Proof of Correctness
Prim’s Algorithm

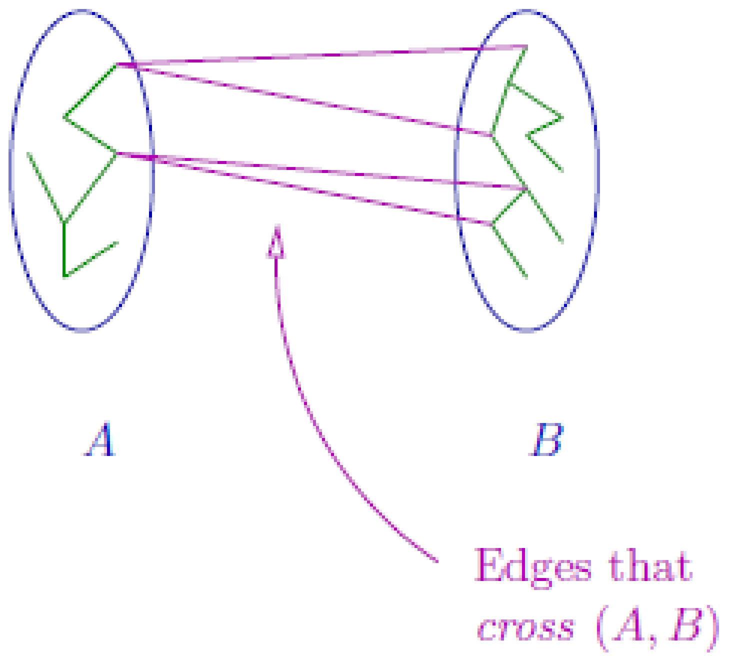
Prim's Algorithm (MinHeap with vertices)

- T = Ø [invariant: X = vertices spanned by tree-so-far T]
- for each u ∈ V
  - Key[u] = ∞
  - Pred[u] = nill
- Key[s] = 0 // select any random vertex and make its cost 0
- Heap Q is initialized with all vertices
- While heap ≠ Ø
  - u = ExtractMin from Heap
  - Add u to X
  - for each v ∈ adj[u]
    - If (v ∈ Q and cost[u,v] < key[v])
      - DecreaseKey(Q, v, cost[u,v]) // DecreaseKey will update key of vertex v only if cost[u,v] < key [v]
      - Pred[v] = u

Cuts

Claim: Prim's algorithm outputs a spanning tree.

Definition: A cut of a graph $G = (V, E)$ is a partition of V into 2 non-empty sets.



Quiz on Cuts

Question: Roughly how many cuts does a graph with n vertices have?

a) n
b) 2^n
c) n^2
d) n^n

Quiz on Cuts

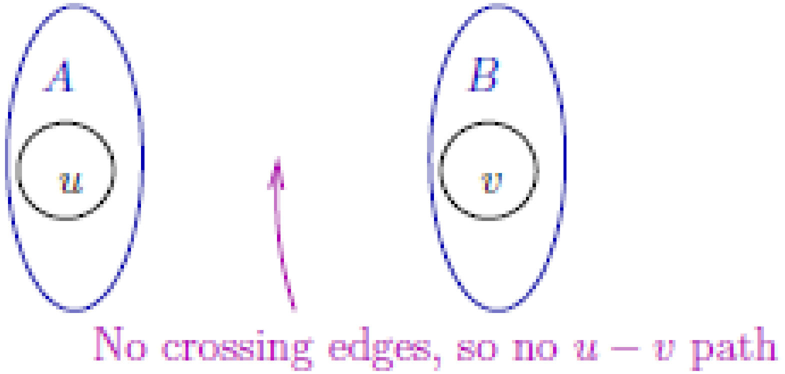
Question: Roughly how many cuts does a graph with n vertices have?

a) n
b) 2^n (for each vertex, choose whether in A or in B)
c) n^2
d) n^n

Empty Cut Lemma

Empty Cut Lemma: A graph is not connected $\iff$ $\exists$ cut (A, B) with no crossing edges.

Proof: $(\Leftarrow)$ Assume the RHS. Pick any $u \in A$ and $v \in B$. Since no edges cross (A, B) there is no u, v path in G. $\Rightarrow$ G not connected.



