Problem analysis: Identify, formulate

Page :
Date : / /

ASSIGNMENT 6

Aim: Consider a friend's network on facebook social website. Model it as a graph to represent each node as a user and a link to represent the friend relationship between them. Store data such as date of birth, number of ~~comments~~ comments for each user.

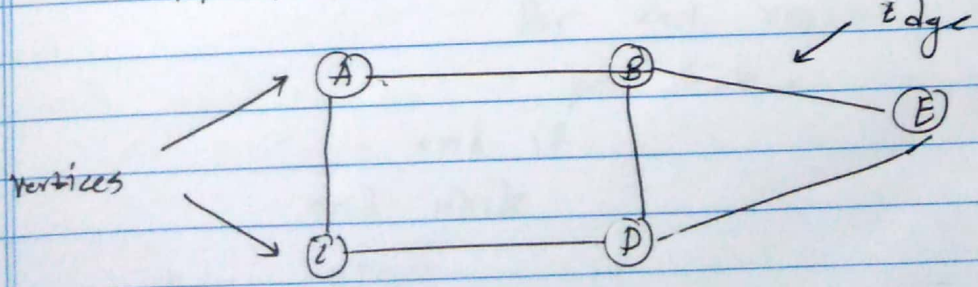
1. Find who is having maximum friends
2. Find who has post maximum and minimum comments
3. Find users having birthday in this month.

Objective:
- To study Graph theory
- To study different graph traversal methods
- To understand the real time applications of graph theory.

Theory:

Graph: A graph G consist of two sets V and E where, V is finite non-empty set of vertices and E is set of pair of vertices called edges. V(G) represents set of edges in graph G.

eg: This graph G can be defined as G = (V, E) where V = {A, B, C, D, E} and E = {(A, B), (A, C), (B, D), (C, D), (B, E), (E, D)}



Page :
Date :

Graph presentation:

Graph can be represented in two ways:
- Adjacency Matrix
- Adjacency List

Traversal of a graph:

Let $G = (V, E)$ be an undirected graph and a vertex v in $V(G)$. There are two ways to visit all the vertices of graph:
- Depth first search (DFS)
- Breadth first search (BFS)

DFS: A depth first search is an algorithm for traversing a finite graph. DFS visits the child vertices before visiting the sibling vertices; that is, it traverses the depth of any particular path before exploring its breadth. A stack is generally used when implementing the algorithm. When there are no adjacent, or visited nodes, then we proceed backwards (backtrack), where in repeat the process.

BFS: Bfs algorithm traverses a graph in a breadth ward motion and uses a queue to remember to get the next vertex to start a search, when a dead end occurs in any iteration. A breadth-first search is another technique for traversing a finite graph. BFS visits the neighbor vertices before visiting the child vertices, and a queue is used in the search process. This algorithm is often used to find the shortest path from one vertex to another.

Algorithm:

The DFS Algorithm:

```
DFS (G, v)    (v is the vertex where search starts)

    Stack S := { };  (Stack with an empty stack)
    for each vertex u, set visited[u] := false;
    push S, v;
    while (S is not empty) do
        u := pop S;
        if (not visited[u]) then
            visited[u] := true;
            for each unvisited neighbour w of u
                push S, w;
        end if
    end while
END DFS()
```

The BFS algorithm:

1. Accept the number of vertices from the user, say n
2. Create a list having n nodes. This list is called as a header list. It will be connected by the down pointer whereas the adjacency list will be connected by the next pointer. Remember that these two lists will follow different structures.
3. Initialise the visited array v to zeroes.
4. Accept the graph
   - Accept an edge say i, j.
   - Search in the header list for vertex i, and in the adjancency list of that vertex i, attach a node of vertex j.
   - Search in the header list for vertex j, and in the adjacency list of that vertex j, attach a node of vertex i.
   - if more edges then goto step (a).
5. Accept the starting vertex say i.
6. Push i to the queue, and mark it as visited i.e. v[i] = 1
7. Pop a vertex from queue, say i.
8. Print i.
9. Move in the adjacency list of i^{th} vertex and for every node say j which is not visited push it to the queue and mark it as visited, i.e v[j] = 1
10. If queue is not empty repeat from step 7
11. Now check whether all vertices are visited.

a. Initialize j=0 and flag = -1
b. if j^{th} vertex is not visited, set flag to j.
c. Increment j, and if number of vertices is not over, repeat from step b.
12. If flag != 1 the graph is not a connected graph, otherwise it is a connected graph.
13. Stop.

Advantages of DFS:

- Requires less memory than BFS since only need to remember the current path.
- If lucky, can find a solution without examining much of the state space.
- With cycle-checking, looping can be avoided.

Advantages of BFS:

- Guaranteed to find a solution if one exists - in addition, find optimal solution first.
- It will not get lost in a blind alley
- Can add cycle checking with very little cost.

Page :
Date : / /

Conclusion: BFS and DFS algorithm is implemented for graph traversal.