Link Analysis Proposal

Ynah Sebastian

Link Analysis

Link: a relationship between two entities

Link Analysis: Using links to establish higher-order relationships among entities, useful for search engines analyzing webpage links.

Goal: Associate a relative quantitative assessment with each entity using link-based measures

Link Analysis Problem through Graphing

Links are like edges in a graph, the objects that links connect are like nodes or vertices in a graph.

Any network of links can be represented as a graph G = (V,E), where V denotes the set of vertices (nodes) and E denotes the set of edges (links).

The goal is to associate a relative quantitative assessment with each node.

- PageRank uses linear algebra and iteration to return a value for each node.
- HIT returns 2 values for each node based on in-degree & out-degree.

PageRank

PageRank is an algorithm that addresses the Link-based Object Ranking (LOR) problem.

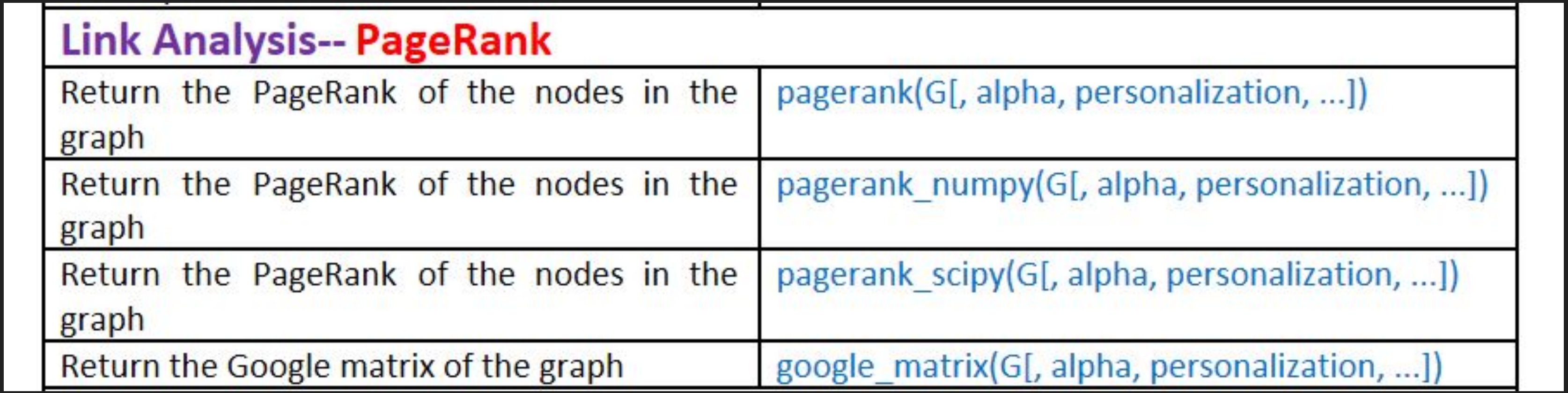
The algorithm considers a model in which a user starts at a vertex and performs a “random walk” by following edges from the vertex he is currently in. PageRank of a vertex is the probability of that vertex being visited on a particular random walk.

If a node has k outgoing edges, it will pass on 1/k of its importance to each of the nodes that it links to.

The Python procedure takes its argument in a similar format to tsort: the graph argument is an iterable of two-element sequences. It also accepts optional arguments for the damping factor (to take into account the probability of a user beginning a new random walk) and the amount of convergence since PageRank values are assigned using a converging iterative power method.

The probability that page i will be visited after k steps is equal to $A^k x$. The sequence Ax, A^2x, A^3x, ..., A^kx, converges in this case to a unique probabilistic vector v^*. In this context v^* is the PageRank vector.

