INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH & DEVELOPMENT

ISSN: 2278 – 0211 (Online)

Searching Techniques In Computing

Dr. G. Srinivasu
Department Of Mathematics, R.S.R. Engineering College, Kadanuthala
SPSR Nellore District, A. P., India
B. V. V. Prasad
Department Of Mathematics, M.B.P.S. Govt., Polytechnic College
Nallapadu, Guntur District, A. P., India

Abstract:
In mathematics and computer science, graph theory is the study of graphs. Mathematical structures are used to model pairwise relations between objects from a certain collection. A graph in this context refers to a collection of vertices or nodes and a collection of edges that connect pairs of vertices. A graph may be undirected, which means that there is no distinction between the two vertices associated with each edge or its edges may be directed from one vertex to another vertex. The development of graphs therefore of major interest in computer science. In this paper , for mathematical computing, the two search methods namely BFS, DFS are thoroughly analyzed by means of definitions, analysis of the algorithms followed by findings and observations.

1.Introduction
A search of a graph is a methodical exploration of all the vertices and edges. It must run in "linear time", i.e., in one pass or a small number of passes over the graph. Even with this restriction, a surprisingly large number of fundamental graph properties can be tested and identified.
This paper examines the two most important search methods. Breadth-first search gives an efficient way to compute distances. Depth-first search is useful for checking many basic connectivity properties, for checking planarity, and also for data flow analysis for compilers. A treatment of at least some aspects of both these methods can be found in almost any algorithms text . All the algorithms of this paper run in linear time or very close to it. Since it takes linear time just to read the graph, the algorithms are essentially as efficient as possible.

2.Notation
Throughout this paper, the number of vertices and edges of a graph G = (V, E) is denoted n and m, respectively. Time bounds for algorithms are given using asymptotic notation, e.g., 0(n) denotes a quantity that, for sufficiently large values of n, is at most c_n ,which denotes for some constant c that is independent of n.

3.Convention
In all algorithms, we assume that the graph G is given as an adjacency list representation. If G is undirected, this means that each vertex has a list of all its adjacent neighbor. The list can be sequentially allocated or linked. If G is directed, then each vertex has a list of all its adjacent neighbors.

4.Breadth-First Search
The breadth-first search method finds shortest paths from a given vertex of a graph to other vertices. It generalizes to Dijkstra's algorithm, which allows numerical edge-lengths. Throughout this paper, the given graph G can be directed or undirected.

4.1.Definitions

A length function on a graph specifies the numerical length of each edge. Each edge is assumed to have length one, unless there is an explicitly declared length function.

The distance from vertex a to vertex v in a graph, denoted d(u, v), is the length of a shortest path from a to v.

The diameter of a graph is the maximum value of d for u not equal to v.

A shortest-path tree T from a vertex s is a tree, rooted at s, that contains all the vertices that are reachable from s. The path in T from s to any vertex x is a shortest path in G, i.e., it has length d(s, x)

4.2.Examples

Figure 1 gives a shortest path tree from vertex s.

