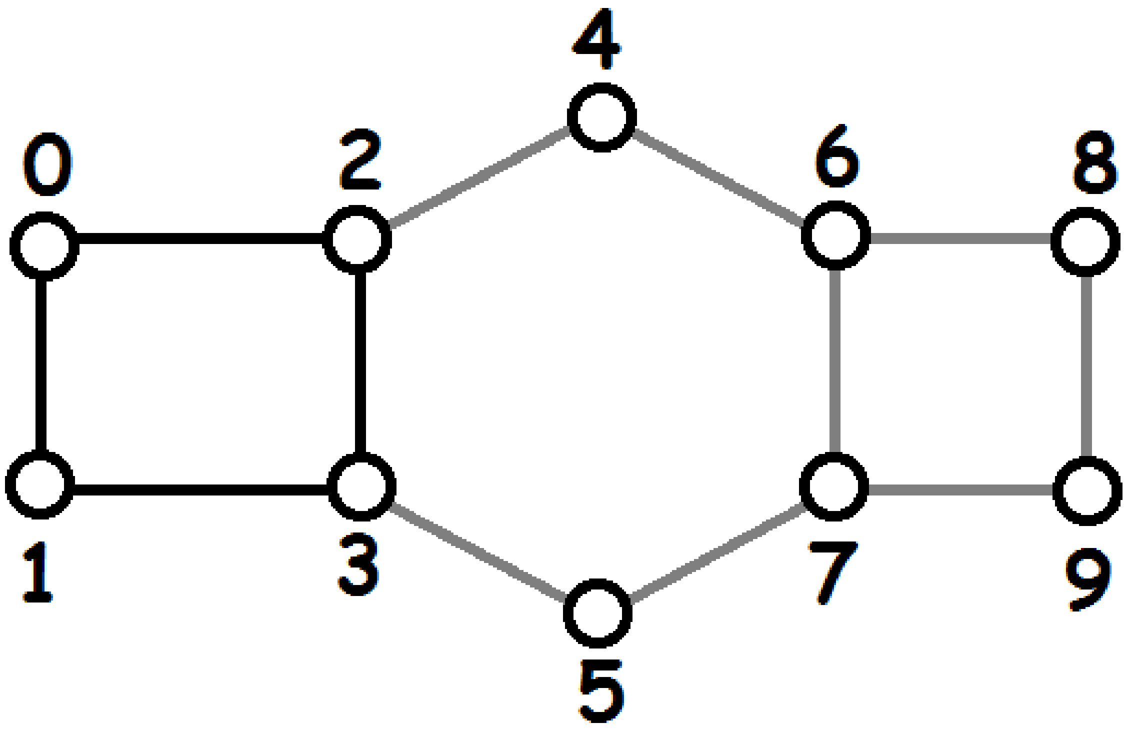
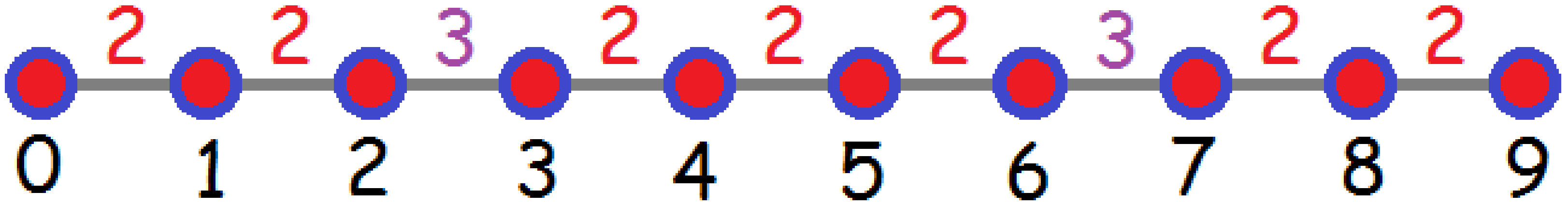
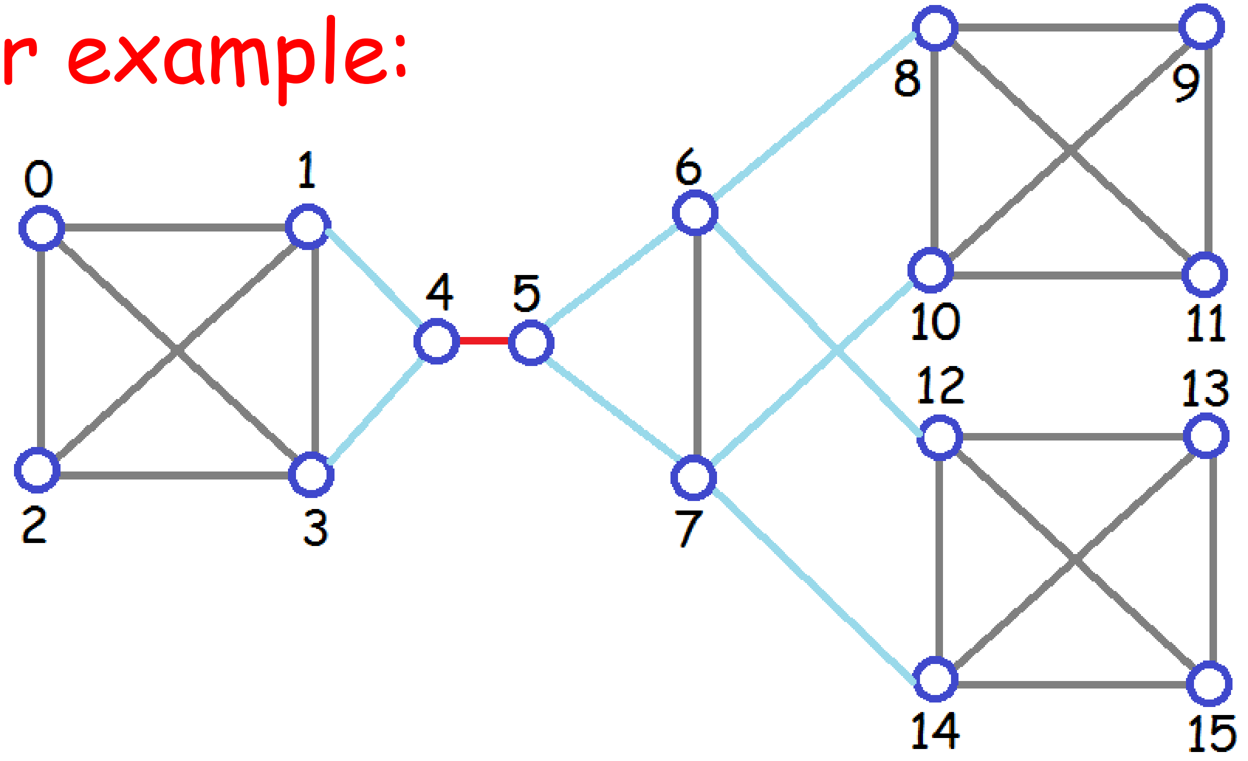
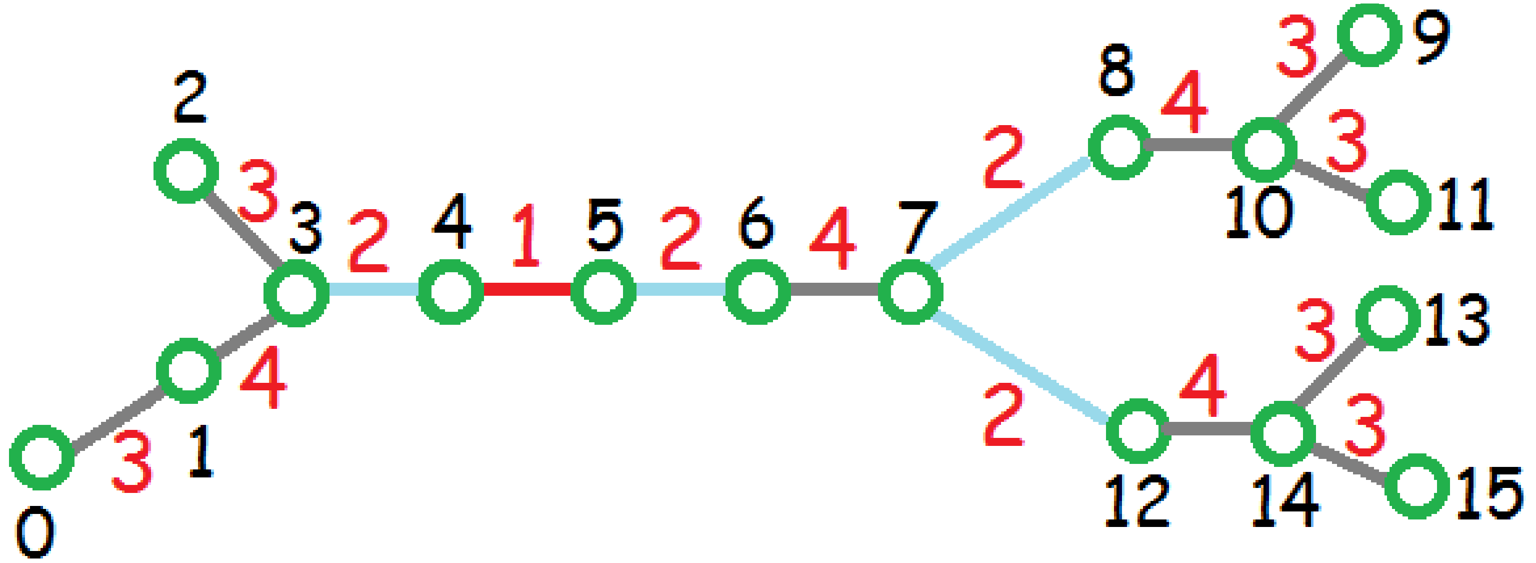
These two graphs have the same sized minimum cuts between every pair of vertices:





A bigger example:





Crossing Cuts

Cuts (X, X') and (Y, Y') cross if all 4 of:

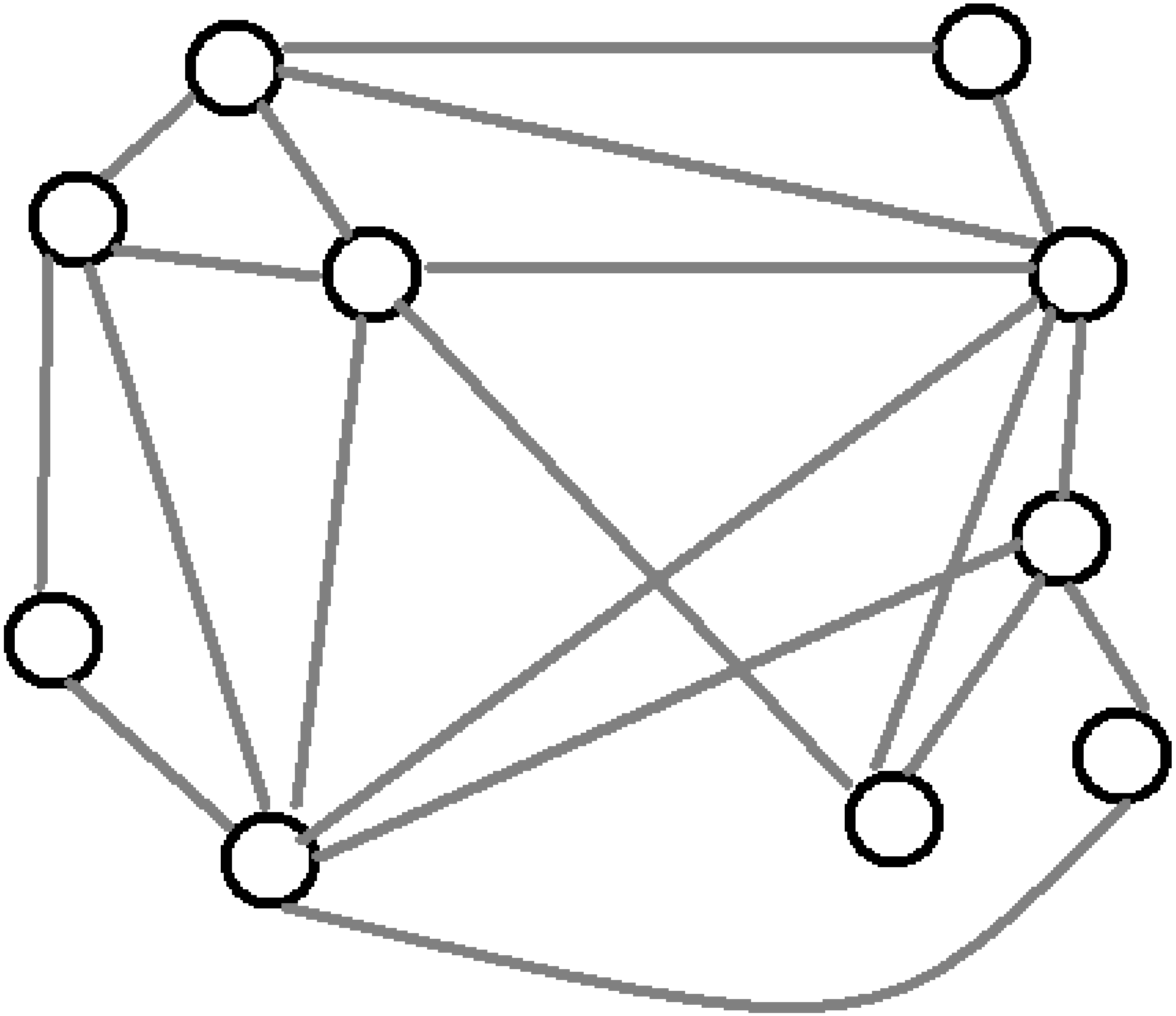
$X \cap Y$

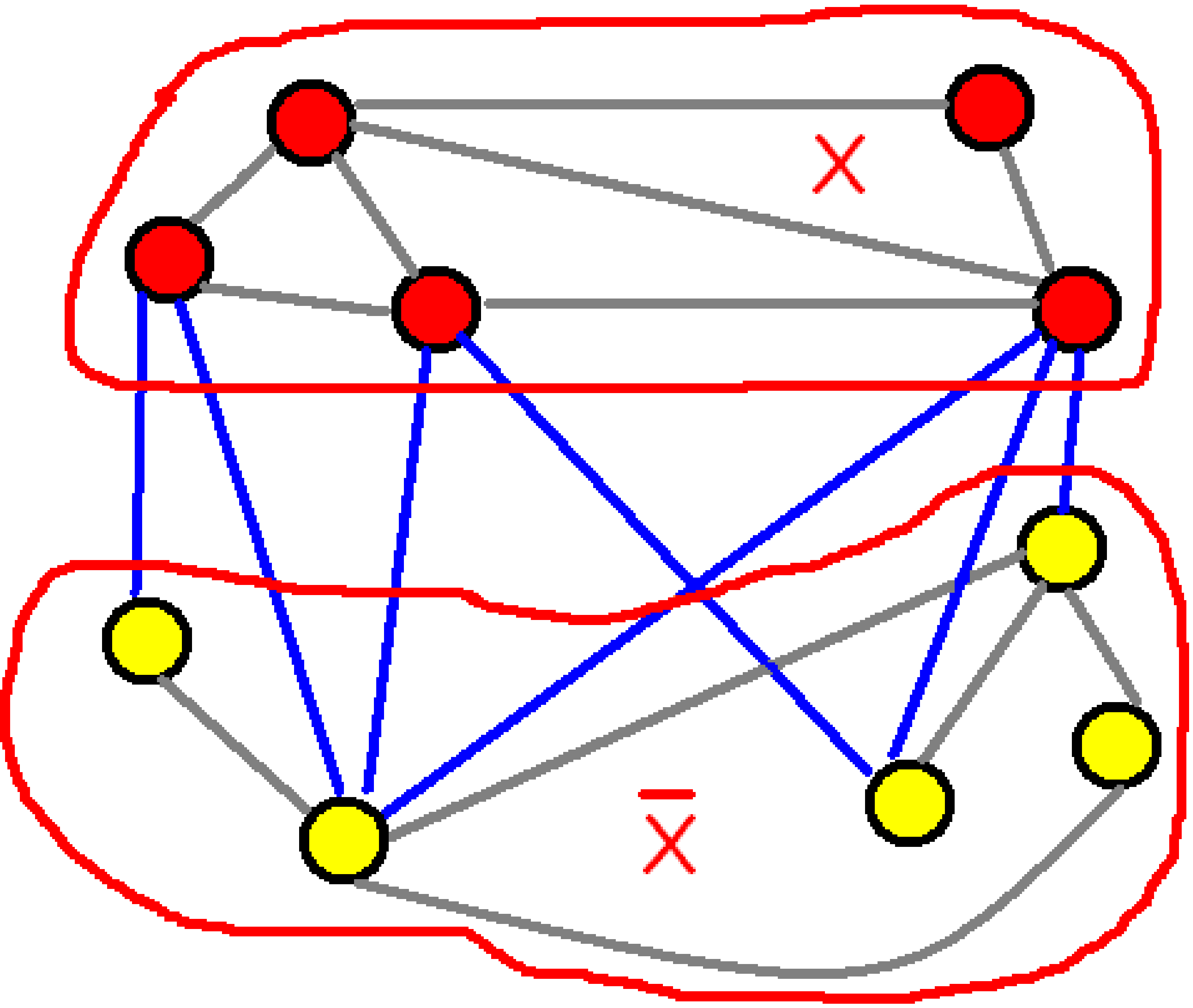
