26,27
28,29,30

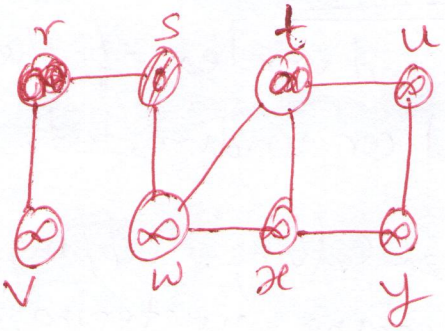
Graph Searching Algorithms

Systematic search of every edge and vertex of the graph.
Graph $G = (V, E)$ is either directed or undirected.
Applications
  - Compilers, Graphics, Maze-Solving, Mapping,
  - Networks: routing, searching, clustering, etc.
Assume an adjacency list representation.

Breadth First Search (BFS)

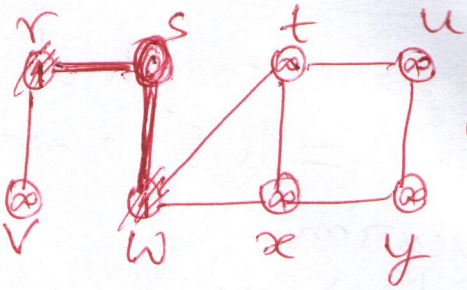
A Breadth-first search (BFS) traverses a connected component of a graph, and in doing so defines a spanning tree with several useful properties.
BFS in an undirected graph G is like wandering in a labyrinth with a string.
The starting vertex s, it is assigned a distance 0.
In the first round, the string is unrolled the length of one edge, and all of the vertices that are only one edge away from the anchor are visited (discovered), and assigned distances of 1.
In the second round, all the new vertices that can be reached by unrolling the string 2 edges are visited and assigned a distance of 2.
This continues until every vertex has been assigned a level.
The ~~vertex of~~ label of any vertex v corresponds to the length of the shortest path (in terms of edges) from s to v.

BFS Example.

