NAME: Meith Navlakha
SAP: 60004190068
DWM Practical

EXPERIMENT 9

AIM: To implement HITS Algorithm

THEORY:

HITS:

Hyperlink-Induced Topic Search (HITS) is a link analysis algorithm that rates Web pages. It could discover and rank the webpages relevant for a particular search. HITS uses hubs and authorities to define a recursive relationship between webpages. This algorithm is used on the web link-structures to discover and rank the webpages relevant for a particular search. The idea of this algorithm originated from the fact that an ideal website should link to other relevant sites and also being linked by other important sites.

The scheme therefore assigns two scores for each page:

- Authority value, which estimates the value of the content of the page
- Hub value, which estimates the value of its links to other pages.

Algorithm

In the HITS algorithm, the first step is to retrieve the most relevant pages to the search query. This set is called the *root set* and can be obtained by taking the top pages returned by a text-based search algorithm.

A *base set* is generated by augmenting the root set with all the web pages that are linked from it and some of the pages that link to it. The web pages in the base set and all hyperlinks among those pages form a focused subgraph.

The HITS computation is performed only on this *focused subgraph*
Authority and hub values are defined in terms of one another in a mutual recursion.

An authority value is computed as the sum of the scaled hub values that point to that page.

A hub value is the sum of the scaled authority values of the pages it points to. Some implementations also consider the relevance of the linked pages.

The algorithm performs a series of iterations, each consisting of three basic steps:

- **Authority update:** Update each node's *authority score* to be equal to the sum of the *hub scores* of each node that points to it. That is, a node is given a high authority score by being linked from pages that are recognized as Hubs for information.

$$\mathrm{auth}(p) = \sum_{q \in P_{\mathrm{to}}} \mathrm{hub}(q)$$

- **Hub update:** Update each node's *hub score* to be equal to the sum of the *authority scores* of each node that it points to. That is, a node is given a high hub score by linking to nodes that are considered to be authorities on the subject.

$$\mathrm{hub}(p) = \sum_{q \in P_{\mathrm{from}}} \mathrm{auth}(q)$$

- **Normalize:** Normalize the values by dividing each Hub score the sum of the all Hub scores, and dividing each Authority score by the sum of all Authority scores.
- Repeat the steps for k times.

HITS algorithm differences with respect to PageRank:

- It is query dependent, that is, the (Hubs and Authority) scores resulting from the link analysis are influenced by the search terms.
- As a corollary, it is executed at query time, not at indexing time, with the associated hit on performance that accompanies query-time processing.
- It computes two scores per document, hub and authority, as opposed to a single score.
- It is processed on a small subset of ‘relevant’ documents (a 'focused subgraph' or base set), not all documents as was the case with PageRank.

CODE:

```
import java.util.*;
import java.util.Arrays;
import java.lang.Integer;
import java.lang.Math;

public class Hits{
   public static int MAX_NODES = 10;
   public static int[][] incoming = new int[MAX_NODES][MAX_NODES] ;
   public static int[][] outgoing = new int[MAX_NODES][MAX_NODES] ;
```