$X \cap Y'$

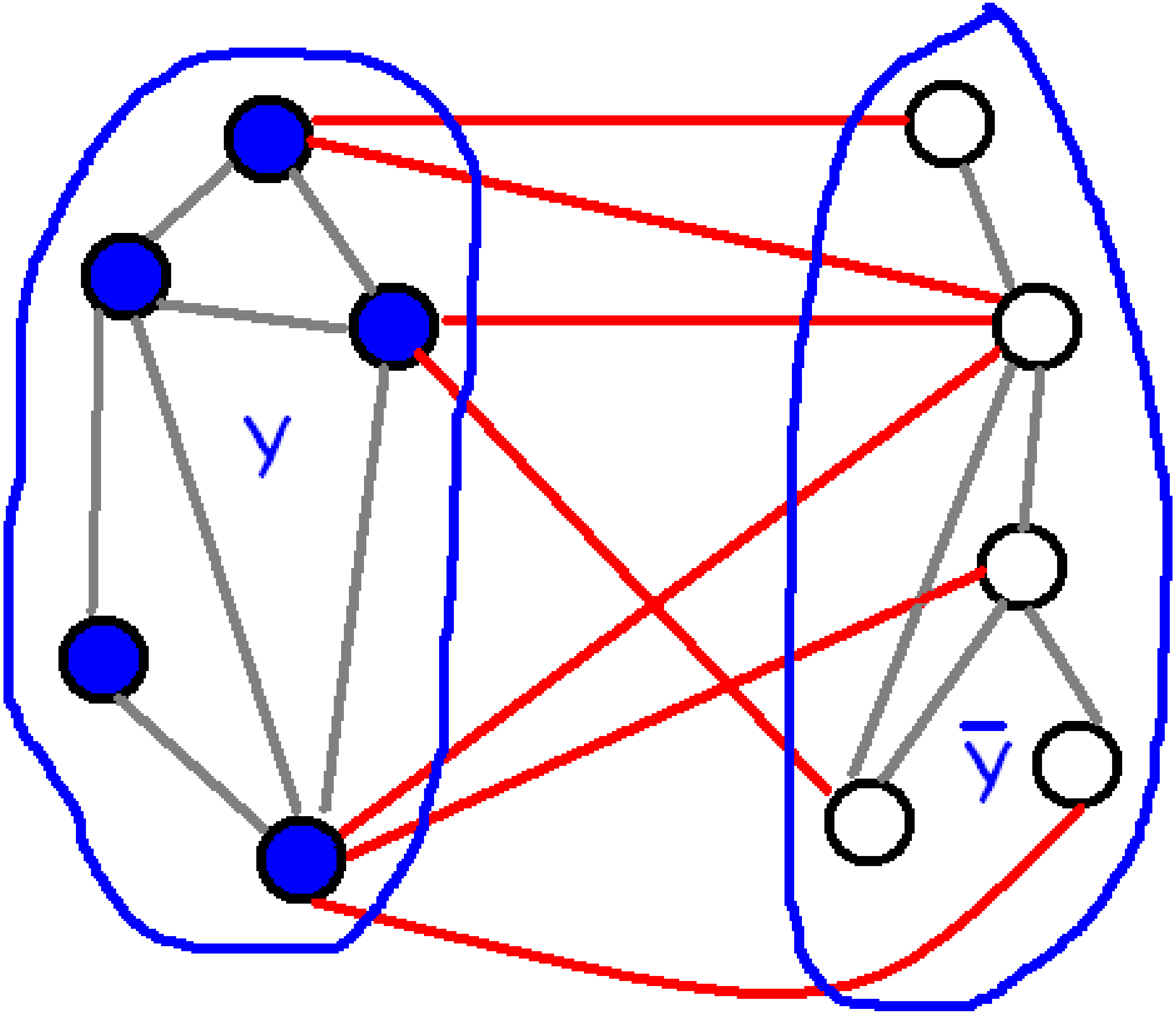
$X' \cap Y$

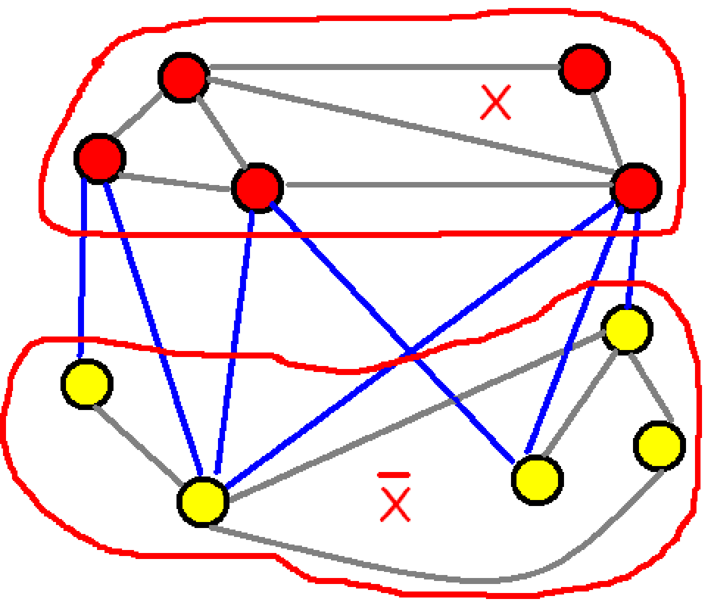
$X' \cap Y'$

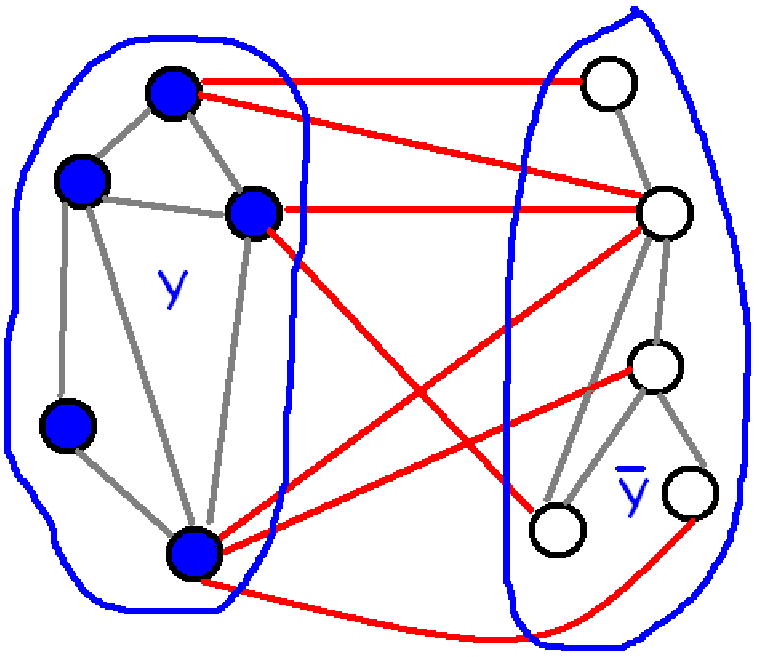
are non-empty.