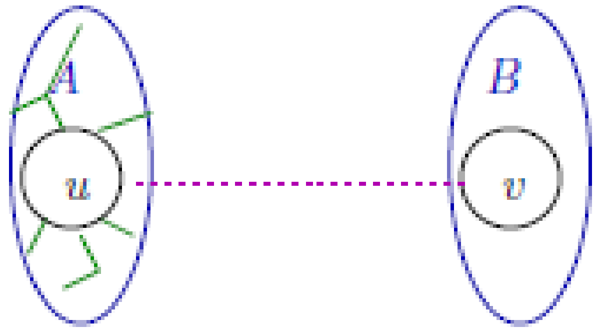
Empty Cut Lemma

($\Rightarrow$) Assume the LHS. Suppose G has no $u - v$ path. Define

$A = \{$Vertices reachable from u in $G\}$ (u's connected component)

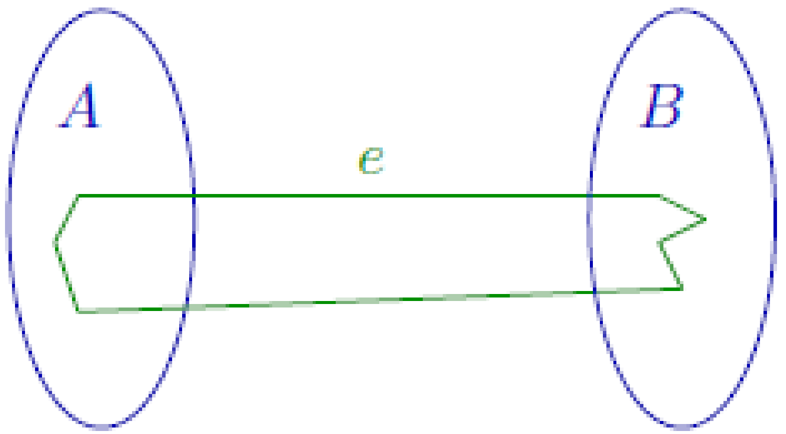
$B = \{$All other vertices$\}$ (all other connected components)

Note: No edges cross cut (A, B) (otherwise A would be bigger!)



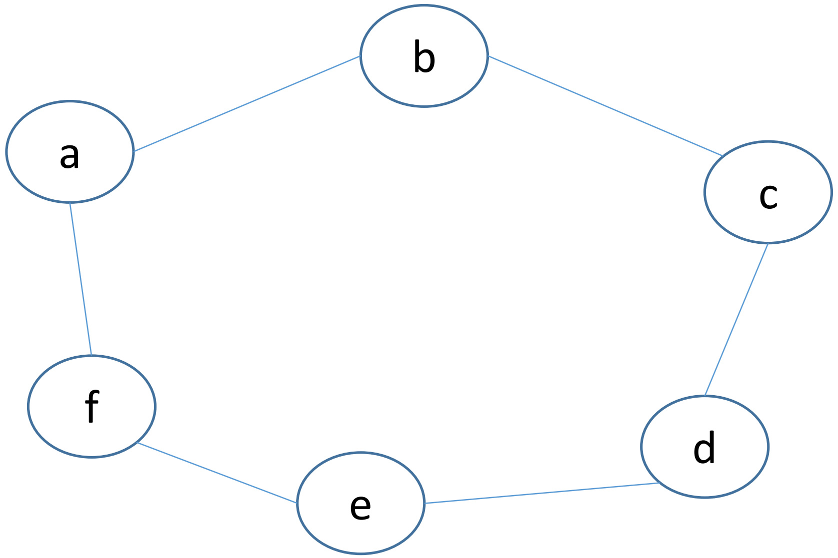
Two Easy Facts

Double-Crossing Lemma: Suppose the cycle $C \subseteq E$ has an edge crossing the cut (A, B): then so does some other edge of C.

