2 Minimum Spanning Tree

2.1 Motivation and Definition

- Consider the oasis problem:
  - Desert state consists of seven oasis which are connected through trade routes
  - Trade routes are damaged because of wind and sun $\Rightarrow$ renovation is necessary
  - Not every trade road should be repaired, it's sufficient to make sure that we can reach every oasis from every other oasis
  - Which roads should be renovated?

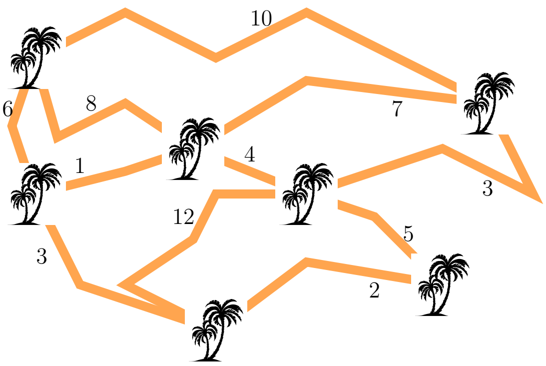

Fig. 2.1: The oasis problem (No. 538)

  - In order to obtain a good infrastructure, a spanning tree with minimum cost is the best choice

Definition 23. Minimum Spanning Tree (MST) Problem

Given: Undirected, connected graph $G = (V, E)$ and edge cost $c : E \to \mathbb{R}$

Find: A spanning tree T with minimum cost $c(T)$ with

$$c(T) := \sum_{e \in E(T)} c(e)$$

- In 1889, Caley proves that enumeration isn't a smart method to find an MST:
  - There exist n^{n-2} different spanning trees in a complete graph on n vertices
  - Assume that we can enumerate 10^6 trees per second:
    for $n = 30$ we have $\frac{30^{28}}{10^6}$sec $\approx 7.25 \cdot 10^{27}$years

2.2 Optimality Criterion

- Can we somehow characterize minimum spanning trees?

- Or: Is there a simple criterion with which we can decide whether a given tree is minimum?
- Such a criterion is called an optimality criterion

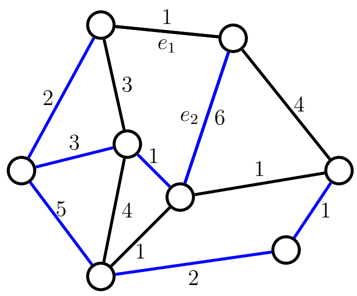

Fig. 2.2: Is the blue spanning tree an MST? (No. 535)

- If we can add a (cheap) edge and delete a more expensive one but keep a tree, our tree is not optimal
- More formally:

Definition 24 (Fundamental Cycle)**.** Let $T = (V, E(T))$ be a spanning tree in a graph $G = (V, E)$ and $e = E \backslash E(T)$ be a *non-tree edge.* The created *fundamental cycle* C_e w.r.t. T is the simple unique cycle which results from $T + e$.

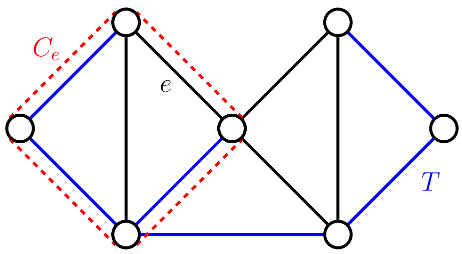

Fig. 2.3: Fundamental cycle (No. 540)

- We first need to prove that this definition makes sense

Lemma 25 (Fundamental Cycle-Lemma)**.** *Let T be a spanning tree of $G = (V, E)$. The fundamental cycle C_e w.r.t. T which is created by the non-tree edge $e \in E(G) \backslash E(T)$ is well-defined.*

Proof. Properties of a tree

- Let $e = uv$ be a non-tree edge
- *Claim 1*: If we add e to T, we obtain a cycle.
  Proof of Claim:

- T is a spanning tree $\Rightarrow \exists$ a path p from u to v in T

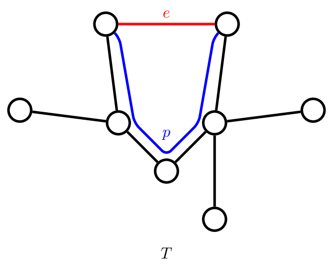

Fig. 2.4: Fundamental cycle exists (No. 657)

- $p + e$ is a cycle which contains e □C1

- $\Rightarrow$ there exists a fundamental cycle in $T + e$
- *Claim 2*: The fundamental cycle C_e is unique.
  Proof of Claim:
  - Assume there exist two different cycles C_1 and C_2 in $T + e$
  - Both cycles have to contain the edge e since T is a tree
  - Delete edge e from both cycles

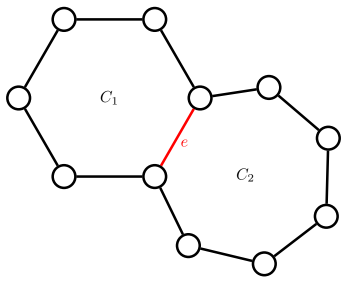

Fig. 2.5: A fundamental cycle is unique (No. 658)

  - $\Rightarrow$ there remain two different simple paths from u to v in T
  - Contradiction to T being a tree □C2

□

- Next, we need to formulate the "exchange" argument (cheap non-tree edge vs. expensive tree edge)

Lemma 26. *[Cycle Criterion] Let T be a spanning tree of G. If T is a minimum spanning tree, then for all non-tree edges $e \in E(G) \backslash E(T)$ this edge is the most expensive edge in the corresponding fundamental cycle, i.e., $c(e') \leq c(e)$ $\forall e' \in C_e$.*

Proof. Edge-exchange

- Assume there exists an edge $e \in E(G) \backslash E(T)$ and an edge $e' \in C_e$ with $c(e') > c(e)$
- Define the graph $T' = T - e' + e$
- T' contains $n - 1$ edges and no cycles, as the unique cycle in $T + e$ is destroyed

- $\Rightarrow T'$ is a tree (Theorem Important characteristics of trees)
- $\Rightarrow c(T') < c(T)$, contradiction

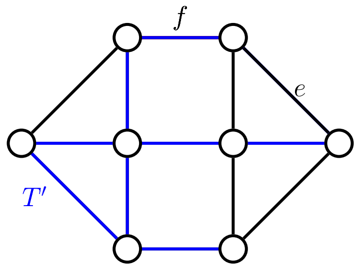

Fig. 2.6: T' is cheaper than T (No. 539)

□

- Is there another criterion to make sure that we have an MST?

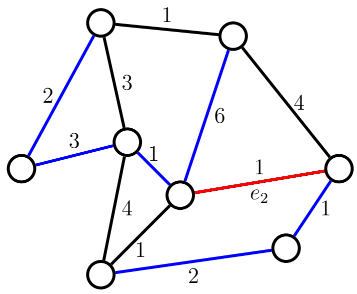

Fig. 2.7: Is the above spanning tree an MST? (No. 536)

- If we can delete an edge from the tree and add a cheaper one to reconnect the two parts, our tree is not optimal
- Let $\emptyset \neq X \subsetneq V$. A *cut* $\delta(X)$ is the set of edges with exactly one end vertex in X, i.e., $\delta(X) = \{uv \in E \mid u \in X, v \notin X\}$
- We call the cut $\delta(X)$ *induced by* X

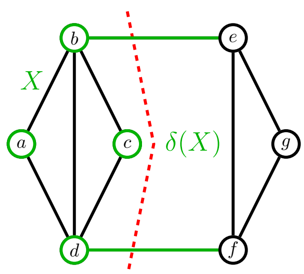

Fig. 2.8: A cut is the edge set that divides the set of vertices X from the other vertices (No. 543)

Definition 27 (Fundamental Cut)**.** Let $T = (V, E(T))$ be an MST of graph $G = (V, E)$ and $e = uv \in E(T)$ be a tree edge. Let X_e be the set of vertices which are reachable from u in $T - e$. The *fundamental cut* w.r.t. T which is created by e is the cut $\delta(X_e)$ in G.

