```
#include <iostream>
using namespace std;

class WebCrawler {
private:
    int vertex;          // number of web pages
    int edge;            // number of hyperlinks
    int adj[20][20];     // adjacency matrix
    bool visited[20];    // visited array

public:
    void input() {
        cout << "Enter number of web pages (vertices): ";
        cin >> vertex;

        cout << "Enter number of hyperlinks (edges): ";
        cin >> edge;

        // Initialize adjacency matrix and visited array
        for (int i = 0; i < vertex; i++) {
            for (int j = 0; j < vertex; j++) {
                adj[i][j] = 0;
            }
            visited[i] = false;
        }

        cout << "\nEnter hyperlinks (from to) format (0-indexed):\n";
        for (int i = 0; i < edge; i++) {
            int u, v;
            cin >> u >> v;
            adj[u][v] = 1; // directed edge
        }
    }

    void display() {
        cout << "\nAdjacency Matrix (Web Page Links):\n";
        for (int i = 0; i < vertex; i++) {
            for (int j = 0; j < vertex; j++) {
                cout << adj[i][j] << " ";
            }
            cout << endl;
        }
    }
```

```
    void BFS(int start) {
        int queue[20];     // array to simulate a queue
        int front = 0, rear = 0;

        visited[start] = true;
        queue[rear++] = start;  // enqueue start vertex

        cout << "\nBFS Traversal (Indexing Order): ";

        while (front < rear) {  // while queue not empty
            int current = queue[front++];  // dequeue
            cout << "Page " << current << " -> ";

            for (int i = 0; i < vertex; i++) {
                if (adj[current][i] == 1 && !visited[i]) {
                    visited[i] = true;
                    queue[rear++] = i;   // enqueue neighbor
                }
            }
        }

        cout << "End\n";
    }
};

int main() {
    WebCrawler crawler;
    crawler.input();
    crawler.display();

    int start;
    cout << "\nEnter starting web page (seed URL index): ";
    cin >> start;

    crawler.BFS(start);

    return 0;
}
```

OUTPUT:-

Enter number of web pages (vertices): 5
Enter number of hyperlinks (edges): 6
Enter hyperlinks (from to) format (0-indexed):

0 1
0 2
1 3
2 3
3 4
4 0
Enter starting web page (seed URL index): 0

Adjacency Matrix (Web Page Links):
0 1 1 0 0
0 0 0 1 0
0 0 0 1 0
0 0 0 0 1
1 0 0 0 0

BFS Traversal (Indexing Order):
Page 0 -> Page 1 -> Page 2 -> Page 3 -> Page 4 -> End