Networks: structure, evolution & processes

Internet Analytics - Lab 2

In [1]:

```
import networkx as nx
from networkx.readwrite import json_graph
import json
import epidemics_helper
import numpy as np
import matplotlib.pyplot as plt
import random
```

In [2]:

```
def read_json_file(filename):
    with open(filename) as f:
        js_graph = json.load(f)
    return json_graph.node_link_graph(js_graph)
```

2.3 Epdemics

Exercise 2.9: Simulate an epidemic outbreak

In [3]:

```
# ... WRITE YOUR CODE HERE...
G = read_json_file('../data/nyc_augmented_network.json')
```

In [4]:

```
sir = epidemics_helper.SimulationSIR(G, beta=10.0, gamma=0.1)
```

In [5]:

```
node_source = 23654
sir.launch_epidemic(source=node_source, max_time=100.0)
```

```
Epidemic stopped after 100.94 days | 100.94 days elapsed | 0.1% susceptible, 0.0% in
fected, 99.9% recovered
```

In [6]:

```
sir.rec_time
```

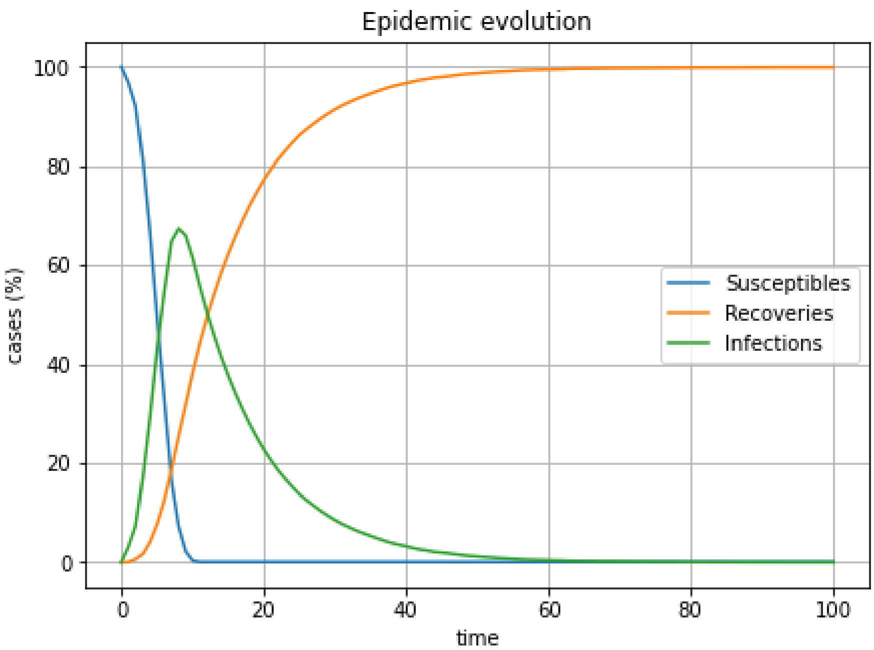
Out[6]:

```
array([24.76681827, 11.63935766,  9.71325497, ..., 24.7443039 ,
        9.88141307,  8.13479434])
```

In [7]:

```
def get_stats(G,sir):
    infections = np.zeros(101)
    recoveries = np.zeros(101)

    for t in range(0,101):
        for n in range(0,len(G.nodes)):
            rec = int(sir.rec_time[n] < t)
            recoveries[t] += rec
            infections[t] += -rec
            infections[t] += int(sir.inf_time[n] < t)

    susceptibles = len(G.nodes) - infections - recoveries
    return infections, recoveries, susceptibles
```

In [8]:
```
infections, recoveries, susceptibles = get_stats(G,sir)
x = range(0,101)

plt.figure('figure1',figsize=(7,5))
plt.plot(x, susceptibles / len(G.nodes) * 100, label = "Susceptibles")
plt.plot(x, recoveries / len(G.nodes) * 100, label = "Recoveries")
plt.plot(x, infections / len(G.nodes) * 100, label = "Infections")
plt.legend()
plt.xlabel('time')
plt.ylabel('cases (%)')
plt.grid()
plt.title('Epidemic evolution')
plt.show()
```