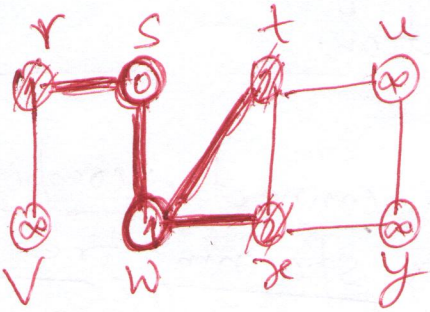


Queue

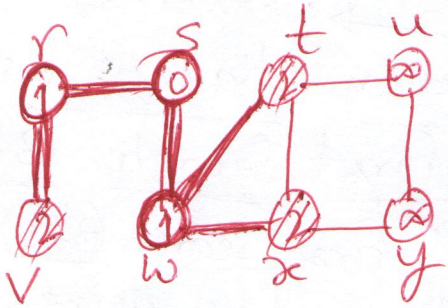
Q | s |
0



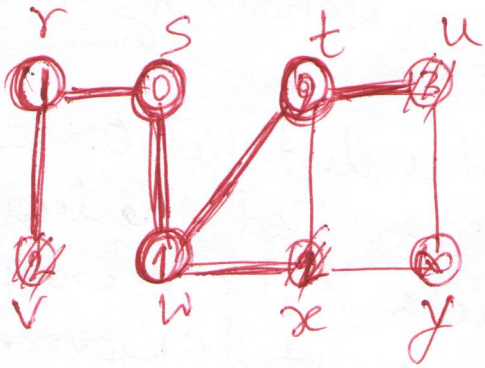
Q | w | r |
1 1



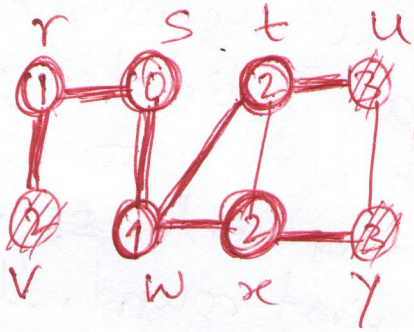
Q | r | t | x |
1 2 2



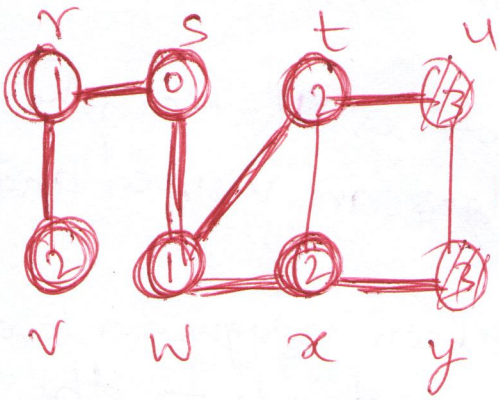
Q | t | x | v |
2 2 2



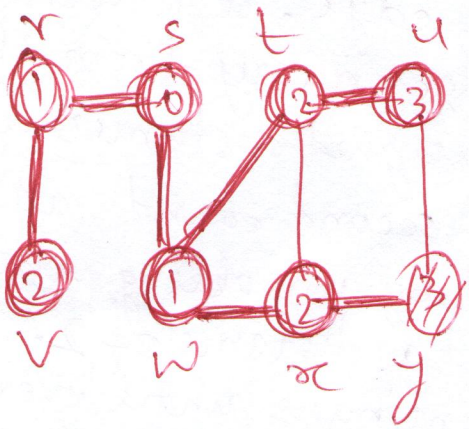
Q | x | v | u |
2 2 3



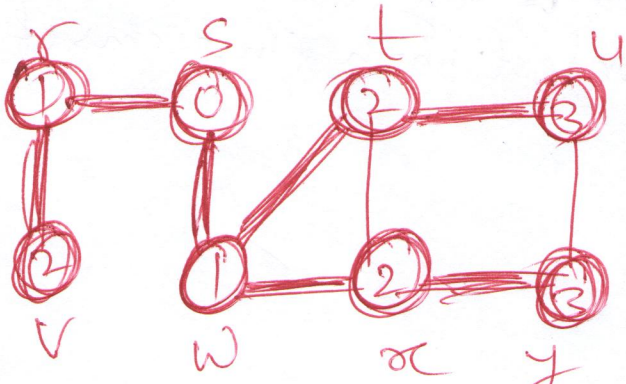
Q | v | u | y |
2 3 3



Q | u | y |
3 3



Q | y |
3



Q | |
↑ Empty

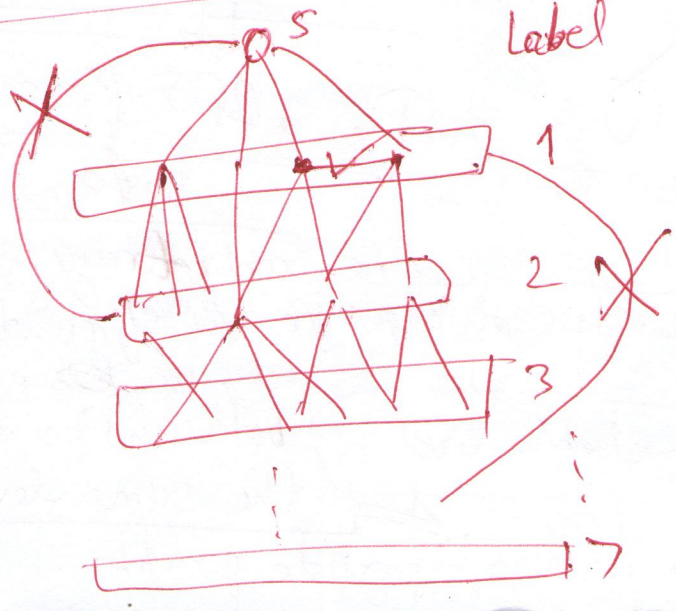
procedure will stop.

BFS Running Time

Given a graph $G=(V,E)$
Vertices are enqueued if there color is white.
Assuming that en- and dequeuing takes $O(1)$ time the total cost of this operation is $O(V)$.
Adjacency list of a vertex is scanned when the vertex is dequeued (and only then....)
The sum of the lengths of all lists is $\theta(E)$.
Consequently, $O(E)$ time is spent on scanning them.
Initializing the algorithm takes $O(V)$.
Total running time $O(V+E)$ (linear in the size of the adjacency list representation of G).

