Department of Computer Science and Engineering (Data Science)

Machine Learning – IV

Experiment 5

Name: Umang Kirit Lodaya **SAP ID: 60009200032**

Batch: D11

Aim:

To implement and gain a comprehensive understanding of the PageRank algorithm, a fundamental algorithm for ranking web pages based on their importance.

Theory:

Introduction to PageRank Algorithm:

- Developed by Larry Page and Sergey Brin at Google, PageRank is a link analysis algorithm used to assign a numerical weight to each element of a hyperlinked set of web pages.
- The algorithm assumes that more important pages are likely to receive more links from other pages.

Algorithm Overview:

- **Random Surfer Model:**
  - PageRank models a user who starts on a random page and follows links with a certain probability.
- **Link Matrix and Transition Probability:**
  - Represent the web as a matrix where each element (i, j) corresponds to the link from page i to page j.
  - Normalize the matrix to obtain the transition probability matrix.
- **PageRank Calculation:**
  - Iteratively compute the PageRank vector until convergence using the formula:
    - $PR(A)=(1-d)+d(L(B)PR(B)+L(C)PR(C)+\ldots)$
    - where $PR(A)$ is the PageRank of page A, $PR(B)$ is the PageRank of page B, $L(B)$ is the number of outbound links on page B, and d is the damping factor (typically set to 0.85).

Step-by-Step Implementation:

- **Step 1: Graph Representation**
  - Represent the web pages and their links using a graph structure.
- **Step 2: Transition Probability Matrix**
  - Create a matrix representing the transition probabilities between pages.
- **Step 3: Iterative PageRank Calculation**
  - Implement the iterative algorithm to compute PageRank until convergence.
- **Step 4: Damping Factor**
  - Integrate the damping factor into the algorithm.
- **Step 5: Convergence Criteria**
  - Define a convergence criterion to stop the iteration when PageRank values stabilize.

Implementation Tips:

- Use efficient data structures for the graph representation.
- Experiment with different damping factors and observe their impact on results.

Lab Experiments to be Performed in This Session:

Execute the PageRank algorithm on a dataset to gain insights into its functionality and operation.

NAME: UMANG KIRIT LODAYA

SAP ID: 60009200032

BATCH: K - K1 / D11

PAGE RANK ALGORITHM

In [1]:
```
import numpy as np
```

In [2]:
```
class PageRank:
    def __init__(self):
        self.__adj_list = {}
        self.__parent_list = {}
        self.__n = 0

    def addNode(self, node):
        self.__n += 1
        self.__adj_list[node] = self.__adj_list.get(node, [])
        self.__parent_list[node] = self.__parent_list.get(node, [])

    def addPath(self, u, v):
        self.__adj_list[u].append(v)
        self.__parent_list[v].append(u)

    def printAdjList(self):
        print("ADJCENCY LIST:")
        for key, value in self.__adj_list.items():
            print(f"{key} -> {value}")

    def __getM(self):
        M = [[0 for i in range(self.__n)] for j in range(self.__n)]
        for u in range(self.__n):
            n = len(self.__adj_list[u])
            for v in range(self.__n):
                if v in self.__adj_list[u]:
                    M[v][u] = 1 / n
                else:
                    M[v][u] = 0

        return np.array(M)

    def rankPages(self, iterations):
        R = [[1 / self.__n] for i in range(self.__n)]
        R = np.array(R)

        M = self.__getM()

        for i in range(iterations):
            R = np.matmul(M, R)
            for i in R:
                i[0] = round(i[0], 3)

        R = [i[0] for i in R]
```

```
        print("FINAL R VECTOR:")
        print(*list(map(list, enumerate(R))), sep="\n")
        print()

        print(f"PAGE WITH MAXIMUM PAGERANK SCORE: {np.argmax(R)}")

    def rankPagesWithBeta(self, iterations, beta):
        R = [1 / self.__n for i in range(self.__n)]
        R = np.array(R)

        for _ in range(iterations):
            for i in range(self.__n):
                r = (1 - beta)
                for j in self.__parent_list[i]:
                    r += (beta * R[j] / len(self.__adj_list[j]))

                R[i] = round(r, 3)

        print("FINAL R VECTOR:")
        print(*list(map(list, enumerate(R))), sep="\n")
        print()

        print(f"WITH B = {beta}, PAGE WITH MAXIMUM PAGERANK SCORE: {np.argma
```

In [3]:

```
PR = PageRank()
PR.addNode(0)
PR.addNode(1)
PR.addNode(2)

PR.addPath(0, 0)
PR.addPath(0, 1)
PR.addPath(0, 2)

PR.addPath(1, 0)
PR.addPath(1, 2)

PR.addPath(2, 1)
PR.addPath(2, 2)

PR.printAdjList()
print()
PR.rankPages(3)
```

```
ADJCENCY LIST:
0 -> [0, 1, 2]
1 -> [0, 2]
2 -> [1, 2]

FINAL R VECTOR:
[0, 0.235]
[1, 0.304]
[2, 0.462]

PAGE WITH MAXIMUM PAGERANK SCORE: 2
```

In [4]:

```
PR = PageRank()
PR.addNode(0)
PR.addNode(1)
PR.addNode(2)
PR.addNode(3)
PR.addNode(4)
PR.addNode(5)
```

```python
PR.addPath(0, 1)
PR.addPath(0, 2)

PR.addPath(1, 0)
PR.addPath(1, 3)
PR.addPath(1, 4)
PR.addPath(1, 5)

PR.addPath(2, 0)
PR.addPath(2, 5)

PR.printAdjList()
print()
PR.rankPagesWithBeta(2, 0.8)
```

```
ADJCENCY LIST:
0 -> [1, 2]
1 -> [0, 3, 4, 5]
2 -> [0, 5]
3 -> []
4 -> []
5 -> []

FINAL R VECTOR:
[0, 0.392]
[1, 0.357]
[2, 0.357]
[3, 0.271]
[4, 0.271]
[5, 0.414]

WITH B = 0.8, PAGE WITH MAXIMUM PAGERANK SCORE: 5
```