In [9]:
```
inf_60 = round(0.6*len(G.nodes))
inf_60
```

Out[9]: 15889

In [10]:
```
def calc_60(infections,recoveries):
    t_inf, t_rec = -1, -1
    for t in range(0,101):
        if infections[t] >= inf_60 and t_inf == -1:
            t_inf = t
        if recoveries[t] >= inf_60 and t_rec == -1:
            t_rec = t
    return t_inf, t_rec
```

In [11]:
```
t_inf, t_rec = calc_60(infections,recoveries)
print(f'The 60% of people will be infected on day = {t_inf}')
print(f'The 60% of people will be recovered on day = {t_rec}')
```

```
The 60% of people will be infected on day = 7
The 60% of people will be recovered on day = 15
```

In [12]:
```
dict_coord = {i:G.nodes[i]['coordinates'] for i in range(len(G.nodes))}
```

In [13]:

```
def select_color(node,t):
    if sir.rec_time[node] < t:
        return 'green'
    if sir.inf_time[node] < t:
        return 'red'
    else:
        return 'yellow'

plt.figure('figure1',figsize=(10,10))
color_map = []
for i in range(0,len(G.nodes)):
    color_map.append(select_color(i,1))

nx.draw_networkx(G,pos=dict_coord, node_color=color_map, with_labels=False, node_siz
plt.title('Epidemics Graph at Day 1')
plt.axis('off')
```

Out[13]: (-74048875.915, -73831169.085, 40623760.05, 40926247.95)

Epidemics Graph at Day 1

At day 1, there are almost all susceptibles (in yellow color) and we can observe how the epidemic is expanding between the neighbours of the source node (red colour).

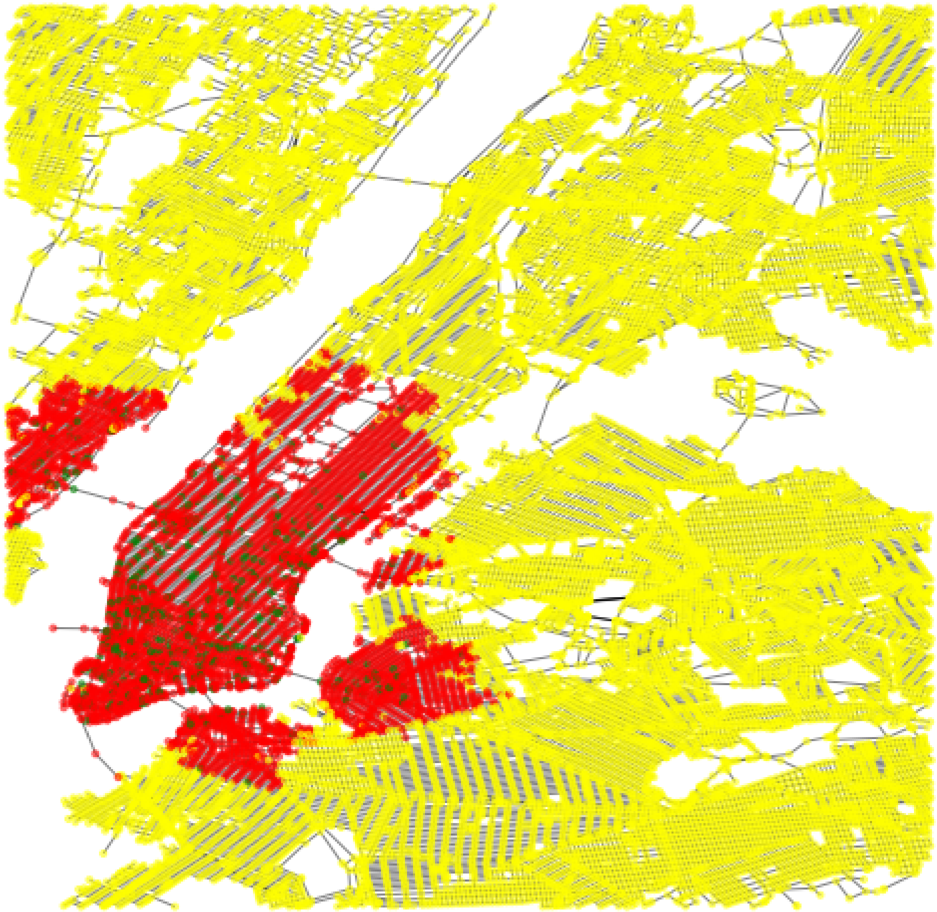
In [14]:

```
plt.figure('figure1',figsize=(10,10))
color_map = []
for i in range(0,len(G.nodes)):
    color_map.append(select_color(i,3))
```

```
nx.draw_networkx(G,pos=dict_coord, node_color=color_map, with_labels=False, node_siz
plt.title('Epidemics Graph at Day 3')
plt.axis('off')
```