PageRank Functions

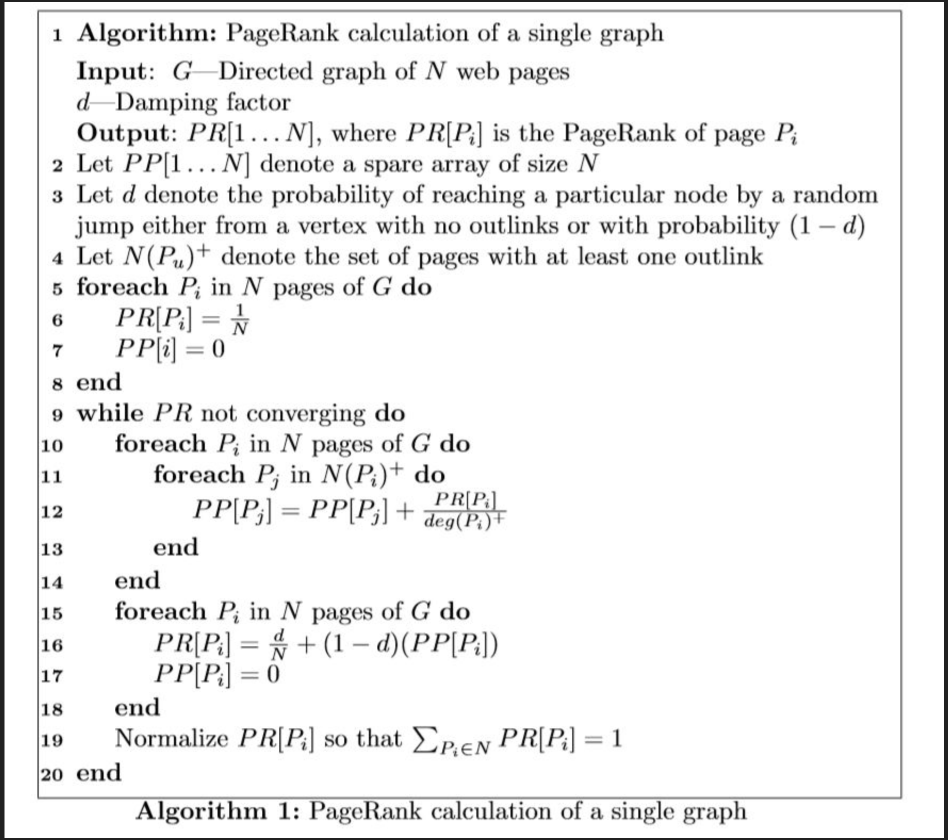


Link Analysis-- PageRank	
Return the PageRank of the nodes in the graph	pagerank(G[, alpha, personalization, ...])
Return the PageRank of the nodes in the graph	pagerank_numpy(G[, alpha, personalization, ...])
Return the PageRank of the nodes in the graph	pagerank_scipy(G[, alpha, personalization, ...])
Return the Google matrix of the graph	google_matrix(G[, alpha, personalization, ...])

pagerank(G, alpha=0.85, personalization=None, max_iter=100, tol=1e-06, nstart=None, weight='weight', dangling=None)

PageRank computes a ranking of the nodes in the graph G based on the structure of the incoming links.

PageRank Algorithm



```
Algorithm: PageRank calculation of a single graph
Input: $G$—Directed graph of $N$ web pages
$d$—Damping factor
Output: $PR[1 \ldots N]$, where $PR[P_i]$ is the PageRank of page $P_i$
Let $PP[1 \ldots N]$ denote a spare array of size $N$
Let $d$ denote the probability of reaching a particular node by a random
jump either from a vertex with no outlinks or with probability $(1 - d)$
Let $N(P_u)^+$ denote the set of pages with at least one outlink
foreach $P_i$ in $N$ pages of $G$ do
    $PR[P_i] = \frac{1}{N}$
    $PP[i] = 0$
end
while $PR$ not converging do
    foreach $P_i$ in $N$ pages of $G$ do
        foreach $P_j$ in $N(P_i)^+$ do
            $PP[P_j] = PP[P_j] + \frac{PR[P_i]}{deg(P_i)^+}$
        end
    end
    foreach $P_i$ in $N$ pages of $G$ do
        $PR[P_i] = \frac{d}{N} + (1 - d)(PP[P_i])$
        $PP[P_i] = 0$
    end
    Normalize $PR[P_i]$ so that $\sum_{P_i \in N} PR[P_i] = 1$
end
```

