

Minimum-weight spanning trees

Motivation: Create a connected network as cheaply as possible.

- ▶ Think: Setting up electrical grid or road network.
- ▶ Some connections are cheaper than others.
- ▶ Only need to minimally connect the vertices.

Minimum-weight spanning trees

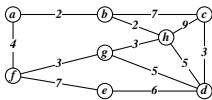
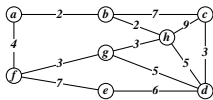
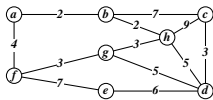
Motivation: Create a connected network as cheaply as possible.

- ▶ Think: Setting up electrical grid or road network.
- ▶ Some connections are cheaper than others.
- ▶ Only need to minimally connect the vertices.

Definition: A **weighted graph** consists of a graph $G = (V, E)$ and **weight function** $w : E \rightarrow \mathbb{R}$ defined on the edges of G .

The **weight** of a subgraph H of G is the **sum** of the edges in H .

Example.



Minimum-weight spanning trees

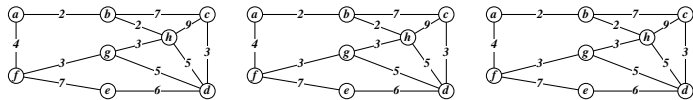
Motivation: Create a connected network as cheaply as possible.

- ▶ Think: Setting up electrical grid or road network.
- ▶ Some connections are cheaper than others.
- ▶ Only need to minimally connect the vertices.

Definition: A **weighted graph** consists of a graph $G = (V, E)$ and **weight function** $w : E \rightarrow \mathbb{R}$ defined on the edges of G .

The **weight** of a subgraph H of G is the **sum** of the edges in H .

Example.



Definition: For a graph G , a **spanning tree** T is a subgraph of G which is a tree and contains every vertex of G .

Goal: For a weighted graph G , find a minimum-weight spanning tree.

Kruskal's algorithm

Kruskal's Algorithm finds a minimum-weight spanning tree in a weighted graph.

1 Initialization: Order the edges from lowest to highest weight:

$$w(e_1) \leq w(e_2) \leq w(e_3) \leq \cdots \leq w(e_k).$$

Kruskal's algorithm

Kruskal's Algorithm finds a minimum-weight spanning tree in a weighted graph.

- 1 Initialization: Order the edges from lowest to highest weight:

$$w(e_1) \leq w(e_2) \leq w(e_3) \leq \cdots \leq w(e_k).$$

- 2 **Step 1:** Define $T = \{e_1\}$ and grow the tree as follows:

Kruskal's algorithm

Kruskal's Algorithm finds a minimum-weight spanning tree in a weighted graph.

- 1 Initialization: Order the edges from lowest to highest weight:

$$w(e_1) \leq w(e_2) \leq w(e_3) \leq \cdots \leq w(e_k).$$

- 2 **Step 1:** Define $T = \{e_1\}$ and grow the tree as follows:
- 3 **Step i :** Determine if adding e_i to T would create a cycle.
 - ▶ If not, add e_i to the set T .
 - ▶ If so, do nothing.

Kruskal's algorithm

Kruskal's Algorithm finds a minimum-weight spanning tree in a weighted graph.

- 1 Initialization: Order the edges from lowest to highest weight:

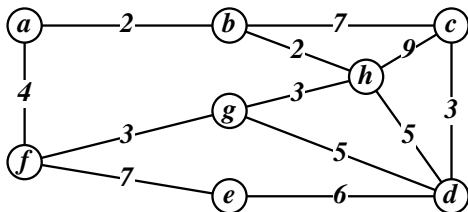
$$w(e_1) \leq w(e_2) \leq w(e_3) \leq \cdots \leq w(e_k).$$

- 2 **Step 1:** Define $T = \{e_1\}$ and grow the tree as follows:
- 3 **Step i :** Determine if adding e_i to T would create a cycle.
 - ▶ If not, add e_i to the set T .
 - ▶ If so, do nothing.

If you have a spanning tree, STOP. You have a m.w.s.t.
Otherwise, continue onto step $i + 1$.