Out[14]: (-74048875.915, -73831169.085, 40623760.05, 40926247.95)

Epidemics Graph at Day 3



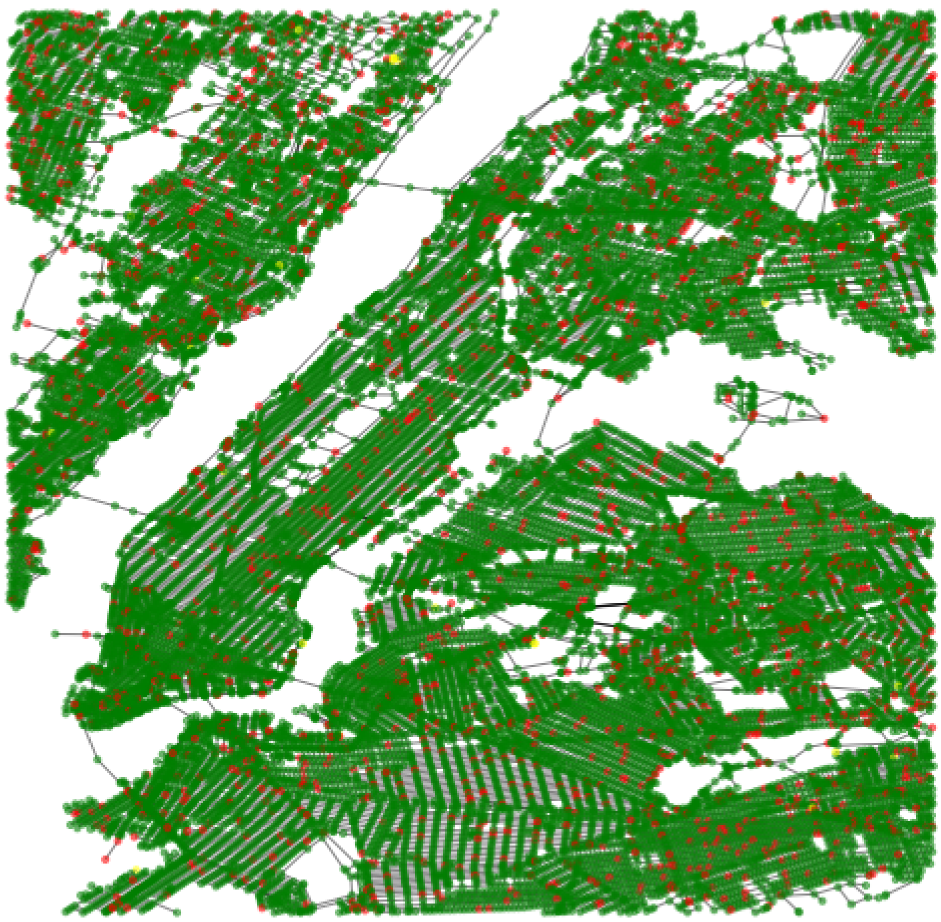
At day 3, the epidemic has started to infect the bridges and therefore other clusters. Some nodes have started to recover (in green).

In [15]:

```
plt.figure('figure1',figsize=(10,10))
color_map = []
for i in range(0,len(G.nodes)):
    color_map.append(select_color(i,30))

nx.draw_networkx(G,pos=dict_coord, node_color=color_map, with_labels=False, node_siz
plt.title('Epidemics Graph at Day 30')
plt.axis('off')
```

Out[15]: (-74048875.915, -73831169.085, 40623760.05, 40926247.95)

Epidemics Graph at Day 30



At day 30, the epidemic has infected almost every node and the majority of them have already recovered from it.