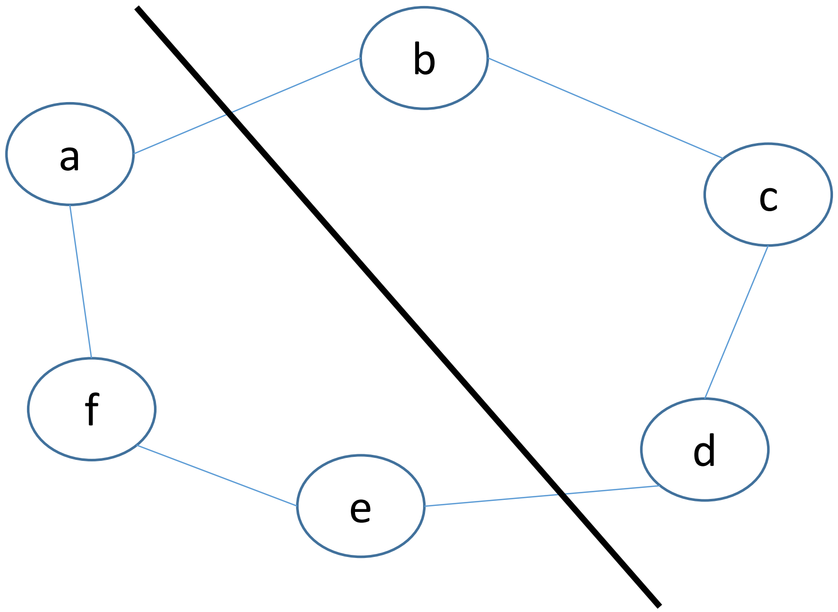


Lonely Cut Corollary: If e is the only edge crossing some cut (A, B), then it is not in any cycle. [If it were in a cycle, some other edge would have to cross the cut!]