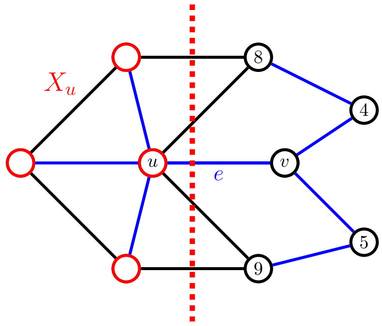

Fig. 2.9: Fundamental cut (No. 537)

- As for fundamental cycles, we need to prove that the definition of fundamental cuts is valid

Lemma 28 (Fundamental Cut-Lemma)**.** *Let T be a spanning tree of $G = (V, E)$. The induced fundamental cut $\delta(X_e)$ w.r.t. T which is created by the tree edge $e = uv \in E(T)$ is well-defined. Furthermore, e is the unique tree edge in $\delta(X_e)$, i.e., $\{\delta(X_e) \cap E(T)\} = \{e\}$.*

Proof. Properties of a tree

- Let $e \in E(T)$ be a tree edge
- *Claim 1*: $X_e \neq \emptyset$ and $X_e \subsetneq V$.
  Proof of Claim:
  - There exists a unique path between two vertices in a tree T
  - Delete edge $e = uv \Rightarrow$ there exist no longer a path between vertices u, v
  - W.l.o.g. $v \notin X_e$ and $u \in X_e$

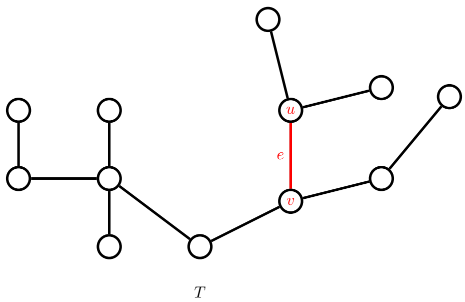

Fig. 2.10: Set X_e induces a cut (No. 659)

  - $\Rightarrow \emptyset \neq X_e \subsetneq V$ □C1
- $\Rightarrow$ the cut induced by X_e is well defined
- *Claim 2*: e is the unique edge of the tree in $\delta(X_e)$.
  Proof of Claim:
  - Assume $e' = xy \in E(T) \cap \delta(X_e)$ and $e' \neq e$
  - W.l.o.g $x \in X_e$ and $y \notin X_e$

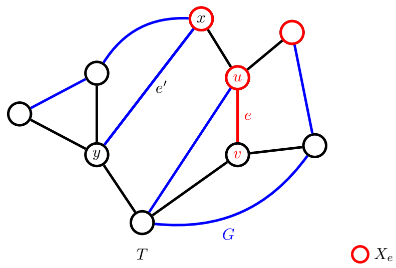

Fig. 2.11: e is the only tree edge (No. 660)

- $\Rightarrow x$ is reachable in $T - e$ from vertex u by definition of X_e
- $\Rightarrow y \in X_e$, contradiction □C2

□

- With these two concepts we can define two optimality criteria

Theorem 29 (Optimality criteria for MST)**.** *Let $G = (V, E)$ be a graph, $c : E \to \mathbb{R}$ an edge cost function and T a spanning tree of G. Then the following is equivalent:*

1. *The tree T is a minimum spanning tree.*
2. *For every non-tree edge e, it is true that e is one of the most expensive edges in the fundamental cycle w.r.t. T which is created by e. (Cycle-criterion)*
3. *For every tree edge e, it is true that e is one of the cheapest edges in the fundamental cut w.r.t. T which is created by e. (Cut-criterion)*