X
$\overline{X}$


y
$\bar{y}$

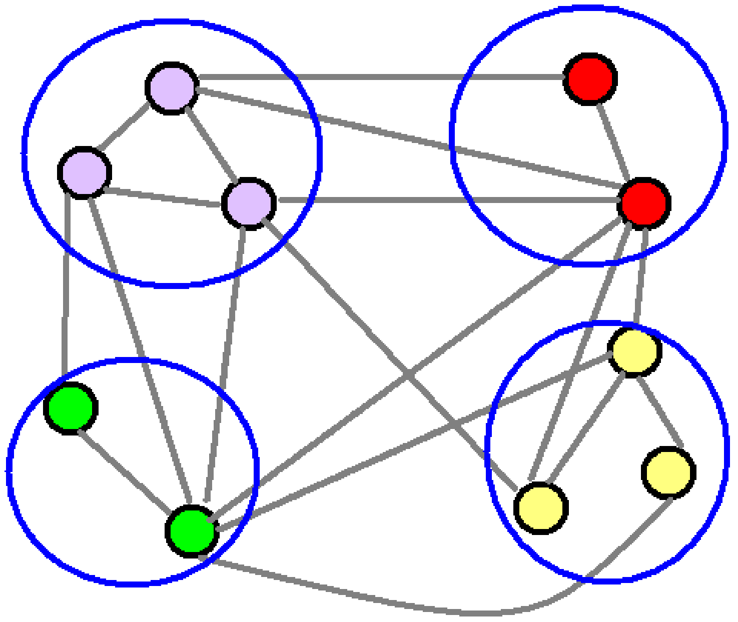


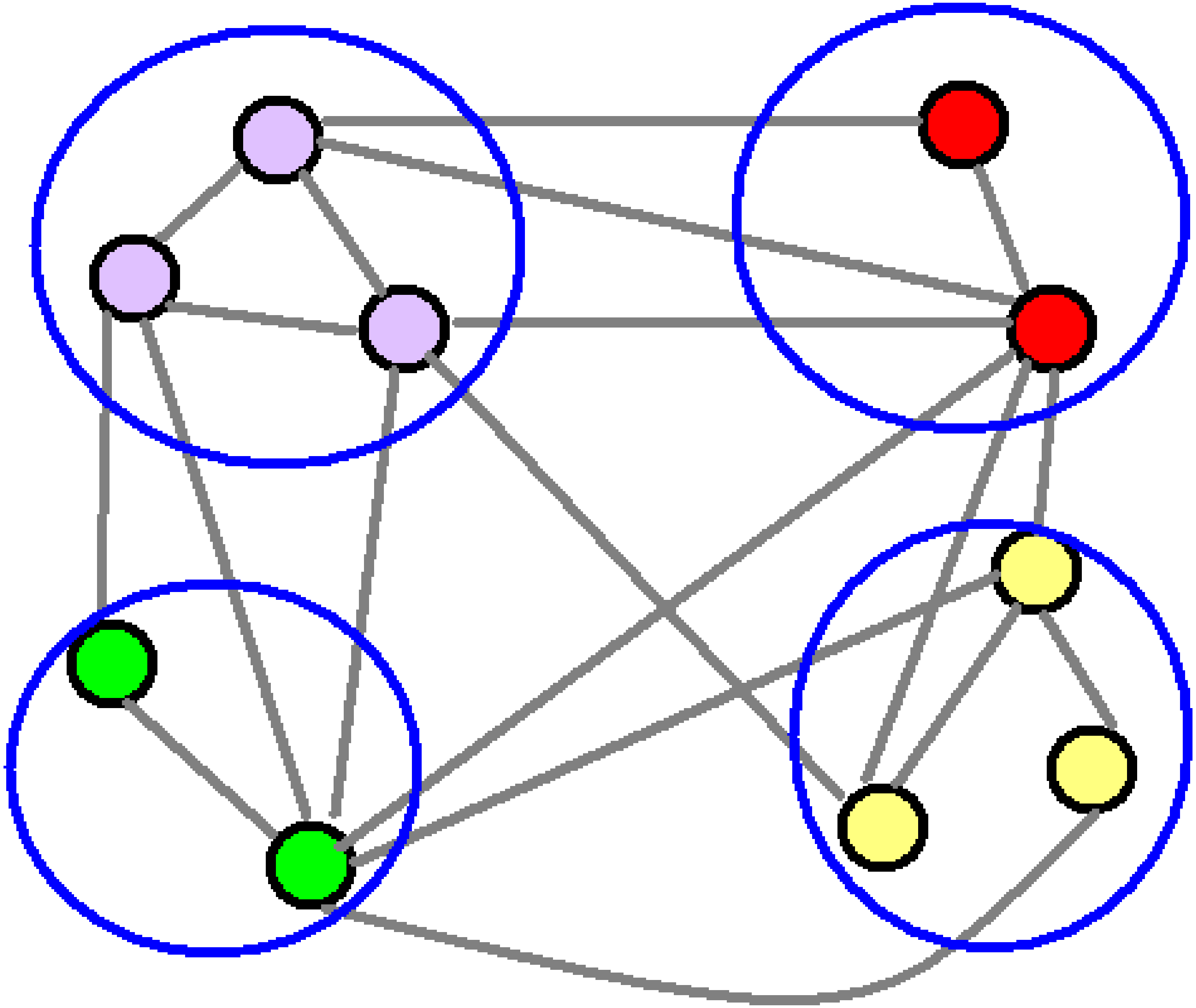


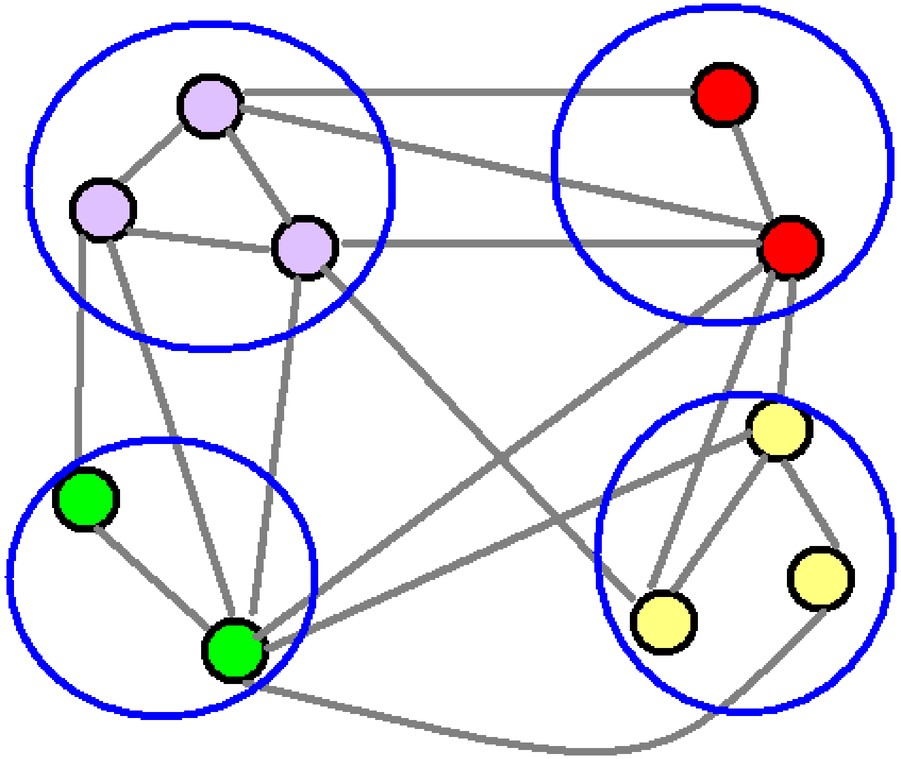
These

2 cuts

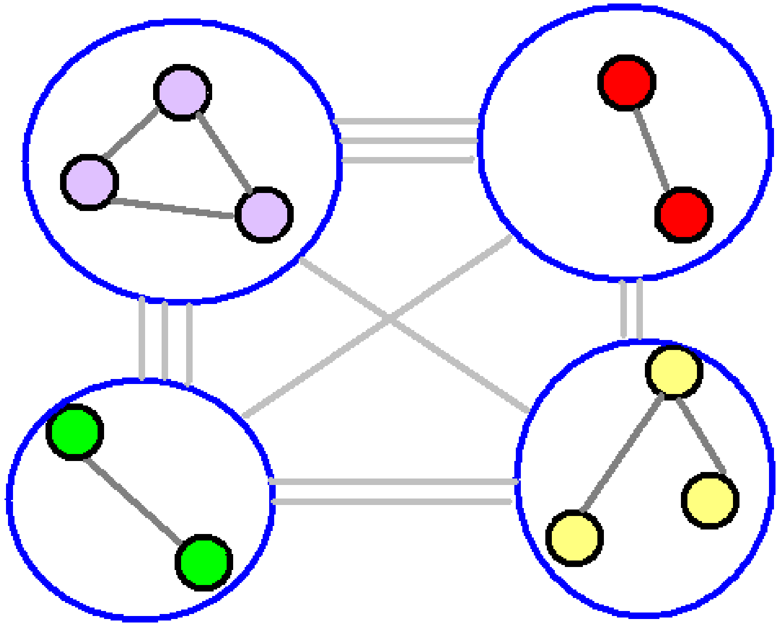
cross

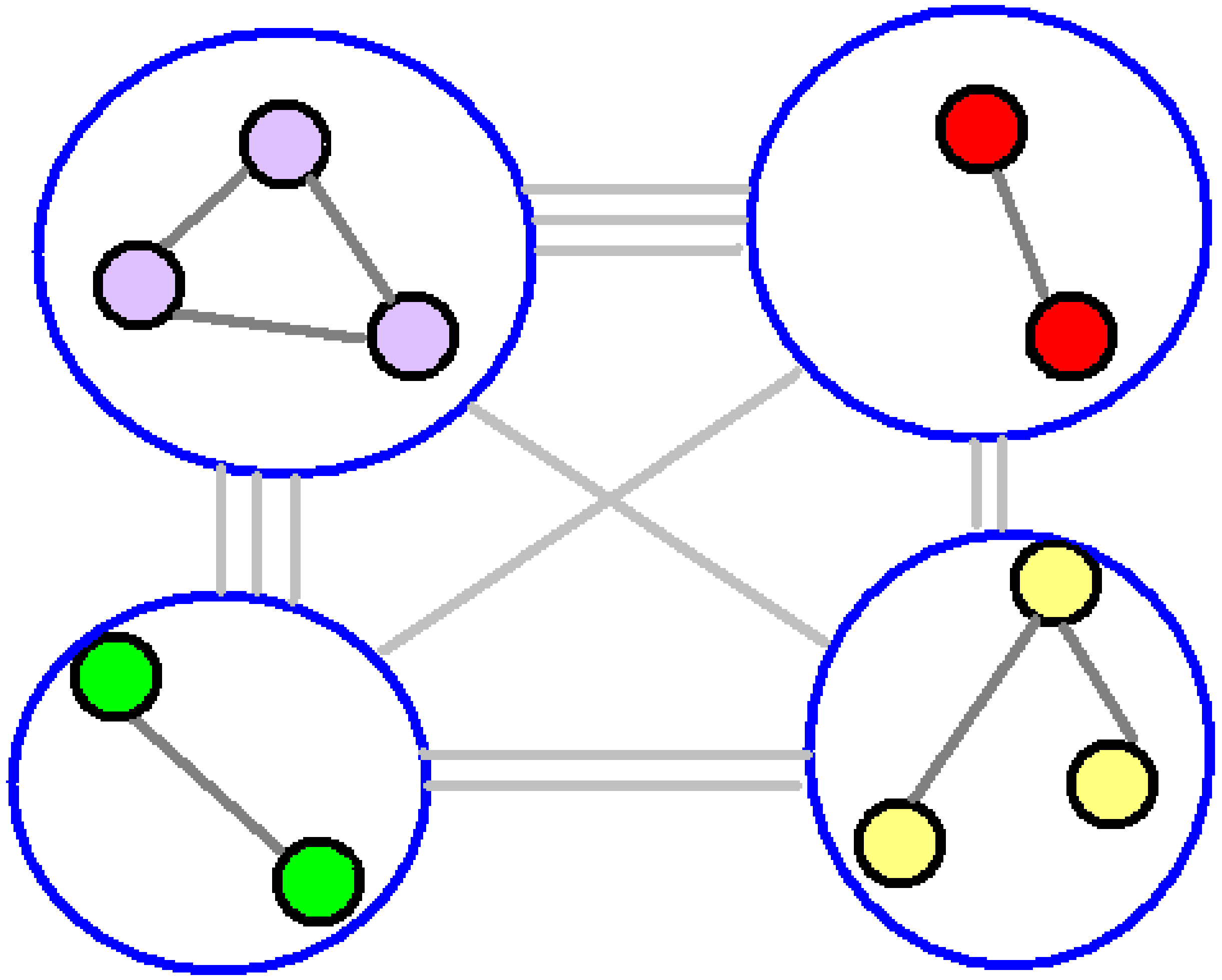


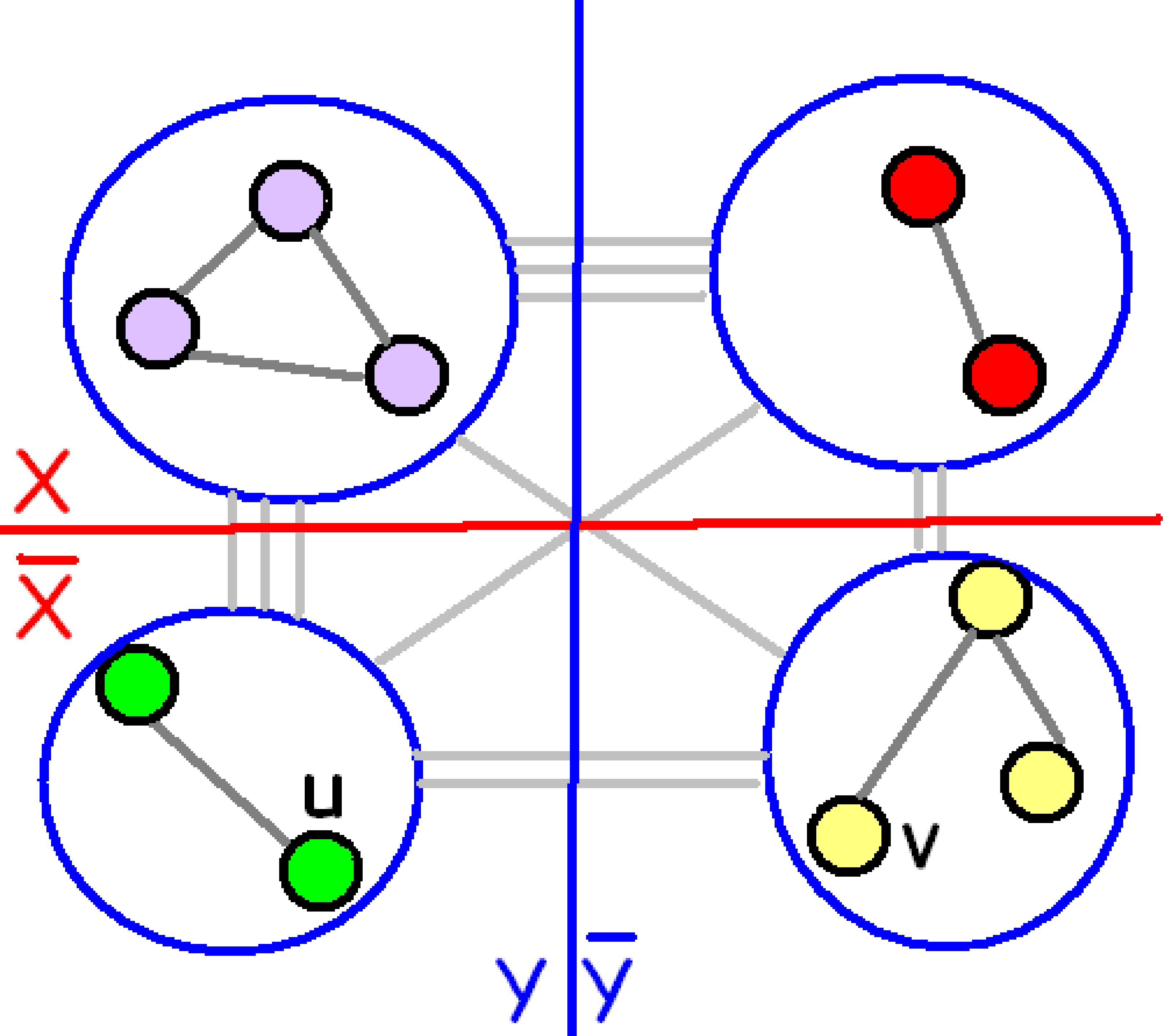


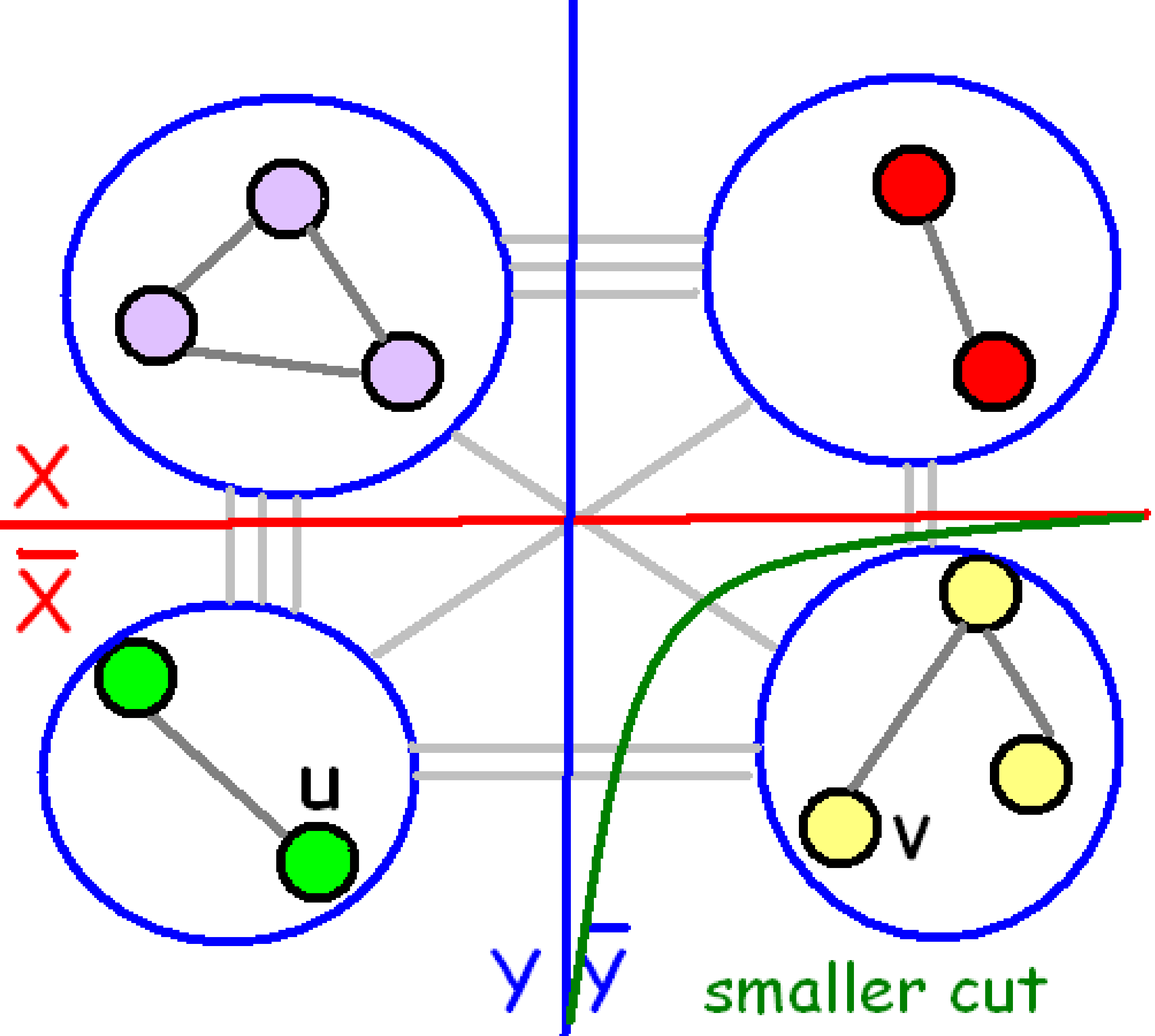