BFS Properties

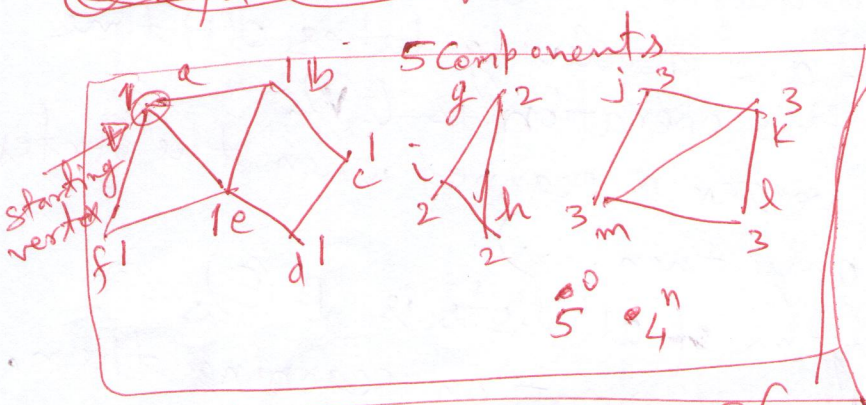
Given an undirected graph $G=(V,E)$, BFS discovers all vertices reachable from a source vertex s.
For each vertex at level i, the path of the BFS tree T between s and v has i edges, and any other path of G between s and v has at least i edges.
If (u,v) is an edge then the level numbers of u and v differ by at most one.
It computes the shortest distance to all reachable vertices.



① Two Applications of BFS

① Finding Connected Component in a graph.

~~Bipartite Graphs~~



for selection of compon

a	j	b	g	k	c	h	l	d	i	e	m	f	o
1	3	1	2	3	1	2	3	1	2	1	4	3	1	5

label

If two vertices have same label, means they are in the same connected components.

Total Running Time = $O(m+n)$

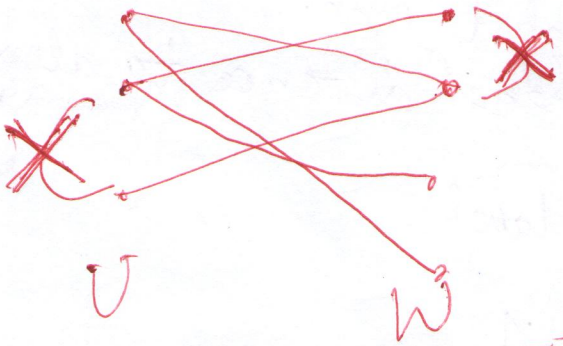
② Bipartite Graphs.

$G = (V, E)$ undirected Graphs

\# G is bipartite graph if $\exists$ a partition of V into U, W

$$V = U \cup W$$
$$U \cap W = \phi$$

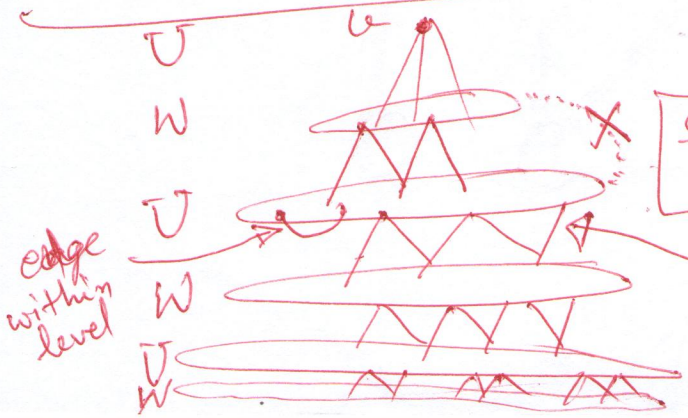
s.t.
every edge has one end point in U and the other in W.



~~Given a graph~~ graph connected

⇒ Given a graph (G), is G bipartite?

Do a BFS from any node (i.e. arbitrary vertex v).



Suppose all edges go b/w adjacent levels.
i.e., no edge has both end points in the same level.

