1. Use the algorithm from last class to walk all the faces for the embedding represented by this rotation system.

2. Does it represent a planar embedding?

Recall: $g = (2 - n + m - f)/2$

0: 1 3 5

1: 0 2

2: 1 5 3

3: 0 2 4

4: 3 5

5: 0 4 2

Now available:

Project literature review specifications.
Due date: Fri. Oct. 24 at 11:55pm.
Late deadline: Fri. Oct. 31 by 11:55pm with a 10% late penalty.
Choose your topic and start work early!

Assignment #2: due Friday Oct. 3.

f_0: (0,1)(1,2)(2,5)(5,0)(0,1)

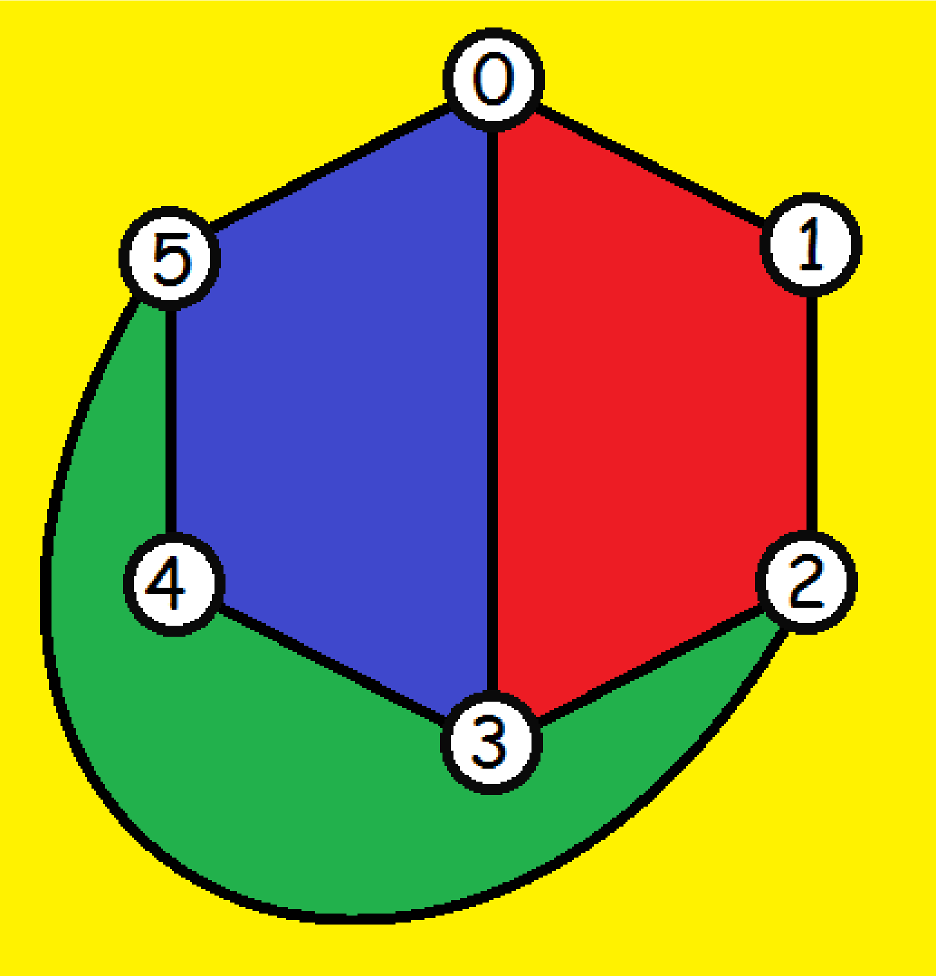
f_1: (0,3)(3,2)(2,1)(1,0)(0,3)

f_2: (0,5)(5,4)(4,3)(3,0)(0,5)

f_3: (2,3)(3,4)(4,5)(5,2)(2,3)

0: 1 3 5
1: 0 2
2: 1 5 3
3: 0 2 4
4: 3 5
5: 0 4 2

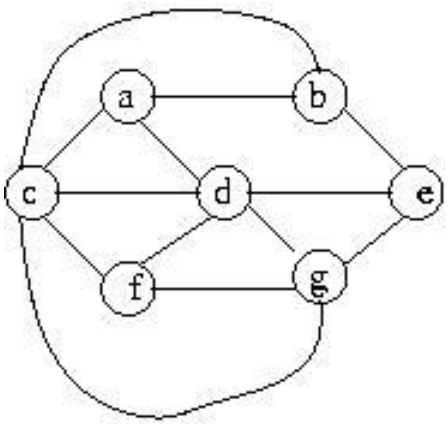
$g = (2 - n + m - f)/2$



Rotation Systems

G connected on an orientable surface:

$g = (2 - n + m - f)/2$



a: b d c
b: a c e
c: a d f g b
d: a e g f c
e: b g d
f: c d g
g: c f d e

0 plane
1 torus
2

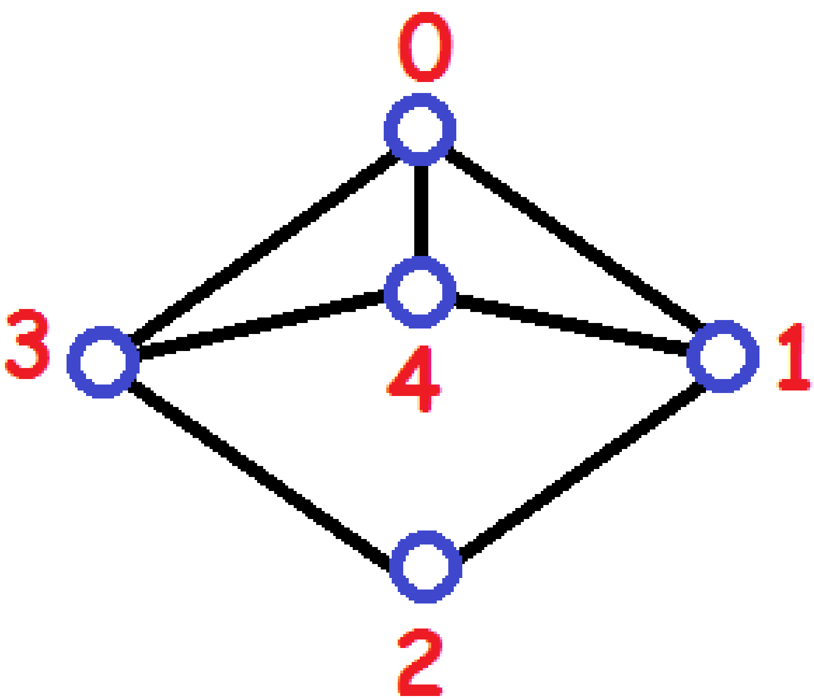
Greg McShane

f_0: (a, b)(b, c)(c, a)(a, b)

f_1: (a, d)(d, e)(e, b)(b, a)(a, d)

How can we find a rotation system that represents a planar embedding of a graph?

Input graph:
0: 1 3 4
1: 0 2 4
2: 1 3
3: 0 2 4
4: 0 1 3



Planar embedding
0: 1 4 3
1: 0 2 4
2: 1 3
3: 0 4 2
4: 0 1 3

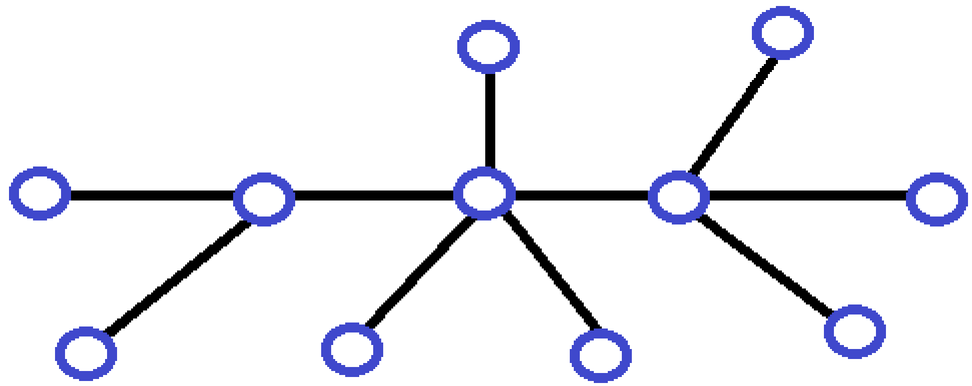
f= number of faces
n= number of vertices
m= number of edges

Euler's formula: For any connected planar graph G, $f = m-n+2$.

Proof by induction:

How many edges must a connected graph on n vertices have?

Euler's formula: For any connected planar graph G, f= m-n+2.



[Basis]

The connected graphs on n vertices with a minimum number of edges are trees.

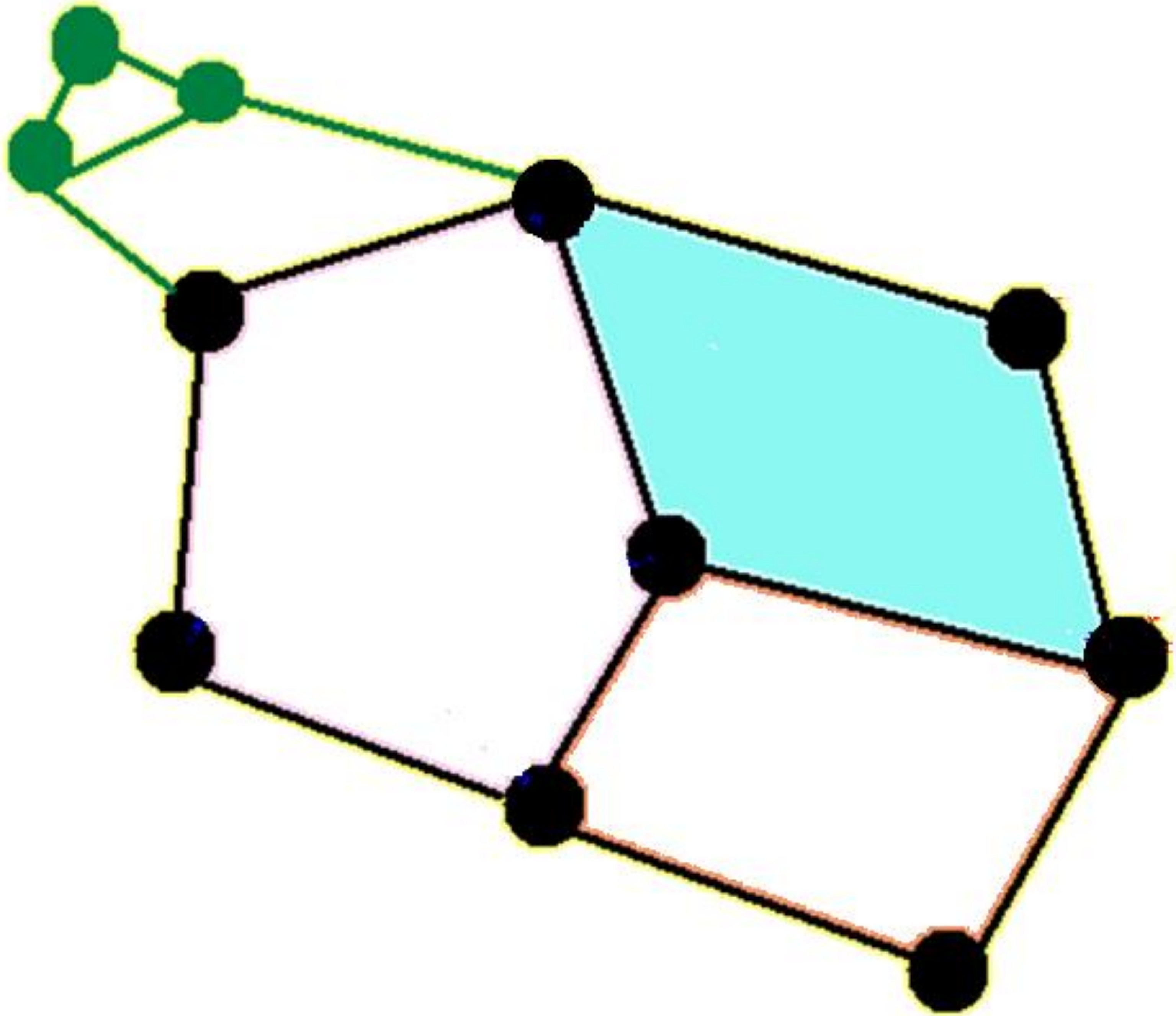
If T is a tree, then it has n-1 edges and one face when embedded in the plane.