Simplifying the picture

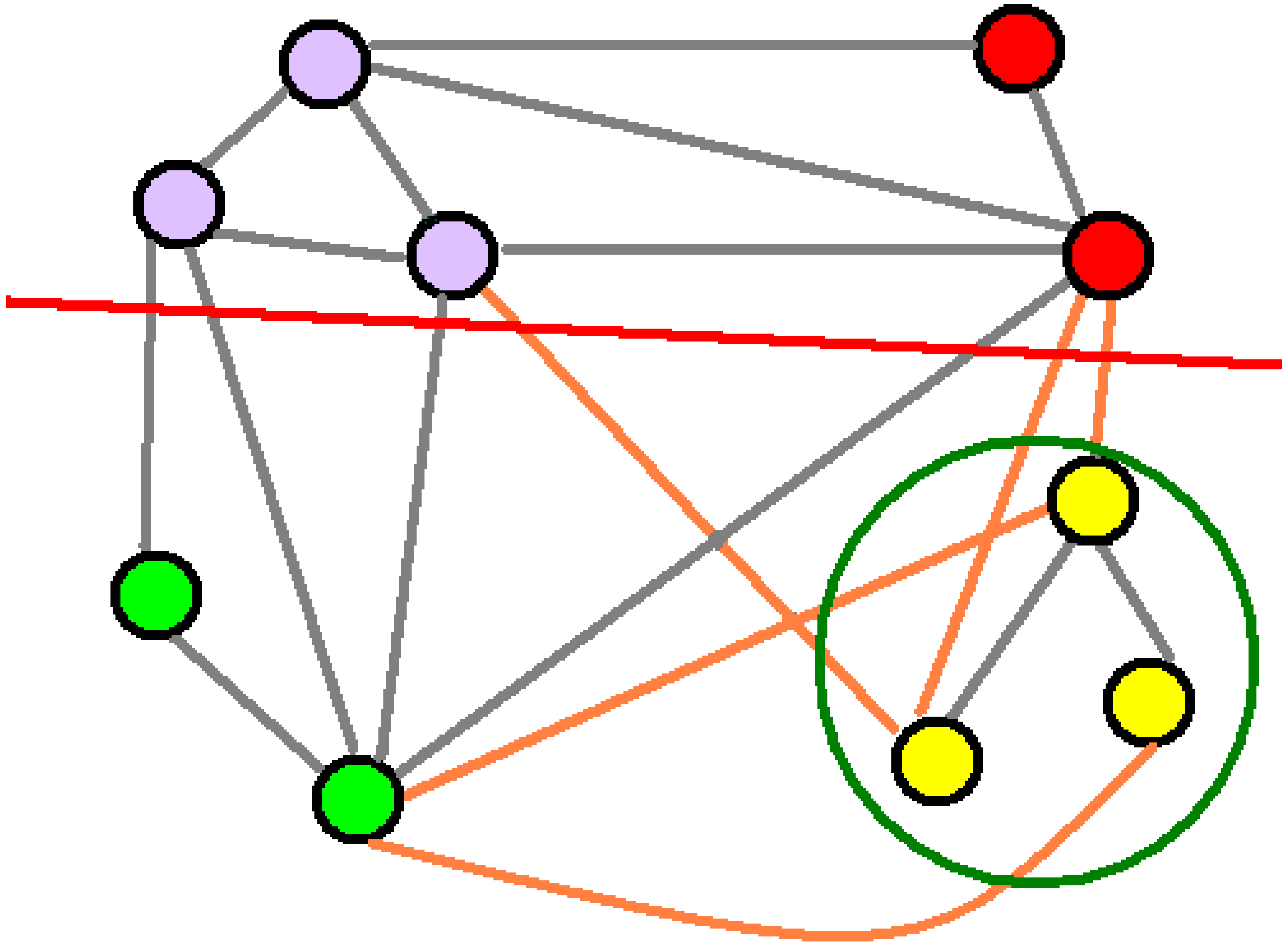





X
$\overline{X}$
u
v
Y
$\overline{Y}$



X
$\overline{X}$
u
v
Y
$\overline{Y}$
smaller cut



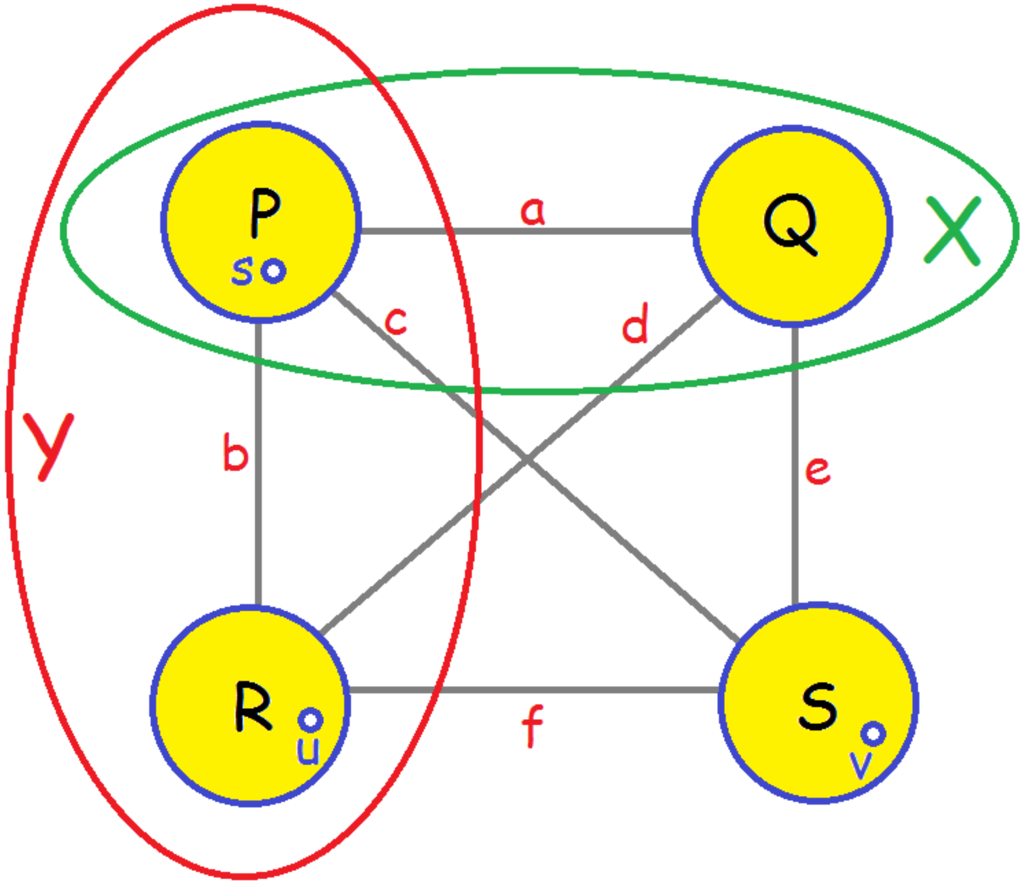
$(X, \bar{X})$ is a minimum s,t-cut with s in X and t in $\bar{X}$.

$(Y, \bar{Y})$ is a minimum u,v-cut with u in Y and v in $\bar{Y}$.