Proof. Ring closure and exchange arguments

- (1)⇒(2): T is a minimum spanning tree.
- Prove: e is the most expensive edge in C_e
  - See Cycle Criterion Lemma 26
- (2)⇒(3): Every non-tree edge is one of the most expensive edge in its fundamental cycle.
- Prove: Every tree edge is one of the cheapest edges in its fundamental cut
  - Let $e = xy$ be a tree edge with fundamental cut X_e, $x \in X_e$
  - Assume there exists an edge $e' = uv \in \delta(X_e)$ with $c(e') < c(e)$
  - By fundamental cut lemma, e' is a non-tree edge in $\delta(X_e)$
  - $\Rightarrow e'$ closes a fundamental cycle $C_{e'}$ w.r.t. T

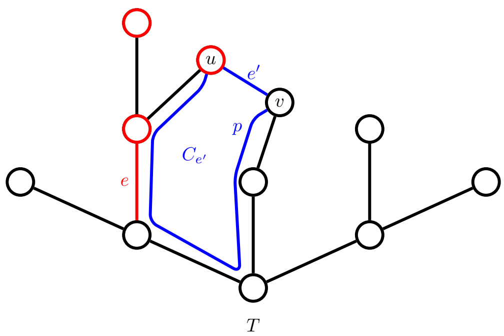

Fig. 2.12: tree edge e is the cheapest in $\delta(X_e)$ (No. 661)

- *Claim*: $e \in C_{e'}$.
  Proof of Claim:
  - $e' = uv \in \delta(X_e)$
  - Let w.l.o.g. $u \in X_e$
  - $\Rightarrow$ (u, v)-path p in T contains an edge in $\delta(X_e)$
  - Since e is the unique tree edge in $\delta(X_e)$, e is part of the (u, v)-path in T
  - $\Rightarrow e \in C_{e'}$ □C
- $\Rightarrow c(e') \geq c(e)$, contradiction to choice of e'

- (3) $\Rightarrow$ (1): Every tree edge is one of the cheapest edges in its fundamental cut.

- Prove: T is a minimum spanning tree.
  - Let T^* be an MST such that $|E(T^*) \cap E(T)|$ is maximum
  - Assume $T^* \neq T$
  - $\Rightarrow \exists\ e = xy \in T$ and $e \notin T^*$
  - $\Rightarrow$ e creates a fundamental cut $\delta(X_e)$ in T and a fundamental cycle C_e in T^*

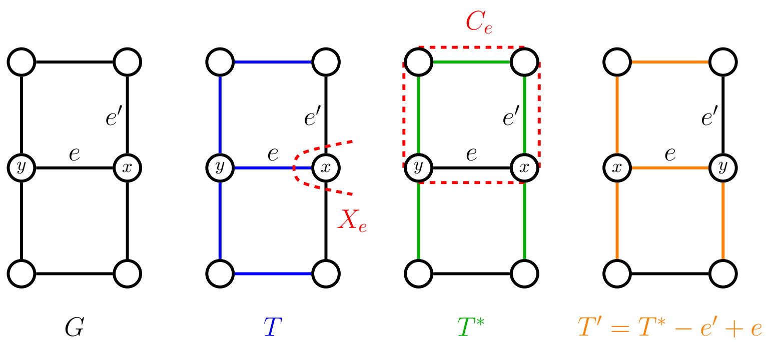

Fig. 2.13: G, T, T^* and T' (No. 542)