Algorithm 1: PageRank calculation of a single graph

PageRank of a node 'u' is defined as the sum of ratios of PageRank of all nodes (v1,v2..vn) providing backlinks to u.

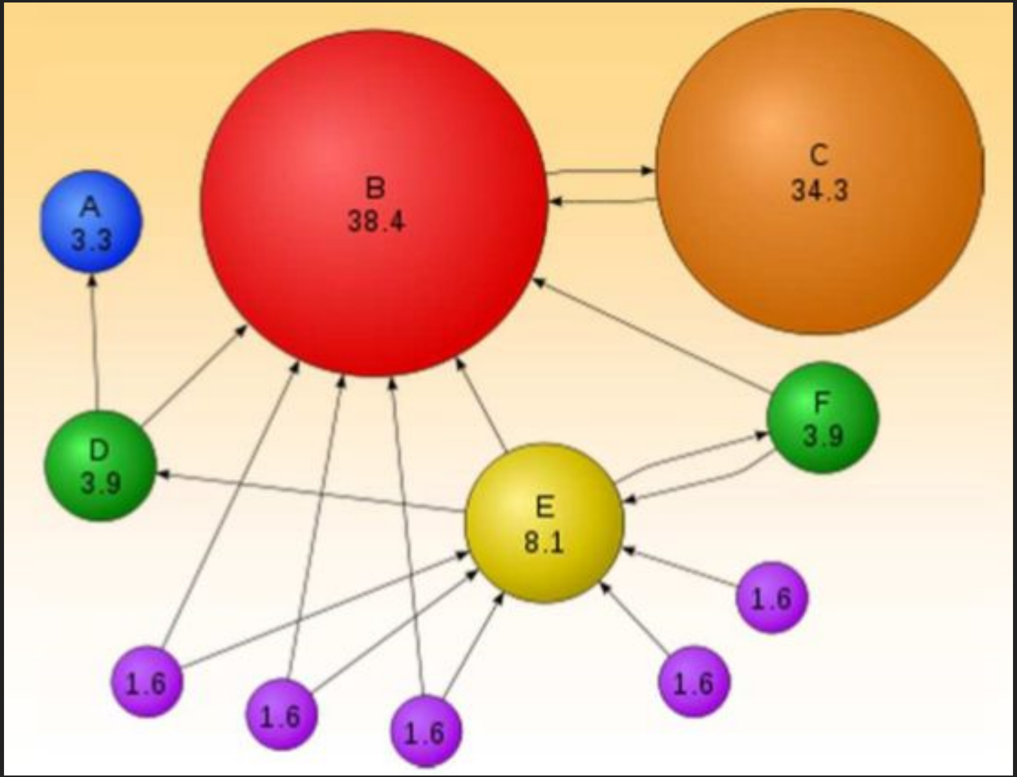
PageRank Output

Input : CSV graph (source, target, weight) converted to object Graph through networkx library.

Returns: pagerank – Dictionary of nodes with PageRank as value

Returns: A – Google matrix of the graph

Output : example image shown right with pagerank probabilities displayed on nodes.



HITS

From the pagerank subproblem, we get the probability of ending up in a node,but to find the more relevant nodes, we must reduce the problem to finding dense subgraphs of hubs and authorities.

The HITS algorithm is another LOR subproblem using the power method and returns two vectors, each of which contains a score for each vertex in the graph.