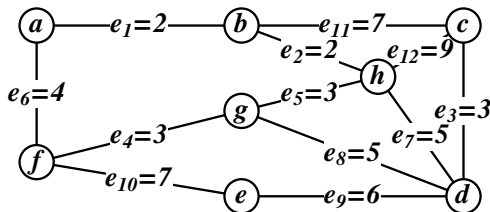
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



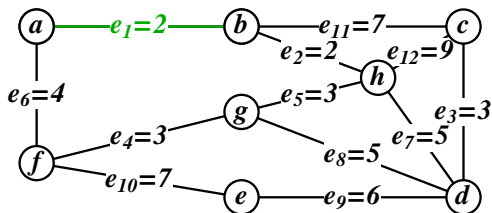
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



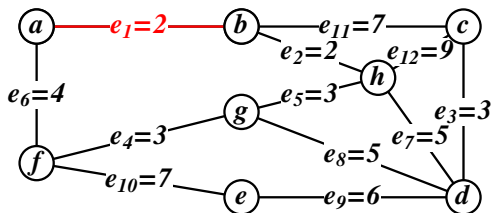
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



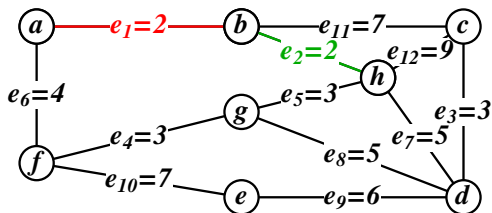
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



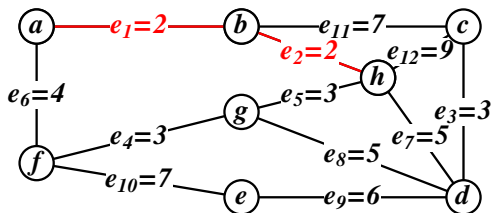
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



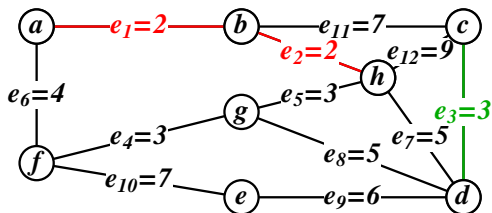
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



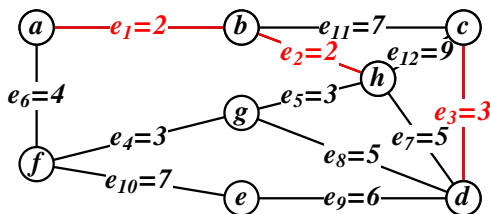
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



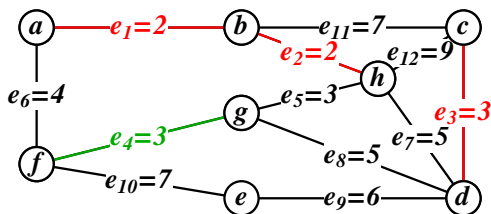
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



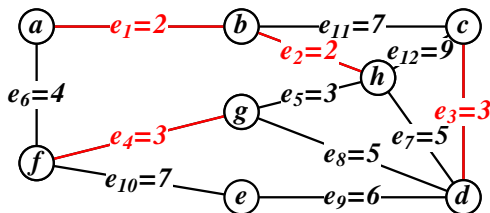
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



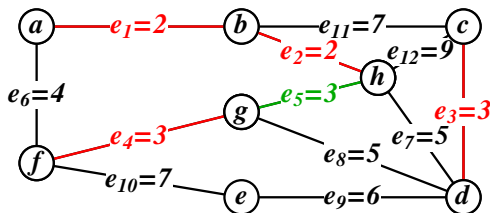
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



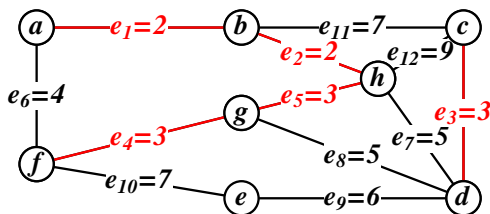
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



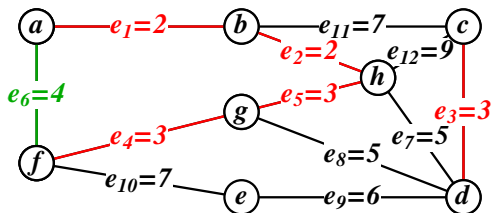
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



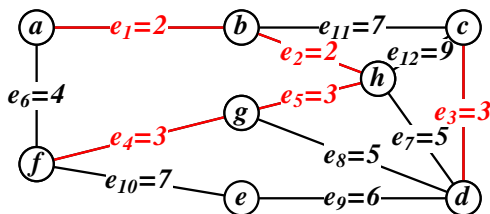
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



