Lec95 Cascading Behavior in network - Programming Intro ¶

- Increase the Pay off - Screencast 1
- Choose the key people - Screencast 2
- Effect of community on cascades - Screencast 3
- Cluster of density > 1-q => Complete cascade is not possible - Screencast 3

Lec96 Cascading Behavior in network - The Base Code

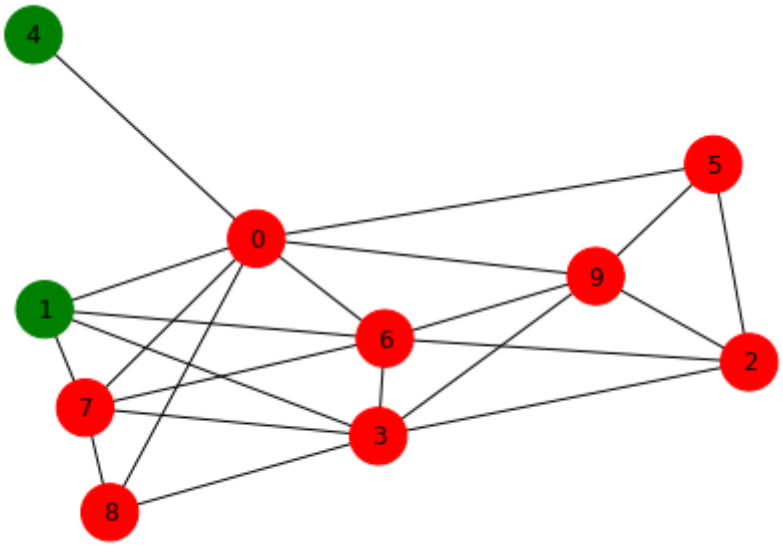
In [11]:

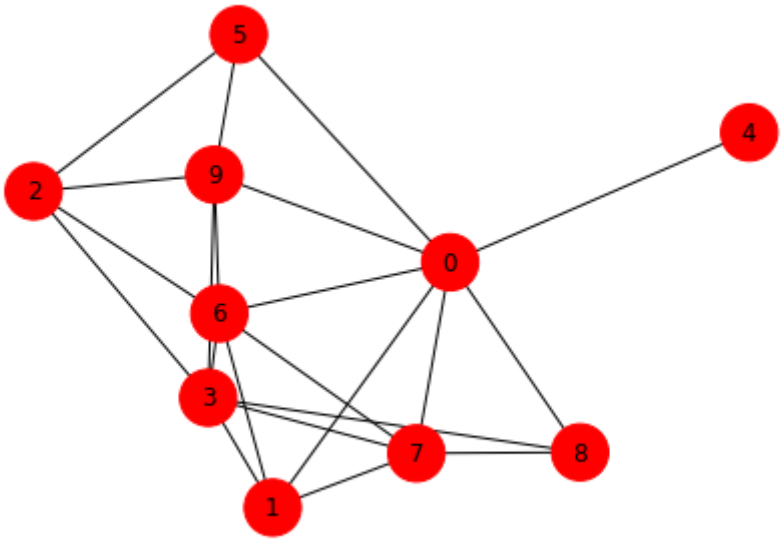
```
import networkx as nx
import matplotlib.pyplot as plt

G = nx.erdos_renyi_graph(10, 0.5)

nx.write_gml(G, 'random_graph.gml')
```