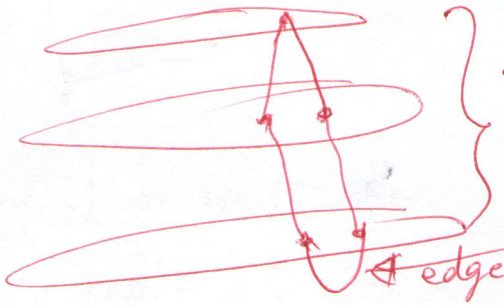
edge b/w level.

G is bipartite graph.

Q) If in the BFS, we find an edge both of whose endpoints are in the same level, then what?

⇒ Then G is not bipartite. But, why?

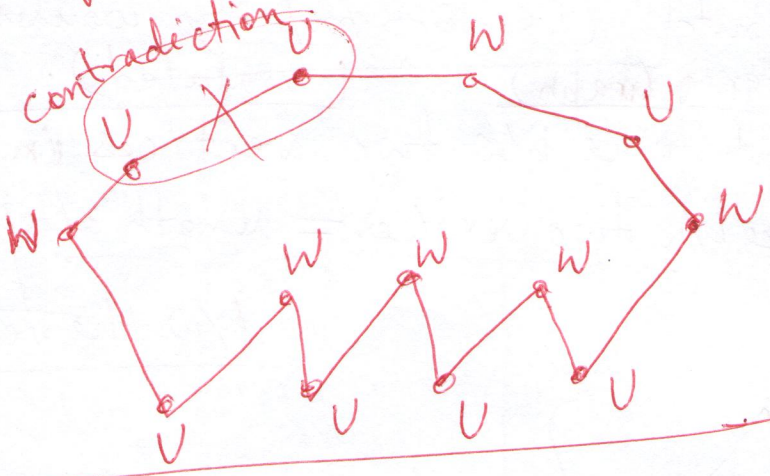
Each edge has a predecessor except source.



The cycle will always has odd length.

Then G has an odd cycle (i.e., odd number of edges in the cycle).

⇒ If G has an odd cycle, then G can not be bipartite.

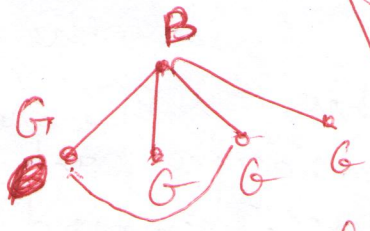


Q) If all cycles in a graph G are even, is G bipartite?

⇒ Do the BFS and observe that there is no edge having both end points in the same level.

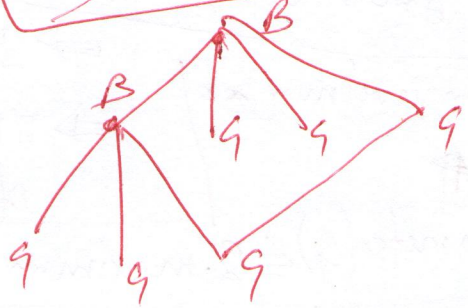
Bipartite

Running Time = O(BFS)

