	Finolex Academy of Management and Technology, Ratnagiri			
	Department of Information Technology			
Subject name: Big Data Analytics				Subject Code: ITC801
Class	BE IT	Semester – VIII (CBGS)		Academic year: 2019-20
Name of Student	**Kazi Jawwad A Rahim**		**QUIZ Score :**	
Roll No	**28**	Assignment/Experiment No.		08
Title: **Implementation of HITS Algorithm**				

1. Course objectives applicable: COB4.Study Page Rank in Link Analysis and concepts of Handling larger datasets

2. Course outcomes applicable:
CO4-Implement use of combiners to consolidate results and ability to handle larger datasets

3. Learning Objectives:

1. To understand concept of HITS
2. To understand Hubs and Authorities
3. To program HITS Score computation in C/C++
4. To prove that HITS Converges after certain iterations

4. Practical applications of the assignment/experiment: HITS Algorithm is used by Ask.com Search engine for indexing of webpages and giving results for search queries

5. Prerequisites:

1. Understanding of Internet Technologies

6. Hardware Requirements:

1. PC with 4GB RAM, 500GB HDD,

7. Software Requirements:

1. Access to C/C++ compiler
2. Internet access if online compiler is used

8. Quiz Questions (if any): (Online Exam will be taken separately batchwise, attach the certificate/ Marks obtained)

1. What is a HITS?
2. What is HUB?
3. What is a Authority Page?
4. What is SCC?

9. Experiment/Assignment Evaluation:

Sr. No.	**Parameters**	**Marks obtained**	**Out of**
1	Technical Understanding (Assessment may be done based on Q & A **or** any other relevant method.) Teacher should mention the other method used -		6
2	Neatness/presentation		2
3	Punctuality		2
Date of performance (DOP)		**Total marks obtained**	**10**
Date of checking (DOC)		**Signature of teacher**	

Theory:

Hyperlink-Induced Topic Search (HITS) is a link analysis algorithm that rates web pages, developed by Jon Kleinberg. The idea behind hubs and authorities stemmed from a particular insight into the creation of web pages when the Internet was originally forming, that is certain webpages, known as hubs served as large directories that were not actually authoritative in the information that they held, but were used as compilations, of a broad catalog of information that led users direct to other authoritative pages. In other words, a good hub represented a page that pointed to many other pages and a good authority represented a page that was liked by many different hubs. The scheme therefore assigns two scores for each page, its authority which estimates the value of the content of the page and its hub value which estimates the value of its links to other pages.

Expanding the root set into a base set

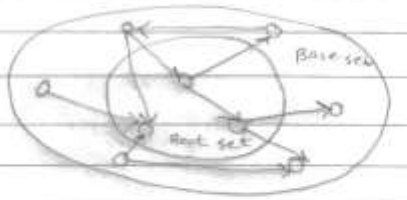
In the HITS algorithm, the first step is to retrieve the most relevant pages to the search query. This set is called the root set and can be obtained by taking the top pages returned by a text-based search algorithm. A base set is generated by augmenting the root set with all the web pages that are

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. According to Kleinberg the reason for constructing a base set is to ensure that most of the strongest authorities are included. 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 two basic steps:

Authority update: Update each node's Authority scores 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.

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.



Expanding the root set into a base set

Precautions:

1) Calculate the adjacency matrix from the web graph accurately.
2) Use separate arrays for Hub Score and Authority score.