In [12]:

```
# All are doing action B. New Action A comes in
def set_all_B(G):
    for each in G.nodes():
        G.node[each]['action'] = 'B'

def set_A(G, list1):
    for each in list1:
        G.node[each]['action'] = 'A'

def get_colors(G):
    list2 = []
    for each in G.nodes():
        if G.node[each]['action'] == 'B':
            list2.append('red')
        else:
            list2.append('green')
    return list2

def find_neigh(each, c, G):
    num = 0
    for each2 in G.neighbors(each):
        if  G.node[each2]['action'] == c:
            num += 1
    return num

def recalculate_options(G):
    dict1 = {}
    # Payoff(A) = a = 4
    # Payoff(B) = b = 3
    a = 6
    b = 5
    for each in G.nodes():
        num_A = find_neigh(each, 'A', G)
        num_B = find_neigh(each, 'B', G)
        payoff_A = a*num_A
        payoff_B = b*num_B
        if payoff_A >= payoff_B:
            dict1[each] = 'A'
        else:
            dict1[each] = 'B'
    return dict1

G = nx.read_gml('random_graph.gml')

def reset_node_attributes(G, action_dict):
    for each in action_dict:
        G.node[each]['action'] = action_dict[each]

set_all_B(G)
list1 = [4, 1]
set_A(G, list1)

colors = get_colors(G)
nx.draw(G, with_labels=1, node_color= colors, node_size=800)
plt.show()

action_dict = recalculate_options(G)
reset_node_attributes(G, action_dict)
```

```
colors = get_colors(G)

nx.draw(G, with_labels=1, node_color= colors, node_size=800)
plt.show()
```