2.3.1 Stop the apocalypse!

Exercise 2.10: Strategy 1

In [16]:
```
# remove edges
def remove_rand_edges(graph,n):
    for i in range(n):
        edges = list(graph.edges)
        chosen_edge = random.choice(edges)
        graph.remove_edge(chosen_edge[0],chosen_edge[1])
    return graph
```

In [17]:
```
G = read_json_file('../data/nyc_augmented_network.json')
sir = epidemics_helper.SimulationSIR(remove_rand_edges(G,1000), beta=10.0, gamma=0.1
```

In [18]:
```
def calc_cv(m):
    mean_rec = np.mean(m)
    std_rec = np.std(m)
```

```
    print(std_rec/mean_rec)
    return std_rec/mean_rec >= 1.0
```

In [19]:

```
def mean_sim(G):

    inf = []
    rec = []
    sus = []
    means = []

    i = 1

    while i<7 or calc_cv(means):
        i += 1
        np.random.seed(i)
        node_source = np.random.randint(0,len(G.nodes))
        sir.launch_epidemic(source=node_source, max_time=100.0)
        infections, recoveries, susceptibles = get_stats(G,sir)
        inf.append(infections)
        rec.append(recoveries)
        sus.append(susceptibles)
        means.append(recoveries[len(recoveries)-1])

    inf_mean = np.array(inf).mean(axis=0)
    rec_mean = np.array(rec).mean(axis=0)
    sus_mean = np.array(sus).mean(axis=0)

    return inf_mean,rec_mean,sus_mean
```

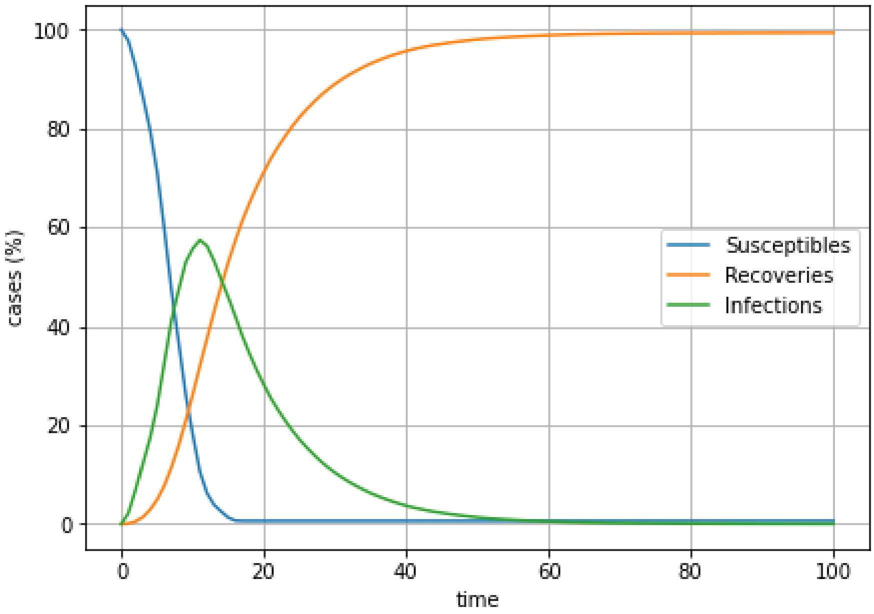
Here we compute the mean of the infections, recoveries and susceptibles in each iteration for each day of the epidemic.
Then, we have decided to fix the number of iterations based on the Coefficient of Variation (CV) of the recoveries at one day (we chose the last day). When CV >= 1, indicates a relatively high variation of the recoveries, while a CV < 1 indicates a low variation. So when we reach CV < 1 we can say that the estimation is accurate.
We have decided to fix a minimum of 7 iterations.

