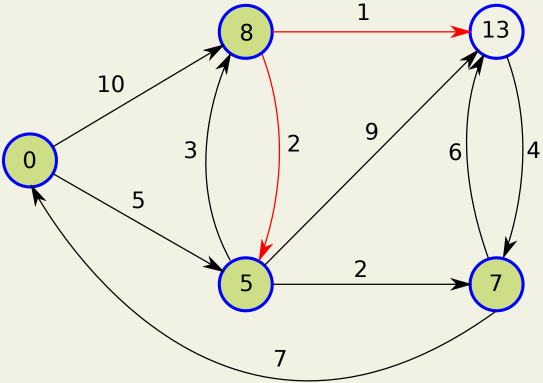
Example graph



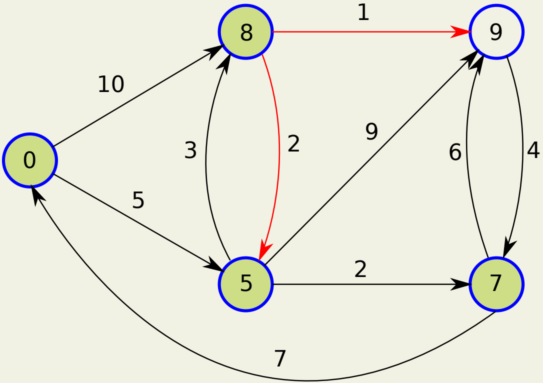
Example graph



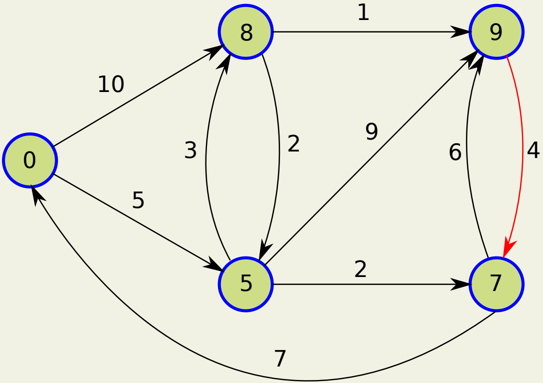
Example graph



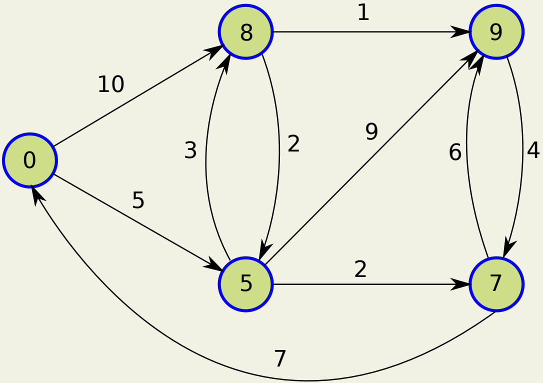
Example graph



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 2 Dijkstra's algorithm

```
For all u ∈ V, d[u] ← ∞, π[u] ← NIL
d[s] ← 0
Initialize min-priority queue Q ← V
S ← ∅
while Q ≠ ∅ do
    u ← EXTRACT-MIN(Q)
    S ← S ∪ {u}
    for each v ∈ 𝒩(u) do
        if d[u] + w(u, v) < d[v] then
            d[v] ← d[u] + w(u, v)
            DECREASE-KEY(v, d[v]).
            π[v] ← u
        end if
    end for
end while
```

Time Complexity of Dijkstra's

- Initialization: $O(|V|)$
- We need to do $|V|$ Extract-Min's and $|E|$ Decrease-Key's
- Depends on the implementation of the priority queue.

Time Complexity of Dijkstra's

- Initialization: $O(|V|)$
- We need to do $|V|$ Extract-Min's and $|E|$ Decrease-Key's
- Depends on the implementation of the priority queue.

