Griffith UNIVERSITY

Lab 08 – Network Data Analytics

Trimester 1 - 2019

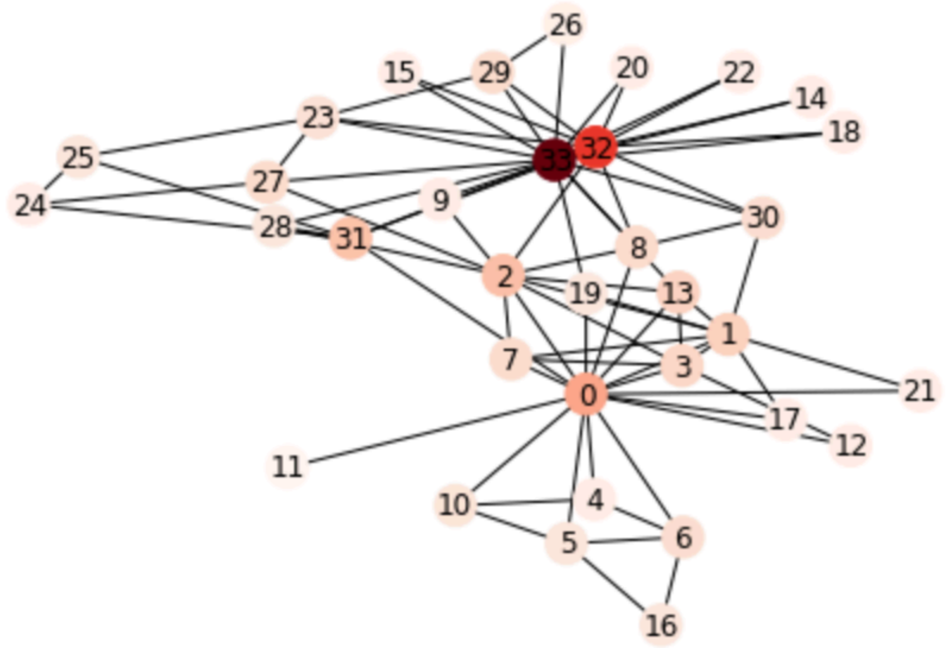
Table of Contents

Complete the code with TODO tag in the Jupyter notebooks.

1. Centrality Analysis

In this exercise, you will implement the pagerank centrality.

```
Name: Zachary's Karate Club
Type: Graph
Number of nodes: 34
Number of edges: 78
Average degree:   4.5882
```

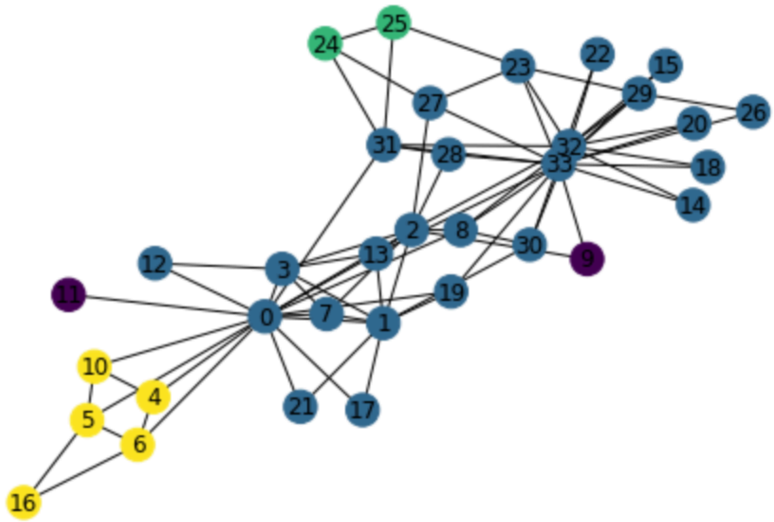


2. Community Analysis [OPTIONAL]

2.1. Clique Percolation Method

One well-known algorithm for detecting overlapping communities is called the Clique Percolation Method (CPM).

