Discrete Mathematics and Logic
Graph Theory
Lecture 2

Andrey Frolov
Professor

Innopolis University

What did we know in the last week?

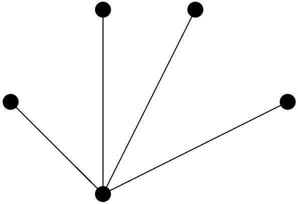
1. My favorite question is "Why?".
2. Books
3. The basic terminology of Graph Theory.
4. Handshaking lemma.
5. Connectivity.

Trees

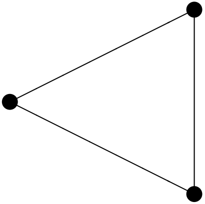
Definition

A connected graph having no cycle is called a **tree**.

a tree



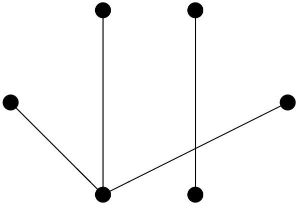
not a tree



Trees

Definition

A graph (it is not necessary to be connected) is called a **forest** if it does not contain any cycle.



Trees

Definition

A graph is called a **forest** if it does not contain any cycle.

A connected forest is called a **tree**.

Proposition

Any graph is a disjoint union of all its connected components.

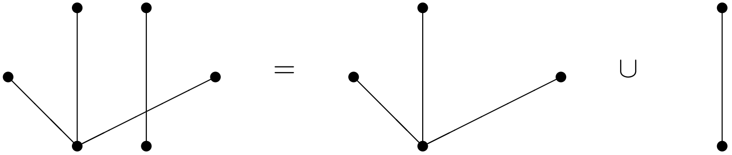
Proposition

Any forest is a disjoint union of trees (which are its components).

Trees

Proposition

Any forest is a disjoint union of trees (which are its components).



Trees

Theorem (equivalent definitions)

The following are equivalent for a graph T:

1) T is a tree,
2) any two vertices of T are connected by a unique path in T,
3) T is minimally connected,
4) T is maximally acyclic.

Trees

Theorem (equivalent definitions)

The following are equivalent for a graph T:

1) T is a tree,
2) any two vertices of T are connected by a unique path in T,
3) T is minimally connected, i.e.,
   - T is connected,
   - $T - e$ is not connected for any its edge e.
4) T is maximally acyclic.

Trees

Theorem (equivalent definitions)

The following are equivalent for a graph T:

1) T is a tree,
2) any two vertices of T are connected by a unique path in T,
3) T is minimally connected,
4) T is maximally acyclic, i.e.,
   - T contains no cycle, but
   - $T + (x, y)$ does for any two non-adjacent vertices $x, y \in V_T$.

Proof

Trees

Theorem (equivalent definitions)

The following are equivalent for a graph T:

1) T is a tree,
2) any two vertices of T are connected by a unique path in T,
3) T is minimally connected,
4) T is maximally acyclic, i.e.,
   - T contains no cycle, but
   - $T + (x, y)$ does for any two non-adjacent vertices $x, y \in V_T$.

Proof

Homework!

Trees

Proposition (home-work)

Any tree has a vertex with degree 1

(even any non-trivial tree has at least two such vertices).

Trees

Theorem (the characteristic property for trees)

Let G be a connected graph with n vertices and e edges.

$$G \text{ is a tree iff } n = e + 1.$$

Trees

Theorem

Let G be a connected graph with n vertices and e edges.

$$G \text{ is a tree iff } n = e + 1.$$

Proof (by the induction)

Base.

Hypothesis.

Inductive step.

Trees

Theorem

Let G be a connected graph with n vertices and e edges.

$$G \text{ is a tree iff } n = e + 1.$$

Proof (by the induction)

Base. Let $n = 1$. G have no edges. It is obvious.

Hypothesis. Suppose that the theorem holds for any $k < n$.

Trees

Theorem

Let G be a connected graph with n vertices and e edges.

$$G \text{ is a tree iff } n = e + 1.$$

Proof (by the induction)

Inductive step. ($\Rightarrow$). Let G be a tree.

- Choose a vertex v having degree 1.
- Then $G - v$ is a tree with $n - 1$ vertices.
- Therefore, (by the induction hypothesis) $G - v$ has $n - 2$ edges.
- So, G has $n - 1$ edges.

Trees

Theorem

Let G be a connected graph with n vertices and e edges.

$$G \text{ is a tree iff } n = e + 1.$$

Proof (by induction)

Inductive step. ($\Leftarrow$). Let G' be a connected graph with $n-1$ edges.

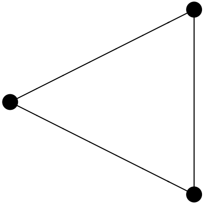
- Suppose that $G' = (V_G, E') \subseteq G$ is a tree (such G' is called a spanning tree).
- Since G' has n vertices and $n-1$ edges,
  and G has n vertices and $n-1$ edges,
  by the first implication it follows that $G' = G$.