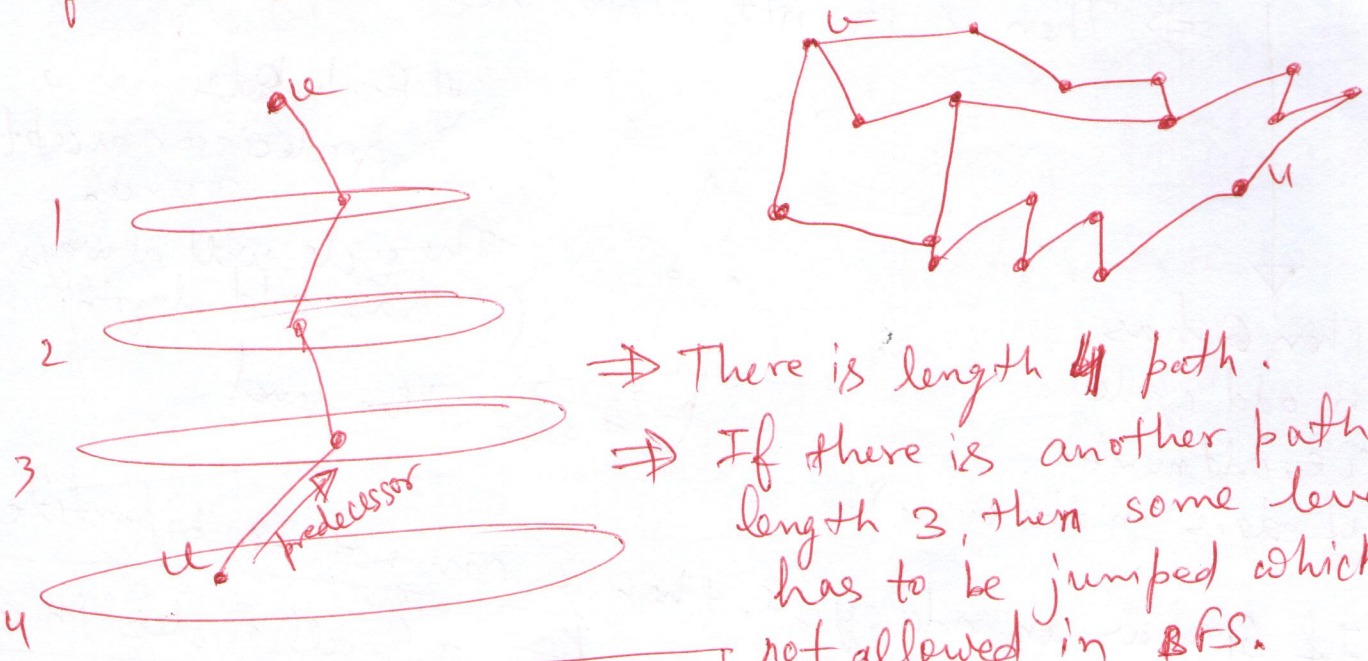


If the level number is same for two vertices of an edge, declare the graph as non-bipartite.

While doing BFS for a node, just check if other end point is gray, it means there will be an edge in the same level, and we can declare the graph as not Bipartite.



⑥ In a BFS starting from vertex v, the level number of vertex u is the length of the shortest path from v to u.



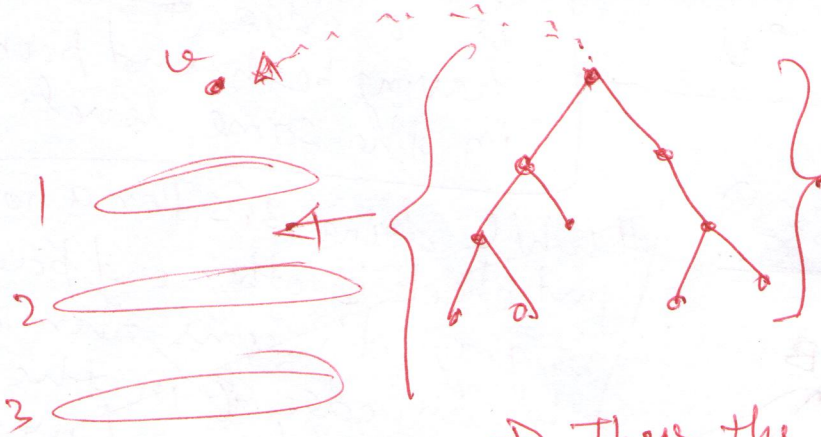
⇒ There is length 4 path.

⇒ If there is another path of length 3, then some level has to be jumped which is not allowed in BFS. (i.e., BFS property would be violated).

Another Application (Compute Diameter of Graph)

diameter(G) = maximum distance b/w two vertices in G.

where, distance b/w two vertices = length of shortest path b/w the vertices.



⇒ Diameter is six.

So BFS at a single node will not give the diameter of a graph.

⇒ Thus the BFS should be done by considering each node as reference once. Then, find the maximum level in all BFS's to get the diameter.

⇒ Diameter is defined only for connected graph.

⇒ The diameter can not be more than twice of maximum level of any BFS of that graph.

Time Complexity = V times BFS Complexity

maximum level number ≤ diameter(G) ≤ 2 · maximum level in any BFS.