- Array: Extract-Min takes $O(|V|)$ and Decrease-Key takes $O(1)$
- Heap: Extract-Min and Decrease-Key both take $O(\log |V|)$
  We need to maintain pointers from vertices to heap entries and vice versa.
- Fibonacci Heap: Decrease-Key takes $O(1)$ amortized time

Proof of Correctness

Theorem

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

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

Proof

Loop Invariant:
At the start of each iteration, we have $\forall v \in S, d[v] = \delta(s, v)$.
Init: At the start of the first iteration, $S = \emptyset$.
Maintenance: Let $u \in V$ be the first vertex for which $d[u] \neq \delta(s, u)$.
If u is not reachable from s, then $d[u] = \delta(s, u) = \infty$, so u must be reachable. Why?
If $u = s$, then the claim holds. So assume $u \neq s$.

Proof of Correctness

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:

$$s \overset{\sigma_1}{\rightsquigarrow} x \rightarrow y \overset{\sigma_2}{\rightsquigarrow} u$$

Claim 1: $d[y] = \delta(s, y)$.

Proof of Correctness

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

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

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

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 1

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

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:
Let $s \rightsquigarrow x \to y$ be a shortest path, and $d[x] = \delta(s, x)$.
Then, relaxing the edge (x, y) sets $d[y] = \delta(s, y)$.

$\square$

Proof of Correctness

Claim 2

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

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

Proof of Correctness

Claim 2

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

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 $d[u] = d[v] + w((v, u))$.
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.

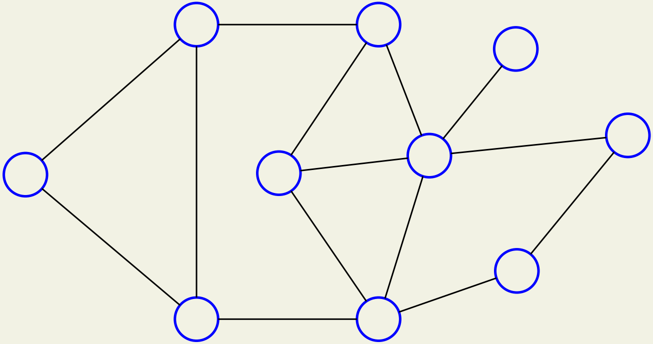
A graph $T = (V, E')$ is a spanning tree of an undirected connected graph $G = (V, E)$ if:

- $E' \subseteq E$.
- T is a *tree*. i.e., T is an acyclic and connected.

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

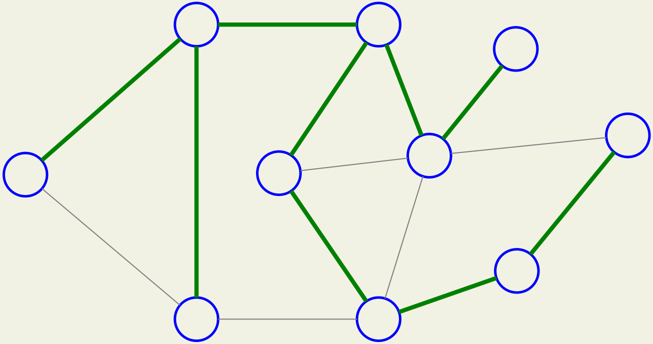
Examples

What are the possible spanning trees for this graph?



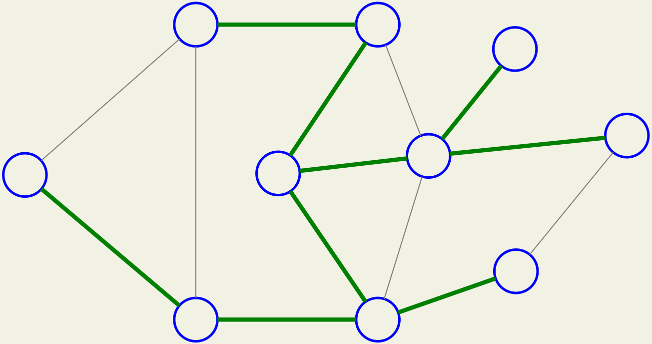
Examples

Is this a spanning tree?



