How can we do it better?

==========

Improved Search Algorithm

Assumptions: Do not allow any node twice

Step1. Explored= {S}, Unexplored= ( ).

Step2. Search for an edge E= (u, v) where u∈ Explored, v ∈ Unexplored

if $\exists E$, mark v and Explored= Explored ∪ v

Step3: Stop where |explored|= |v|

==========

Good news: It is better than the previous version.

But why?

Analysis of the improved graph search algorithm

Assume: adjacency matrix data structure is used

Worst case: When n-1 nodes are already visited, i.e. $|Explored| = n - 1$, to find the next vertex v adjacent to any node $\in$ Explored, we need to scan n-1 rows to find it and each row contains n entries because no order is followed.

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=1}^{i} n = n + 2n + .. + n(n-1) = O(n^2)$$

Running time : $O(n^2)$, n = number of vertices

Assume: Adjacency list data structure is used

When n-1 nodes are already visited, i.e. $|Explored| = n - 1$, to find the next vertex v adjacent to any node $\in$ Explored, we need to scan n-1 rows to find it and each row contains d entries, where d is the degree of the corresponding vertex.

Running time : $O(n^2)$, n = number of vertices

Assume: Adjacency list data structure is used

When n-1 nodes are already visited, i.e. $|Explored| = n - 1$, to find the next vertex v adjacent to any node $\in$ Explored, we need to scan n-1 rows to find it and each row contains d entries, where d is the degree of the corresponding vertex.

$$T(n) = \sum_{i=1}^{n-1} \sum_{j=1}^{i} d_i = \sum_{j=1}^{1} d_i + \sum_{j=1}^{2} d_i + \sum_{j=1}^{2} d_i - - - \sum_{j=1}^{n-1} d_i$$

$$=$$

$$T(n) = \sum_{j=1}^{1} d_i + \sum_{j=1}^{2} d_i + \cdots \sum_{j=1}^{n-2} d_i + \sum_{j=1}^{n} d_i - (n-1)$$

$$< \sum_{j=1}^{1} d_i + \sum_{j=1}^{2} d_i + \cdots \sum_{j=1}^{n-2} d_i + \sum_{j=1}^{n} d_i$$

$$< \sum_{j=1}^{n} d_i + \sum_{j=1}^{n} d_i + \cdots \sum_{j=1}^{n} d_i + \sum_{j=1}^{n} d_i = 2m(n-1) = O(mn)$$

m: number of edges

How can we do it better?

Graph Search using Breadth First Search(BFS) Algorithm

[All nodes are initially unexplored]

Queue data structure(FIFO) is used to keep track of order of searching nodes.

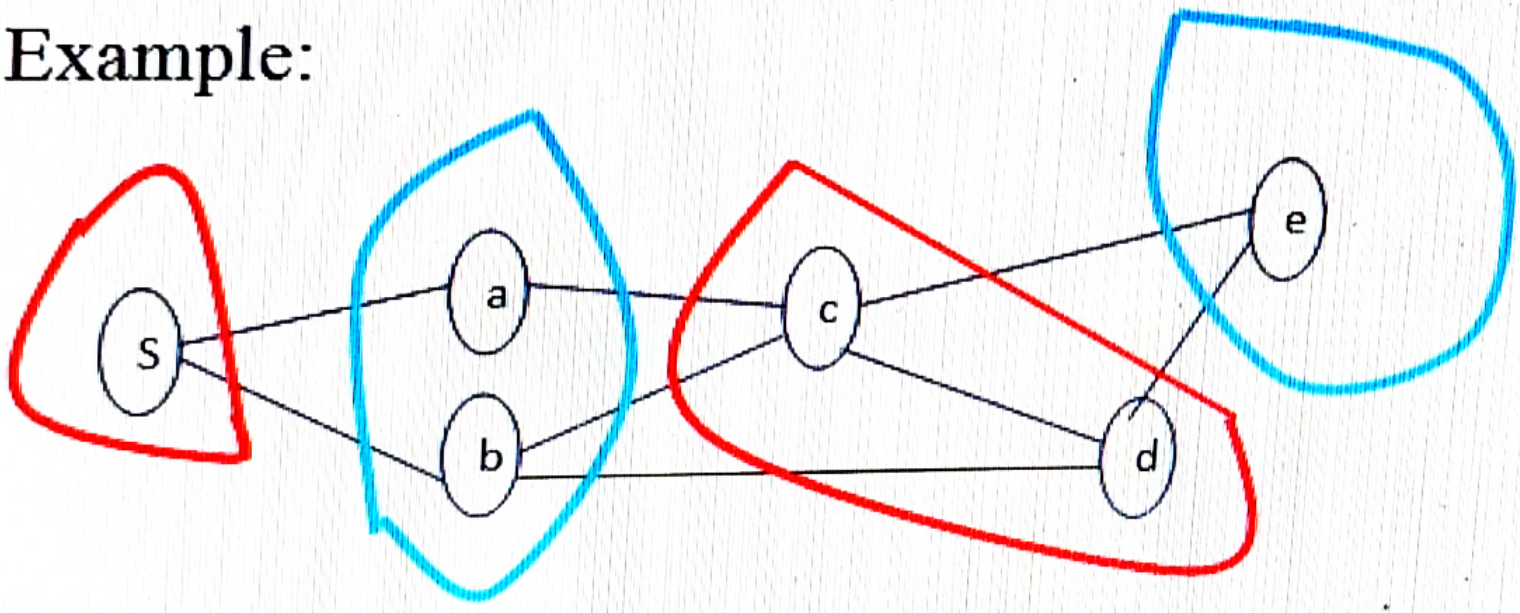
Step1: Mark S as explored.

Let Q = queue data structure (FIFO) initialized with S

Step2: While Q $\neq \emptyset$

- Remove the first node of Q, call it v.
- For each edge (v, w)
  - if w is unexplored
  - Mark w as explored
  - Add w to Q (at the end)

Example:



Example 2

Total number of times, the loop is executed

3+2+2+3= 10= 2 * 5 = 2m, m=5 (number of edges)

Analysis of BFS algorithm

Inner Loop: "*For each edge (v, w)*" is executed in total $\sum_{j=1}^{n} d_i$= 2m times (when Q $\neq \emptyset$ is reached)

The line "*Remove the first node of Q, call it v*" is executed n times

Total running time: O(n+m).