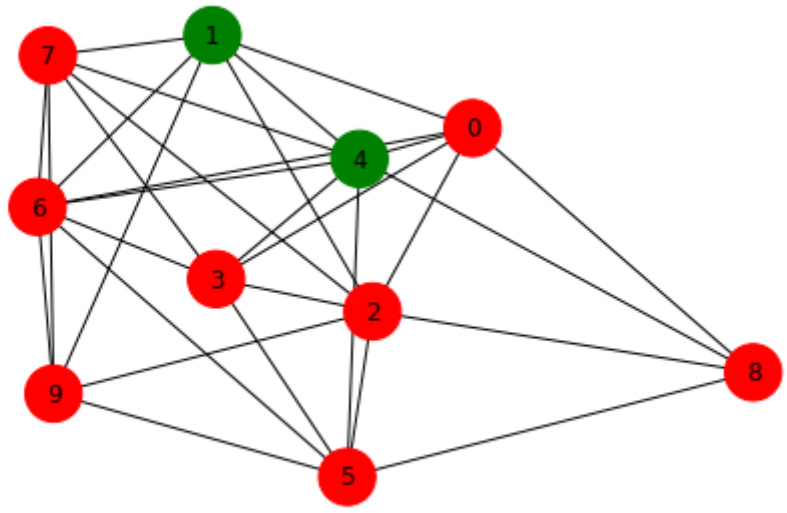


Lec97 Cascading Behavior in network - Increasing the pay off

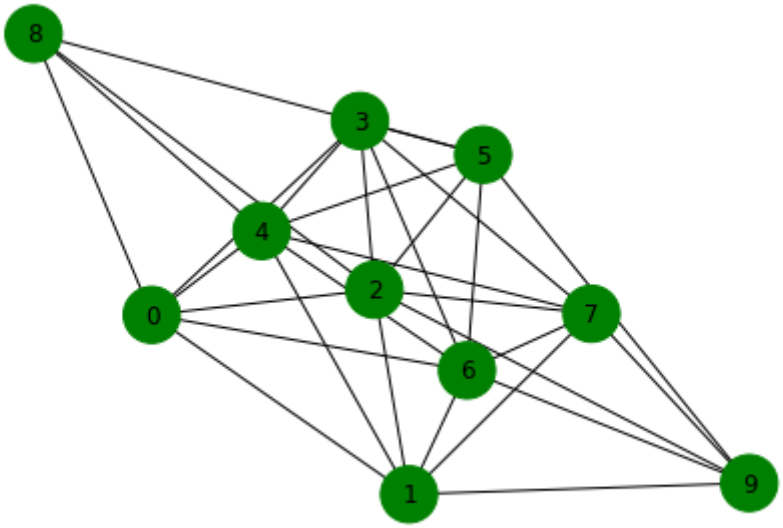
In [3]:

```
# All are doing action B. New Action A comes in
def set_all_B(G):
    for each in G.nodes():
        G.node[each]['action'] = 'B'

def set_A(G, list1):
    for each in list1:
        G.node[each]['action'] = 'A'

def get_colors(G):
    list2 = []
    for each in G.nodes():
        if G.node[each]['action'] == 'B':
            list2.append('red')
        else:
            list2.append('green')
    return list2

def find_neigh(each, c, G):
    num = 0
    for each2 in G.neighbors(each):
        if  G.node[each2]['action'] == c:
            num += 1
    return num

def recalculate_options(G):
    dict1 = {}
    # Payoff(A) = a = 4
    # Payoff(B) = b = 3
    a = 15
    b = 5
    for each in G.nodes():
        num_A = find_neigh(each, 'A', G)
        num_B = find_neigh(each, 'B', G)
        payoff_A = a*num_A
        payoff_B = b*num_B
        if payoff_A >= payoff_B:
            dict1[each] = 'A'
        else:
            dict1[each] = 'B'
    return dict1



def reset_node_attributes(G, action_dict):
    for each in action_dict:
        G.node[each]['action'] = action_dict[each]

def terminate_1(c, G):
    f = 1
    for each in G.nodes():
        if G.node[each]['action'] != c:
            f = 0
            break
    return f

def terminate(G, count):
    flag1 = terminate_1('A', G)
    flag2 = terminate_1('B', G)
```

```
    if flag1 == 1 or flag2 == 1 or count >= 100:

        return 1
    else:
        return 0

G = nx.read_gml('random_graph.gml')


set_all_B(G)
list1 = [4, 1]
set_A(G, list1)

colors = get_colors(G)
nx.draw(G, with_labels=1, node_color= colors, node_size=800)
plt.show()

flag = 0
count = 0
while(1):
    #raw_input("Enter any text to continue..")
    flag = terminate(G, count)
    if flag == 1:
        break
    count += 1
    action_dict = recalculate_options(G)
    reset_node_attributes(G, action_dict)

    #colors = get_colors(G)
    #nx.draw(G, with_labels=1, node_color= colors, node_size=800)
    #plt.show()

c = terminate_1('A', G)
if c == 1:
    print "Complete Cascade"
else:
     print "Incomplete Cascade"
colors = get_colors(G)
nx.draw(G, with_labels=1, node_color= colors, node_size=800)
plt.show()
```