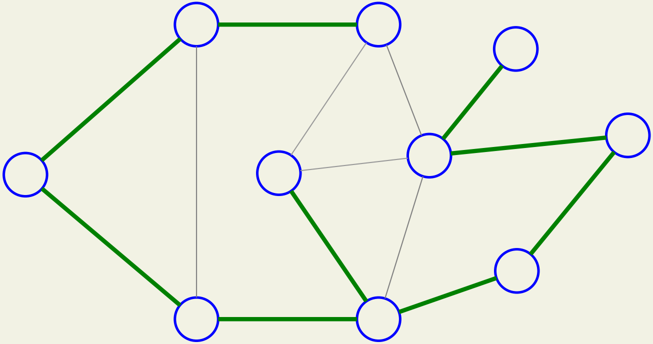
Examples

Is this a spanning tree?



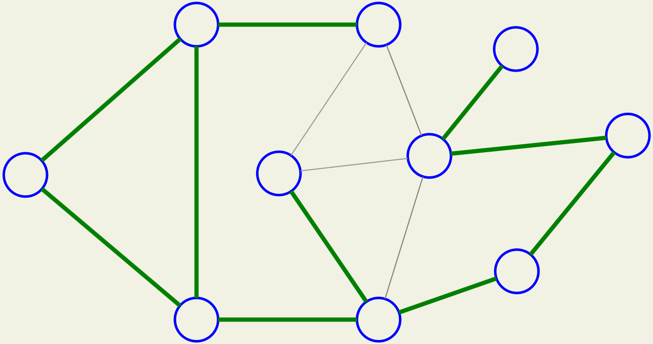
Examples

Is this a spanning tree?



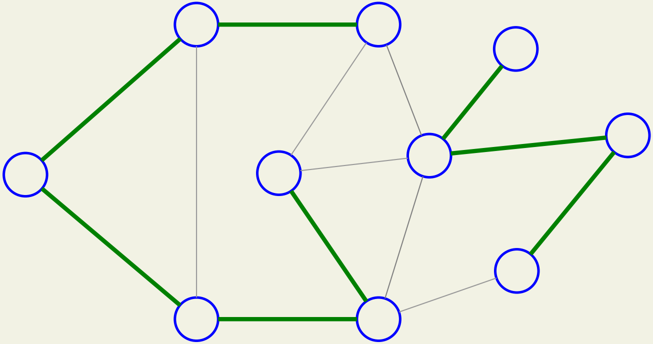
Examples

Is this a spanning tree?



Examples

Is this a spanning tree?



Minimum Spanning Tree Problem

Input

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

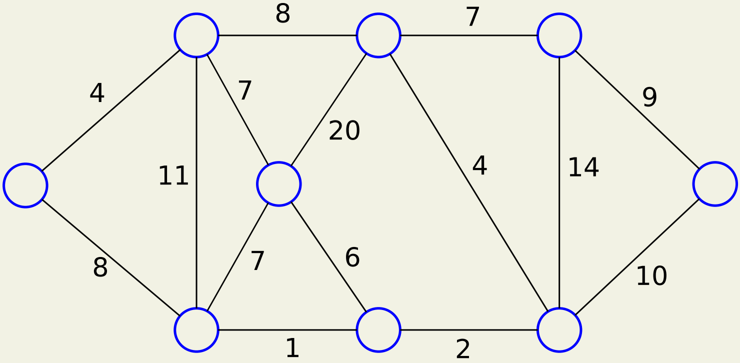
Goal

Compute a spanning tree for G with minimum total weight.

Kruskal's Algorithm (informal)

- Sort the edges in nondecreasing order by weight
- Set $T = \emptyset$
- Choose the lightest edge and add it to T as long as it does not create a cycle in T
- Terminate when T is spanning

Kruskal's algorthm example



Kruskal's Algorithm Pseudocode

Algorithm 3 Kruskal's algorithm

```
A = ∅
for each vertex v ∈ V do
    MAKE-SET(v)
end for
Sort the edges in E into nondecreasing order by weight w
for each edge (u, v) ∈ E taken in nondecreasing order by weight do
    if FIND-SET(u) ≠ FIND-SET(v) then
        A = A ∪ {(u, v)}
        UNION(u, v)
    end if
end for
Return A
```