

The Minimum Spanning Tree Problem

Kruskal and Prim's algorithm

Jesper Larsen, David Pisinger

jesla, pisinger@man.dtu.dk

DTU Management

Technical University of Denmark

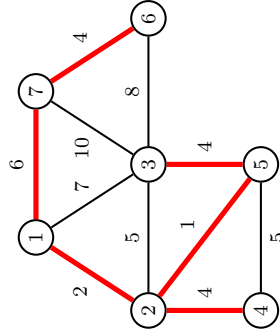
The Minimum Spanning Tree Problem – p.1/23

Prerequisites

Let $G = (V, E, w)$ be a weighted graph, with $w \geq 0$

- A **tree** H is a connected subgraph with no cycles (circuits) of G .
- Spanning: **all vertices** are in the H
- Cost of tree H is $c(H) = \sum_{e \in H} w_e$

The minimum (weight) spanning tree problem consists of finding a spanning tree of minimum cost.



The Minimum Spanning Tree Problem – p.2/23

Cut I

Consider a given graph, $G = (V, E)$.

- A **cut** $C = \{X, X'\}$ is a partition of the vertices into two sets X and X' , where $X \cup X' = V$ and $X \cap X' = \emptyset$.
- Edges having one endpoint in X and the other in $V \setminus X$ are said to **cross the cut**. We define:

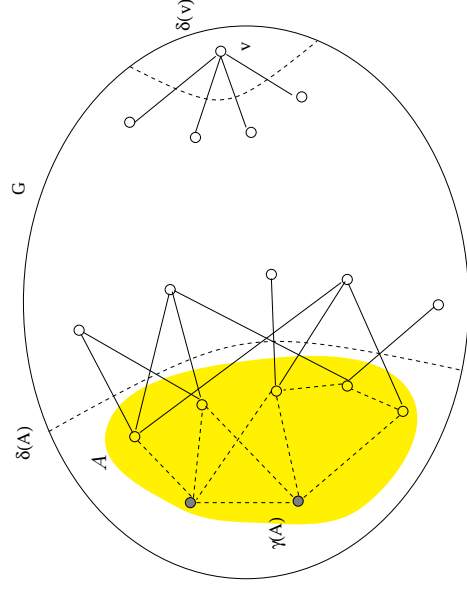
$$\delta(X) = \{(v, w) \in E : v \in X \wedge w \in V \setminus X\}$$

- For $v \in V$, $\delta(\{v\}) = \{(v, w) \in E : w \in V \setminus \{v\}\}$ i.e. $\delta(\{v\})$ consists of all edges with v as one of the vertices.
- $\gamma(X)$ is the set of internal edges in X :

$$\gamma(X) = \{(v, w) \in E : v, w \in X\}$$

The Minimum Spanning Tree Problem – p.3/23

Cut Figure

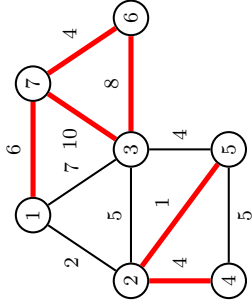


The Minimum Spanning Tree Problem – p.4/23

Cut-set definition of MST

Let $x_e = 1$ if and only if e is in MST

$$\begin{aligned} \min \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & \sum_{e \in E} x_e = n - 1 \\ & \sum_{e \in E} x_e \geq 1 \quad \emptyset \subset S \subset V \\ & x_e \in \{0, 1\} \end{aligned}$$

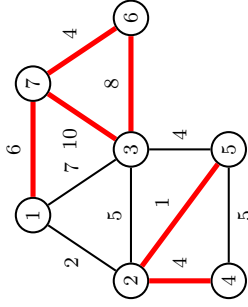


The Minimum Spanning Tree Problem – p.5/23

Subtour elimination definition of MST

Let $x_e = 1$ if and only if e is in MST

$$\begin{aligned} \min \quad & \sum_{e \in E} w_e x_e \\ \text{s.t.} \quad & \sum_{e \in E} x_e = n - 1 \\ & \sum_{e \in \gamma(S)} x_e \leq |S| - 1, \quad \emptyset \subset S \subset V \\ & x_e \in \{0, 1\} \end{aligned}$$

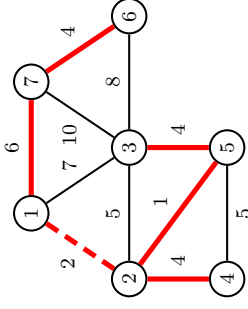


The Minimum Spanning Tree Problem – p.6/23

Relationship cut vs. MST I

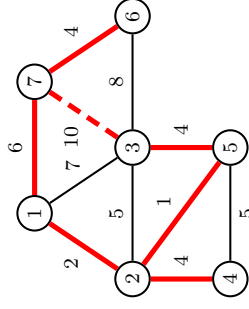
Let T^* be a MST, and let $e \in T^*$. If we remove e from T^* the tree will be split into two components. Let us call them X and $X' = V \setminus X$.

