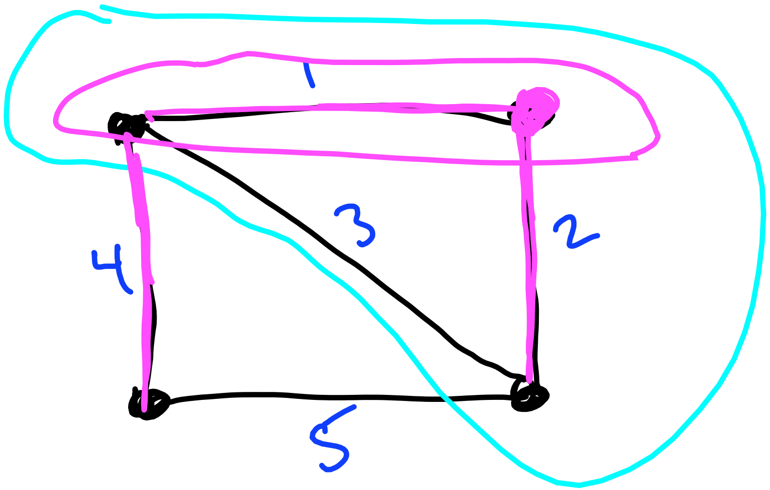
Minimum Spanning Trees

Prim's MST Algorithm

Algorithms: Design and Analysis, Part II

Example



(compare to Dijkstra's shortest-path algorithm)

[pink edges = minimum spanning tree]

Tim Roughgarden

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

i.e., 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 & spanning trees]

useful also for Kruskal's MST algorithm

Part II: T^* is an MST. [will use the "Cut Property"]

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