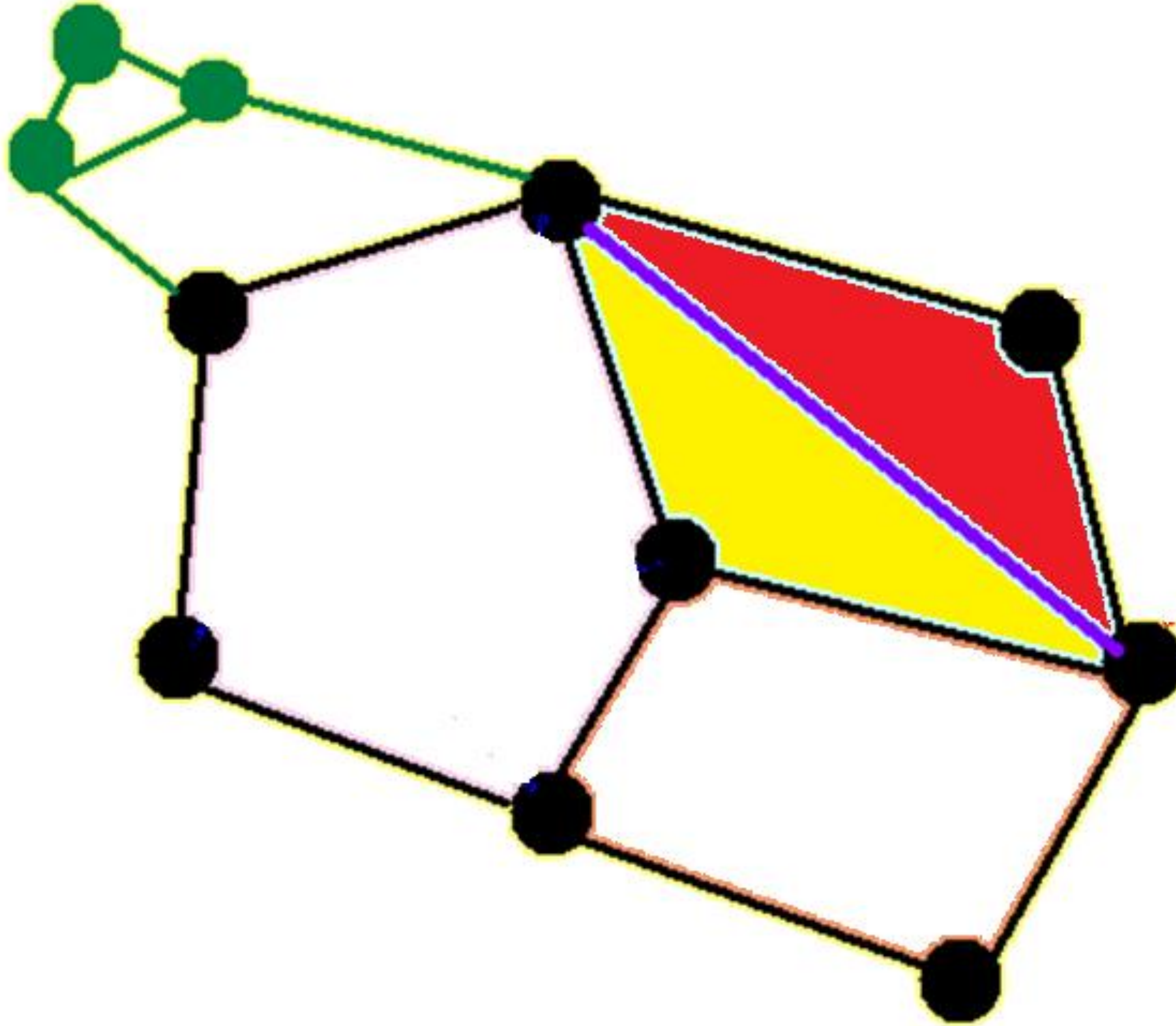
Checking the formula:

1 = (n-1) – n + 2 $\Longrightarrow$ 1 = 1 so the base case holds.