- *Claim*: $\exists\ e' \in C_e \setminus \{e\}$ with $c(e') = c(e)$.
  Proof of Claim:
  - $C_e \backslash \{e\}$ connects x and y
  - $\Rightarrow \exists e' \in C_e \backslash \{e\}$ which is part of $\delta(X_e)$
  - e' is a tree edge of T^* and a non-tree edge of T
  - e is a non-tree edge of T^* and $e' \in C_e \Rightarrow c(e) \geq c(e')$ since T^* is optimal
  - e is tree edge of T and $e' \in \delta(X_e) \Rightarrow c(e) \leq c(e')$ by assumption
  - $\Rightarrow c(e) = c(e')$ □C
- $T' = T^* - e' + e$ is an MST since $c(T^*) = c(T')$
- and $|E(T') \cap E(T)| = 1 + |E(T^*) \cap E(T)|$, contradiction to the choice of T^*

□

- Using these criteria we can test efficiently, whether a given tree T is an optimal minimum spanning tree (or not)

2.3 Kruskal's and Prim's Algorithms

- The most classical algorithms to solve the minimum spanning tree problem are those by Kruskal and Prim
- In both cases, the idea is to start with an empty graph and extend the graph until we obtain a spanning tree
- Such methods are called a greedy algorithm
- After each extension, either the cycle-criterion (Kruskal) or the cut-criterion (Prim) remains satisfied
- Thus, we end up with a minimum spanning tree

Kruskal's Algorithm

- Kruskal, an American mathematician, published his algorithm in 1956

Algo. 2.1 Kruskal's algorithm

Input: Undirected, connected graph $G = (V, E)$ and edge cost $c : E \to \mathbb{R}$

Output: Spanning tree T

Method:

Step 1 Set $T = \emptyset$ and sort all edges by their cost in non-decreasing order, i.e. $c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$
Set $i = 1$

Step 2 **While** $|T| \neq n - 1$ **do**
If e_i does not close a cycle in T **do**
- Add e_i to T
- Set $i = i + 1$

Return T

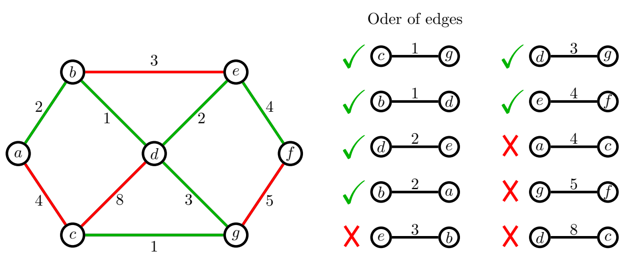

Fig. 2.14: Kruskal's Algorithm (No. 662)

Theorem 30. *Kruskal's algorithm computes a minimum spanning tree.*

Proof. Cycle-criterion

- Let T be the spanning tree computed by Kruskal's algorithm

- Let e be a non-tree edge of T

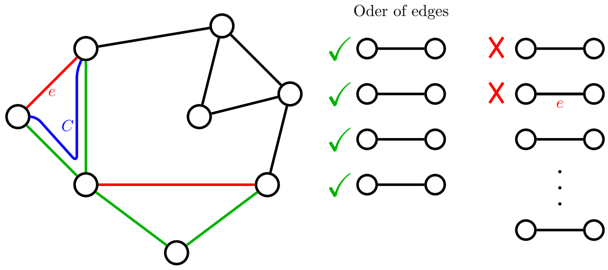

Fig. 2.15: Kruskal's Algorithm constructs MST (No. 663)

- Since e is not part of T, it closes a cycle C when it was considered in the algorithm
- Since the fundamental cycle of e w.r.t. T is unique, C is the fundamental cycle C_e
- All other edges in C_e were added before e
- $\Rightarrow c(e) \geq c(e') \; \forall e' \in C_e$
- $\Rightarrow$ cycle-criterion is fulfilled $\Rightarrow T$ is an MST

□

- The run-time of the algorithm heavily depends on the implementation and the data structure
- In the $\mathcal{O}$-notation, one estimates the number of steps needed (more Details: see Section on Complexity)

Theorem 31. *Kruskal's algorithm can be implemented with a run-time of* $\mathcal{O}(m \cdot \log m + n^2)$.