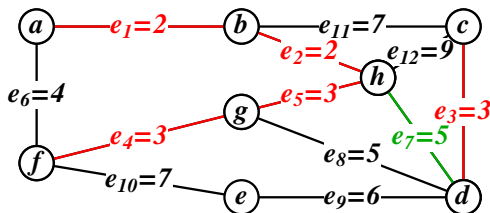
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



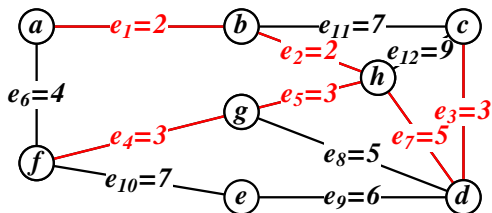
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



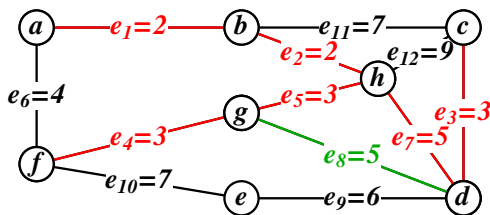
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



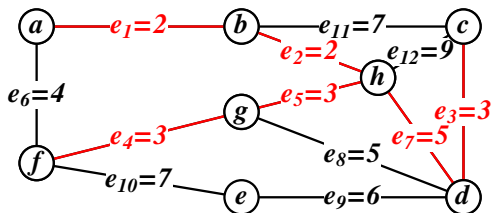
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



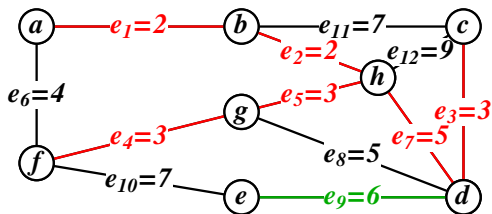
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



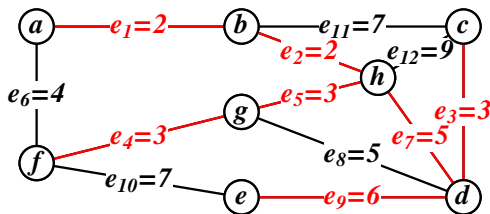
Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



Kruskal's algorithm

Example. Run Kruskal's algorithm on the following graph:



Notes on Kruskal's algorithm

- ▶ Proof of correctness similar to homework. Must additionally verify that the spanning tree is indeed minimum-weight.

Notes on Kruskal's algorithm

- ▶ Proof of correctness similar to homework. Must additionally verify that the spanning tree is indeed minimum-weight.
- ▶ Kruskal's algorithm is an example of a *greedy algorithm*. (It chooses the cheapest edge at each point.)

Notes on Kruskal's algorithm

- ▶ Proof of correctness similar to homework. Must additionally verify that the spanning tree is indeed minimum-weight.
- ▶ Kruskal's algorithm is an example of a *greedy algorithm*. (It chooses the cheapest edge at each point.)
- ▶ Greedy algorithms don't always work.

The traveling salesman problem

Motivation: Visit all nodes and return home as cheaply as possible.

The traveling salesman problem

Motivation: Visit all nodes and return home as cheaply as possible.

- ▶ Least cost trip flying between five major cities.
- ▶ Optimal routes for delivering mail, collecting garbage.
- ▶ Finding a trip to all buildings on campus, return.

The traveling salesman problem

Motivation: Visit all nodes and return home as cheaply as possible.

- ▶ Least cost trip flying between five major cities.
- ▶ Optimal routes for delivering mail, collecting garbage.
- ▶ Finding a trip to all buildings on campus, return.

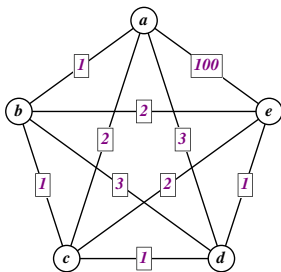
Goal: Find a minimum-weight **Hamiltonian cycle** in a weighted graph.

The traveling salesman problem

Motivation: Visit all nodes and return home as cheaply as possible.

- ▶ Least cost trip flying between five major cities.
- ▶ Optimal routes for delivering mail, collecting garbage.
- ▶ Finding a trip to all buildings on campus, return.

Goal: Find a minimum-weight **Hamiltonian cycle** in a weighted graph.