In [20]:

```
inf_mean,rec_mean,sus_mean = mean_sim(G)
print()
print(f'On average the amount of people healthy people on day 30 is: {sus_mean[30]}'
print(f'On average the amount of people infected people on day 30 is: {inf_mean[30]}
print(f'On average the amount of people recovered people on day 30 is: {rec_mean[30]
```

```
Epidemic stopped after 104.64 days | 104.64 days elapsed | 0.6% susceptible, 0.0% in
fected, 99.4% recovered
Epidemic stopped after 101.67 days | 101.67 days elapsed | 0.6% susceptible, 0.0% in
fected, 99.4% recovered
Epidemic stopped after 103.28 days | 103.28 days elapsed | 0.6% susceptible, 0.0% in
fected, 99.4% recovered
Epidemic stopped after 107.08 days | 107.08 days elapsed | 0.7% susceptible, 0.0% in
fected, 99.3% recovered
Epidemic stopped after 136.44 days | 136.44 days elapsed | 0.6% susceptible, 0.0% in
fected, 99.4% recovered
Epidemic stopped after 102.92 days | 102.92 days elapsed | 0.6% susceptible, 0.0% in
fected, 99.4% recovered
0.00038655007618337026

On average the amount of people healthy people on day 30 is: 163.66666666666666
On average the amount of people infected people on day 30 is: 2754.0
On average the amount of people recovered people on day 30 is: 23563.333333333332
```

In [21]:
```
x = range(0,101)

plt.figure('figure1',figsize=(7,5))
plt.plot(x, sus_mean / len(G.nodes) * 100, label = "Susceptibles")
plt.plot(x, rec_mean / len(G.nodes) * 100, label = "Recoveries")
plt.plot(x, inf_mean / len(G.nodes) * 100, label = "Infections")
plt.legend()
plt.xlabel('time')
plt.ylabel('cases (%)')
plt.grid()
plt.show()
```



In [22]:
```
t_inf, t_rec = calc_60(inf_mean,rec_mean)
print(f'The 60% of people will be infected at time = {t_inf}')
print(f'The 60% of people will be recovered at time = {t_rec}')
```

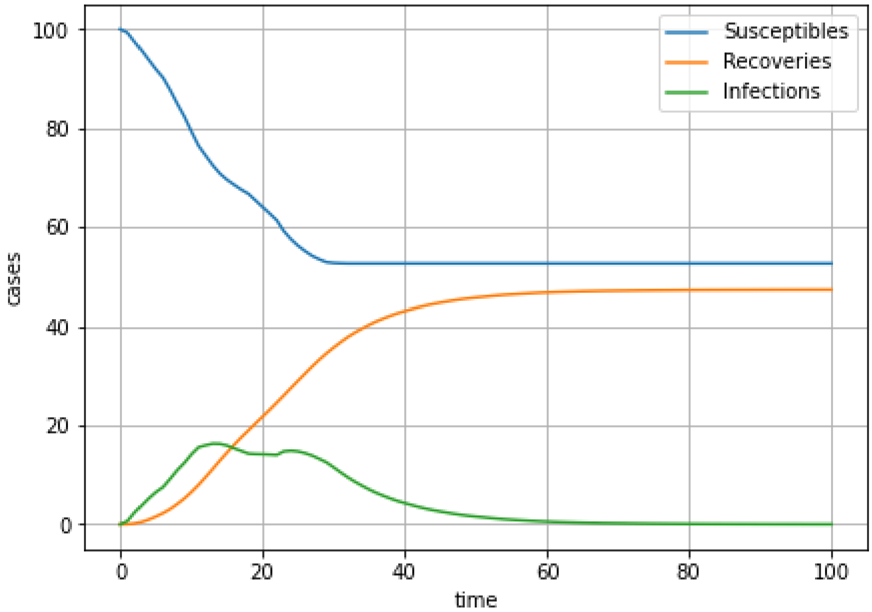
```
The 60% of people will be infected at time = -1
The 60% of people will be recovered at time = 17
```

This strategy has decreased the number of people that gets infected (~3000 people less) by the COVID-19 because the number of edges between the nodes (people) have decreased.