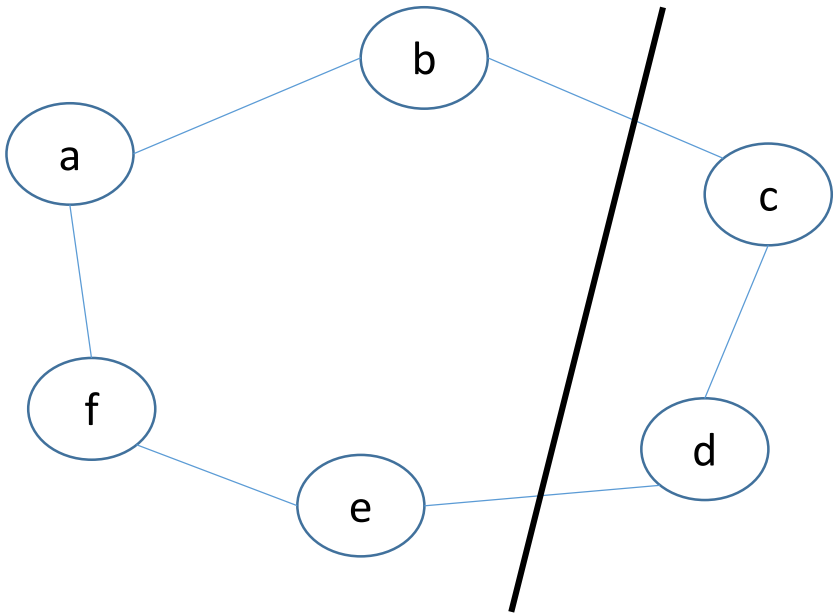
Double Crossing Lemma



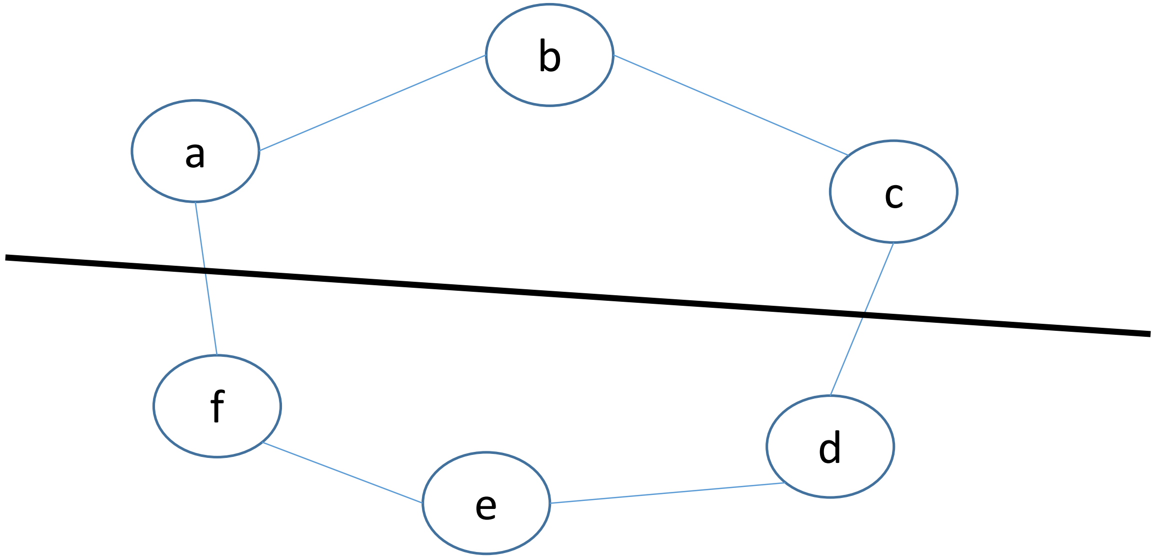
Double Crossing Lemma



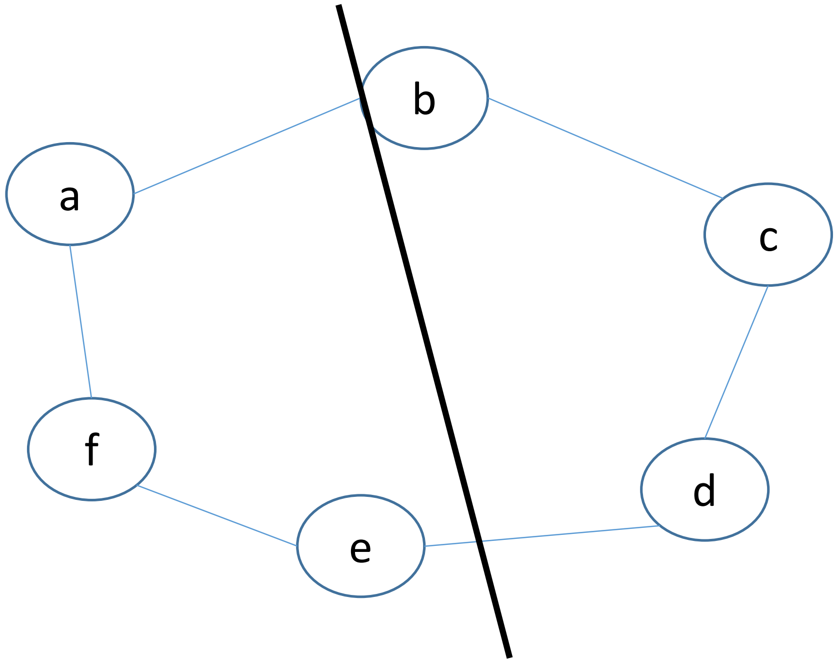
Double Crossing Lemma



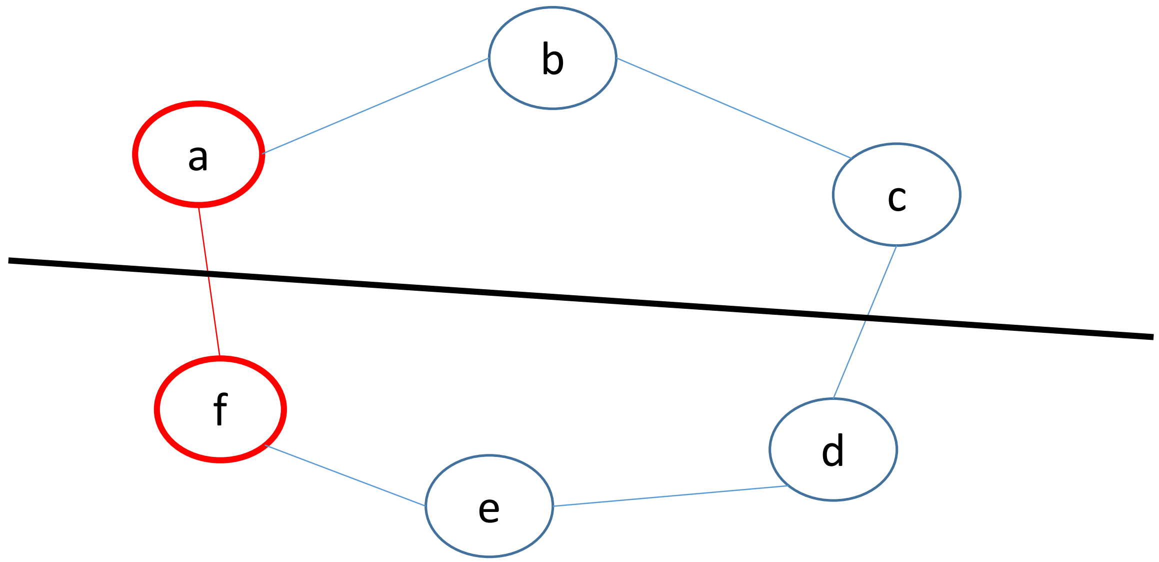
Double Crossing Lemma



Double Crossing Lemma



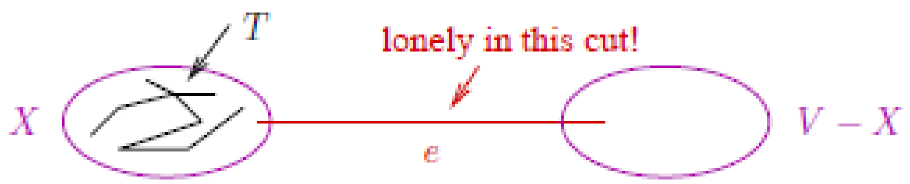
Double Crossing Lemma



Proof Part 1

Claim: Prim's algorithm outputs a spanning tree.
[Not claiming MST yet]

Proof: (1) Algorithm maintains invariant that T spans X



(2) Can't get stuck with $X \neq V$