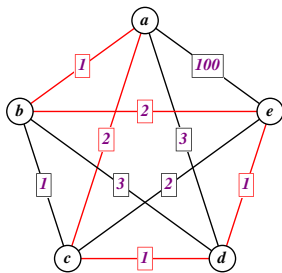


The traveling salesman problem

Motivation: Visit all nodes and return home as cheaply as possible.

- ▶ Least cost trip flying between five major cities.
- ▶ Optimal routes for delivering mail, collecting garbage.
- ▶ Finding a trip to all buildings on campus, return.

Goal: Find a minimum-weight **Hamiltonian cycle** in a weighted graph.

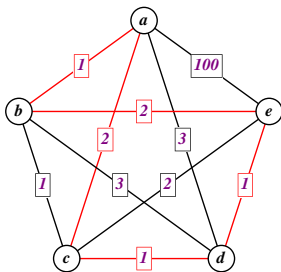


The traveling salesman problem

Motivation: Visit all nodes and return home as cheaply as possible.

- ▶ Least cost trip flying between five major cities.
- ▶ Optimal routes for delivering mail, collecting garbage.
- ▶ Finding a trip to all buildings on campus, return.

Goal: Find a minimum-weight **Hamiltonian cycle** in a weighted graph.



We can not use a greedy algorithm to find this **TSP tour**!

The traveling salesman problem

- ▶ It is *hard* to find an optimum solution.

The traveling salesman problem

- ▶ It is *hard* to find an optimum solution.
- ▶ Goal: Create an easy-to-find *pretty good* solution.

The traveling salesman problem

- ▶ It is *hard* to find an optimum solution.
- ▶ Goal: Create an easy-to-find *pretty good* solution.

Theorem. When the edge weights satisfy the triangle inequality, the *tree shortcut algorithm* finds a tour that costs at most twice the optimum tour.

Recall. The **triangle inequality** says that if x , y , and z are vertices, then $\text{wt}(xy) + \text{wt}(yz) \leq \text{wt}(xz)$.

The traveling salesman problem

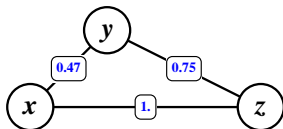
- ▶ It is *hard* to find an optimum solution.
- ▶ Goal: Create an easy-to-find *pretty good* solution.

Theorem. When the edge weights satisfy the triangle inequality, the *tree shortcut algorithm* finds a tour that costs at most twice the optimum tour.

Recall. The **triangle inequality** says that if x , y , and z are vertices, then $\text{wt}(xy) + \text{wt}(yz) \leq \text{wt}(xz)$.

Example: Euclidean distances

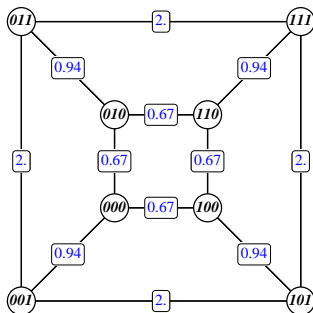
Non-example: Airfares



Finding a good TSP-tour

The **Tree Shortcut Algorithm** to find a good TSP-tour

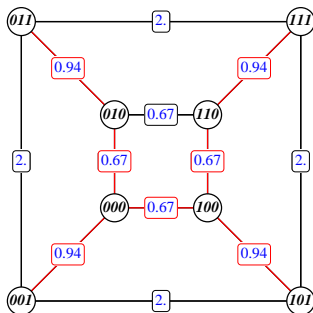
- 1 Find a minimum-weight spanning tree (Use Kruskal's Algorithm)
- 2 Walk in a circuit around the edges of the tree.
- 3 Take shortcuts to find a tour.



Finding a good TSP-tour

The **Tree Shortcut Algorithm** to find a good TSP-tour

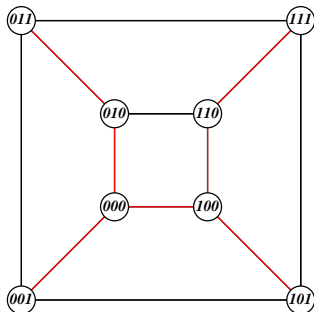
- 1 Find a minimum-weight spanning tree (Use Kruskal's Algorithm)
- 2 Walk in a circuit around the edges of the tree.
- 3 Take shortcuts to find a tour.