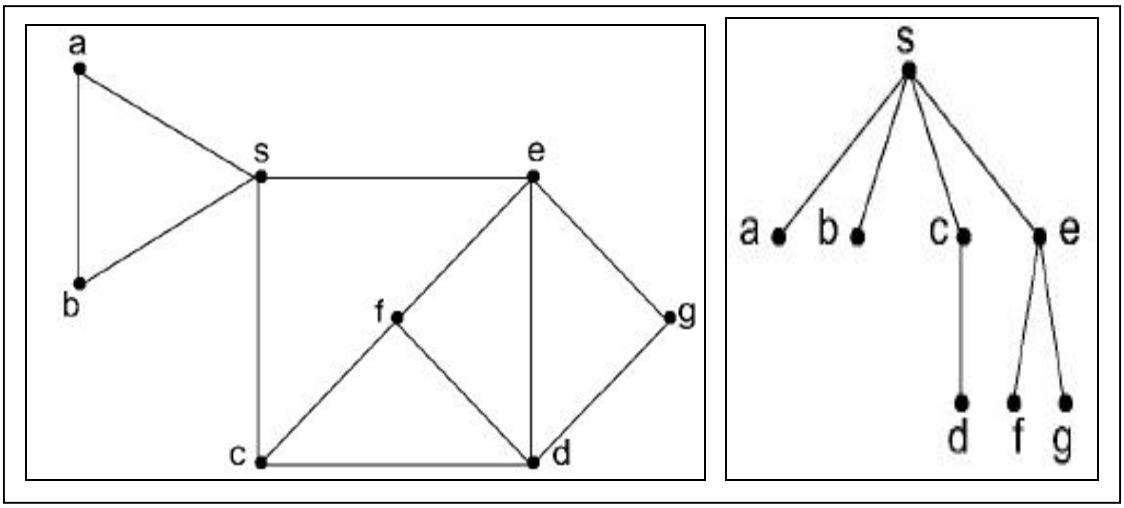

Figure 1: Undirected Graph And Shortest Path Tree

The small-world phenomenon [2],[3] occurs when relatively sparse graphs have low diameter. Studies have shown that the graphs of movie actors, neural connections in the c.-elegans worm, and the electric power grid of the western United States all exhibit the small-world phenomenon. The world-wide web is believed to have this structure too.

For several decades, mathematicians have computed their Erdds number as their distance from the prolific mathematician Paul Erdos, in the graph where an edge joins two mathematicians who have co-authored a paper.

The premise of the 6 Degrees of Kevin Bacon game is that the graph whose vertices are movie actors and whose edges join two actors appearing in the same movie has diameter at most 6.

In computer and communications networks, a message is typically broadcast from one site s to all others by passing it down a shortest path tree from s.

To solve a puzzle like Sam Lloyd's "15 puzzle" [4], we can represent each position by a vertex. A directed edge (i, j) exists if we can legally move from i to j. We seek a shortest path from the initial position to a winning position.

5.Ordered Trees

5.1.Definitions

In a rooted tree, a vertex x is an ancestor of a vertex y, and y is a descendant of x, if there is a path from x to y whose edges all go from parent to child. By convention x is an ancestor and descendant of itself. In the tree of Figure 1vertex e has 3 descendants.

Vertex x is a proper ancestor or proper descendant of vertex y if it is an ancestor or descendant and x is not equal to y.

An ordered tree is a rooted tree in which the children of each vertex are linearly ordered. In a plane drawing of such a tree, left-to-right order gives the order of the children.

Vertex x is to the left of vertex y if some vertex has children c and d, with c to the left of d, c an ancestor of x and d an ancestor of y.

In a graph G, a breadth-first tree T from a vertex s contains the vertices that are reachable from s. It is an ordered tree, rooted at s. If x is a vertex at depth S in the tree T, then the children of x in T are the vertices of G that are adjacent in G to x, but not adjacent in G to any vertex in T at depth less than S, or to any vertex at depth S in T that is at the left of x.

5.2.Inference
Any breadth-first tree is a shortest-path tree.
A high level bfs algorithm is given below as Algorithm 1.1.1. It constructs a breadth-first tree. It starts from s, finds the vertices at distance 1 from s, then the vertices at distance 2, etc.

Algorithm1.1: Breadth-first Search
Input: directed or undirected graph G = (V, E), vertex s Output: breadth-first tree T from s
= {all vertices at distance i from s}
Vo={s}
make s the root of T
i=0
while V not equal 2pi do /* construct vi+1 */
Vi+1=pi
for each vertex v E Vi do /* "scan" v */
for each edge (v, w) do if w U3 . 1 then
make w the next child of v in T
add w to V+l
i=i+1

The high-level algorithm can be implemented to run in total time O(n+m).The main data structure is a queue of vertices that have been added to T, but whose children in T have not been computed.
In general we verify that an algorithm takes time O(n+m)by checking that it spends constant time i.e., 0(1) time on each vertex and edge of G.
Not every shortest path tree is a breadth-first tree . This does not cause any problems in applications.
The diameter can be found by doing a breadth-first search from each vertex.
Dijkstra's algorithm computes a shortest path tree from s in a graph with a nonnegative length function. It generalizes breadth-first search. Like BFS it finds the set Vd of all vertices at distance dfrom s, for increasing values of d.An appropriate data structure implements the algorithm in time O(m+nlog n).