[otherwise the cut $(X, V - X)$ must be empty; by Empty Cut Lemma input graph G is disconnected]

(3) No cycles ever get created in T. Why? Consider any iteration, with current sets X and T. Suppose e gets added.
Key point: e is the first edge crossing $(X, V - X)$ that gets added to $T \Rightarrow$ its addition can't create a cycle in T (by Lonely Cut Corollary).

Correctness of Prim’s Algorithm

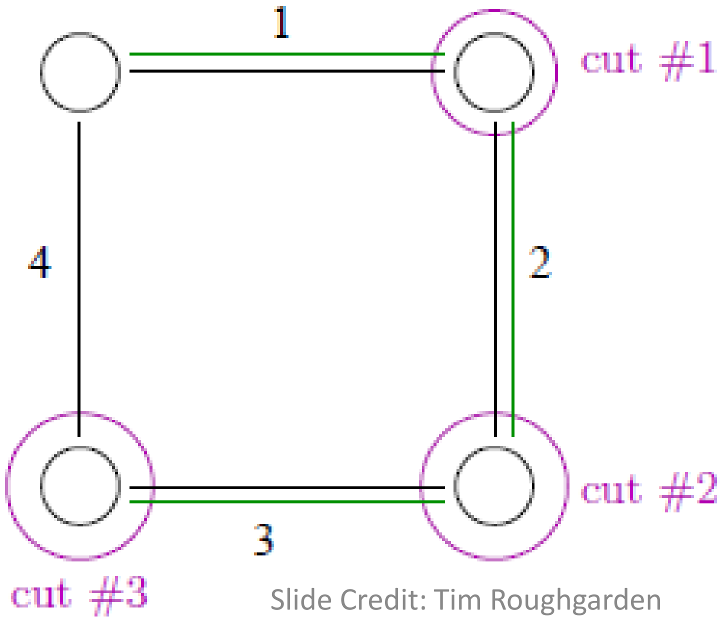
Theorem: Prim's algorithm always outputs a minimum-cost spanning tree.

