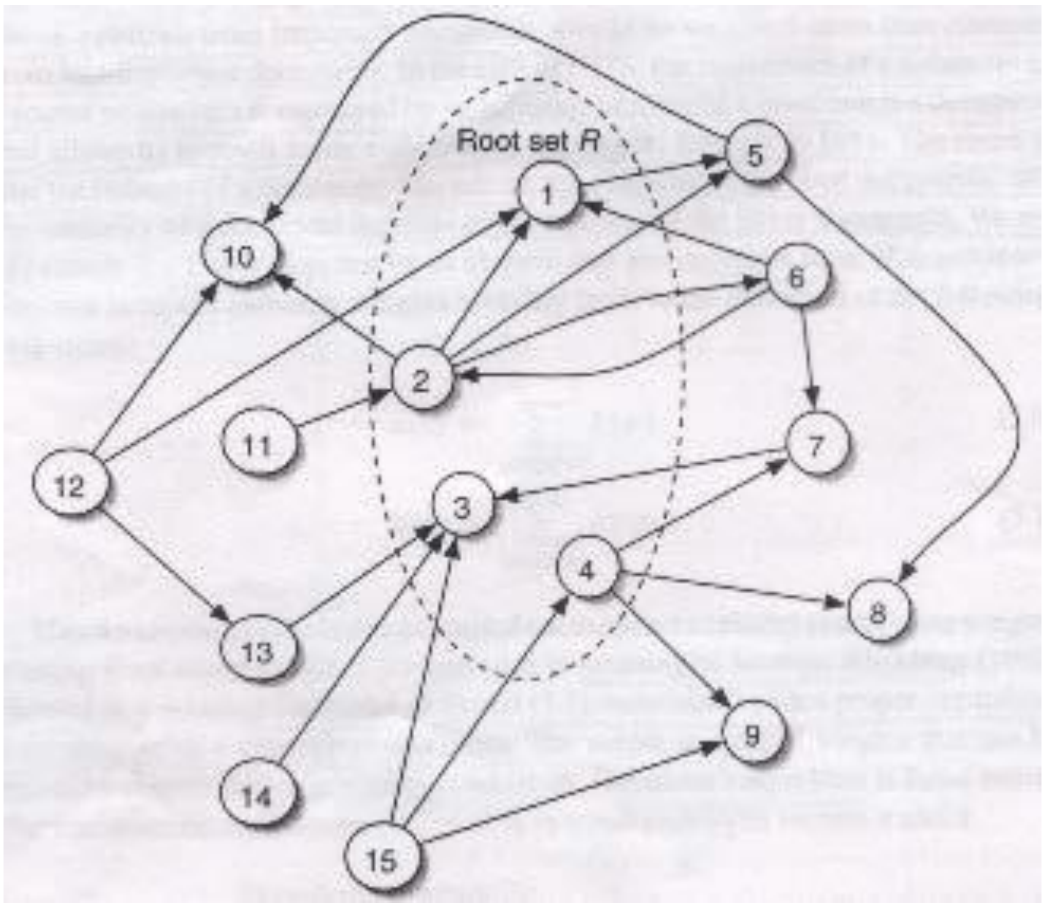
Data Analytics - Assignment 8

Jing Pang, Tian Xue, Chuqian Ma, Jiaxiang Leng

1. Apply the HITS algorithm to the following network (there is no one correct answer)



Root Set R={1,2,3,4}

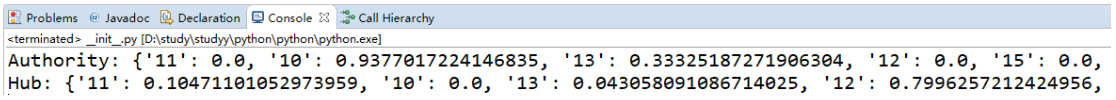
Extend it to form the base set S

As the main idea of HITS algorithm tells us, given the root set R, we add to S all pages pointed to by any page in R and all pages that point to any page in R. We extend the root set R to form our base set S, which is the whole network of all these 15 nodes.