Authority –A vertex is considered an authority if it has many pages linking to it (High Indegree)

Hub –A vertex is considered a hub if it points to many other vertices (High Outdegree)

It is a runtime algorithm that continues to update these 2 vectors for max_iter iterations.

It first initializes input matrix A s.t :

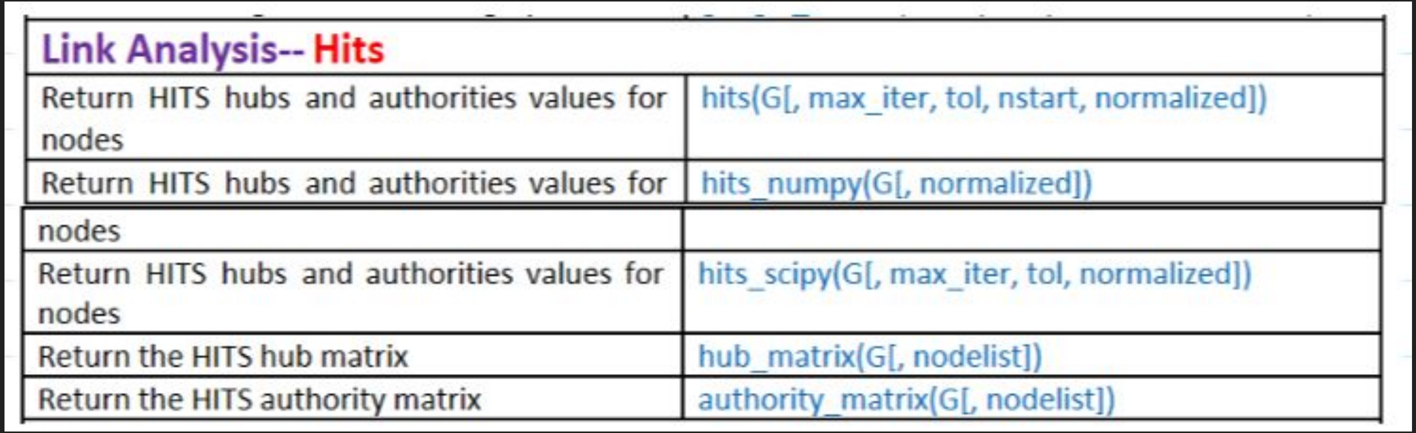
$$\mathbf{A}_{\{ij\}} = \begin{cases} 1, & \text{if } e_{\{ij\}} \in E \\ 0, & \text{otherwise} \end{cases}$$

authority (a) and hub (h) vectors are then recursively calculated using A

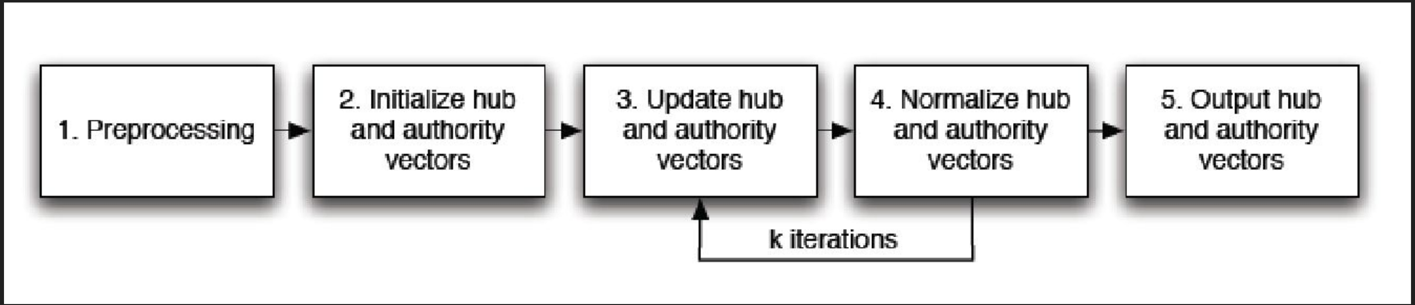
$$\mathbf{a}^{(k)} \leftarrow \mathbf{A}^{\mathbf{T}} \cdot \mathbf{h}^{(k-1)}.$$