12. Installation Steps / Performance Steps –

PageRank Program with Teleportation:

```
import java.util.*;
import java.io.*;
import java.lang.*;
import static java.lang.Math.*;
public class hits341 {
      int iter;
      int initval;
      String filename;
      int n;      // number of vertices in the graph
      int m;      // number of edges in the graph
      int[][] L;  // adjacency matrix
      double[] h0;
      double[] a0;
      final double errorrate = 0.00001;
   hits341() {} //default constructor
   hits341(int iter, int initval, String filename)     // 3 argument constructor to initialize class variables with provided command line arguments
   {
      this.iter = iter;
      this.initval = initval;
      this.filename = filename;
      try {
         Scanner scanner = new Scanner(new File(filename));
         n = scanner.nextInt();
         m = scanner.nextInt();
         //System.out.println("n = " + n + " m = " + m);
         //Adjacency matrix representation of graph
         L = new int[n][n];
         for(int i = 0; i < n; i++)
          for(int j = 0; j < n; j++)
                    L[i][j] = 0;
         while(scanner.hasNextInt())
         {
            L[scanner.nextInt()][scanner.nextInt()] = 1;
            //System.out.println(scanner.nextInt());
         }
         h0 = new double[n];
         a0 = new double[n];
         switch(initval) {
         case 0:
           for(int i = 0; i < n; i++) {
             h0[i] = 0;
             a0[i] = 0;
           }
           break;
         case 1:
           for(int i = 0; i < n; i++) {
             h0[i] = 1;
             a0[i] = 1;
           }
           break;
         case -1:
           for(int i =0; i < n; i++) {
             h0[i] = 1.0/n;
             a0[i] = 1.0/n;
           }
           break;
```

```
            case -2:
              for(int i =0; i < n; i++) {
                h0[i] = 1.0/Math.sqrt(n);
                a0[i] = 1.0/Math.sqrt(n);
              }
              break;
            }
        }
        catch(FileNotFoundException fnfe){}
    }

    public static void main(String[] args)
    {
        if(args.length != 3) {
            System.out.println("Usage: hits3416 iterations initialvalue filename");
            return;
        }
        //command line arguments
        int iterations = Integer.parseInt(args[0]);
        int initialvalue = Integer.parseInt(args[1]);
        String filename = args[2];
        if( !(initialvalue >= -2 && initialvalue <= 1) ) {
            System.out.println("Enter -2, -1, 0 or 1 for initialvalue");
            return;
        }
        hits3416 ht = new hits3416(iterations, initialvalue, filename);
        ht.hitsAlgo3416();
    }
 boolean isConverged(double[] p, double[] q)
    {
      for(int i = 0 ; i < n; i++) {
         if ( abs(p[i] - q[i]) > errorrate )
          return false;
      }
      return true;
    }

    public void hitsAlgo3416()
    {
        double[] h = new double[n];
        double[] a = new double[n];
        double a_scale_factor = 0.0;
        double a_sum_square = 0.0;
        double h_scale_factor = 0.0;
        double h_sum_square = 0.0;
        double[] aprev = new double[n]; //last iterations values of a, used for convergence
        double[] hprev = new double[n]; //last iterations values of h, used for convergence
        //If the graph has N greater than 10, then the values for iterations, initialvalue revert to 0 and -1
respectively
        if(n > 10) {
            iter = 0;
            for(int i =0; i < n; i++) {
                h[i] = 1.0/n;
                a[i] = 1.0/n;
                hprev[i] = h[i];
                aprev[i] = a[i];
            }
          int i = 0;
          do {
              for(int r = 0; r < n; r++) {
```

```
                aprev[r] = a[r];
                hprev[r] = h[r];
            }
            //A step starts
            for(int p = 0; p < n; p++) {
                a[p] = 0.0;
            }
            for(int j = 0; j < n; j++) {