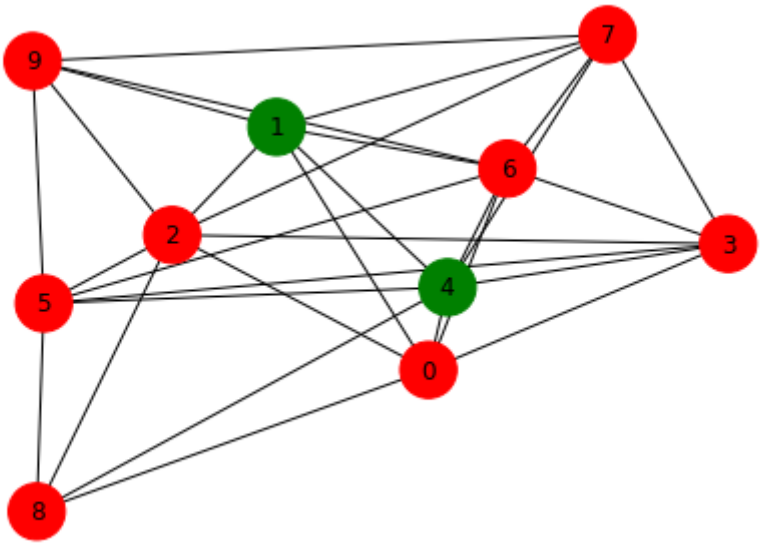

```
Complete Cascade
```


8
3
5
4
2
0
7
6
1
9

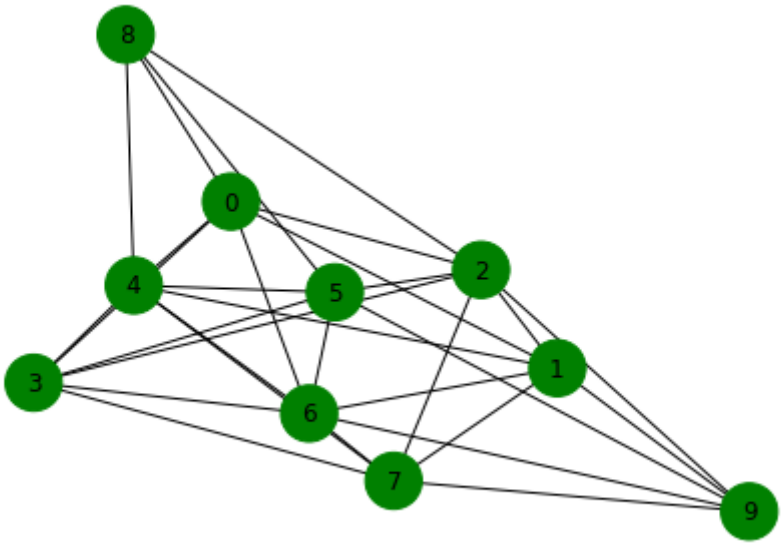
In [4]:

```
# All are doing action B. New Action A comes in
def set_all_B(G):
    for each in G.nodes():
        G.node[each]['action'] = 'B'

def set_A(G, list1):
    for each in list1:
        G.node[each]['action'] = 'A'

def get_colors(G):
    list2 = []
    for each in G.nodes():
        if G.node[each]['action'] == 'B':
            list2.append('red')
        else:
            list2.append('green')
    return list2

def find_neigh(each, c, G):
    num = 0
    for each2 in G.neighbors(each):
        if  G.node[each2]['action'] == c:
            num += 1
    return num

def recalculate_options(G):
    dict1 = {}
    # Payoff(A) = a = 4
    # Payoff(B) = b = 3
    a = 13
    b = 5
    for each in G.nodes():
        num_A = find_neigh(each, 'A', G)
        num_B = find_neigh(each, 'B', G)
        payoff_A = a*num_A
        payoff_B = b*num_B
        if payoff_A >= payoff_B:
            dict1[each] = 'A'
        else:
            dict1[each] = 'B'
    return dict1

G = nx.read_gml('random_graph.gml')

def reset_node_attributes(G, action_dict):
    for each in action_dict:
        G.node[each]['action'] = action_dict[each]

def terminate_1(c, G):
    f = 1
    for each in G.nodes():
        if G.node[each]['action'] != c:
            f = 0
            break
    return f

def terminate(G, count):
    flag1 = terminate_1('A', G)
    flag2 = terminate_1('B', G)
```

```
    if flag1 == 1 or flag2 == 1 or count >= 100:

        return 1
    else:
        return 0


set_all_B(G)
list1 = [4, 1]
set_A(G, list1)

colors = get_colors(G)
nx.draw(G, with_labels=1, node_color= colors, node_size=800)
plt.show()

flag = 0
count = 0
while(1):
    #raw_input("Enter any text to continue..")
    flag = terminate(G, count)
    if flag == 1:
        break
    count += 1
    action_dict = recalculate_options(G)
    reset_node_attributes(G, action_dict)

    #colors = get_colors(G)
    #nx.draw(G, with_labels=1, node_color= colors, node_size=800)
    #plt.show()

c = terminate_1('A', G)
if c == 1:
    print "Complete Cascade"
else:
     print "Incomplete Cascade"
colors = get_colors(G)
nx.draw(G, with_labels=1, node_color= colors, node_size=800)
plt.show()
```




```
Complete Cascade
```