Both u and v are in $\bar{X}$ (u could be t).

Theorem:

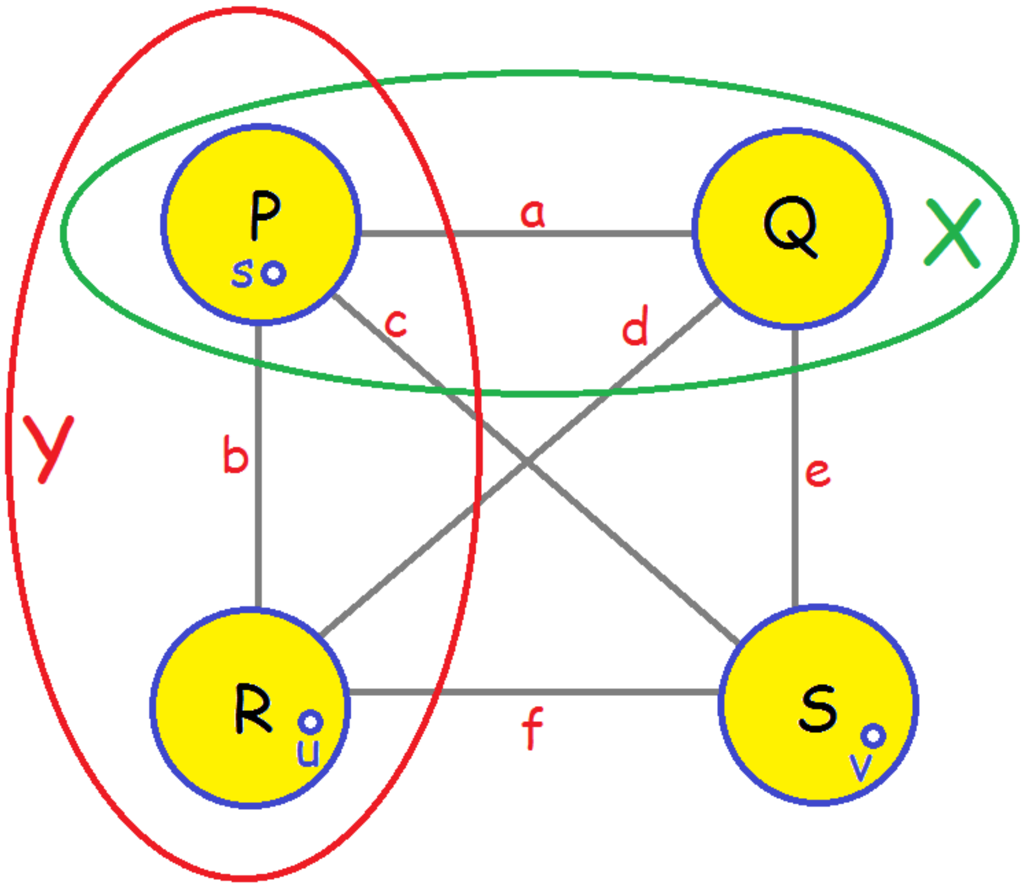
Suppose that $(X, \bar{X})$ and $(Y, \bar{Y})$ cross. Then there exists some minimum u,v-cut $(Z, \bar{Z})$ which does not cross $(X, \bar{X})$.



$(X,\bar{X})$= min s,t-cut.

$(Y,\bar{Y})$= min u,v-cut.