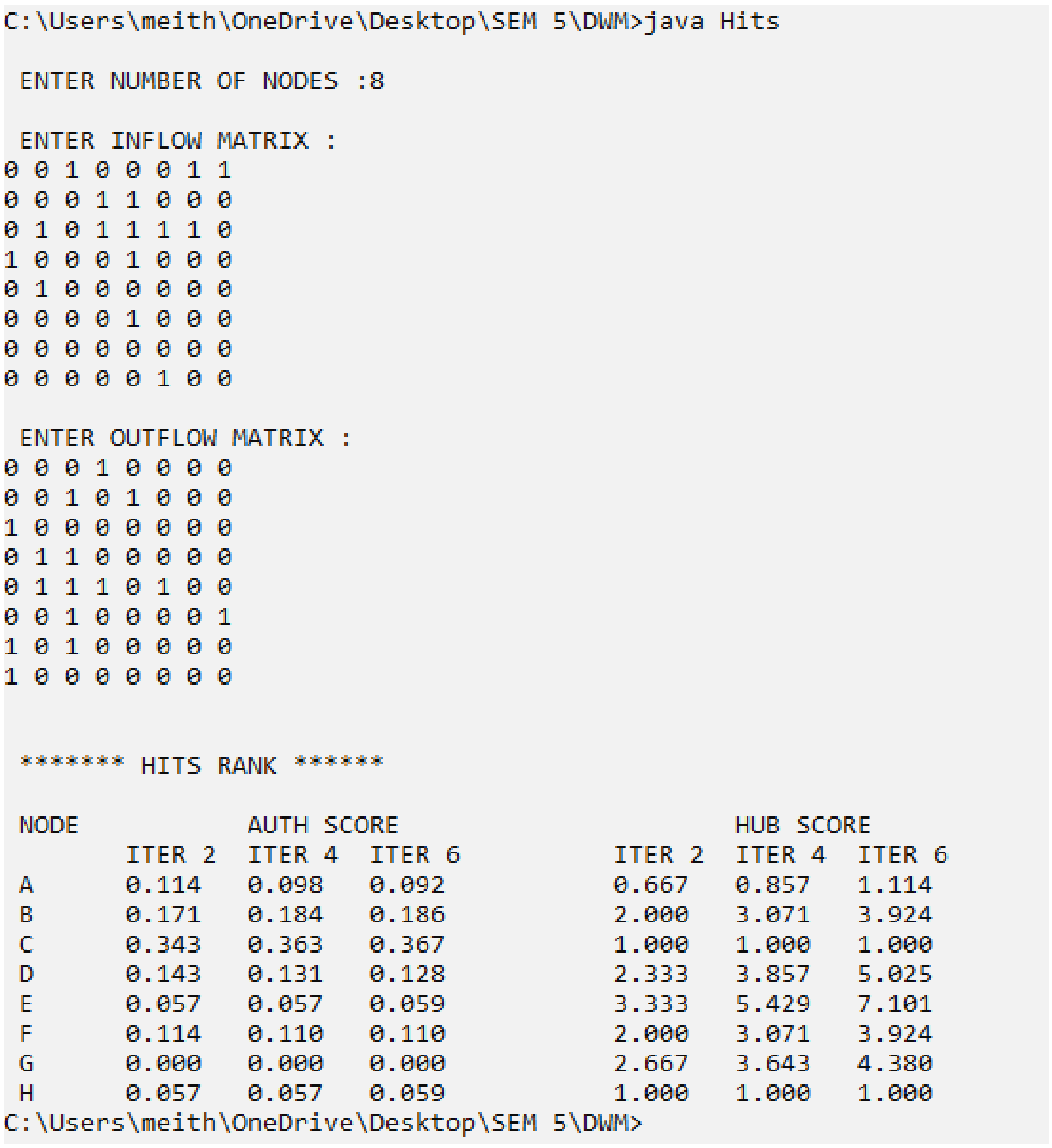
```
    public static float[][] overAllHubScore = new float[MAX_NODES][MAX_NODES] ;
    public static float[][] overAllAuthScore = new float[MAX_NODES][MAX_NODES] ;
    public static float[] AuthScore = new float[MAX_NODES] ;
    public static float[] HubScore = new float[MAX_NODES] ;
    public static float[] New_AuthScore = new float[MAX_NODES] ;
    public static float[] New_HubScore = new float[MAX_NODES] ;

    public static void displayOutput(int nodes){

      System.out.print("\n\n ******* HITS RANK ******\n") ;
      System.out.print("\n NODE\t\tAUTH SCORE\t\t\tHUB SCORE") ;
      System.out.print("\n \tITER 2\tITER 4\tITER 6\t\tITER 2\tITER 4\tITER 6") ;



      for(int i =0; i<nodes;i++){
          System.out.printf("\n %s \t%.3f \t%.3f\t%.3f \t\t%.3f \t%.3f \t%.3f" ,
              (char)(i+65) ,overAllAuthScore[1][i],overAllAuthScore[3][i],
overAllAuthScore[5][i] ,
              overAllHubScore[1][i], overAllHubScore[3][i], overAllHubScore[5][i]) ;
      }
    }

    public static float calcAuthScore(int curNode , int nodes){
        float val=0.0f;
        for(int i=0; i<nodes;i++){

            if(incoming[curNode][i]==1){
                val += HubScore[i] ;
            }
        }
        return val;

    }

    public static float calcHubScore(int curNode , int nodes){
        float val=0.0f;
        for(int i=0; i<nodes;i++){
            if(outgoing[curNode][i]==1){
                val += AuthScore[i] ;
            }
        }
        return val;

    }
```

```
public static void calcHitsRank(int nodes){
    int MAX_ITER = 6;
    float total_auth=0.0f, total_hub=0.0f ;

    for(int i=0; i<MAX_ITER; i++){

        // ALL NODES
        for(int n=0; n<nodes; n++){
            New_AuthScore[n] = calcAuthScore(n, nodes);
            total_auth += New_AuthScore[n];
        }
        for(int n=0; n<nodes; n++){
            New_HubScore[n] = calcHubScore(n, nodes);
            total_hub =+ New_HubScore[n];
        }

        // NORMALISE
        for(int n=0; n<nodes; n++){
            HubScore[n] = (float)(New_HubScore[n])/(total_hub) ;
            AuthScore[n] = (float)(New_AuthScore[n])/(total_auth) ;

            overAllHubScore[i][n] =  HubScore[n];
            overAllAuthScore[i][n] =  AuthScore[n];
        }

        // RE-INITIALISE
        total_auth=0;
        total_hub=0;
    }

    displayOutput(nodes);

}
public static void initialise(int nodes){

    for(int j=0; j<nodes; j++){
        AuthScore[j] =1.0f;
        HubScore[j] =1.0f;

    }
}
```

```
    public static void main(String args[]){

        Scanner sc = new Scanner(System.in) ;

        System.out.print("\n ENTER NUMBER OF NODES :");
        int nodes = sc.nextInt();

        // int nq=0;
        System.out.print("\n ENTER INFLOW MATRIX :\n");
        for(int i=0; i<nodes;i++){
            for(int j=0; j<nodes; j++){
                incoming[i][j] = sc.nextInt();

                // if(outgoing[i][j] ==1){
                //     nq++;
                // }
            }

            // AuthScore[i]= nq;
            // nq=0;
        }

        System.out.print("\n ENTER OUTFLOW MATRIX :\n");
        for(int i=0; i<nodes;i++){
            for(int j=0; j<nodes; j++){
                outgoing[i][j] = sc.nextInt();

            }
        }

        initialise(nodes) ;
        calcHitsRank(nodes);

        sc.close();
    }
}
```