$$\mathbf{h}^{(k)} \leftarrow \mathbf{A} \cdot \mathbf{a}^{(k)}.$$

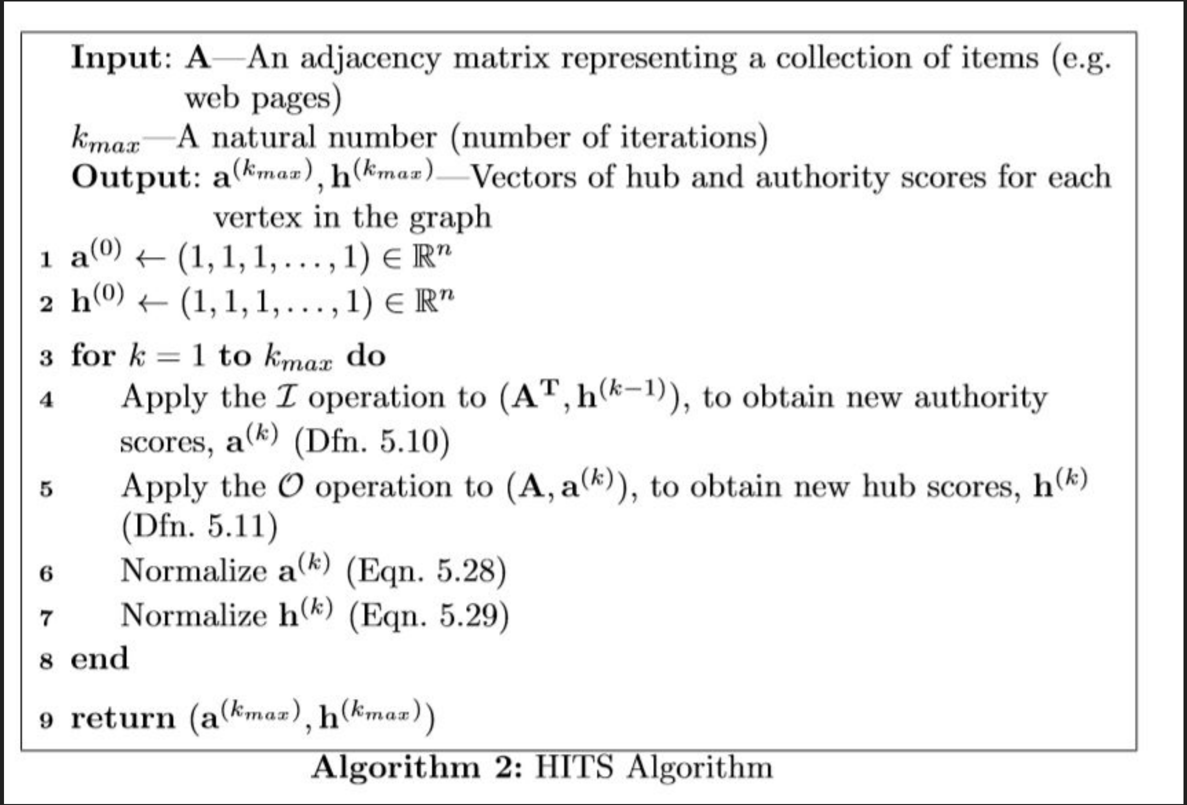
HITS functions



Link Analysis-- Hits	
Return HITS hubs and authorities values for nodes	hits(G[, max_iter, tol, nstart, normalized])
Return HITS hubs and authorities values for nodes	hits_numpy(G[, normalized])
Return HITS hubs and authorities values for nodes	hits_scipy(G[, max_iter, tol, normalized])
Return the HITS hub matrix	hub_matrix(G[, nodelist])
Return the HITS authority matrix	authority_matrix(G[, nodelist])