6.Depth-First Search
Depth-first search was investigated in the 19th century as a strategy for exploring a maze .The fundamental properties of the depth-first search tree was discovered by Hopcroft and Tarjan [1], [5]. Tarjan also developed many other elegant and efficient dfs algorithms. The idea of depth-first search is to scan repeatedly an edge incident to the most recently discovered vertex that still has unscanned edges.

6.1.Definitions
Two vertices in a tree are related if one is an ancestor of the other.
In an undirected graph G=(V,E),a depth-first tree from a vertex s is a tree subgraph T, rooted at s, that contains all the vertices of Gthat are reachable from s.
Edges of E(T)and E(G)-E(T)are called tree edges and nontree edges, respectively.
Each nontree edge is also called a back edge.
The crucial property is that the two endpoints of each back edge are related.
In an undirected graph G,a depth-first spanning forest is a collection of depth-first trees, one for each connected component of G.Each vertex of Gbelongs to exactly one tree of the forest.
Let G=(V,E)be a directed graph where every vertex is reachable from a designated vertex s. A depth-first tree from s is an ordered tree in G,rooted at s that contains all vertices V. Each edge of T is called a tree edge. Each nontree edge (x, y) belongs to E-T can be classified into one of three types:

- A back edge has y an ancestor of x.
- A forward edge has y a descendant of x.
- A cross edge joins two unrelated vertices.

The crucial property is that each cross edge (x, y) has x to the right of y.
Let G be a directed graph, in which we no longer assume that some vertex can reach all others. A depth-first forest is an ordered collection of trees in G so that each vertex of G belongs to exactly one tree. The edges of G are classified into the 4 types of edges in definition mentioned above with one additional possibility:
A cross edge can join 2 vertices in different trees as long as it goes from right to left i.e., from a higher numbered tree to a lower numbered tree.

6.2.Examples

Figure 2 illustrates a depth-first search of an undirected graph. In drawings of depth-first spanning trees, tree edges are solid and nontree edges are dashed. There can be many depth-first trees with the same root. For instance the tree edge (5, 6) could be replaced by (5, 7).

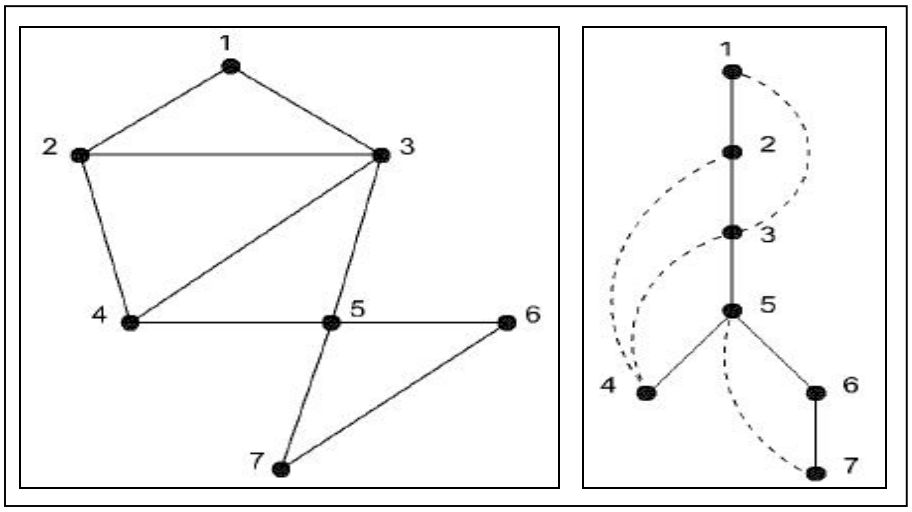

Figure 2: Undirected Graph And Depth-First Spanning Tree

Figure 3 illustrates a depth-first search of a directed graph. There is 1 forward edge, 2 back edges and 2 cross edges.