                for(int k = 0; k < n; k++) {
                    if(L[k][j] == 1) {
                        a[j] += h[k];
                    }
                }
            }//A step ends
            //H step starts
            for(int p = 0; p < n; p++) {
                h[p] = 0.0;
            }
            for(int j = 0; j < n; j++) {
                for(int k = 0; k < n; k++) {
                    if(L[j][k] == 1) {
                        h[j] += a[k];
                    }
                }
            }//H step ends
            //Scaling A starts
            a_scale_factor = 0.0;
            a_sum_square = 0.0;
            for(int l = 0; l < n; l++) {
                a_sum_square += a[l]*a[l];
            }
            a_scale_factor = Math.sqrt(a_sum_square);
            for(int l = 0; l < n; l++) {
                a[l] = a[l]/a_scale_factor;
            }//Scaling A ends
            //Scaling H starts
            h_scale_factor = 0.0;
            h_sum_square = 0.0;
            for(int l = 0; l < n; l++) {
                h_sum_square += h[l]*h[l];
            }
            h_scale_factor = Math.sqrt(h_sum_square);
            for(int l = 0; l < n; l++) {
                h[l] = h[l]/h_scale_factor;
            }// Scaling H ends
            i++; // incr the interation counter
        } while( false == isConverged(a, aprev) || false == isConverged(h, hprev));
        System.out.println("Iter:    " + i);
        for(int l = 0; l < n; l++) {
            System.out.printf("
A/H[%d]=%.6f/%.6f\n",l,Math.round(a[l]*1000000.0)/1000000.0,Math.round(h[l]*1000000.0)/1000000.0)
;
        }
        return;
    }
    //Initialization
    for(int i = 0; i < n; i++)
    {
        h[i] = h0[i];
        a[i] = a0[i];
```

```
            hprev[i] = h[i];
            aprev[i] = a[i];
        }

        //Base Case
        System.out.print("Base:    0 :");
        for(int i = 0; i < n; i++) {
          System.out.printf("
A/H[%d]=%.4f/%.4f",i,Math.round(a0[i]*1000000.0)/1000000.0,Math.round(h0[i]*1000000.0)/1000000.0)
;
          //System.out.println("a0[" + i + "]= " + a0[i]);
        }
        if (iter != 0) {
            for(int i = 0; i < iter; i++) { //iteration starts
                //A step starts
                for(int p = 0; p < n; p++) {
                    a[p] = 0.0;
                }
                for(int j = 0; j < n; j++) {
                    for(int k = 0; k < n; k++) {
                        if(L[k][j] == 1) {
                            a[j] += h[k];
                        }
                    }
                }//A step ends
               //H step starts
                for(int p = 0; p < n; p++) {
                    h[p] = 0.0;
                }
                for(int j = 0; j < n; j++) {
                    for(int k = 0; k < n; k++) {
                        if(L[j][k] == 1) {
                            h[j] += a[k];
                        }
                    }
                }//H step ends
                //Scaling A starts
                a_scale_factor = 0.0;
                a_sum_square = 0.0;
                for(int l = 0; l < n; l++) {
                    a_sum_square += a[l]*a[l];
                }
                a_scale_factor = Math.sqrt(a_sum_square);
                for(int l = 0; l < n; l++) {
                    a[l] = a[l]/a_scale_factor;
                }//Scaling A ends
                //Scaling H starts
                h_scale_factor = 0.0;
                h_sum_square = 0.0;
                for(int l = 0; l < n; l++) {
                    h_sum_square += h[l]*h[l];
                }
                h_scale_factor = Math.sqrt(h_sum_square);
                for(int l = 0; l < n; l++) {
                    h[l] = h[l]/h_scale_factor;
                }// Scaling H ends
                System.out.println();
                System.out.print("Iter:    " + (i+1) + " :");
                for(int l = 0; l < n; l++) {
                    System.out.printf("
```