[Induction step ($m \rightarrow m+1$)]

Assume that for a planar embedding $\widetilde{G}$ of a connected planar graph G with n vertices and m edges that f= m-n+2.
We want to prove that adding one edge (while maintaining planarity) gives a new planar embedding $\widetilde{H}$ of a graph H such that f' (the number of faces of H)
satisfies $f' = m' - n + 2$
where $m' = m+1$ is the number of edges of H.





Adding one edge adds one more face.

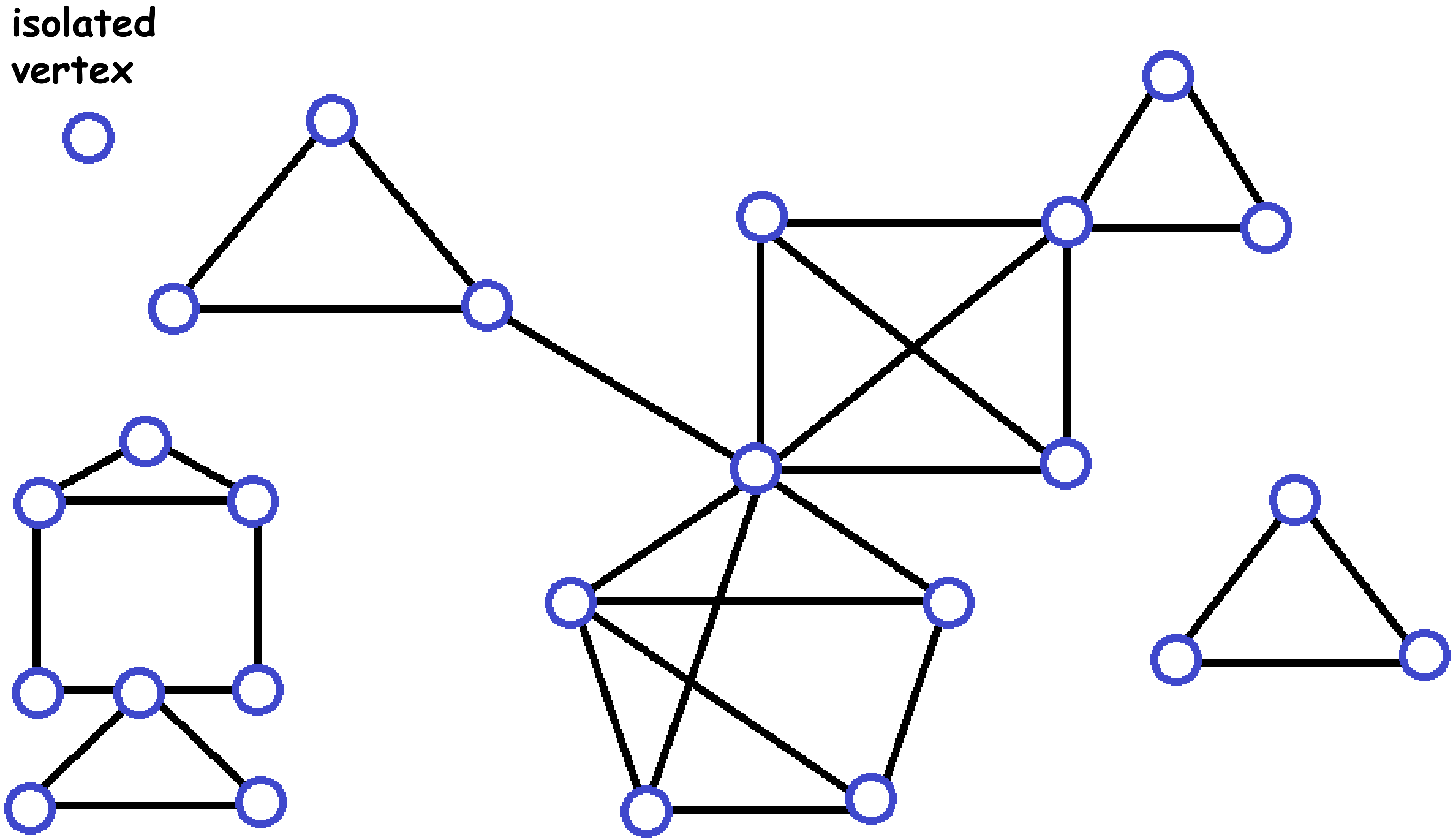
Therefore, $f' = f + 1$. Recall $m' = m+1$.

Checking the formula:
$f' = m' - n + 2$
means that
$f+1 = m+1 - n + 2$
subtracting one from both sides gives
$f = m - n + 2$ which we know is true by induction.