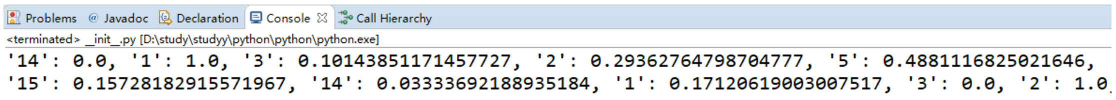
We choose Python to apply the HITS algorithm to this network and we use sparse matrices to reduce memory usage. As the main code shows below, first, we initialize the authority vector, whose values are all 1. The clean_pages method is to remove links to the outside the set of pages we are running HITS. Then, we initialize the pages dictionary, L. After that, we implement normalization, which means that we take a vector and divide all components by max value, which controls the max value in the vector to be 1. Next step is to get the sum of all the differences between components in vector 1 and 2, which gives us the authority and hub of each node. At last, we create the page network and run HITS algorithm on it.

```python
def initialize_authority(pages):

    return dict(zip(pages.keys(), [1]*len(pages)))

def clean_pages(pages):
    for page in pages:
        outside_links = []
        for i in range(len(pages[page])):
            if pages[page][i] not in pages or pages[page][i] == page: outside_links.append(i)
        outside_links.reverse()
        for outside_link in outside_links:
            pages[page].pop(outside_link)
    return pages

def initialize_L_matrices(pages):
    L_matrix = pages
    Lt_matrix = {}
    for page in pages:
        Lt_matrix[page] = []
    for page in pages:
        for link in pages[page]:
            Lt_matrix[link].append(page)
    return L_matrix, Lt_matrix

def multiply_matrix_vector(matrix, vector):
    result_matrix = {}
    for row in matrix:
        result_matrix[row] = 0
        for item in matrix[row]:
            result_matrix[row] += vector[item]
    return result_matrix
def normalize(vector):
    max = 0
    for component in vector:
        if vector[component] > max:
            max = vector[component]
    if max == 0:
        return vector
    for component in vector:
        vector[component] = float(vector[component]) / max
    return vector

def vector_difference(vector1, vector2):
    if not (vector1 and vector2): return float("inf")
    total = 0
    for component in vector1:
        total += abs(vector1[component] - vector2[component])
    return total

def HITS(pages):
    pages = clean_pages(pages)
    authority_old = None
    authority = initialize_authority(pages)
    (L_matrix, Lt_matrix) = initialize_L_matrices(pages)
    while vector_difference(authority_old, authority) > 0.1:
        authority_old = authority
        hub = normalize(multiply_matrix_vector(L_matrix, authority))
        authority = normalize(multiply_matrix_vector(Lt_matrix, hub))
    return authority, hub

def main():
    pages = {"1":["5"], "2":["1","5","6","10"], "3":[],"4":["7", "8","9"], "5":["8","10"],
             "6":["1","2","7"],"7":["3"], "8":[], "9":[], "10":[],"11":["2"], "12":["1","10","13"],
             "13":["3"],"14":["3"],"15":["3","4","9"]}
    (authority, hub) = HITS(pages)
    print "Authority: " + str(authority)
    print "Hub: " + str(hub)

if __name__ == "__main__":
    main()
```

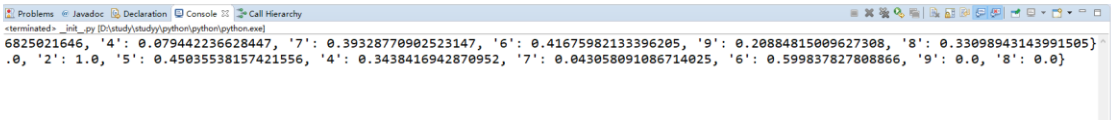
After running the HITS algorithm on this network, we got results like this:



```
<terminated> _init_.py [D:\study\studyy\python\python\python.exe]
Authority: {'11': 0.0, '10': 0.9377017224146835, '13': 0.33325187271906304, '12': 0.0, '15': 0.0,
Hub: {'11': 0.10471101052973959, '10': 0.0, '13': 0.043058091086714025, '12': 0.7996257212424956,
```