Spanning trees

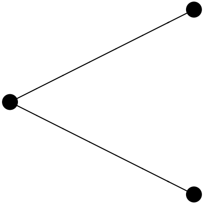
Definition

Let $G = (V, E)$ be a connected graph. A graph $G' = (V, E') \subseteq G$ is called a **spanning tree**, if G' is a tree.

a given graph



its spanning tree



Spanning trees

Theorem

Any connected graph has a spanning tree.

Proof

You need to remove (step by step) edges from cycles until the graph becomes a tree.

Weighted graphs

Definition

A graph is called weighted if each vertex or edge has an associated numerical value, i.e.,

- a vertex-weighted graph has weights on its vertices,
- an edge-weighted graph has weights on its edges.

Weighted graphs

Definition

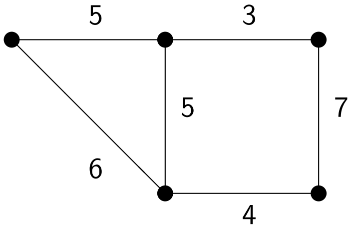
Formally, a graph $G = (V, E)$ is vertex-weighted,
if there is a total function $f : V \to \mathbb{R}$ ($\mathbb{Z}$ or $\mathbb{N}$).

G is edge-weighted if there is a total function $f : E \to \mathbb{R}$ ($\mathbb{Z}$ or $\mathbb{N}$).

Usually, edge weights are non-negative.

Minimal spanning trees

The total **weight** of a graph is the **sum** of the weights of its edges.



The weight of the graph $=$

$$= 5 + 3 + 6 + 5 + 7 + 4 = 30$$

Minimal spanning trees

Let G be a edge-weighted graph.

Problem

We need to build a spanning tree T with minimal total weight.

Minimal spanning trees

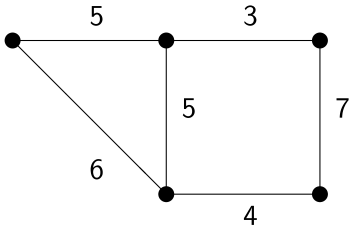
The simplest (but not optimal) **minimal spanning tree algorithm**

We start from $(V_G, \emptyset)$.

Repeat as long as possible. If T is not a tree, add to T a new edge e with minimal possible weight from $G - T$ such that $T + e$ has no cycle.

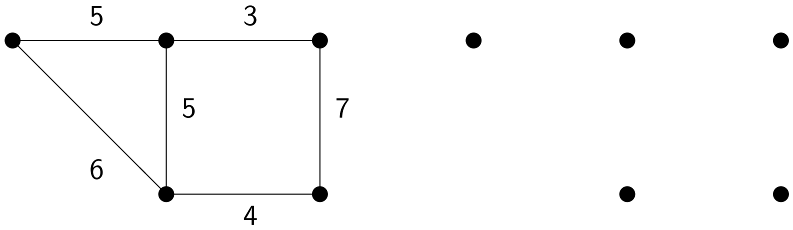
Minimal spanning tree

The simplest (but not optimal) **minimal spanning tree algorithm**



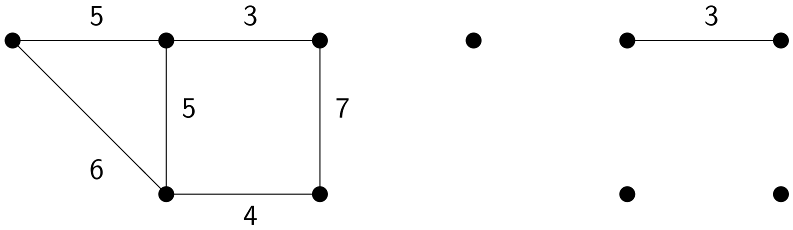
Minimal spanning tree

The simplest (but not optimal) **minimal spanning tree algorithm**



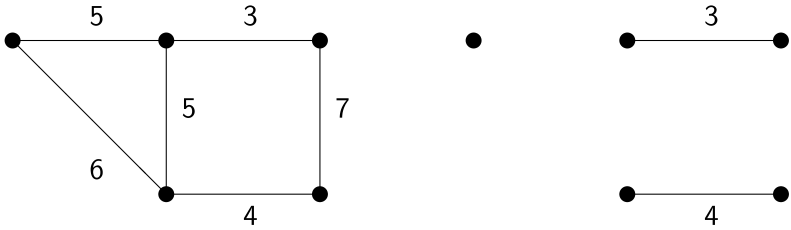
Minimal spanning tree

The simplest (but not optimal) **minimal spanning tree algorithm**



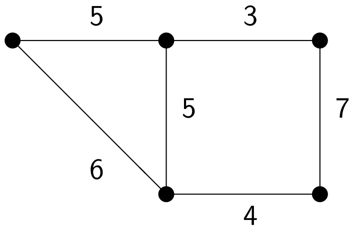
Minimal spanning tree

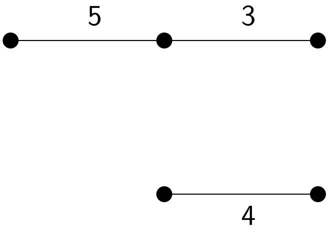
The simplest (but not optimal) **minimal spanning tree algorithm**