Key Question: When is it “safe” to include an edge in the tree-so-far?

The Cut Property

CUT PROPERTY: Consider an edge e of G. Suppose there is a cut (A, B) such that e is the cheapest edge of G that crosses it. Then e belongs to the MST of G.

Turns out MST is unique if edge costs are distinct



Slide Credit: Tim Roughgarden

Cut Property Implies Correctness

Claim: Cut Property $\Rightarrow$ Prim's algorithm is correct.

Proof: Prim's algorithm outputs a spanning tree T^*.

Key point: Every edge $e \in T^*$ is explicitly justified by the Cut Property.

$\Rightarrow$ T^* is a subset of the MST

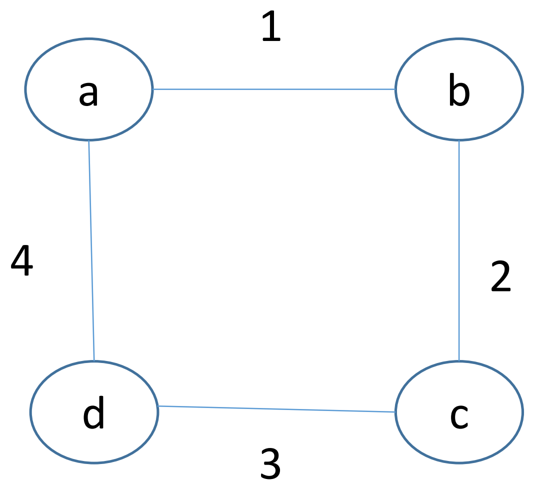
$\Rightarrow$ Since T^* is already a spanning tree, it must be the MST

The Cut Property

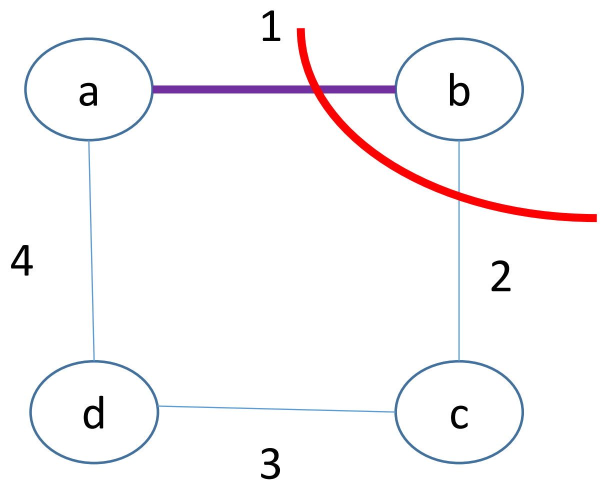
Assumption: Distinct edge costs.

CUT PROPERTY: Consider an edge e of G. Suppose there is a cut (A, B) such that e is the cheapest edge of G that crosses it. Then e belongs to the MST of G.