In [23]:
```
# n = 10000
G = read_json_file('../data/nyc_augmented_network.json')
sir = epidemics_helper.SimulationSIR(remove_rand_edges(G,10000), beta=10.0, gamma=0.

inf_mean,rec_mean,sus_mean = mean_sim(G)

x = range(0,101)

plt.figure('figure1',figsize=(7,5))
plt.plot(x, sus_mean / len(G.nodes) * 100, label = "Susceptibles")
plt.plot(x, rec_mean / len(G.nodes) * 100, label = "Recoveries")
plt.plot(x, inf_mean / len(G.nodes) * 100, label = "Infections")
plt.legend()
plt.xlabel('time')
plt.ylabel('cases')
plt.grid()
plt.show()
```

```
Epidemic stopped after 108.59 days | 108.59 days elapsed | 24.9% susceptible, 0.0% i
nfected, 75.1% recovered
```

```
Epidemic stopped after 101.05 days | 101.05 days elapsed | 24.9% susceptible, 0.0% i
nfected, 75.0% recovered
Epidemic stopped after 100.69 days | 100.69 days elapsed | 57.4% susceptible, 0.0% i
nfected, 42.6% recovered
Epidemic stopped after 102.68 days | 102.68 days elapsed | 24.7% susceptible, 0.0% i
nfected, 75.3% recovered
Epidemic stopped after 105.17 days | 105.17 days elapsed | 83.9% susceptible, 0.0% i
nfected, 16.1% recovered
Epidemic stopped after 0.16 days | 0.16 days elapsed | 100.0% susceptible, 0.0% infe
cted, 0.0% recovered
0.6427662477726126
```



On average there is a considerable decreasing on the number of infected people, but it depends a lot on which is the source node (the first infected person), if the nodes is part of a big cluster of nodes or if it is in a small one, in the first case there will be more infections than in the second case, or also if it has been affected for many removed edges.

Exercise 2.11: Strategy 2

In [25]:

```python
def remove_bridge_edges(graph,n):
    bridges = nx.local_bridges(graph)
    bridge_sorted = sorted(bridges, key=lambda tup: tup[2])[-n:]
    print(len(bridge_sorted))
    for b in bridge_sorted:
        graph.remove_edge(b[0],b[1])
    return graph
```

In [26]:

```python
G = read_json_file('../data/nyc_augmented_network.json')
sir = epidemics_helper.SimulationSIR(remove_bridge_edges(G,2500), beta=10.0, gamma=0
```

```
2500
```

In [27]:

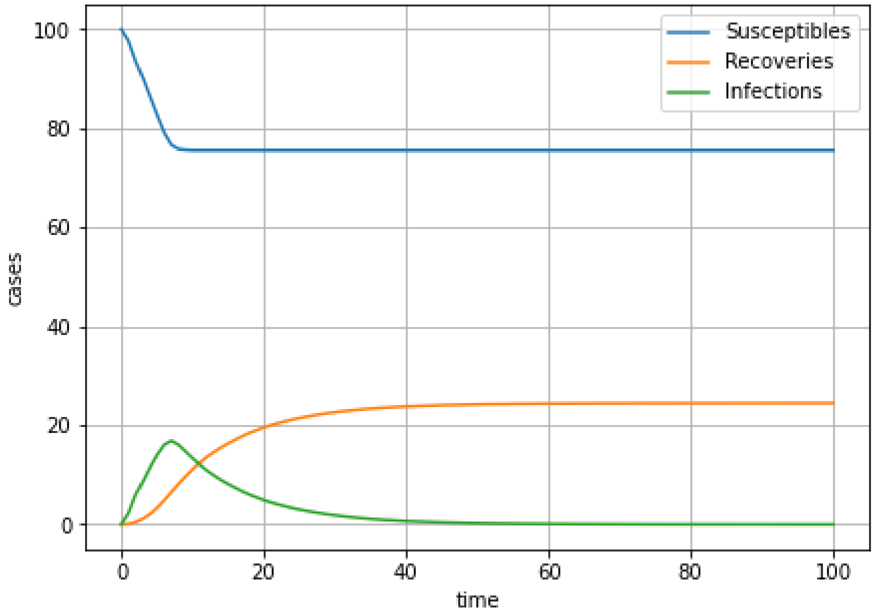
```python
inf_mean,rec_mean,sus_mean = mean_sim(G)
```

```
Epidemic stopped after 113.56 days | 113.56 days elapsed | 83.6% susceptible, 0.0% i
nfected, 16.4% recovered
Epidemic stopped after 79.65 days | 79.65 days elapsed | 83.8% susceptible, 0.0% inf
ected, 16.2% recovered
Epidemic stopped after 93.79 days | 93.79 days elapsed | 58.0% susceptible, 0.0% inf
ected, 42.0% recovered
Epidemic stopped after 78.55 days | 78.55 days elapsed | 83.8% susceptible, 0.0% inf
ected, 16.2% recovered
```

```
Epidemic stopped after 98.26 days | 98.26 days elapsed | 86.2% susceptible, 0.0% inf
ected, 13.8% recovered
Epidemic stopped after 103.69 days | 103.69 days elapsed | 58.0% susceptible, 0.0% i
nfected, 42.0% recovered
0.5096256227820168
```