Lec98 Cascading Behavior in network - Key People

Note : Don't run the last two steps to view this result

In [13]:

```
# All are doing action B. New Action A comes in
def set_all_B(G):
    for each in G.nodes():
        G.node[each]['action'] = 'B'

def set_A(G, list1):
    for each in list1:
        G.node[each]['action'] = 'A'

def get_colors(G):
    list2 = []
    for each in G.nodes():
        if G.node[each]['action'] == 'B':
            list2.append('red')
        else:
            list2.append('green')
    return list2

def find_neigh(each, c, G):
    num = 0
    for each2 in G.neighbors(each):
        if  G.node[each2]['action'] == c:
            num += 1
    return num

def recalculate_options(G):
    dict1 = {}
    # Payoff(A) = a = 4
    # Payoff(B) = b = 3
    a = 19
    b = 5
    for each in G.nodes():
        num_A = find_neigh(each, 'A', G)
        num_B = find_neigh(each, 'B', G)
        payoff_A = a*num_A
        payoff_B = b*num_B
        if payoff_A >= payoff_B:
            dict1[each] = 'A'
        else:
            dict1[each] = 'B'
    return dict1

def reset_node_attributes(G, action_dict):
    for each in action_dict:
        G.node[each]['action'] = action_dict[each]

def terminate_1(c, G):
    f = 1
    for each in G.nodes():
        if G.node[each]['action'] != c:
            f = 0
            break
    return f

def terminate(G, count):
    flag1 = terminate_1('A', G)
    flag2 = terminate_1('B', G)
    if flag1 == 1 or flag2 == 1 or count >= 100:
        return 1
```

```
    else:

        return 0

G = nx.read_gml('random_graph.gml')

for u in G.nodes():
    for v in G.nodes():
        if u<v:
            print u, v, ' : ',
            list1 = []
            list1.append(v)
            list1.append(v)

            set_all_B(G)
            set_A(G, list1)

            #colors = get_colors(G)
            #nx.draw(G, with_labels=1, node_color= colors, node_size=800)
            #plt.show()

            flag = 0
            count = 0
            while(1):
                #raw_input("Enter any text to continue..")
                flag = terminate(G, count)
                if flag == 1:
                    break
                count += 1
                action_dict = recalculate_options(G)
                reset_node_attributes(G, action_dict)

                #colors = get_colors(G)
                #nx.draw(G, with_labels=1, node_color= colors, node_size=800)
                #plt.show()

            c = terminate_1('A', G)
            if c == 1:
                print "Complete Cascade"
            else:
                 print "Incomplete Cascade"
            #colors = get_colors(G)
            #nx.draw(G, with_labels=1, node_color= colors, node_size=800)
            #plt.show()
```

```
0 1  :  Incomplete Cascade
0 2  :  Incomplete Cascade
0 3  :  Complete Cascade
0 4  :  Incomplete Cascade
0 5  :  Incomplete Cascade
0 6  :  Complete Cascade
0 7  :  Complete Cascade
0 8  :  Incomplete Cascade
0 9  :  Complete Cascade
1 2  :  Incomplete Cascade
1 3  :  Complete Cascade
1 4  :  Incomplete Cascade
1 5  :  Incomplete Cascade
1 6  :  Complete Cascade
1 7  :  Complete Cascade
1 8  :  Incomplete Cascade
1 9  :  Complete Cascade
```

2 3 : Complete Cascade

2 4 : Incomplete Cascade
2 5 : Incomplete Cascade
2 6 : Complete Cascade
2 7 : Complete Cascade
2 8 : Incomplete Cascade
2 9 : Complete Cascade
3 4 : Incomplete Cascade
3 5 : Incomplete Cascade
3 6 : Complete Cascade
3 7 : Complete Cascade
3 8 : Incomplete Cascade
3 9 : Complete Cascade
4 5 : Incomplete Cascade
4 6 : Complete Cascade
4 7 : Complete Cascade
4 8 : Incomplete Cascade
4 9 : Complete Cascade
5 6 : Complete Cascade
5 7 : Complete Cascade
5 8 : Incomplete Cascade
5 9 : Complete Cascade
6 7 : Complete Cascade
6 8 : Incomplete Cascade
6 9 : Complete Cascade
7 8 : Incomplete Cascade
7 9 : Complete Cascade
8 9 : Complete Cascade