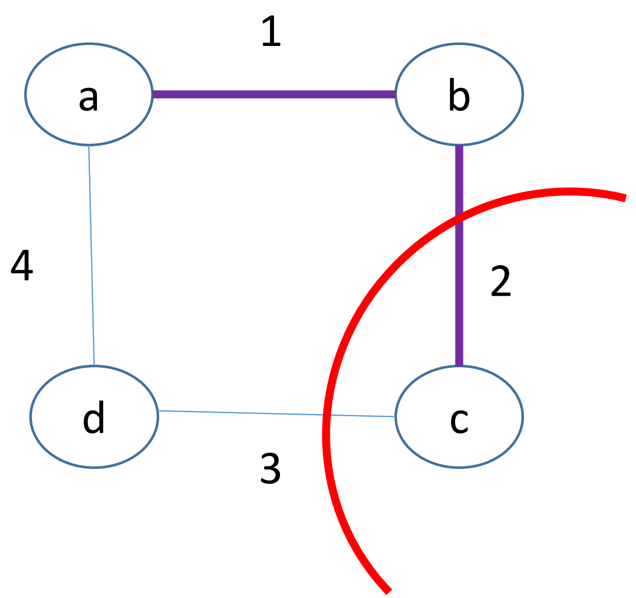
The Cut Property



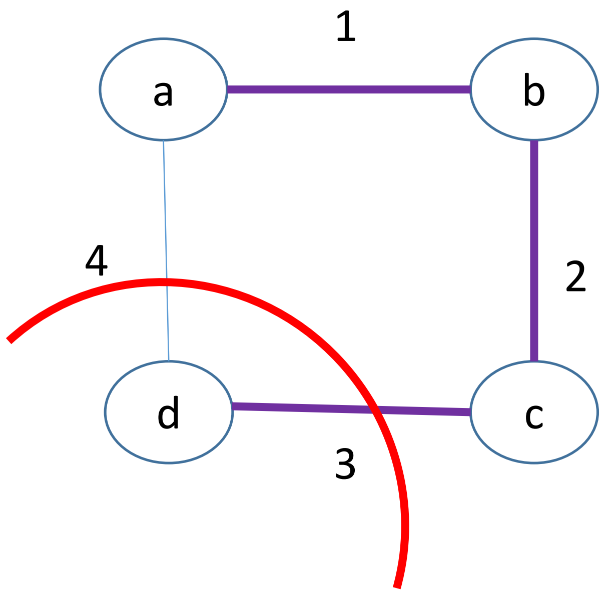
The Cut Property



The Cut Property



The Cut Property



Proof Plan

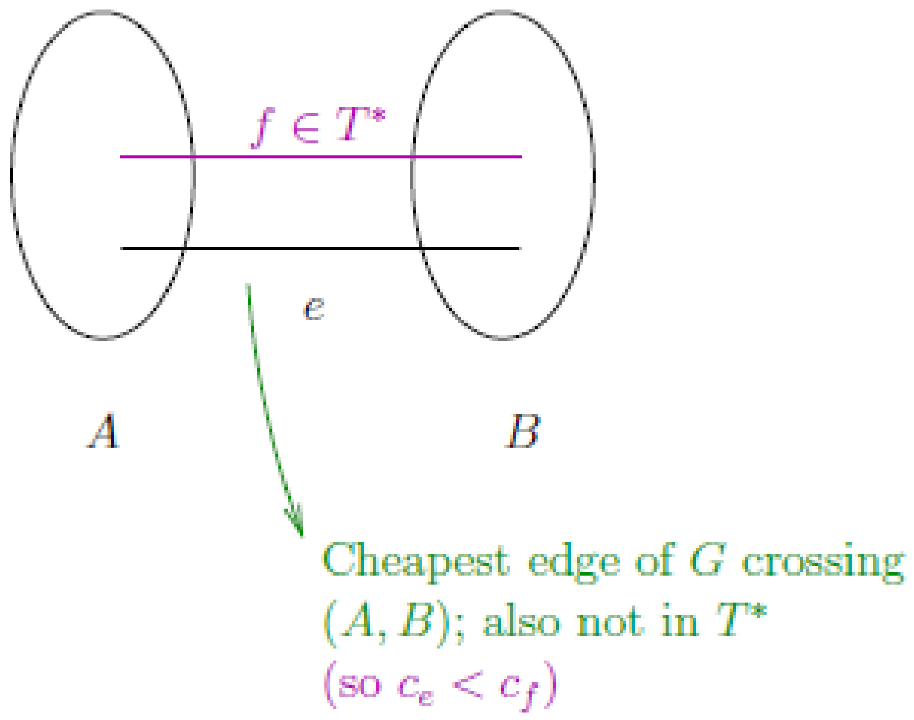
Will argue by contradiction, using an exchange argument.

Suppose there is an edge e that is the cheapest one crossing a cut (A, B), yet e is not in the MST T^*.

Idea: Exchange e with another edge in T^* to make it even cheaper (contradiction).

Question: Which edge to exchange e with?

Attempted Exchange



Note: Since T^* is connected, must construct an edge $f (\neq e)$ crossing (A, B).

Idea: Exchange e and f to get a spanning tree cheaper than T^* (contradiction).

Exchanging Edges

Question: Let T^* be a spanning tree of G, $e \notin T^*$, $f \in T^*$. Is $T^* \cup \{e\} - \{f\}$ a spanning tree of G?