Proof. Data structure to label nodes of the same connected component

- The run-time depends on how fast, we can
  1. Sort all edges
  2. Test if a cycle is closed
  3. Add an edge to the tree
- Sorting of m elements can be done in $\mathcal{O}(m \cdot \log m)$
- 2. and 3. can be done nicely in the following way:

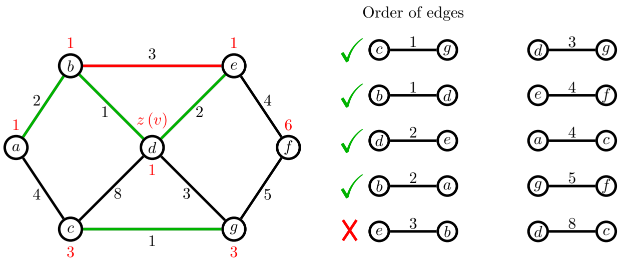

Fig. 2.16: Labeling with Kruskal (No. 664)

- label each vertex with $z(v)$
- The label $z(v)$ corresponds to a connected component, i.e., all vertices v with $z(v) = x$ are at that step in the algorithm in the same connected component of the current tree
- Initialization: set $z(v_i) = i$ for $i = 1, \dots, n$ (some numbering of the vertices)
- Let $e = uv$ some edge we consider
- If $z(u) = z(v) \Rightarrow u$ and v are already connected $\Rightarrow e$ closes a cycle
- If $z(u) \neq z(v) \Rightarrow u$ and v are in different components $\Rightarrow$ we need to add e to our tree and update z
- Update z according to $e = uv$: Set $z(w) = z(u)$ for all $w \in V$ with $z(w) = z(v)$

- In total: for each update, we need $\mathcal{O}(n)$ time
- Updating is done $m = (n-1)$-times
- Total run-time is: $\mathcal{O}(m \log n + n^2)$

$\square$

- For an efficient implementation see Corman, Leiserson, Rivest, Stein 2002
  - $O(m \log n)$ if sorting is needed
  - $O(m \cdot \alpha(m, n))$ if sorting is given, whereby $\alpha(m, n)$ is the inverse of the Ackerman function, i.e., a quasi constant

Prim's Algorithm

- A different algorithm is based on the cut-criterion
- The algorithm was developed in 1930 by the Czech mathematician Vojtěch Jarník
- Later it was rediscovered and republished by the computer scientists Robert C. Prim in 1957 and Edsger W. Dijkstra in 1959.
- Therefore, it is also sometimes called the Jarník's algorithm, Prim-Jarník algorithm, Prim-Dijkstra algorithm or the DJP algorithm

Algo. 2.2 Prim's algorithm

Input: Undirected, connected graph $G = (V, E)$ and edge cost $c : E \to \mathbb{R}$

Output: Minimum spanning tree T

Method:

Step 1 Choose an arbitrary $v \in V(G)$ and set $T = (\{v\}, \emptyset)$

Step 2 **While** $V(T) \neq V(G)$ **do**
- Choose $e \in \delta(V(T))$ with minimum cost $c(e)$
- Set $T = T + e$

Return T

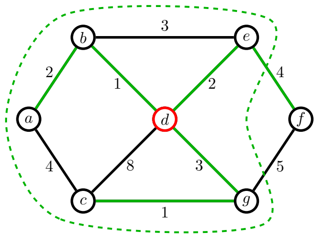

Fig. 2.17: Prim's Algorithm (No. 665)

Theorem 32. *Prim's algorithm computes a minimum spanning tree.*

Proof. Exercise ☐

- The run-time depends again on a "good" data structure i.e., that we can chose $e \in \delta(V(T))$ with minimum cost efficiently

Theorem 33. *Prim's algorithm can be implemented in $\mathcal{O}(n^2)$.*

Proof. Exercise ☐

- There are several extensions of the above algorithms to speed them up