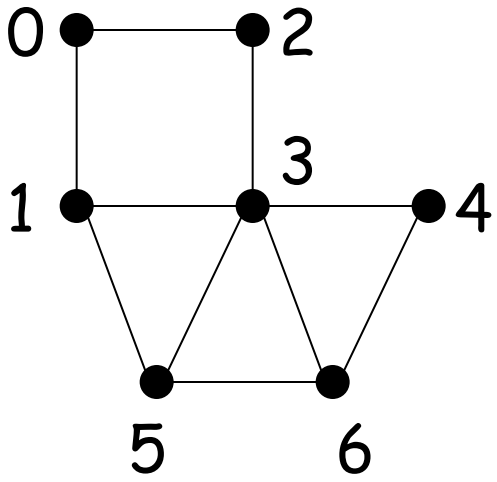
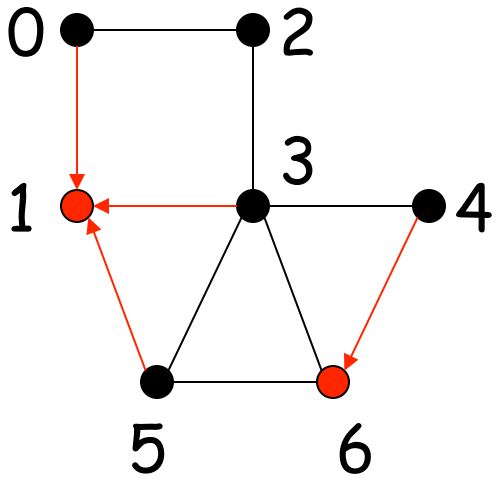
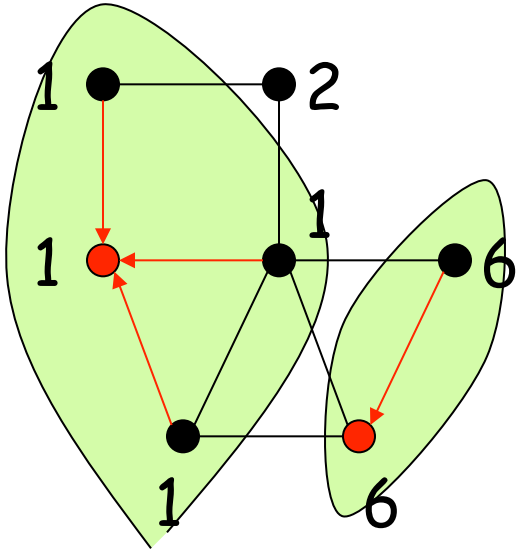
Contraction : Graph Connectivity

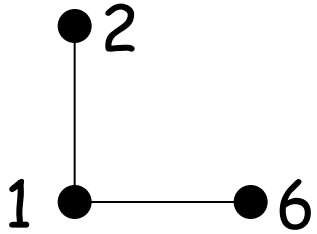




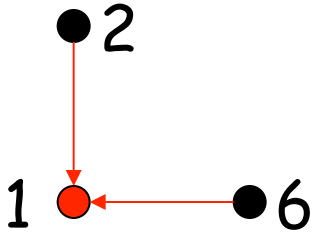
Form stars

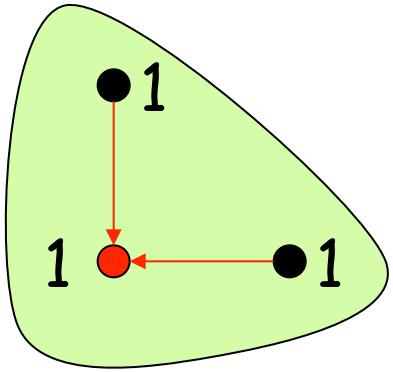


relabel



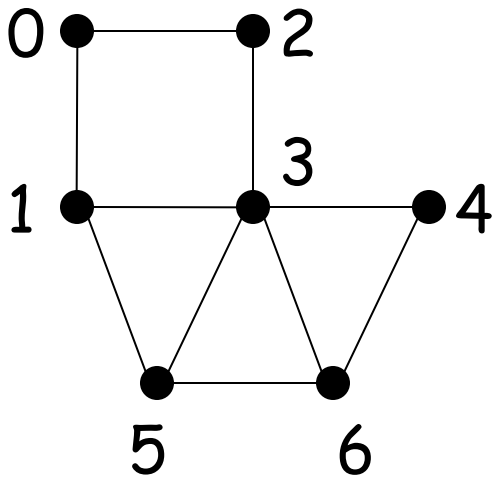
contract





Graph Connectivity

Representing a graph as an edge list:

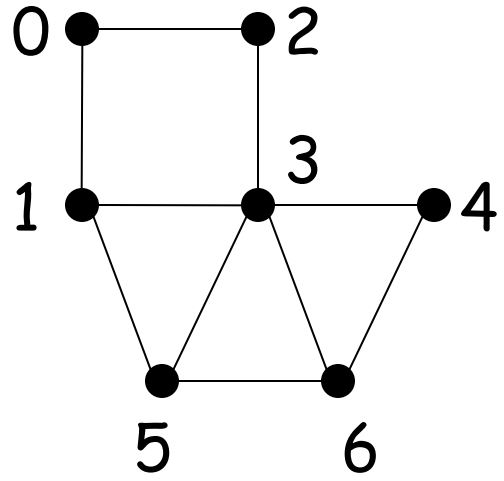



```
E = [(0,1),(0,2),(1,0),(1,3),(1,5),(2,0),(2,3),
     (3,1),(3,2),(3,4),(3,5),(3,6),(4,3),(4,6),
     (5,1),(5,3),(5,6),(6,3),(6,4),(6,5)]
```

Here every edge is represented once in each direction

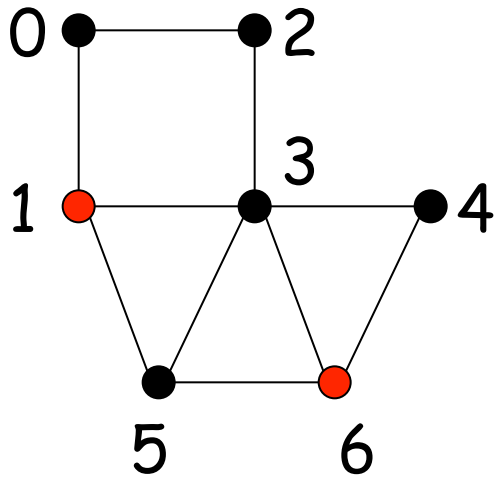
Graph Connectivity

Use an array of pointers, one per vertex to point to parent in connected tree. Initially everyone points to self.




```
L = [0,1,2,3,4,5,6]   (initially)
L = [1,1,1,1,6,1,1]   (possible final)
```

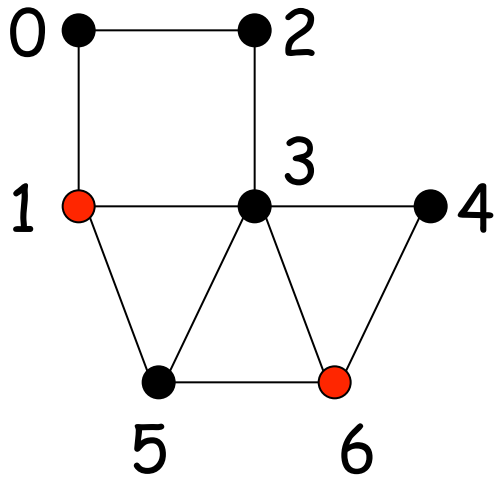
Graph Connectivity



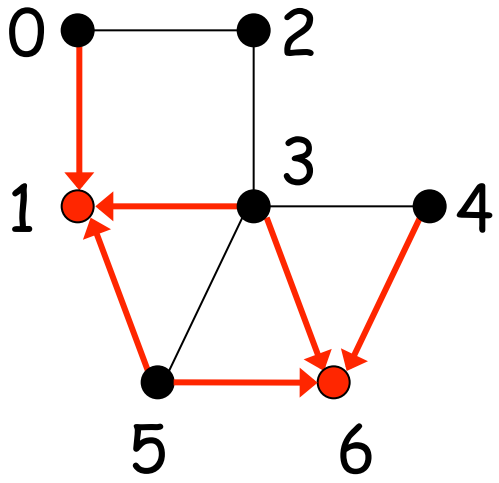
Randomly flip coins

```
FL = {coinToss(.5) : x in [0:#L]};
   FL = [0, 1, 0, 0, 0, 0, 1]
```

Graph Connectivity



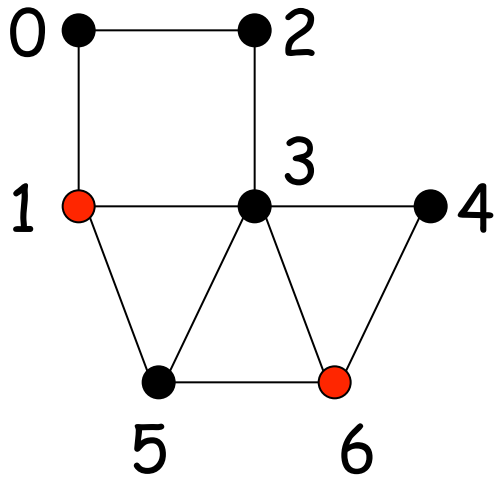
Randomly flip coins



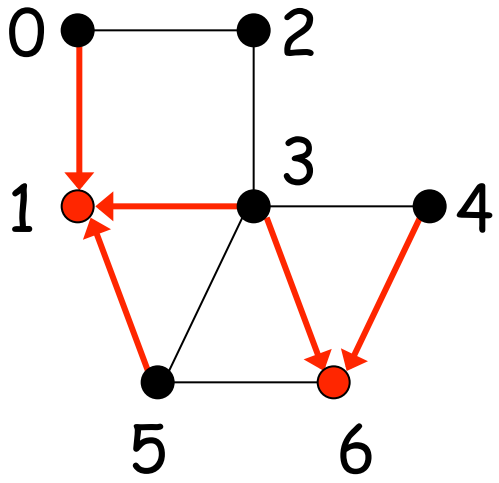
Every edge link
from black to red

```
FL = [0, 1, 0, 0, 0, 0, 1]
H = {(u,v) in E | not(Fl[u]) and Fl[v]}
 H = [(0,1), (3,1), (5,1), (3,6), (4,6), (5,6)]
```