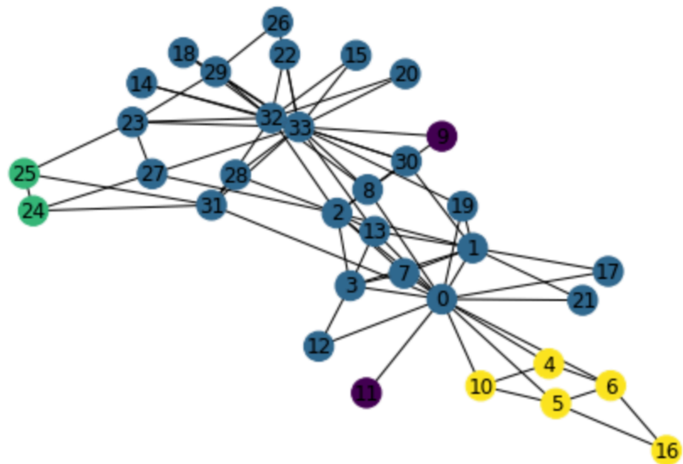
```
Name: Zachary's Karate Club
Type: Graph
Number of nodes: 34
Number of edges: 78
Average degree:   4.5882
---0.000205993652344 seconds---
```



2.2.Efficient Implementation

That implementation is correct but expensive---it requires O(N^2) clique comparisons, where N is the number of cliques (which is often much larger than the number of nodes!). If we use a python dictionary to index which nodes belong to which cliques, then we can easily compare only those cliques that share at least one node in common. This implementation is a bit longer but should be more efficient.

```
---0.0001540184021 seconds---
```



2.3. Test with large dataset

Now we test with a real large-scale network data at https://snap.stanford.edu/data/com-Amazon.html

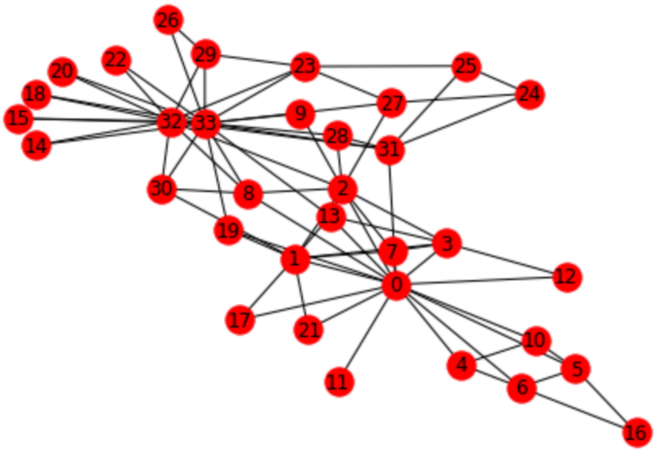
```
Name:
Type: Graph
Number of nodes: 334863
Number of edges: 925872
Average degree:   5.5299
---0.0001220703125 seconds---
---0.000126123428345 seconds---
```

3. Information Diffusion

It is also known as graph activation process, e.g. http://ncase.me/crowds/

Further readings:

- https://stackoverflow.com/questions/31815454/animate-graph-diffusion-with-networkx
- https://stackoverflow.com/questions/27475211/animating-a-network-graph-to-show-the-progress-of-an-algorithm/



3.1. Diffusion process

Now we implement the diffusion process. Each active node will cause other nodes in the graph to become active over time. The diffusion rule is that a node gets active if at least a certain percentage of its neighbours become active. The process continues until convergence (i.e. has no new node activated).

OPTIONAL: Can you implement a data visualization to illustrate the diffusion process?

```
diffusion(G, {0,1})
```

```
{0, 1, 2, 3, 7, 9, 11, 12, 13, 17, 19, 21}
```

3.2. Influence maximization

Now we find a minimal set of seeds that maximize the influence (i.e. the number of active nodes). The influence maximization problem is NP-hard in general. Here, we use a greedy algorithm that iteratively chooses a seed such that the gain of influence is maximal.

```
seeds = greedy(G,3)
print(seeds)
print(utility(G, seeds))
```

```
set([0, 33, 4])
34
```