- 1) $\{X, X'\}$ defines a cut.
- 2) e is one of the arcs with minimal weight crossing the cut defined by X .



The Minimum Spanning Tree Problem – p.7/23

Relationship cut vs. MST II



- If we insert an edge (i, j) into a spanning tree we create a unique cycle.

The Minimum Spanning Tree Problem – p.8/23

Safe edge

- A strategy for solving the MST is to maintain a set A that is a subset of some MST.
- In each iteration of our algorithm we add one edge e to A maintaining that the “new” A , that is, $A \cup \{e\}$, also fulfills the property.
- An edge with this property is called a **safe edge**.

The Minimum Spanning Tree Problem – p.9/23

Light edge

- A **light edge** is defined as an edge with minimum weight of any edge crossing a cut.
- Note that there can be more than one light edge crossing a cut.
- We have: Given a cut C respecting a set A . If (i, j) is a light edge crossing the cut C then it is also a safe edge.

The Minimum Spanning Tree Problem – p.11/23

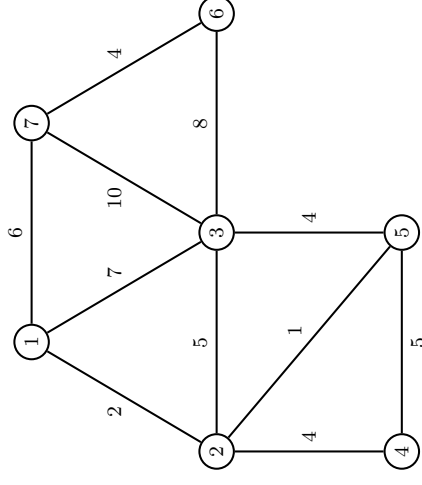
The Selection rule for MST

- 1 select a cut in G that does not contain any selected edges;
- 2 determine a safe edge of the cut-set and select it;
- 3 if there are more than one safe edge break tie arbitrarily;

- The “generic” algorithm is often denoted the selection rule, and is applied $n - 1$ times to produce a MST.
- So how do we find “safe edges”?

The Minimum Spanning Tree Problem – p.10/23

Example



The Minimum Spanning Tree Problem – p.12/23

Proof of correctness I

- Need to prove that the method is full-filling its purpose, so we need to show, that the method based on the selection rule only finds MSTs.
- When we cannot use the selection rule anymore we know that the solution T , we have build is connected. So we need to show that T is the spanning tree with the minimum weight.

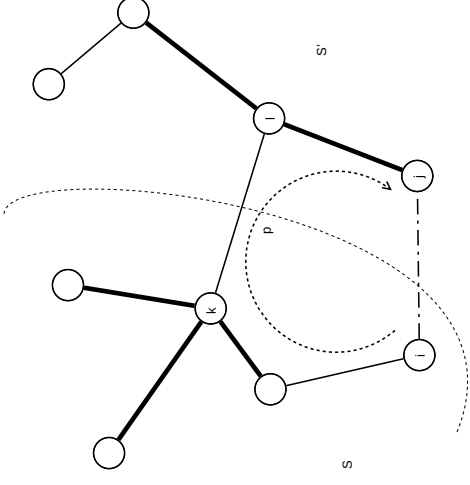
The Minimum Spanning Tree Problem – p.13/23

Proof of correctness II

- Let T be a MST that includes A . Let $\mathcal{C} = \{S, \bar{S}\}$ be any cut that respects A , and let (i, j) be a light edge.
- If T contains the light edge (i, j) we are done, so let us assume this is **not** the case.
- The general idea now is to construct a new MST T' that includes $A \cup \{(i, j)\}$. This will then prove that (i, j) is a safe edge.