Minimal spanning tree

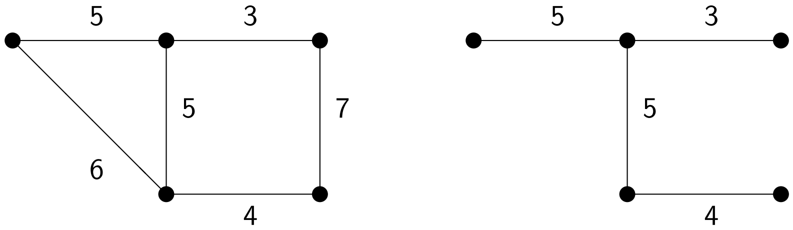
The simplest (but not optimal) **minimal spanning tree algorithm**





Minimal spanning tree

The simplest (but not optimal) **minimal spanning tree algorithm**



The Prim's algorithm

Let G be a edge-weighted graph.

Problem

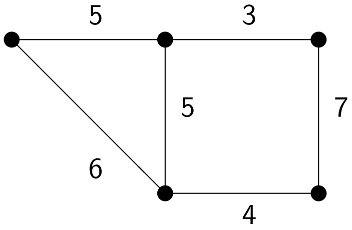
We need to build a spanning tree T with minimal total weight.

An optimal decision is the **Prim's algorithm**

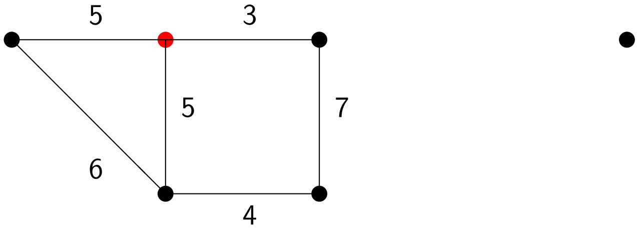
The Prim's algorithm

1. We start from a **single vertex**, chosen arbitrarily from the graph.
2. Grow the tree T by one edge:
   - find the edges that connect T to vertices not yet in T,
   - choose from them the minimum-weight edge,
   - transfer it to the tree T.
3. Repeat step 2 (until all vertices are in the tree T).

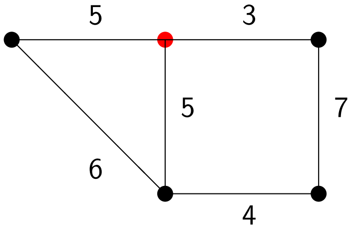
The Prim's algorithm

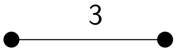


The Prim's algorithm

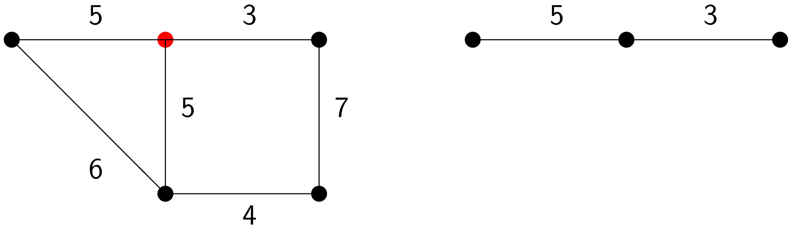


The Prim's algorithm

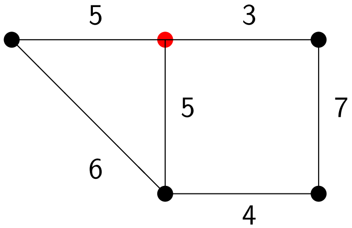


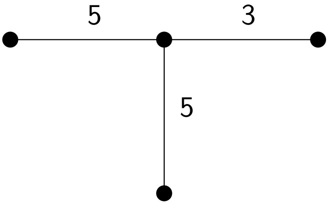


The Prim's algorithm

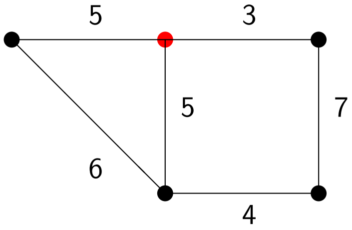


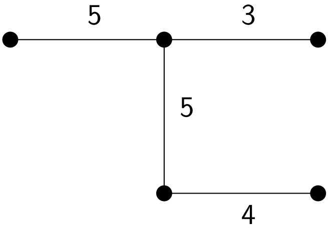
The Prim's algorithm





The Prim's algorithm





What we knew today?

1. Trees and forests
2. The characteristic property for trees
3. Spanning trees
4. Weighted graphs
5. Minimal spanning trees
6. The Prim's algorithm
7. Don't forget that my favorite question is "Why?"!

Thank you for your attention!