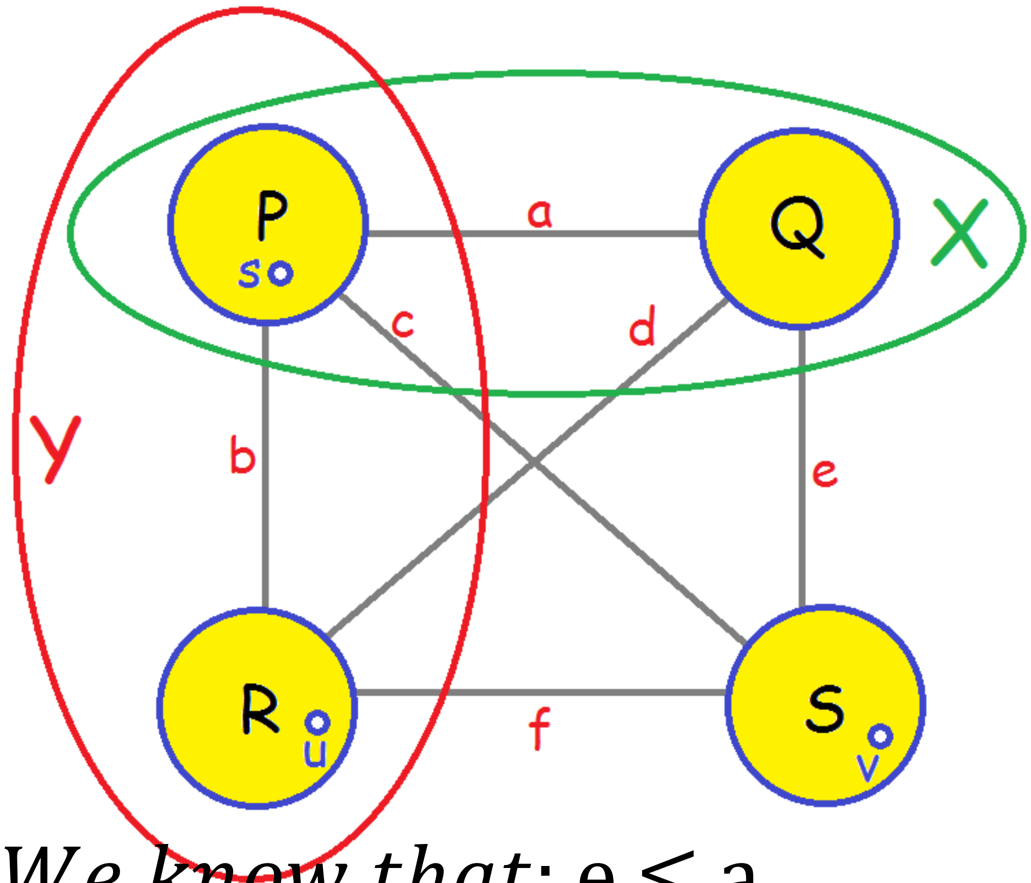
X= P ∪ Q, Y= P ∪ R.
Without loss of generality, s is in P.
Assume u ∈ R and v ∈ S.
The vertex t might be u or v.



$(X, \bar{X})$= min s,t-cut.

$(Y, \bar{Y})$= min u,v-cut.

$capacity(X, \bar{X})$= b + c + d + e ≤ a + b +c since otherwise (P, $\bar{P}$) is a smaller s,t-cut.
So d+e ≤ a $\Rightarrow$ e ≤ a.



$(X, \bar{X})$= min s,t-cut.

$(Y, \bar{Y})$= min u,v-cut.

$We\ know\ that$: e ≤ a.

$c + f + e \leq c + f + a + d = capacity(Y, \bar{Y})$

So $(S, \bar{S})$ is a non-crossing u,v-cut.

This cut is either smaller (contradicting that the original cut was minimum) or the same size.

The previous theorem is the basis for the proof of correctness of the Gomory-Hu cut tree algorithm.

It means that if we have found a minimum st-cut $(X, \bar{X})$ and both u and v are in $\bar{X}$, then we are guaranteed to find some minimum u,v-cut by restricting attention to just $\bar{X}$. The vertices in X can be contracted together.

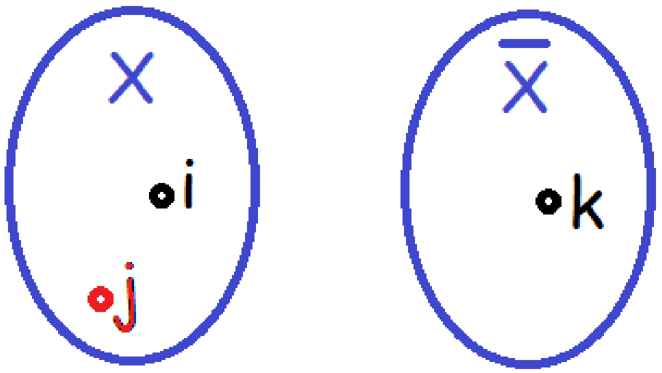
Theorem [Triangle inequality for cutsets]

Let $f_{i,j}$ be the maximum amount of flow possible from vertex i to vertex j. From our flow theory we know that this also equals the capacity of a minimum i,j-cut.

Then $f_{i,k} \geq \text{Min}\,\{f_{ij}, f_{j,k}\}$ for all i, j, k.,

$f_{i,k} \geq \text{Min}\,\{f_{ij}, f_{j,k}\}$ for all i, j, k.

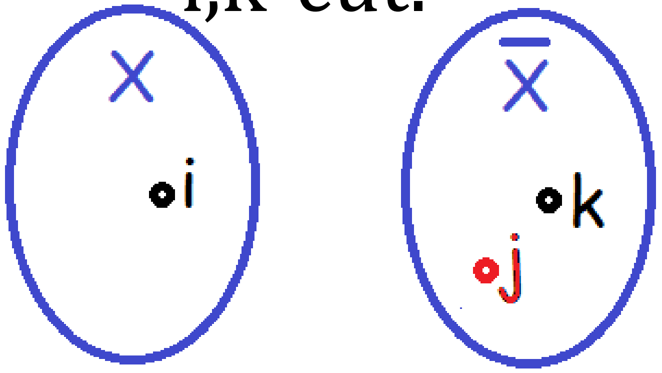
Case 1:



$f_{j,k} \leq f_{i,k}$

$(X, \bar{X}) =$ min i,k-cut.

Case 2:



$f_{i,j} \leq f_{i,k}$

Theorem [Triangle inequality for cutsets]

Let $f_{i,j}$ be the maximum amount of flow possible from vertex i to vertex j. From our flow theory we know that this also equals the capacity of a minimum i,j-cut.

Then $f_{i,k} \geq \mathrm{Min}\,\{f_{ij}, f_{j,k}\}$ for all i, j, k.,

Corollary:

$f_{1,k} \geq \mathrm{Min}\,\{ f_{1,2}, f_{2,3}, f_{3,4}, \dots f_{k-1,\,k} \}$

Proof by induction.

Theorem: A graph G has at most n-1 distinct flow values between pairs of vertices.

Idea in proof:

Start with a complete graph on n vertices with each edge (u,v) labelled with f_{uv}. Choose a maximum weight spanning tree T of G. Any chord (w,x) of T must have $f_{w,x}$ equal to the minimum weight edge on the path connecting w to x.

Gomory-Hu algorithm: Constructs a maximum weight spanning tree T.

Recall: Any chord (w,x) of T must have $f_{w,x}$ equal to the minimum weight of an edge on the path connecting w to x.

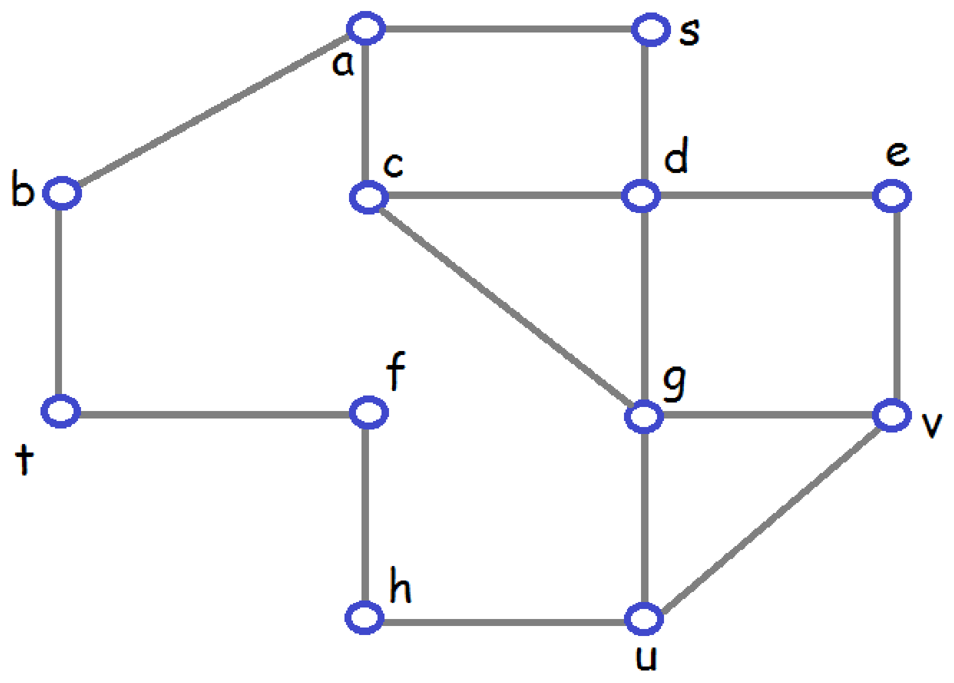
Furthermore, if (u,v) is a minimum weight edge on the path from w to x, then one minimum w,x-cut of G can be found by considering T- (u,v) and setting

X= subset of vertices in same component as vertex w in T-(u,v), and

X'= the rest of the vertices

where T is the Gomery-Hu cut tree.