In [28]:

```
x = range(0,101)

plt.figure('figure1',figsize=(7,5))
plt.plot(x, sus_mean / len(G.nodes) * 100, label = "Susceptibles")
plt.plot(x, rec_mean / len(G.nodes) * 100, label = "Recoveries")
plt.plot(x, inf_mean / len(G.nodes) * 100, label = "Infections")
plt.legend()
plt.xlabel('time')
plt.ylabel('cases')
plt.grid()
plt.show()
```



In [29]:

```
print(f'On average the % of people healthy people on day 30 is: {sus_mean[30] / len(
print(f'On average the % of people infected people on day 30 is: {inf_mean[30] / len
print(f'On average the % of people recovered people on day 30 is: {rec_mean[30] / le
```

```
On average the % of people healthy people on day 30 is: 75.56109411779515
On average the % of people infected people on day 30 is: 1.8566771144090732
On average the % of people recovered people on day 30 is: 22.58222876779578
```

As we can see in this graph, we have managed to reach more than 70% of healthy nodes on average.

Here we are going to plot the evolution of the epidemic in our last simulation (58% of the people stayed healthy).

As we can see, it depends a lot in which giant component is the source node. In this case it is in a giant component with a lot of nodes that it much more connected than the giant components where the source nodes of the other simulations are.

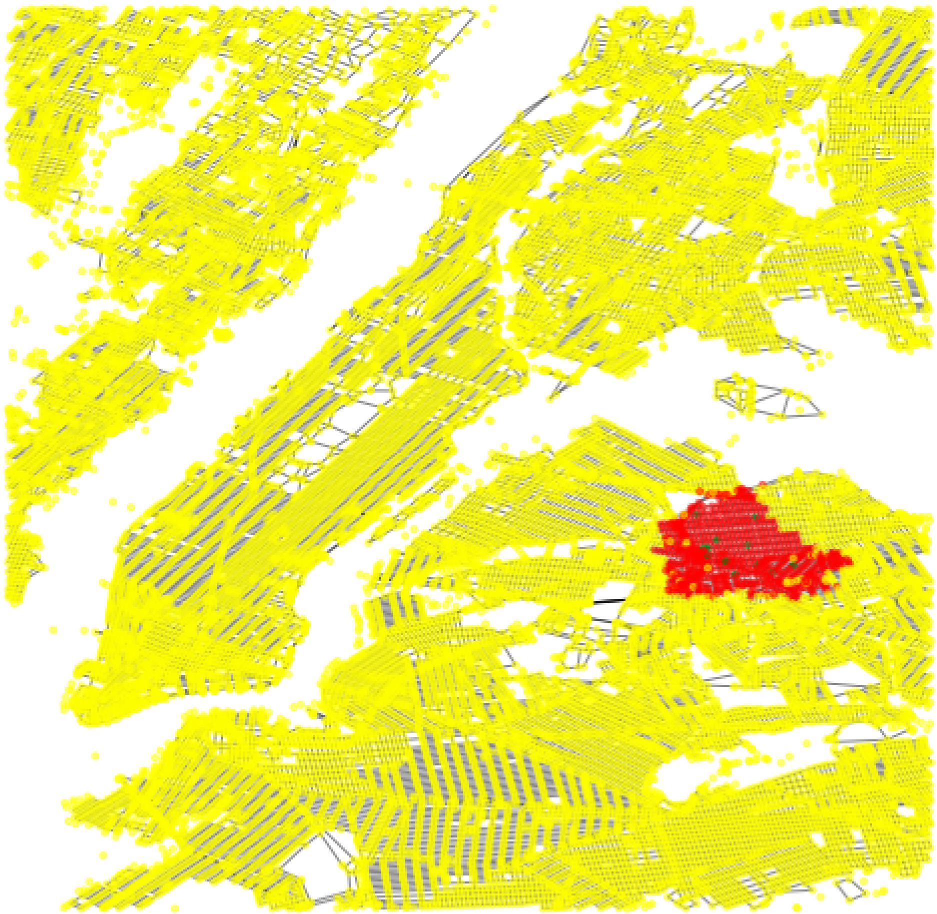
We can also see how the removing of the local bridges has affected the connections between the giant components, isolating from of each other.