NAME: Meith Navlakha
SAP: 60004190068
DWM Practical

OUTPUT:



```
C:\Users\meith\OneDrive\Desktop\SEM 5\DWM>java Hits

 ENTER NUMBER OF NODES :8

 ENTER INFLOW MATRIX :
0 0 1 0 0 0 1 1
0 0 0 1 1 0 0 0
0 1 0 1 1 1 1 0
1 0 0 0 1 0 0 0
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0

 ENTER OUTFLOW MATRIX :
0 0 0 1 0 0 0 0
0 0 1 0 1 0 0 0
1 0 0 0 0 0 0 0
0 1 1 0 0 0 0 0
0 1 1 1 0 1 0 0
0 0 1 0 0 0 0 1
1 0 1 0 0 0 0 0
1 0 0 0 0 0 0 0


 ******* HITS RANK ******

 NODE           AUTH SCORE                      HUB SCORE
        ITER 2  ITER 4  ITER 6          ITER 2  ITER 4  ITER 6
 A      0.114   0.098   0.092           0.667   0.857   1.114
 B      0.171   0.184   0.186           2.000   3.071   3.924
 C      0.343   0.363   0.367           1.000   1.000   1.000
 D      0.143   0.131   0.128           2.333   3.857   5.025
 E      0.057   0.057   0.059           3.333   5.429   7.101
 F      0.114   0.110   0.110           2.000   3.071   3.924
 G      0.000   0.000   0.000           2.667   3.643   4.380
 H      0.057   0.057   0.059           1.000   1.000   1.000
C:\Users\meith\OneDrive\Desktop\SEM 5\DWM>
```

CONCLUSION: In this experiment, I implemented HITS Algorithm used for analysis web pages rank using two scores Authority_Score and Hub_Score. The above example displays the Authority_Score and Hub_Score for iteration 2 ,4 and 6. The values of the scores changes with iterations but after a larger number of iterations they converge to a stable value. One major advantage of HITS algorithm over PageRank is that it considers only a subset of network instead of the entire network.