Minimum Spanning Trees

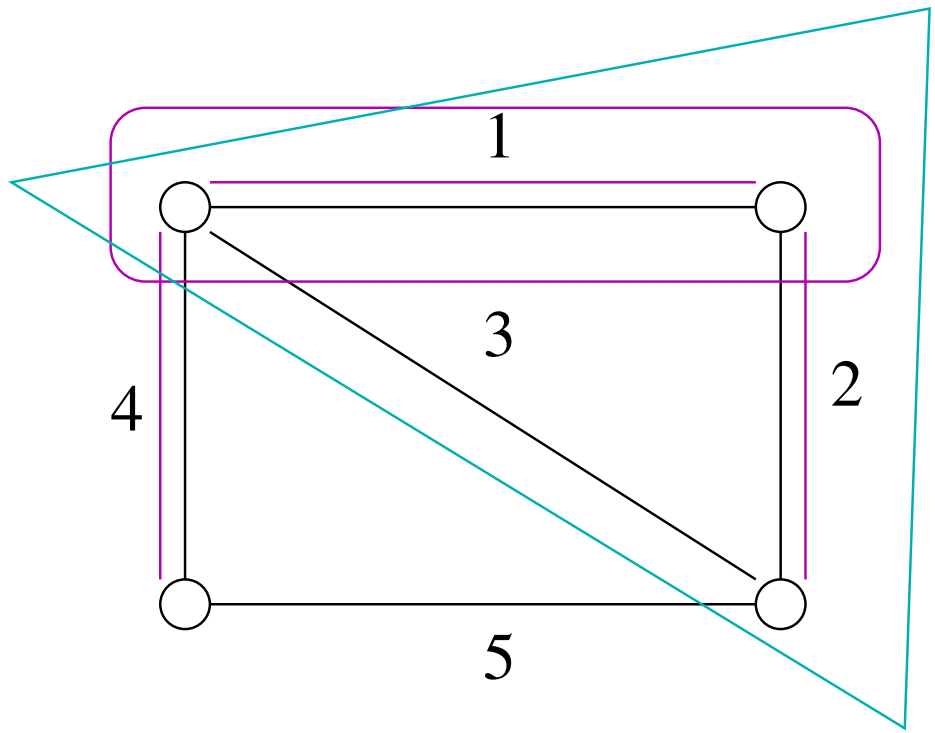
Algorithms: Design and Analysis, Part II

Prim's MST Algorithm

Example

[Purple edges = minimum spanning tree]

(Compare to Dijkstra's shortest-path algorithm)



Prim's MST Algorithm

- Initialize $X = \{s\}$ [$s \in V$ chosen arbitrarily]
- $T = \emptyset$ [invariant: $X =$ vertices spanned by tree-so-far T]
- While $X \neq V$
  - Let $e = (u, v)$ be the cheapest edge of G with $u \in X$, $v \notin X$.
  - Add e to T
  - Add v to X.

While loop: Increase # of spanned vertices in cheapest way possible.

Correctness of Prim's Algorithm

Theorem: Prim's algorithm always computes an MST.

Part I: Computes a spanning tree T^*.
[Will use basic properties of graphs and spanning trees] (Useful also in Kruskal's MST algorithm)

Part II: T^* is an MST.
[Will use the "Cut Property"] (Useful also in Kruskal's MST algorithm)

Later: Fast $[O(m \log n)]$ implementation using heaps.