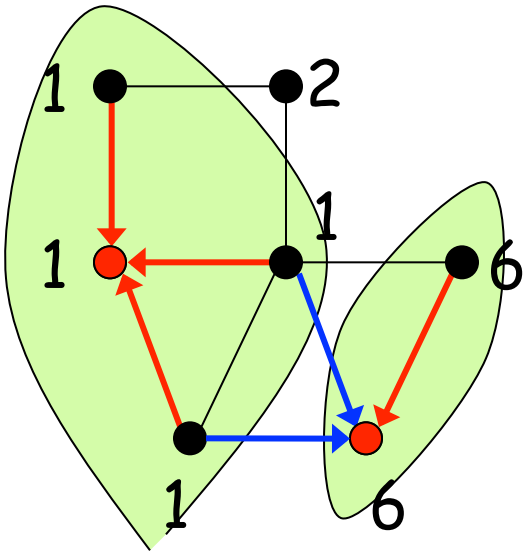
Graph Connectivity



Randomly flip coins



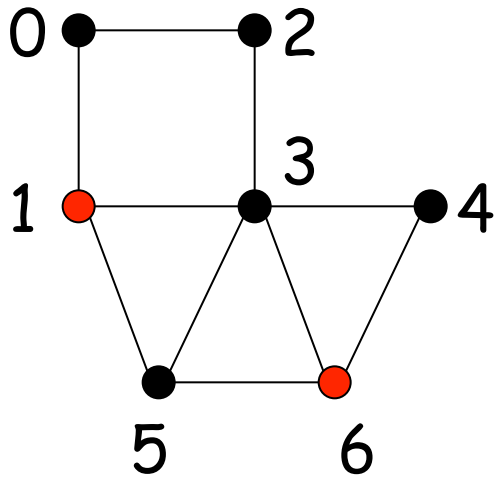
Every edge link from black to red



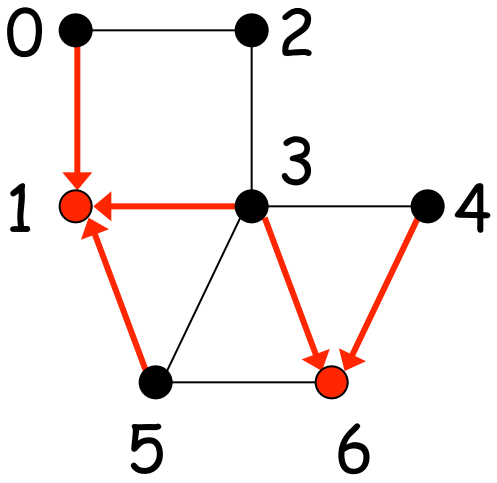
"Hook"

```
H = [(0,1), (3,1), (5,1), (3,6), (4,6), (5,6)]
L = L <- H
    L = [1, 1, 2, 1, 6, 1, 6]
```

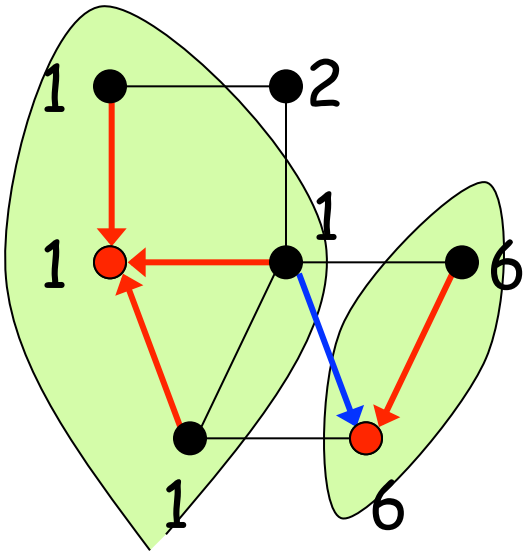
Graph Connectivity



Randomly flip coins



Every edge link from black to red



Relabel edges and remove self edges

```
L = [1, 1, 2, 1, 6, 1, 6]
E = {(L[u],L[v]): (u,v) in E | L[u]\=L[v]}
  E = [(1,2),(2,1),(2,1),(1,2),(1,6),(1,6),
       (6,1),(1,6),(6,1),(6,1)]
```

Graph Connectivity

L = Vertex Labels, E = Edge List

```
function connectivity(L, E) =
if #E = 0 then L
else let
  FL = {coinToss(.5) : x in [0:#L]};
  H = {(u,v) in E | not(Fl[u]) and Fl[v]}
  L = L <- H;
  E = {(L[u],L[v]): (u,v) in E | L[u]\=L[v]};
in connectivity(L,E);
```

D = $O(\log n)$
W = $O(m \log n)$