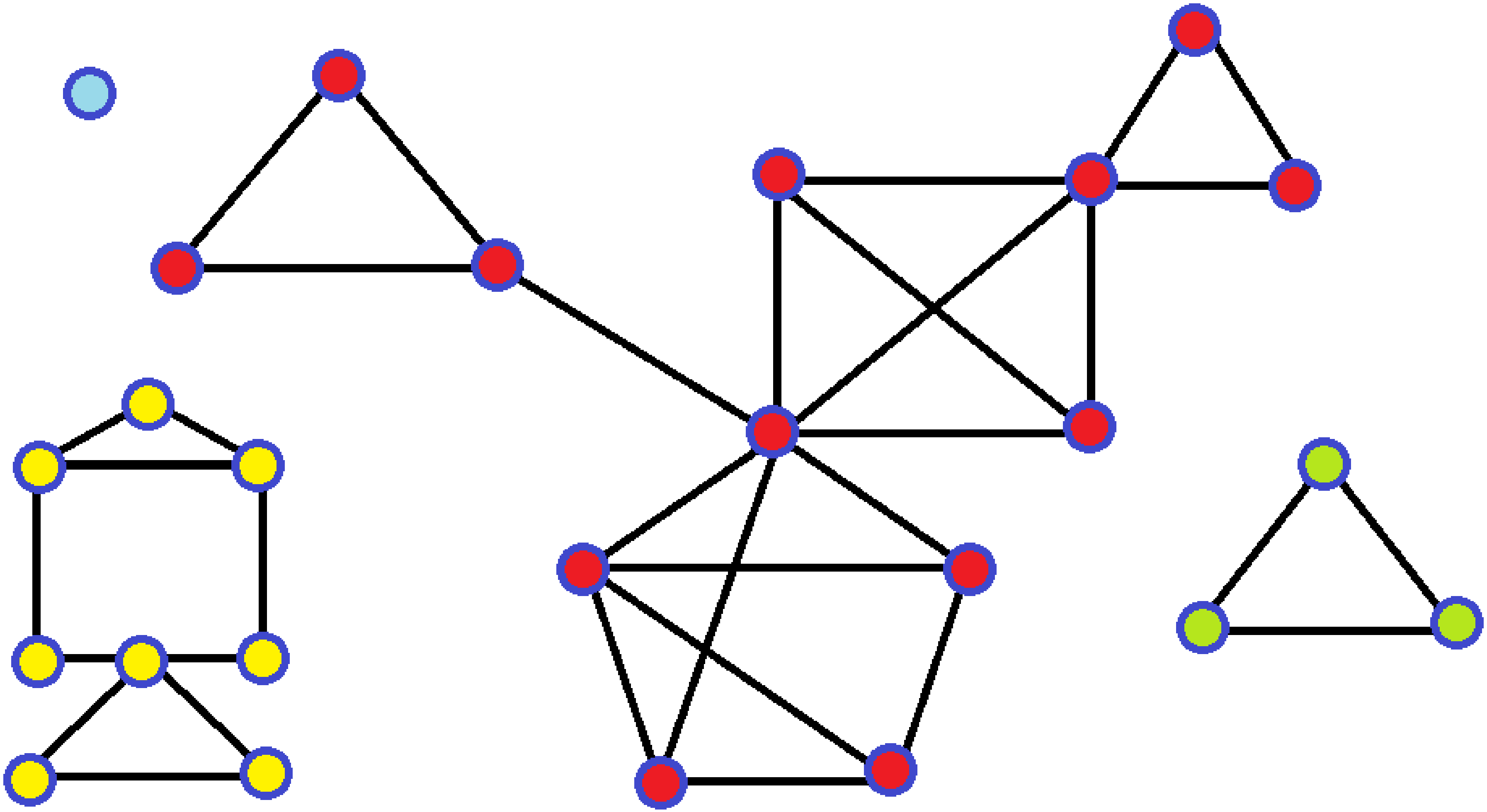
Pre-processing for an embedding algorithm.

1. Break graph into its connected components.
2. For each connected component, break it into its 2-connected components (maximal subgraphs having no cut vertex).

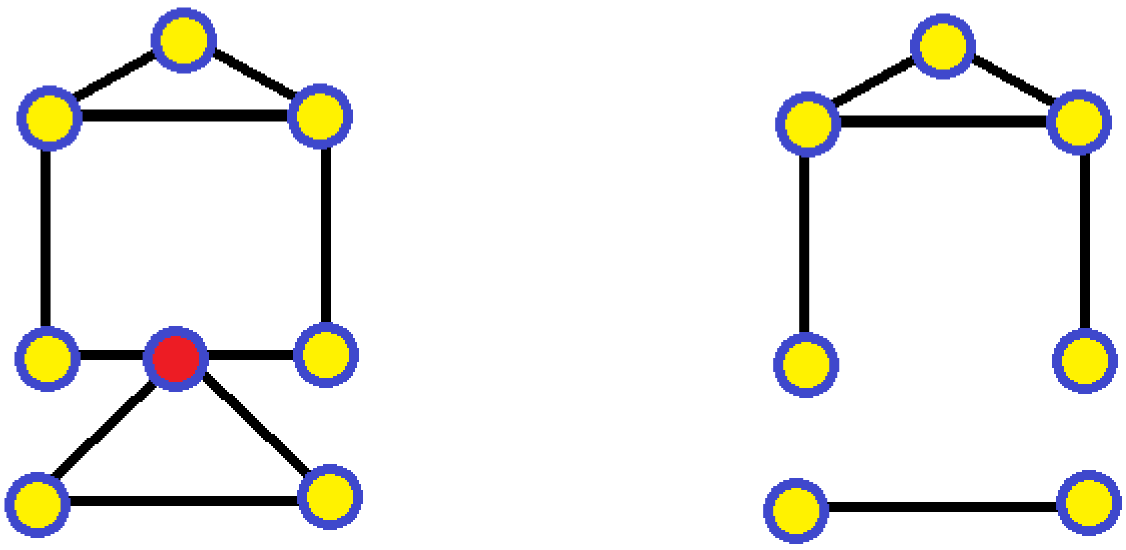
A disconnected graph:



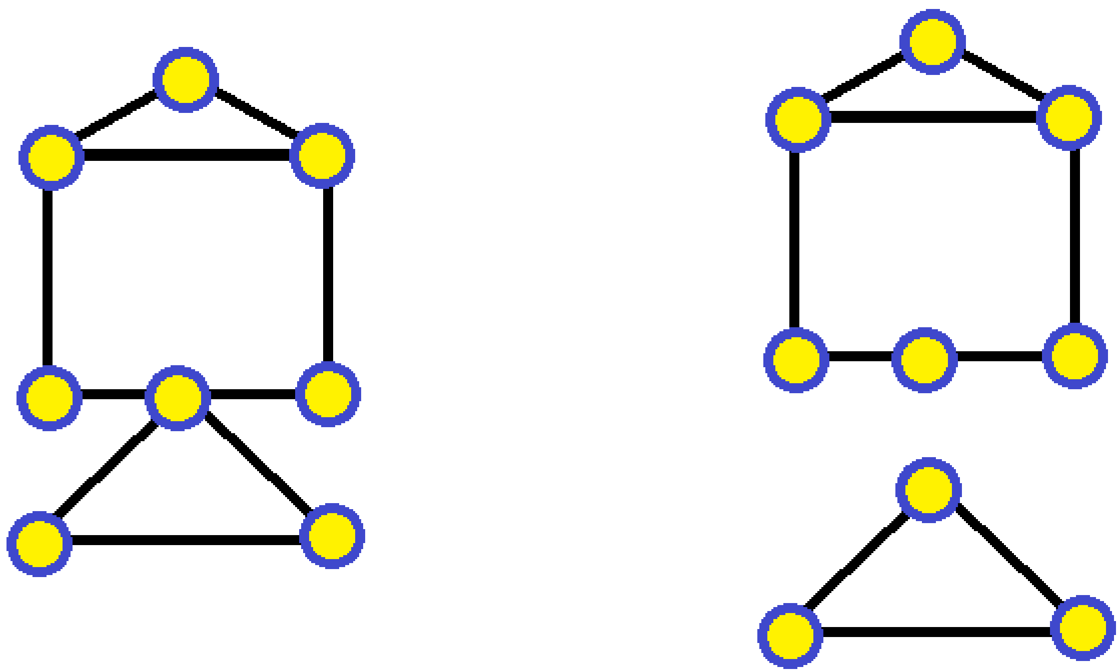
First split into its 4 connected components:



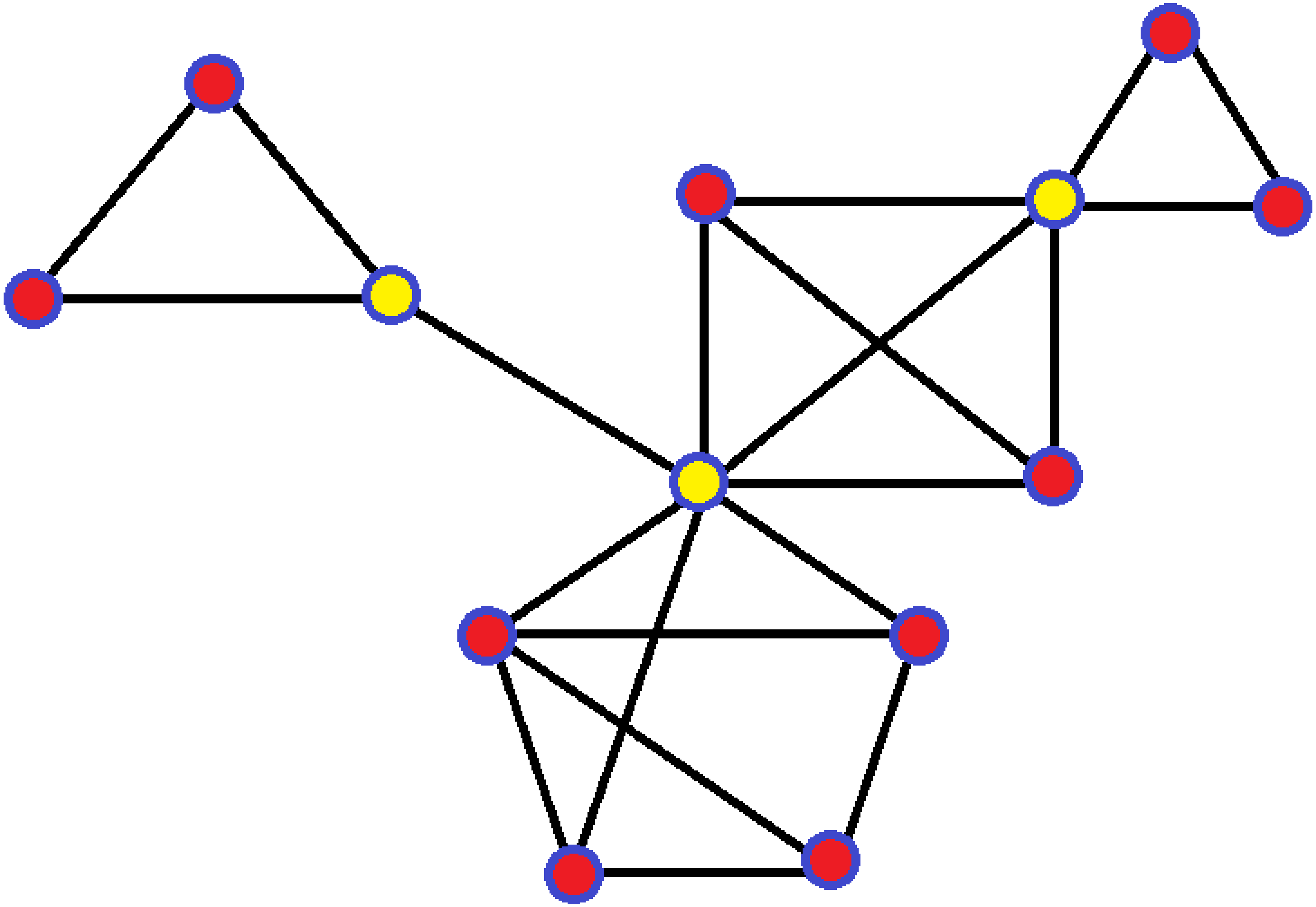
The yellow component has a cut vertex:



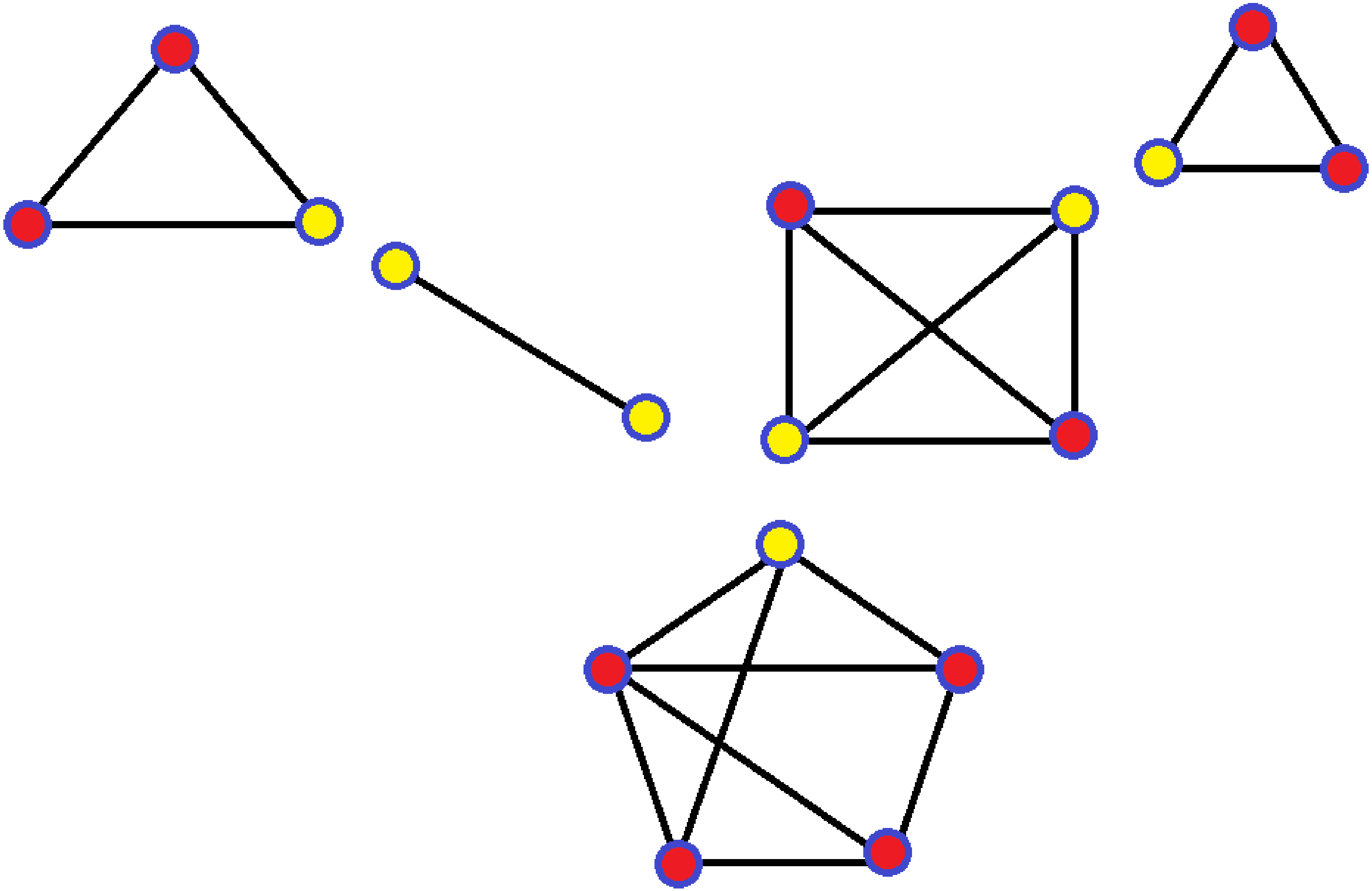
The 2-connected components of the yellow component:

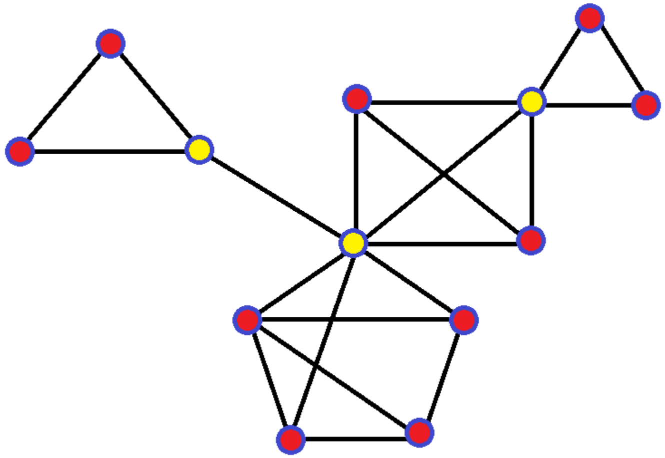


The red component: the yellow vertices are cut vertices.

