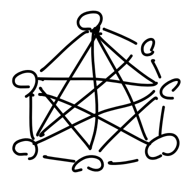
Applications : – networks (computers, sensors, electrical, ...)
– machine learning (clustering)
– computer vision (object detection)
– data mining
– subroutine in other (approximation) alg.

How difficult is it?

How many spanning trees can a graph have?



complete graph: has all the $\binom{n}{2}$ possible edges

a complete graph has n^{n-2} different spanning trees
exponential

However, MST can be solved in near-linear time!
Not only: greedy algorithms $\Rightarrow$ simple to implement in practice

Prim Kruskal

They both apply (in $\neq$ ways) a generic greedy algorithm

Invariant maintained:

- at each iteration, A is a subset of edges of some MST

At each iteration the algorithm adds an edge that does not violate the invariant

"safe" edge for A

```
GENERIC-MST(G)
  A = ∅
  while A does not form a spanning tree
      find an edge (u,v) that is safe for A   \\ crucial step
      A = A ∪ {(u,v)}
  return A        \\ A is an MST
```

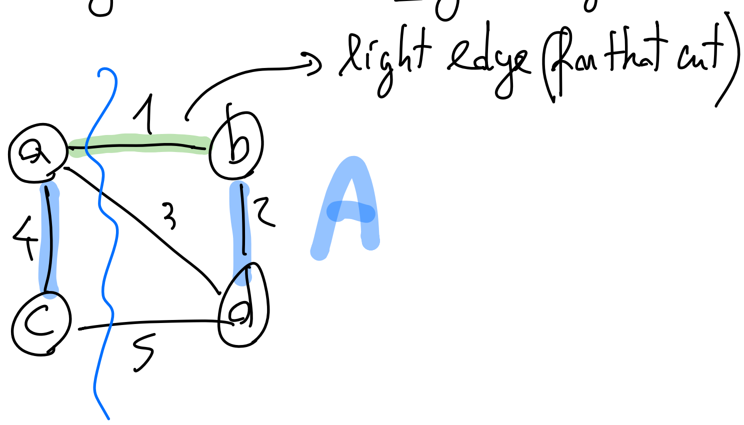
How to find a safe edge? Luckily, MSTs enjoy the following structural property. First, some definitions:

- A cut of a graph $G=(V,E)$ is a partition of V
  $\hookrightarrow (S, V\setminus S)$
- An edge $(u,v) \in E$ crosses a cut $(S, V\setminus S)$ if

$u \in S$ and $v \in V \setminus S$ (or vice versa)

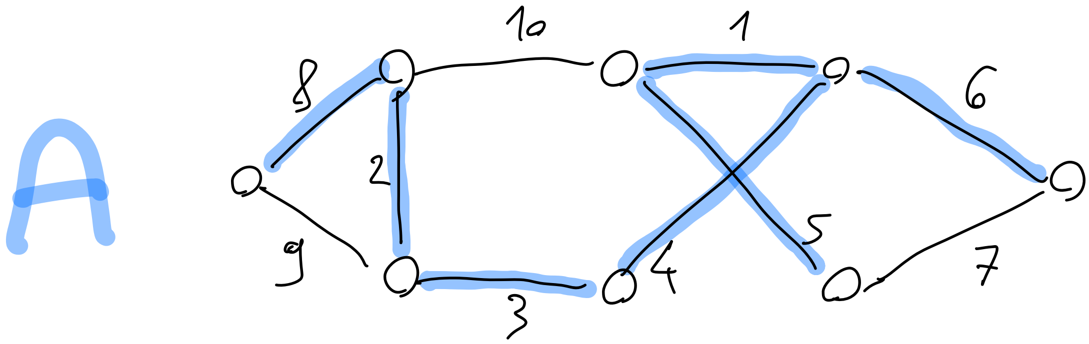
- A cut <u>respects</u> a set of edges A if no edge of A crosses the cut
- Given a cut, an edge that crosses the cut and is of minimum weight is called <u>light edge</u>

example:



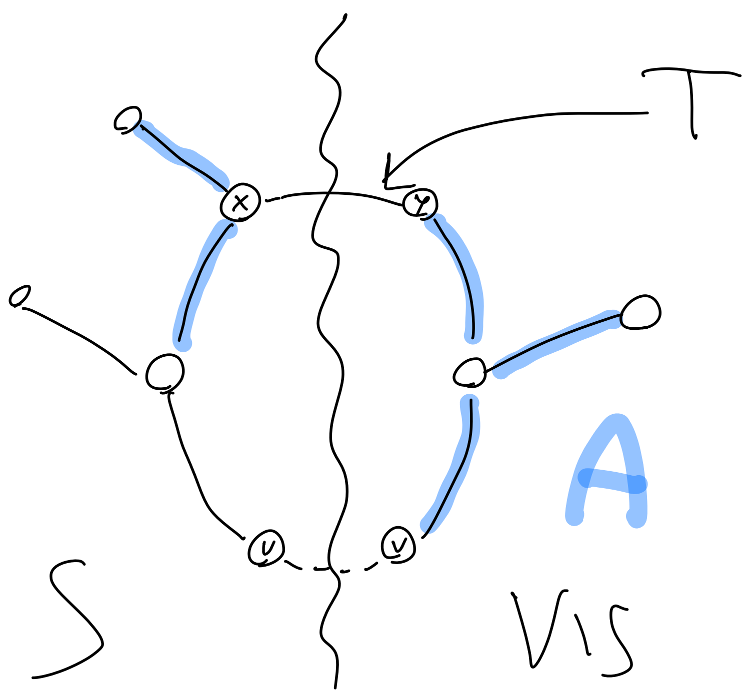
Theorem: Let $G=(V,E)$ be an undirected, connected and weighted graph. Let A be a subset of E included in some MST of G, let $(S, V \setminus S)$ be a cut that respects A, and let (u,v) be a light edge for $(S, V \setminus S)$. Then (u,v) is safe for A.