1. Preprocessing
2. Initialize hub and authority vectors
3. Update hub and authority vectors
4. Normalize hub and authority vectors
5. Output hub and authority vectors
k iterations

HITS algorithm



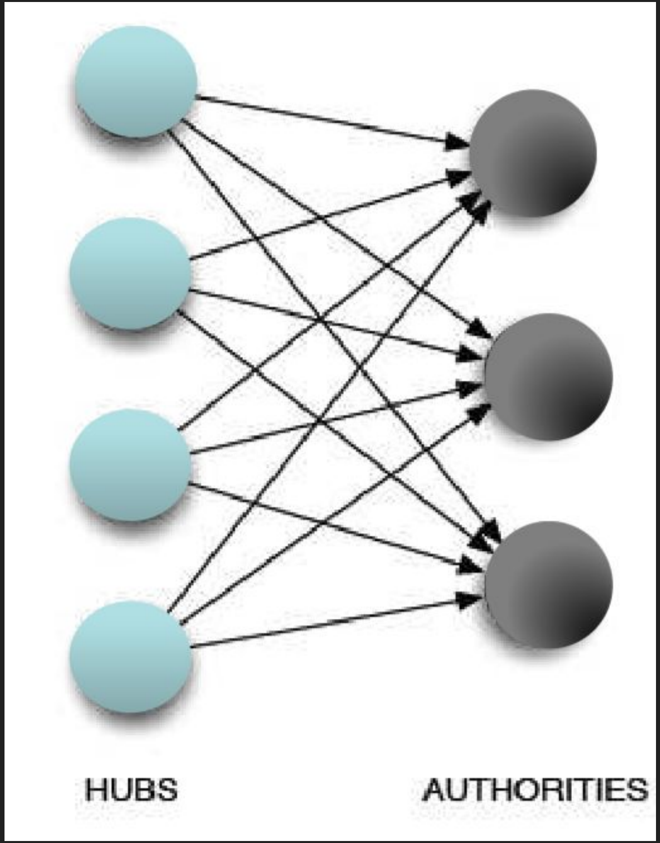
```
Input: $\mathbf{A}$—An adjacency matrix representing a collection of items (e.g.
        web pages)
$k_{max}$—A natural number (number of iterations)
Output: $\mathbf{a}^{(k_{max})}, \mathbf{h}^{(k_{max})}$—Vectors of hub and authority scores for each
        vertex in the graph
1 $\mathbf{a}^{(0)} \leftarrow (1, 1, 1, \ldots, 1) \in \mathbb{R}^n$
2 $\mathbf{h}^{(0)} \leftarrow (1, 1, 1, \ldots, 1) \in \mathbb{R}^n$
3 for $k = 1$ to $k_{max}$ do
4     Apply the $\mathcal{I}$ operation to $(\mathbf{A}^\mathbf{T}, \mathbf{h}^{(k-1)})$, to obtain new authority
      scores, $\mathbf{a}^{(k)}$ (Dfn. 5.10)
5     Apply the $\mathcal{O}$ operation to $(\mathbf{A}, \mathbf{a}^{(k)})$, to obtain new hub scores, $\mathbf{h}^{(k)}$
      (Dfn. 5.11)
6     Normalize $\mathbf{a}^{(k)}$ (Eqn. 5.28)
7     Normalize $\mathbf{h}^{(k)}$ (Eqn. 5.29)
8 end
9 return $(\mathbf{a}^{(k_{max})}, \mathbf{h}^{(k_{max})})$
```

