Computational Time Analyse

Traditional method:

For the traditional way of calculating PageRank, we're using a matrix to represent graphs and using a vector to represent PageRank values for each node. This may raise two issues that may lead the algorithm to fail. The first issue is: that given the size of the Web, there may be millions of nodes. If we still use the matrix to represent this graph, the main memory of our computer cannot fit. The space complexity is O(n^2) where n is the number of nodes in the graph. The second issue is: that since each node's PageRank value gets updated through the multiplication of matrix and vector, the time complexity would be extremely high. The time complexity would essentially be O(kv^2) where K is the number of iterations and v is the number of nodes in the graph.

Proposed method:

In order to reduce high time complexity, we take advantages of quick retrieval nature of dictionary data structure. By observing the traditional method for updating PageRank values, matrix-vector multiplication is very time wasteful. By using dictionary, we can directly access the out links and in links of the current node with O(1) time complexity, and then we can just using the algorithm:

In order to reduce the high space complexity mentioned above, a dictionary of this data structure is used to represent a graph instead of a matrix, to be more specific, `defaultdict` is used. The `defaultdict` is a subdivision of the `dict` in python, it allows each new key to be given a default value based on the type of dictionary being created. In our algorithm, we define the value of each key to be a list, and in the list, we store the parent node and the number of children for each parent node. For example: {1: [2,3], 2: [1,4], 3: [1,4], 4: [1, 2]}. In this example. node 1 has one parent which is node 2 and the children number(out links) of node 2 is 3, meaning there are 3 websites that website 2 is pointing to. In this way, all the nodes and their associative edges have been stored, the space complexity has been reduced to O(V+E). By comparing the traditional method in which a lot of space is wasted due to 0s present in the matrix, this method is more space-efficient.
In order to reduce high time complexity, we take advantage of the quick retrieval nature of dictionary data structure. By observing the traditional method for updating PageRank

values, matrix-vector multiplication is very time wasteful. By using the dictionary, we can directly access the out links and in links of the current node with O(1) time complexity, and then we can just use the algorithm:

$$PR(x) = \frac{(1-\beta)}{N} + \sum_{y \to x} \frac{PR(y)}{Out(y)}$$

to update each node's PageRank value.

In this way, the time complexity can be reduced to O(V+E) since we only need to go through each node and their edges once. This time complexity is way smaller than O(V^2) in sparse graph.