Example of GENERIC-MST:



Proof of Theorem: technique: "cut & paste" (standard in greedy algorithms)

Let T be an MST that includes A. Assume that $(u,v) \notin T$ (otherwise, we'd be done).
We'll build a new MST T' that includes $A \cup \{(u,v)\}$.



By hypothesis (u,v) crosses $(S, V\setminus S)$ $\Rightarrow \exists$ another edge of T that crosses that cut (x,y)

By hypothesis $(S, V\setminus S)$ respects A $\Rightarrow (x,y) \notin A$

$\Rightarrow$ removing (x,y) from T and adding (u,v) we obtain a new spanning tree $T' = T \setminus \{(x,y)\} \cup \{(u,v)\}$ that includes $A \cup \{(u,v)\}$.

Now we need to show that T' not only is a ST, but also a MST. (x,y) and (u,v) both cross $(S, V\setminus S)$ but by hypothesis (u,v) is light $\Rightarrow w(u,v) \leq w(x,y)$

$$\Rightarrow w(T') = w(T) \underbrace{- w(x,y) + w(u,v)}$$

$$\leq w(T)$$

but T is a MST $\Rightarrow w(T') = w(T)$.

We'll now see two MST algorithms that organize the choice of those "respectful" cuts.

Prim's algorithm (1957)

How does Prim's alg. apply GENERIC-MST(G):

- A : a single tree
- safe edge : a light edge that connects the tree with a vertex that does not belong to the tree

$\text{Prim}(G, s)$ \\ s = source vertex

$X = \{s\}$

$A = \emptyset$

while there is an edge (u, v) with $u \in X$ and $v \notin X$ do

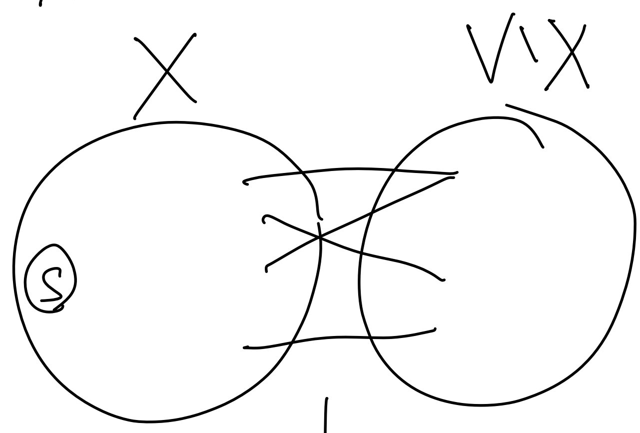
(u^*, v^*) = a minimum-weight such edge \\ light edge $\rightarrow O(m)$

add vertex v^* to X

add edge (u^*, v^*) to A

return A

$n-1$



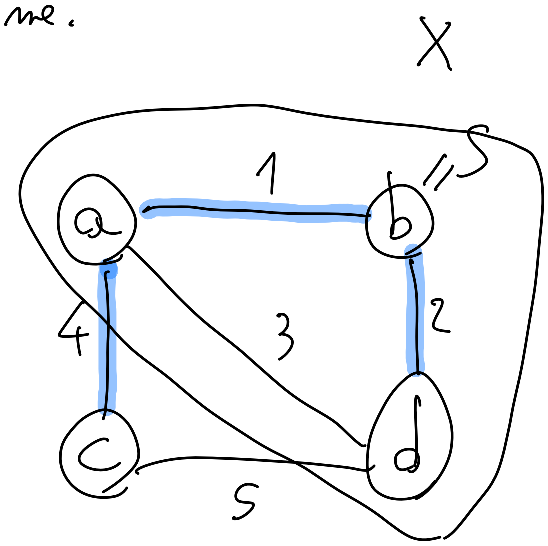
processed vertices

not yet processed

Candidates for (u^*, v^*)

This algorithm "grows" a spanning tree from a source vertex s (doesn't matter who s is) by adding one edge at a time.

Example:

$G =$ 

Correctness: follows from the Theorem

Complexity: (assume that G is represented with an adjacency list)

$$O(m \cdot n)$$

polynomial time $\Rightarrow$ efficient