Algorithm 2: HITS Algorithm

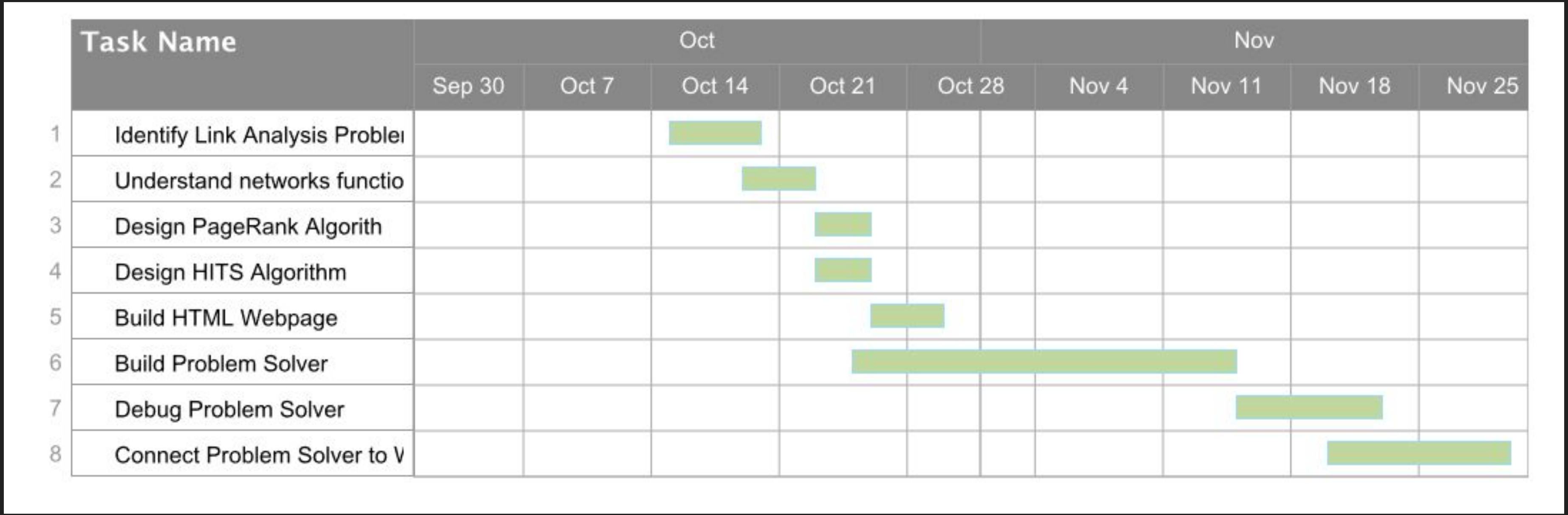
Authority and hub vectors are initialized to all 1s with length n (number of vertices in the graph)

HITS Output

Authority_matrix and hub_matrix networkX functions return outputs to categorize nodes based on their backlinks or forwardlinks. Knowing the highest ranking authority or hub nodes show their contribution in the graph



Gantt Chart


Task Name
Oct
Nov
Sep 30
Oct 7
Oct 14
Oct 21
Oct 28
Nov 4
Nov 11
Nov 18
Nov 25
1 Identify Link Analysis Proble
2 Understand networks functio
3 Design PageRank Algorith
4 Design HITS Algorithm
5 Build HTML Webpage
6 Build Problem Solver
7 Debug Problem Solver
8 Connect Problem Solver to V

Citations

https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.link_analysis.hits_alg.authority_matrix.html

https://www.csc2.ncsu.edu/faculty/nfsamato/practical-graph-mining-with-R/slides/pdf/Link_Analysis.pdf

https://www.techopedia.com/definition/30336/link-analysis

https://www.csc2.ncsu.edu/faculty/nfsamato/practical-graph-mining-with-R/sample/chapter_5_LinkAnalysis.pdf

https://cs7083.wordpress.com/2013/01/31/demystifying-the-pagerank-and-hits-algorithms/

http://dpk.io/pagerank

http://pi.math.cornell.edu/~mec/Winter2009/RalucaRemus/Lecture3/lecture3.html