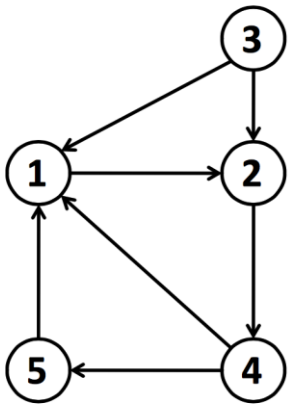
```
<terminated> _init_.py [D:\study\studyy\python\python\python.exe]
'14': 0.0, '1': 1.0, '3': 0.10143851171457727, '2': 0.29362764798704777, '5': 0.4881116825021646,
'15': 0.15728182915571967, '14': 0.03333692188935184, '1': 0.17120619003007517, '3': 0.0, '2': 1.0
```



```
<terminated> _init_.py [D:\study\studyy\python\python\python.exe]
6825021646, '4': 0.079442236628447, '7': 0.39328770902523147, '6': 0.41675982133396205, '9': 0.20884815009627308, '8': 0.33098943143991505}
.0, '2': 1.0, '5': 0.45035538157421556, '4': 0.3438416942870952, '7': 0.043058091086714025, '6': 0.599837827808866, '9': 0.0, '8': 0.0}
```

From the results we have got, it is obvious that node 1 has the biggest authority score and node 2 has the biggest hub score.

2. Find the Hubs and Authorities of the graphs below given by HITS. Are the results consistent with the notions of Hubs and Authorities?

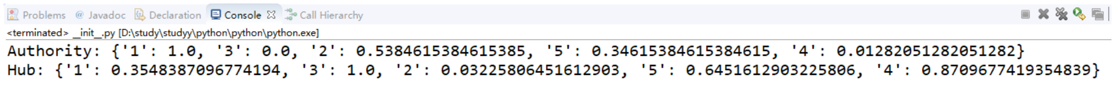


Graph 1

We also run our HITS algorithm to this graph, we create a new network as graph 1 like this:

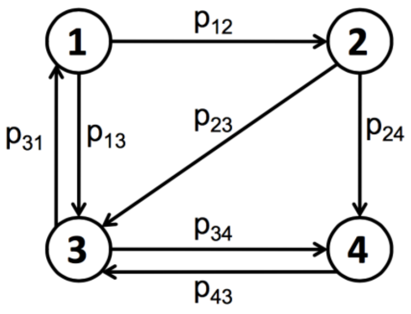
```
def main():
    pages = {"1":["2"], "2":["4"], "3":["1","2"],"4":["1","5"], "5":["1"]}
    (authority, hub) = HITS(pages)
    print "Authority: " + str(authority)
    print "Hub: " + str(hub)
```

Then we run it, our results are as follow:



```
<terminated> _init_.py [D:\study\studyy\python\python\python.exe]
Authority: {'1': 1.0, '3': 0.0, '2': 0.5384615384615385, '5': 0.34615384615384615, '4': 0.01282051282051282}
Hub: {'1': 0.3548387096774194, '3': 1.0, '2': 0.03225806451612903, '5': 0.6451612903225806, '4': 0.8709677419354839}
```

From above, we know that node 1 has the biggest authority score, so it is an Authority, but it is not consistent with the notion of Authorities, which is because it also points to another page. Node 3 has the biggest hub score, and it is not pointed by any other pages, so it is a Hub, and it is consistent with the notion of Hubs. Similar to node 1, node 2, 4, 5 have both in-links and out-links, they are not consistent with the notions of Hubs and Authorities.

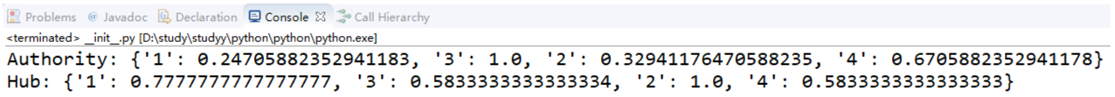


Graph 2

We also run our HITS algorithm to this graph, we create a new network as graph 2 like this:

```python
def main():
    pages = {"1":["2","3"], "2":["3","4"], "3":["1","4"],"4":["3"]}
    (authority, hub) = HITS(pages)
    print "Authority: " + str(authority)
    print "Hub: " + str(hub)
```

Then we run it, our results are as follow:



```
<terminated> __init__.py [D:\study\studyy\python\python\python.exe]
Authority: {'1': 0.24705882352941183, '3': 1.0, '2': 0.32941176470588235, '4': 0.6705882352941178}
Hub: {'1': 0.7777777777777777, '3': 0.5833333333333334, '2': 1.0, '4': 0.5833333333333333}
```

From above, we get to the result that node 3 has the biggest authority score, and node 2 has the biggest hub score. We can say that node 3 is Authority and node 2 is Hub. However, all of them are not consistent with the notions of Hubs and Authorities, including node 1 and 4, which is because that they all both have in-links and out-links.