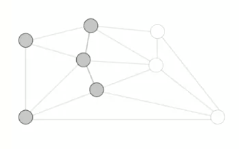
Greedy Algorithm

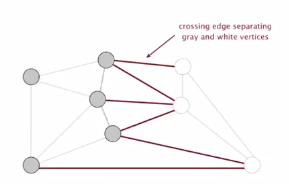
Simplifying assumptions

* Edge weights distinct
* Graph is connected

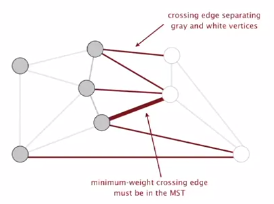
Consequence: MST exists and is unique

Cut (on a graph): a partition of its vertices into two (nonempty) sets

Crossing edge: connects a vertex in one set with a vertex in the other

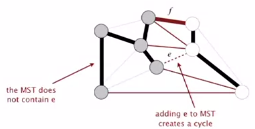


Cut property: given any cut, the crossing edge of min weight is in the MST



Proof: if MST does not contain the minimum weight crossing edge *e*

* This means one of the other crossing edges must be in the MST
* If you add *e* to the MST, you will create a cycle
* Some other edge in *f* must be a crossing edge
* Removing *f* and adding *e* is also a spanning tree
* Since weight of *e* is less than weight of *f*, that spanning tree is a lower weight…
* **Contradiction**

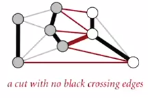


Greedy MST algorithm

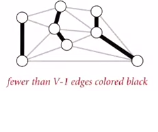
Proposition: compute the MST

Proof:

* Any edge colored black is in the MST (via cut property)



* Fewer than V-1 black edges -> cut with no black crossing edges.   
  (consider cut whose vertices are one connected component)



A number of implementations

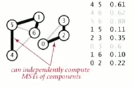
1. Kruskal’s algorithm
2. Prim’s algorithm
3. Boruvka’s algorithm (like a combination of 1 and 2)

What if edge weights are not all distinct?

C:\Users\Zach\AppData\Local\Microsoft\Windows\INetCache\Content.Word\nondistedge.png

Answer: there are multiple MSTs.   
The greedy algorithm is still correct (correctness proof fails, but that can be fixed)

What if graph is not connected?



Answer: Compute minimum spanning forest = MST of each component