1. Find a maximum number of edge disjoint paths and a minimum cut between s and t and between u and v.
2. Find a minimum s,t-cut $(P, \bar{P})$ and a minimum u,v-cut $(Q, \bar{Q})$ that cross. Then find another minimum u,v-cut $(R, \bar{R})$ that does not cross $(P, \bar{P})$.



Gomory-Hu Cut Tree Algorithm

At every step of the algorithm we have constructed some tree T.

Each node of T corresponds to a subset of the vertices of G with each vertex of G in exactly one subset.

I will call the vertices of T supernodes (each supernode corresponds to one or more vertices of the original graph).

Initially, T consists of one supernode which corresponds to the set of all vertices of G. At each step of the algorithm, a supernode containing two or more vertices from G is split into two supernodes which are connected by an edge.

This means at each iteration, the tree gains one more edge and one more vertex.

We stop when each supernode corresponds to exactly one vertex from G.

At each phase of the algorithm:

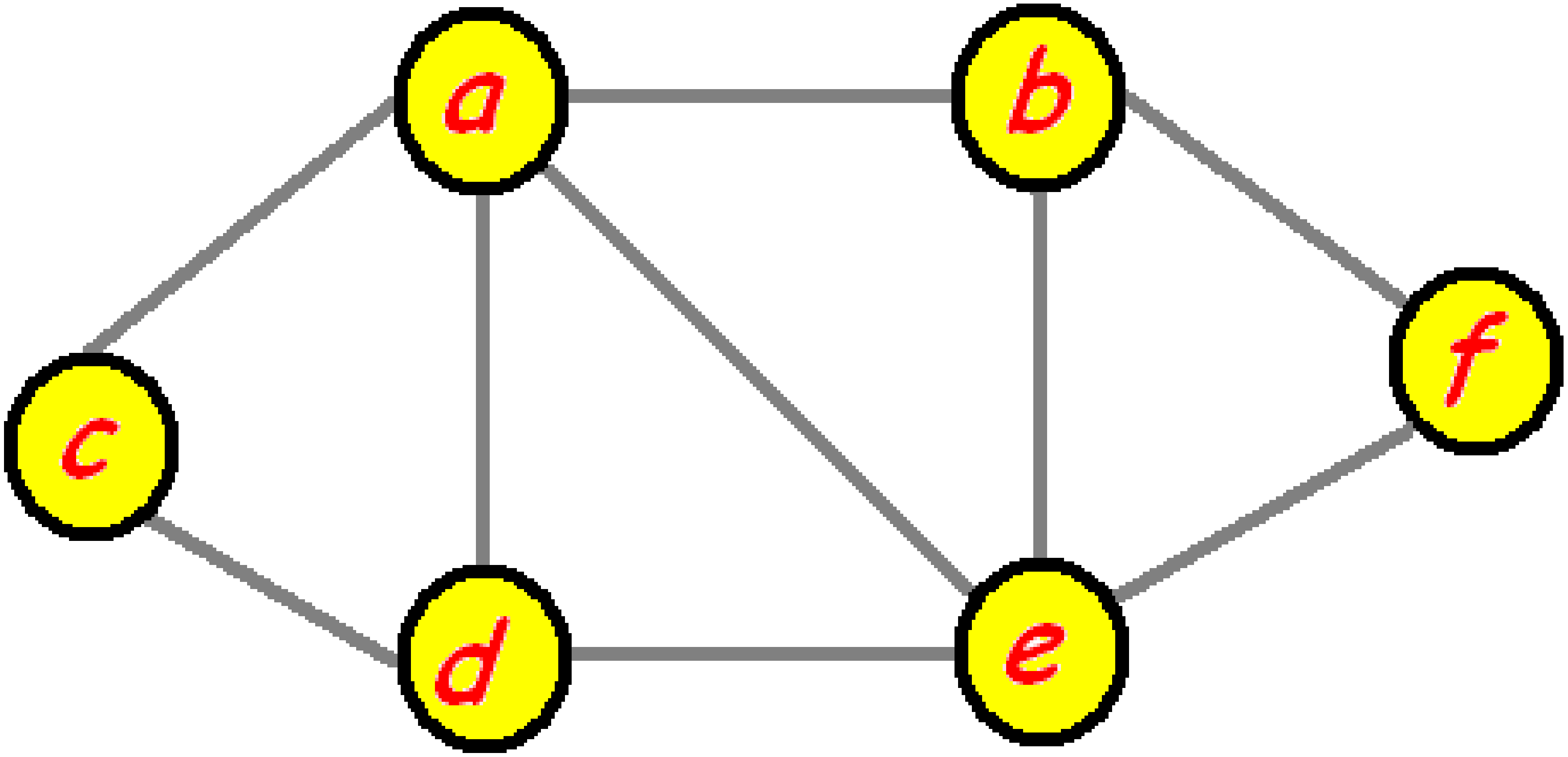
1. Let S be a supernode of T that corresponds two or more vertices in G. Choose two vertices s and t from S. Create *G'* from *G* : for each component T' of T – S, contract together the vertices of T' in *G*.

2. Find a minimum s,t-cut (X, X') in *G'* which has capacity $f_{s,t}$.

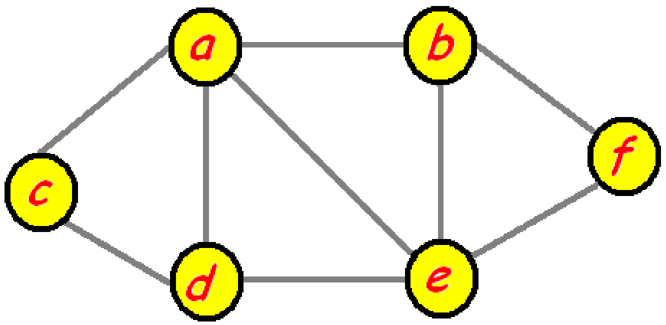
To update the tree T:

1. Delete vertex S (and the edges incident to S).
2. Add two supernodes x and x' where x corresponds to the vertices of G from S which end up on the X side of the cut and x' to the ones from S on the X' side. Add edge (x,x') with weight $f_{s,t}$.
3. For edge (v, S) in the previous tree, vertex v was in a subtree T' of T-S that was contracted to a single vertex v'. If v' ends up on the X side of the cut, then add edge (v, x). Otherwise, add edge (v, x'). The new weight is the same as the weight of (v, S).

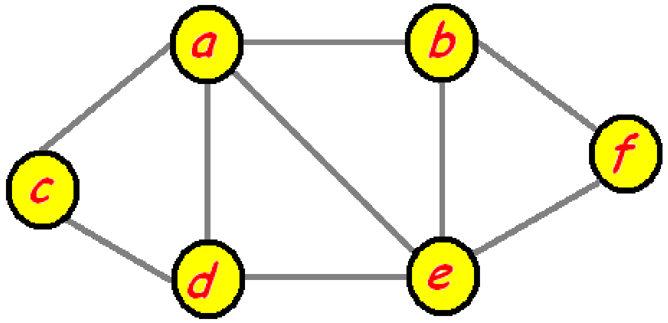
Example:



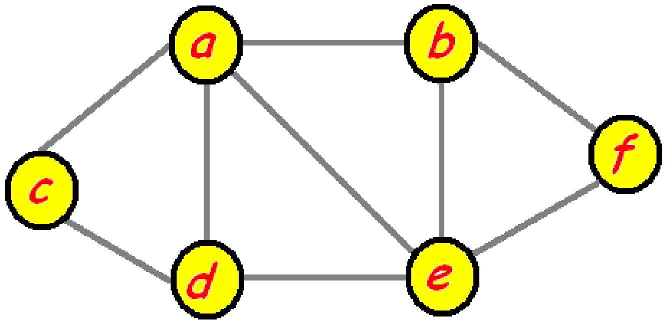
0.



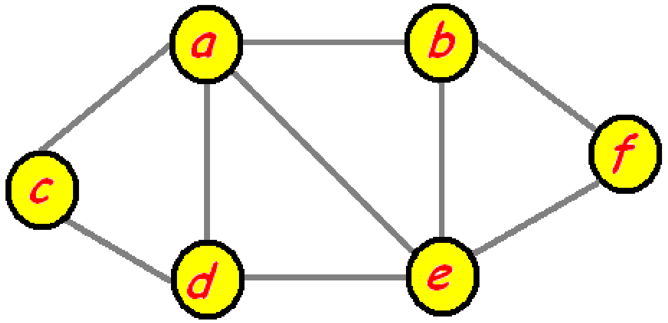
3.



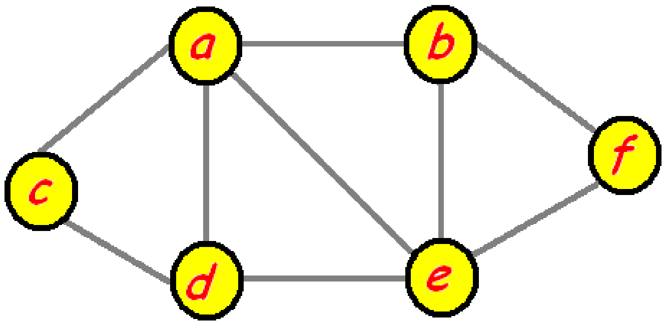
1.



4.



2.



5.