The Minimum Spanning Tree Problem – p.14/23

Proof of correctness III



- $T \cup \{(i, j)\}$ contains a (unique) cycle. Let us call this cycle p .

The Minimum Spanning Tree Problem – p.15/23

Proof of correctness IV

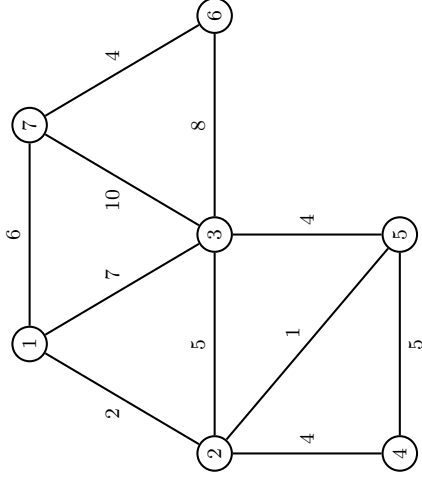
- i and j are on opposite sides of the cut $\{S, \bar{S}\}$, there is at least one edge in T on the path p that also crosses the cut. Call it (k, l) .
- As the cut respects A $(k, l) \notin A$.
- Now produce $T' = T \setminus \{(k, l)\} \cup \{(i, j)\}$.
- Since (i, j) is a light edge and (k, l) also crosses the cut $w_{ij} \leq w_{kl}$, and therefore $w_{T'} \leq w_T$.

The Minimum Spanning Tree Problem – p.16/23

Proof of correctness V

- As T is a minimum spanning tree we have $w_T \leq w_{T'}$. So T' must also be a minimum spanning tree.
- But is it safe for A ? As $A \subseteq T$ and $(k, l) \notin A$ then $A \subseteq T'$, so $A \cup \{(i, j)\} \subseteq T'$. Consequently, since T' is a minimum spanning tree, (i, j) is safe for A .

Example



Kruskal's Algorithm

Starting from a spanning forest (many trees) repeatedly add edges of of minimum weight until forest becomes a tree

- F is set of edges in forest
- L is list of sorted edges
- 1 $F \leftarrow \emptyset$;
- 2 sort edges by nondecreasing cost;
- 3 **for** each $(u, v) \in L$ **do**
- 4 **if** a path from u to v in F does not exist **then**
- 5 $F \leftarrow F \cup (u, v)$;

Time complexity of Kruskal's algorithm

- Time complexity depends on the sorting algorithm and the time it takes to evaluate whether two vertices are in the same set or not (this can be done in almost constant time using disjoint set data structure).
- Ultimately the time complexity of the sorting is the deciding factor. Bubble sort requires $O(m^2)$, while an algorithm like Quicksort with median selection takes $O(m \log m)$.
 - m edges in data structure
 - sort m edges according in time $O(m \log m)$
 - m times consider next edge (u, v)
 - m times check if u and v belong to same set in $O(1)$
 - m times merge-sets in time $O(1)$ if u and v are not in same set

Prim's Algorithm

Start from node 1 and gradually grow tree until spanning tree

- S is set of finished nodes
- ℓ_v is cheapest edge from any $u \in S$ to $v \in V \setminus S$

```

1  $Q \leftarrow V, S \leftarrow \emptyset;$ 
2  $p_u \leftarrow \text{NIL} \quad \forall u;$ 
3  $\ell_1 \leftarrow 0, \ell_j \leftarrow +\infty, \forall j \neq 1;$ 
4 while  $S \neq V$  do
5    $u \leftarrow \arg \min_{Q \setminus S} (\ell);$ 
6   add  $u$  to  $S;$ 
7   for each  $v$  adjacent to  $u$  do
8     if  $v \in Q \setminus S$  and  $w_{uv} < \ell_v$  then
9        $p_v \leftarrow u;$ 
10       $\ell_v \leftarrow w_{uv};$ 
```

Time Complexity of Prim's Algorithm

- Note the similarity with Dijkstra's algorithm for Shortest Path (as we get to next lecture).
- Time complexity depends on the exact data structures. But the basic implementations with list representation gives $O(n^2)$ and with a heap or balanced binary tree data structure we get $O(m \log n)$.
- for each node v in $V \setminus S$ maintain label ℓ_v with cheapest edge weight from any $u \in S$ to v (in total n numbers)
- n times find minimum in time $O(\log n)$
- m times update label by deleting/inserting in time $O(\log n)$

Example

