The object of this project is (1) to implement the PageRank algorithm and the HITS algorithm; (2) to experiment with random graphs of a structure similar to the Web.

The PageRank Algorithm

The PageRank algorithm can be specified, in high-level pseudo-code, as follows below. This implementation assume that, if you reach a page with no outlinks, you jump randomly in the Web. This is not a very efficient implementation, but it will do for this assignment.

```
function PageRank(in G : directed graph of N web pages;
    E : real. Probability of flipping heads.)
    return PR[1 ... N]. /* PR[I] is the page rank of page I */
    var PP[1..N] /* Spare array of size N */
    randomTrans : real; /* Probability of reaching a particular node */
        by a random jump either from a vertex with no outlinks,
        or by flipping tails.
    vOuts : Set of pages with at least one outlink;
    vNoOuts : Set of pages with no outlinks;
    for (I=1 to N) PR[I]=1/N;
    repeat
        randomTrans = \sum_{I\in vOuts} (1 - E) PR[I] / N + \sum_{I\in vNoOuts} PR[I] / N;
        for (J=1 to N) PP[J] = randomTrans + \sum_{I\Rightarrow J} E \cdot PR[I] / (# of outlinks from I)
        if (for every I, |PP[I] - PR[I]| / PR[I] < \epsilon) then exitloop;
        PR := PP;
    } return(PR);
```

The HITS Algorithm

One formulation of the HITS algorithm is as follows:

```
function HITS(in G : directed graph of N web pages;
    E : real. Probability of flipping heads.)
    return AU[1 ... N], /* AU[I] is the authority value of page I */
            HUB[1 ... N]; /* HUB[I] is the hub value of page I */
    var A1[1 .. N], H1[1..N]; /* Spare arrays */
    for (I in Hubs) HUB[I] := 1/|Hubs|;
    Hubs = set of pages with at least one outlink;
    Auths = set of pages with at least one inlink;
    HUB = AU = 0;
```
for (I in Auths) AU[I] := 1/|Auths|;
repeat {
    for (J in Auths) A1[J] = (1−E)/|Auths| + ∑_{I→J} E · HUB[I] / #(outlinks from I)
    for (I in Hubs) H1[I] = (1−E)/|Hubs| + ∑_{I→J} E · AU[J] / #(inlinks to J)
    if ((for every J in Auths, |A1[J] − AU[J]| / AU[J] < ε) and
        (for every I in Hubs, |H1[I] − HUB[I]| / HUB[I] < ε))
        then exitloop;
    HUB := H1;
    AU := A1;
}

Part I

Write a program that reads from input a set of URL’s and then computes the PageRank, hub value, and authority value of each of these pages relative to that set. Do not include any self-links (links from a page to itself) in setting up the adjacency graph.

Run your program on the three test inputs linked on the Web page with E=0.15, E=0.5, and E=0.85. In all these experiments, and in the experiments in part 2, set ε = 0.05

Part II

The following algorithm generates a random graph with approximately N vertices whose structure somewhat resembles the actual structure of the Web.

randomGraph(N);
{ initialize the graph to contain vertices U, V and a link U→V;
    repeat {
        pick a random number R between 0 and 3;
        if ((R == 0) or (R == 1))
            then pick a link at random, and let T be the tail of that link;
        else if (R == 2)
            then pick an existing vertex at random, and let T be that vertex;
            else create a new vertex T;
        pick a random number R between 0 and 3;
        if ((R == 0) or (R == 1))
            then pick a link at random, and let H be the head of that link;
        else if (R == 2)
            then pick an existing vertex at random, and let H be that vertex;
            else create a new vertex H;
        add a link from T→H unless this link already exists or T==H;
    } until there are at least N vertices in the graph.
}

Run and report the following experiments:
For N=20, 200, 2000, ... and continuing by factors of 10 as long as you reasonably can:

- Generate a random graph of size N and plot in-degree versus rank of in-degree (see lecture notes 4).
• Generate 100 random graphs of size N, and run the PageRank algorithm with $E=0.15$, $E=0.5$, $E=0.85$. For each value of N and E report the mean number of iterations of the main loop needed for the algorithm to converge and the standard deviation.