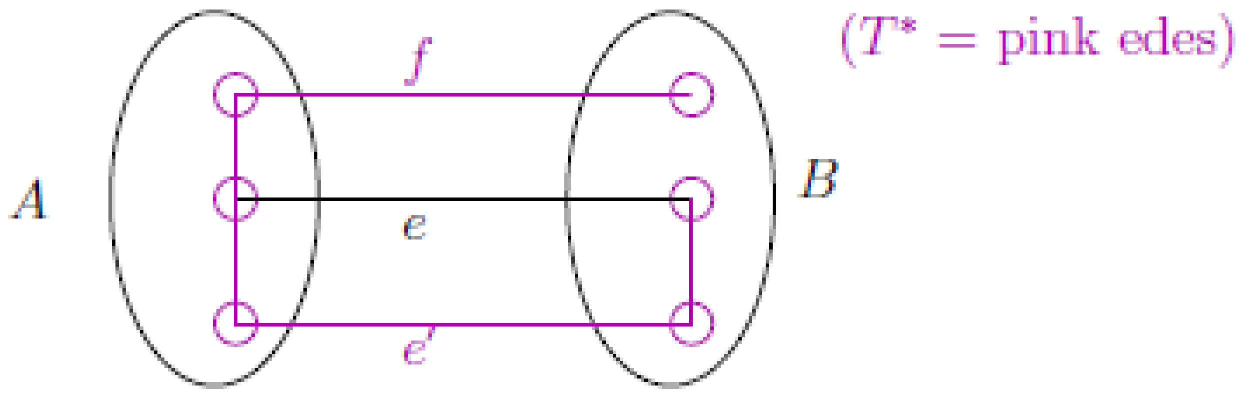
a) Yes always
b) No never
c) If e is the cheapest edge crossing some cut, then yes
d) Maybe, maybe not (depending on the choice of e and f)

Exchanging Edges

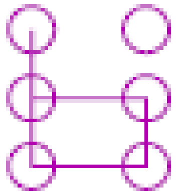
Question: Let T* be a spanning tree of G, $e \notin T^*$, $f \in T^*$. Is $T^* \cup \{e\} - \{f\}$ a spanning tree of G?

a) Yes always
b) No never
c) If e is the cheapest edge crossing some cut, then yes
d) Maybe, maybe not (depending on the choice of e and f)

Exchanging Edges

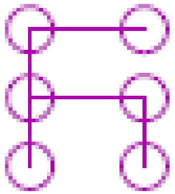


Exchange e, f:



(not a spanning tree)

Exchange e, e':

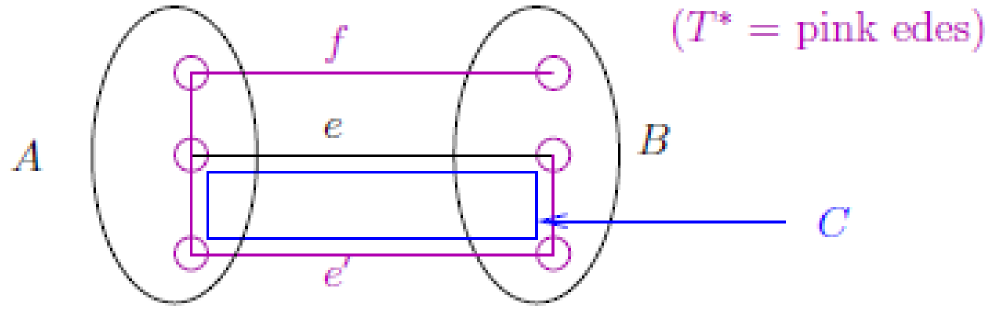


(a spanning tree)

Smart Exchanges

Hope: Can always find suitable edge e' so that exchange yields bona fide spanning tree of G.

How? Let C = cycle created by adding e to T^*.



By the Double-Crossing Lemma: Some other edge e' of C [with $e' \neq e$ and $e' \in T^*$] crosses (A, B).

You check: $T = T^* \cup \{e\} - \{e'\}$ is also a spanning tree.

Since $c_e < c_{e'}$, T cheaper than purported MST T^*, contradiction.