Finding a good TSP-tour

The **Tree Shortcut Algorithm** to find a good TSP-tour

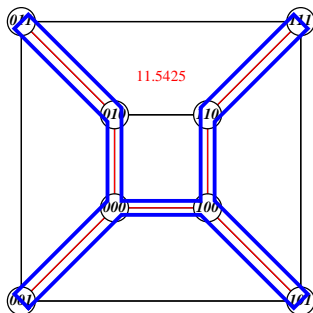
- 1 Find a minimum-weight spanning tree (Use Kruskal's Algorithm)
- 2 Walk in a circuit around the edges of the tree.
- 3 Take shortcuts to find a tour.



Finding a good TSP-tour

The **Tree Shortcut Algorithm** to find a good TSP-tour

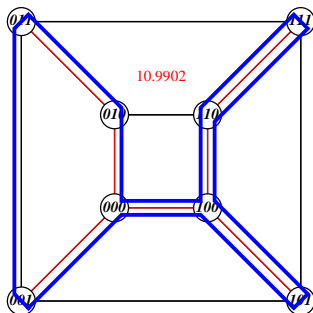
- 1 Find a minimum-weight spanning tree (Use Kruskal's Algorithm)
- 2 Walk in a circuit around the edges of the tree.
- 3 Take shortcuts to find a tour.



Finding a good TSP-tour

The **Tree Shortcut Algorithm** to find a good TSP-tour

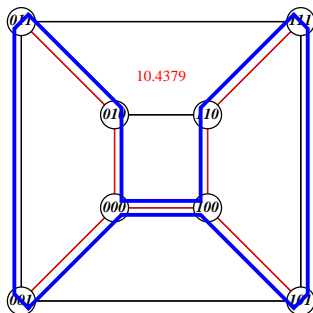
- 1 Find a minimum-weight spanning tree (Use Kruskal's Algorithm)
- 2 Walk in a circuit around the edges of the tree.
- 3 Take shortcuts to find a tour.



Finding a good TSP-tour

The **Tree Shortcut Algorithm** to find a good TSP-tour

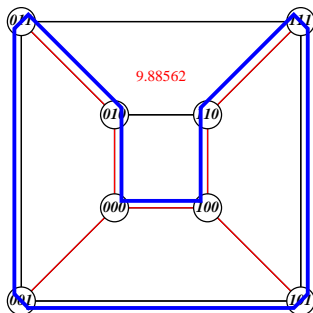
- 1 Find a minimum-weight spanning tree (Use Kruskal's Algorithm)
- 2 Walk in a circuit around the edges of the tree.
- 3 Take shortcuts to find a tour.



Finding a good TSP-tour

The **Tree Shortcut Algorithm** to find a good TSP-tour

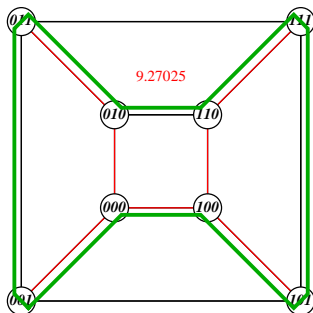
- 1 Find a minimum-weight spanning tree (Use Kruskal's Algorithm)
- 2 Walk in a circuit around the edges of the tree.
- 3 Take shortcuts to find a tour.



Finding a good TSP-tour

The **Tree Shortcut Algorithm** to find a good TSP-tour

- 1 Find a minimum-weight spanning tree (Use Kruskal's Algorithm)
- 2 Walk in a circuit around the edges of the tree.
- 3 Take shortcuts to find a tour.



Proof of theorem

Theorem. When the edge weights satisfy the triangle inequality, the *tree shortcut algorithm* finds a tour that costs at most twice the optimum tour.

Proof of theorem

Theorem. When the edge weights satisfy the triangle inequality, the *tree shortcut algorithm* finds a tour that costs at most twice the optimum tour.

Proof. Define:

- ▶ TSP_A : TSP tour from shortcutting spanning tree
- ▶ $CIRC_A$: Circuit constructed by doubling spanning tree
- ▶ MST : Minimum-weight spanning tree
- ▶ TSP^* : Minimum-weight TSP tour

Proof of theorem

Theorem. When the edge weights satisfy the triangle inequality, the *tree shortcut algorithm* finds a tour that costs at most twice the optimum tour.

Proof. Define:

- ▶ TSP_A : TSP tour from shortcutting spanning tree
- ▶ $CIRC_A$: Circuit constructed by doubling spanning tree
- ▶ MST : Minimum-weight spanning tree
- ▶ TSP^* : Minimum-weight TSP tour

Then,

$$\text{wt}(TSP_A) \leq \text{wt}(CIRC_A) = 2 \text{wt}(MST) \leq 2 \text{wt}(TSP^*).$$