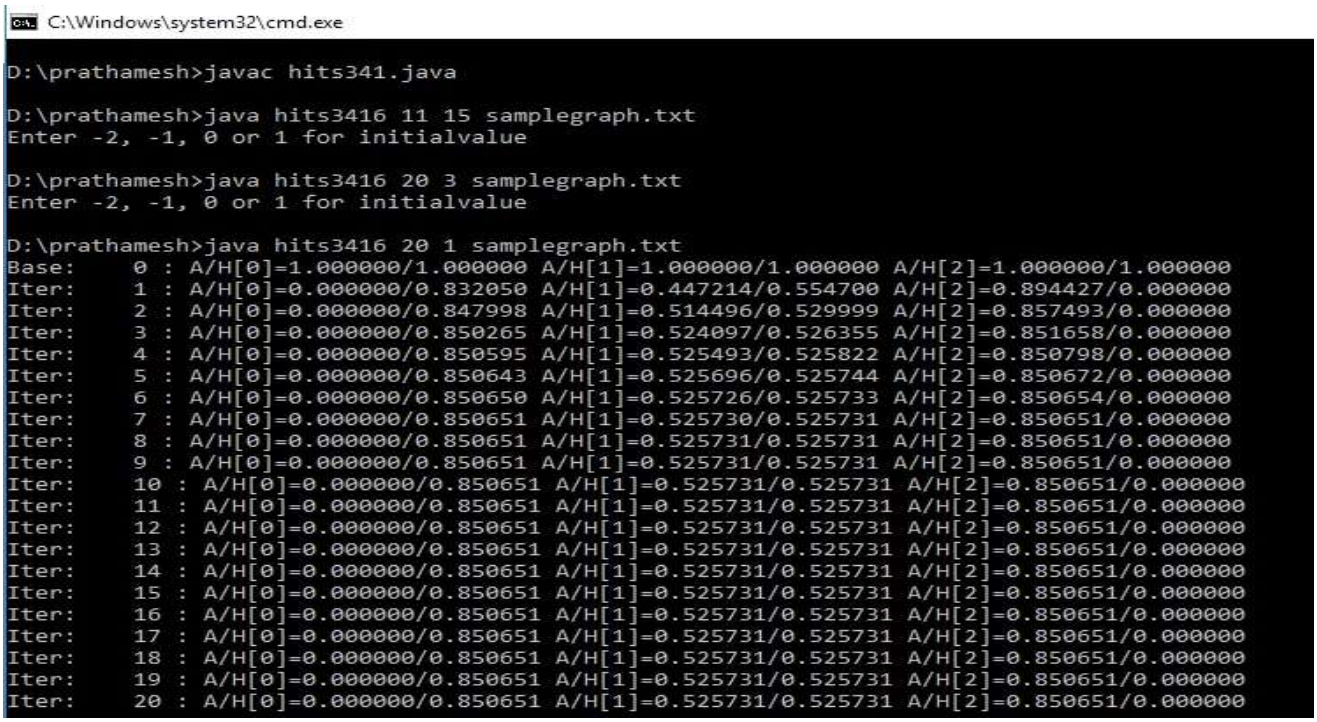
```
A/H[%d]=%.4f/%.4f",l,Math.round(a[l]*1000000.0)/1000000.0,Math.round(h[l]*1000000.0)/1000000.0);
            }
          }//iteration ends
       } // if iter != 0 ends
       else
       {
         int i = 0;
         do {
             for(int r = 0; r < n; r++) {
                aprev[r] = a[r];
                hprev[r] = h[r];
             }
             //A step starts
             for(int p = 0; p < n; p++) {
                a[p] = 0.0;
             }
             for(int j = 0; j < n; j++) {
                for(int k = 0; k < n; k++) {
                   if(L[k][j] == 1) {
                      a[j] += h[k];
                   }
                }
             }//A step ends
             //H step starts
             for(int p = 0; p < n; p++) {
                h[p] = 0.0;
             }
             for(int j = 0; j < n; j++) {
                for(int k = 0; k < n; k++) {
                   if(L[j][k] == 1) {
                      h[j] += a[k];
                   }
                }
             }//H step ends
             //Scaling A starts
             a_scale_factor = 0.0;
             a_sum_square = 0.0;
             for(int l = 0; l < n; l++) {
                a_sum_square += a[l]*a[l];
             }
             a_scale_factor = Math.sqrt(a_sum_square);
             for(int l = 0; l < n; l++) {
                a[l] = a[l]/a_scale_factor;
             }//Scaling A ends
             //Scaling H starts
             h_scale_factor = 0.0;
             h_sum_square = 0.0;
             for(int l = 0; l < n; l++) {
                h_sum_square += h[l]*h[l];
             }
             h_scale_factor = Math.sqrt(h_sum_square);
             for(int l = 0; l < n; l++) {
                h[l] = h[l]/h_scale_factor;
             }// Scaling H ends
             i++; // incr the interation counter
             System.out.println();
             System.out.print("Iter:    " + i + " :");
             for(int l = 0; l < n; l++) {
                System.out.printf("
A/H[%d]=%.4f/%.4f",l,Math.round(a[l]*1000000.0)/1000000.0,Math.round(h[l]*1000000.0)/1000000.0);
```

```
            }
        } while( false == isConverged(a, aprev) || false == isConverged(h, hprev));
      }
      System.out.println();
   }
}
```

13. Observations

1. The scores converged after 20 Iterations

14. Results:



References :

1. *Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze (2008). "Introduction to Information Retrieval". Cambridge University Press. Retrieved 2008-11-09.*
2. **Jump up^** *Kleinberg, Jon (December 1999). "Hubs, Authorities, and Communities". Cornell University. Retrieved 2008-11-09.*

Learning Outcomes Achieved:

1. Students developed logic for HITS implementation
2. The Hub and authority score for the given examples were evaluated.
3. HITS Score computation in Java was implemented
4. The implemented HITS instance converged after 20 iterations.

Conclusion:

1. Applications of the studied Technique in industry.
   a. HITS is used by leading search engines Ask.com.

2. Engineering relevance
   a. HITS is based on Hubs and authority based model.

3. Skills Developed
   a. Implementation of HITS algorithm.