In [30]:

```
plt.figure('figure1',figsize=(10,10))
```

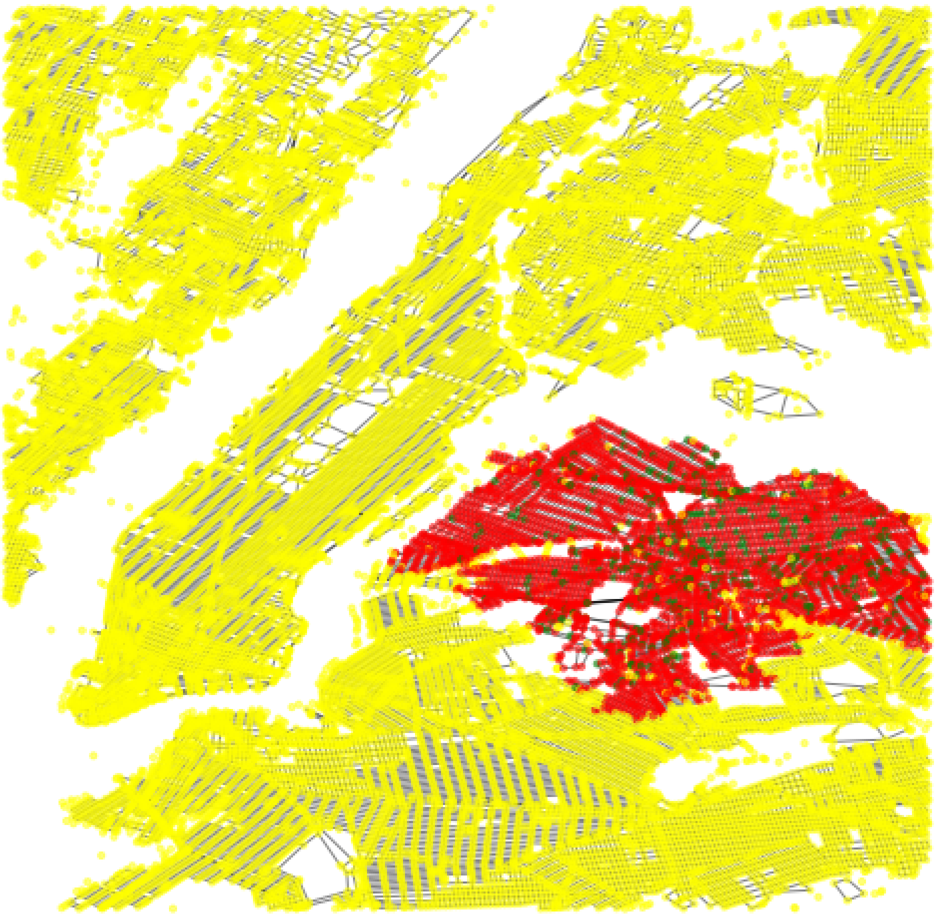
```
color_map = []
for i in range(0,len(G.nodes)):
    color_map.append(select_color(i,1))

nx.draw_networkx(G,pos=dict_coord, node_color=color_map, with_labels=False, node_siz
plt.axis('off')
```

Out[30]: (-74048875.915, -73831169.085, 40623773.31, 40926246.69)



In [31]:
```
plt.figure('figure1',figsize=(10,10))
color_map = []
for i in range(0,len(G.nodes)):
    color_map.append(select_color(i,3))

nx.draw_networkx(G,pos=dict_coord, node_color=color_map, with_labels=False, node_siz
plt.axis('off')
```

Out[31]: (-74048875.915, -73831169.085, 40623773.31, 40926246.69)



In [32]:

```
plt.figure('figure1',figsize=(10,10))
color_map = []
for i in range(0,len(G.nodes)):
    color_map.append(select_color(i,30))

nx.draw_networkx(G,pos=dict_coord, node_color=color_map, with_labels=False, node_siz
plt.axis('off')
```

Out[32]: (-74048875.915, -73831169.085, 40623773.31, 40926246.69)