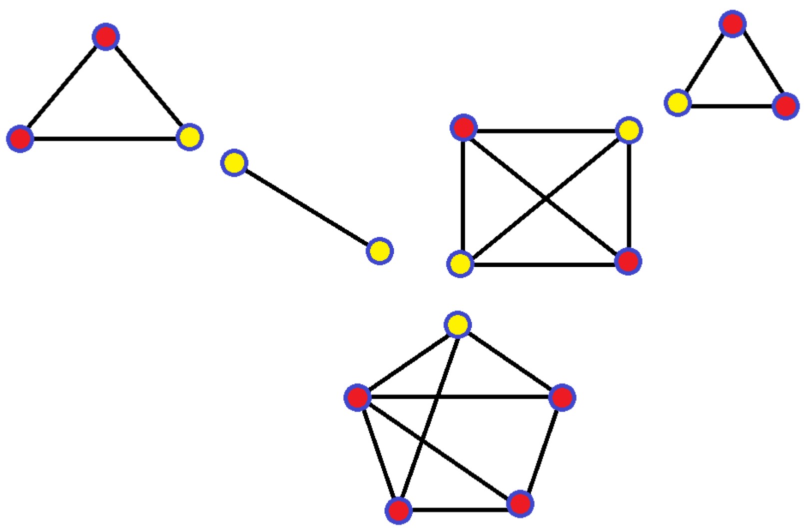


The 2-connected components of the red component:





How do we decompose the graph like this using a computer algorithm?

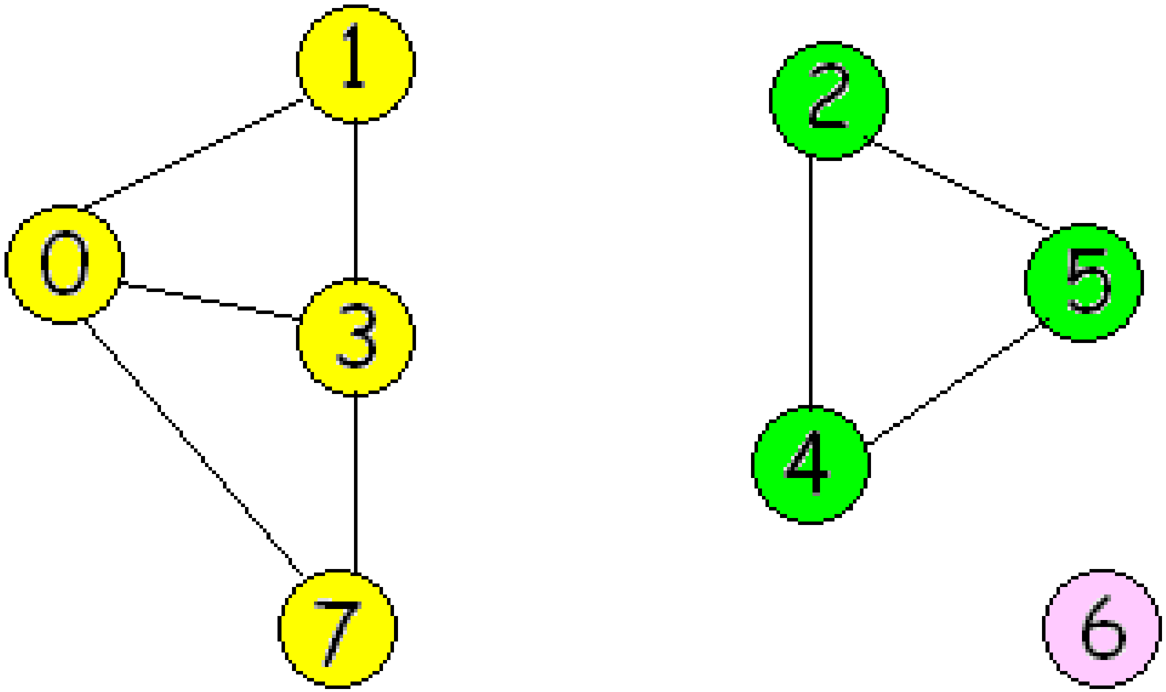


The easiest way:

BFS (Breadth First Search)

One application:

How many connected components does a graph have and which vertices are in each component?



To find the connected components:

```
for (i=0; i < n; i++)
    parent[i]= -1;
nComp= 0;
for (i=0; i < n; i++)
    if (parent[i] == -1)
        nComp++;
        BFS(i, parent, component, nComp);
```

```
BFS(s, parent, component, nComp)

// Do not initialize parent.

// Initialize the queue so that BFS starts at s

qfront=0; qrear=1; Q[qfront]= s;

parent[s]=s;

component[s]= nComp;
```

```
while (qfront < qrear) // Q is not empty
    u= Q[qfront]; qfront++;
    for each neighbour v of u
        if (parent[v] == -1) // not visited
            parent[v]= u; component[v]= nComp;
            Q[qrear]= v; qrear++;
        end if
    end for
end while
```

How could you modify BFS to determine if v is a cut vertex?

-