Introduction to algorithms (IT301)
(July-November, 2010):
Breadth First Search (BFS)

- In BFS the edges along which traversal leads to the discovery of a new vertex are called tree edges. The set of tree-edges induce a tree called the BFS-tree.
- The subgraph induced by all the tree edges is a tree rooted at the source vertex $s$.
- The algorithm outputs the array $d$ which contains for each vertex the length of a shortest path from the source vertex to that vertex.
- An actual shortest path is obtained for each vertex by traversing recursively the parent pointers till we reach the root.
- The algorithm is quite similar to the algorithms of Prim and Dijkstra.
- Prim’s algorithm and Dijkstra’s algorithm use priority queues to pick vertices on the basis of some minimisation of weight of incident edges.
- BFS uses queues to manage vertices in first-in first-out manner.
- The incoming vertices are in order of distance from the source node, enforced by selecting undiscovered nodes in the adjacency list of discovered nodes. Among the nodes at the same level, the queue is used to ensure the discovery from each proceeds in the same order in which the vertices were discovered. This also helps in ensuring that the algorithm finds the shortest path to every vertex reachable from the source.
- Every nontree edge connecting a pair of nodes neither of which is ancestor or descendant of the other are called cross edges.
• Any nontree edge connecting a descendant node to an ancestor is called a **back edge** while one connecting an ancestor node to a descendant node is called a **forward edge**. These two are obviously the same for an undirected graph. Fortunately there is no confusion, because the **BFS** of an undirected graph has only tree-edges and cross-edges.

• In the **BFS** of an undirected graph, there are only tree-edges and cross-edges. Cross-edges in the **BFS** of an undirected graph always connects two vertices which are either at the same level or one level apart. The former are called **horizontal cross-edges** while the latter are called **vertical cross-edges**.

• Adding one cross edge to the **BFS**-tree creates exactly one cycle. This is the **fundamental cycle** with respect to that edge and that **BFS** tree.

• Since the **BFS** algorithm finds the distance to all vertices reachable from the source, those with $d$ field $\infty$ at the end are not reachable from the source. Thus we can find the connected components of a graph by repeatedly running **BFS**. The first execution gets the component containing the source. We remove all vertices with finite $d$ values and recurse on the remaining vertices using a new source vertex in the reduced subgraph.

• We can also find the Least-Common-Ancestor (**LCA**) of a pair of vertices in the tree by following the paths from the two vertices to the root till we reach the first common vertex. The fundamental cycle formed by a cross-edge is obtained this way. It is an odd-cycle if we start with a horizontal cross-edge and an even-cycle if we start with a vertical cross-edge.

• It follows that the **BFS** of a bipartite graph has no horizontal cross-edges.