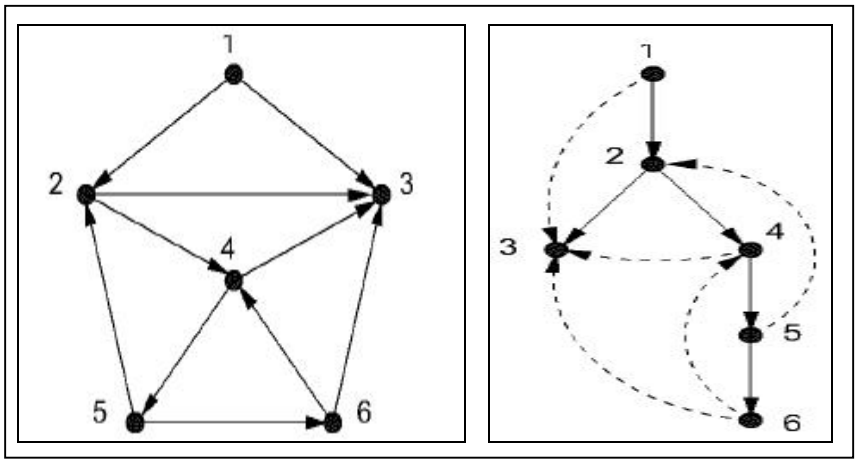

Figure 3: Directed Graph And Depth-First Spanning Tree

6.3.Inference

Any vertex s of an undirected graph has a depth-first tree from s. Any vertex s of a directed graph has a depth-first tree of the subgraph induced by the vertices reachable from s. A high level algorithm to find such a tree is the following.

Algorithm 1.2: Depth-First Search
Input: directed or undirected graph G = (V, E), vertex s Output: depth-first tree T from s
make s the root of T
DFS(s)
procedure DFS(v)
/* vertex v is discovered at this point */
for each edge (v, w) do
/* edge (v, w) is scanned (from v) at this point */ if w has not been discovered then
make w the next child of v
DFS(w)
/* vertex v is finished at this point */

- The procedure DFS is recursive, i.e., it calls itself. The overhead for a recursive call is 0 (1). Algorithm 1.2 uses linear time, O (n + m).
- If scanning the edge (v, w) from the vertex v results in the discovery of the vertex w, then (v, w) is a tree edge.
- Suppose that the graph G is undirected. For the tree T produced by Algorithm 1.2 to be a valid depth-first tree, any edge (v, w) belongs to E -T must have v and w related vertices. Why does T have this property? By symmetry suppose v gets discovered before w. Then w will either be made a child of v, like edge (3, 5) in Figure 1.2 or a nonchild descendant of v ,like edge (3, 4) in Figure 2.
- Suppose that the graph G is directed. For T to be a valid depth-first tree, any edge (v, w) must be one of the 4 possible types. Why does T have this property? First suppose v gets discovered before w. In that case w will be a descendant of v and (v, w) will be a tree or forward edge. Next suppose v is discovered after w. Then either v descends from w or v is to the right of w. In the former case (v, w) is a back edge and in the latter case (v, w) is a cross edge.
- Algorithm 1.2 can be extended to a procedure that constructs a depth-first forest F: The procedure starts with F = 0. It repeatedly chooses a vertex s F, uses DFS(s) to grow a depth-first tree T from s, and adds T to F.
- Algorithm 1.2 uses linear time. For direct graphs a point to note is that a vertex w gets added to only 1 tree of F. This is because once discovered, vertex w remains "discovered" throughout the whole procedure.

6.4.Observation

We can test whether an undirected graph is connected in linear time, by using a depth-first search. The trees of a depth-first search spanning forest give the connected components.

We can test whether all vertices of a directed graph are reachable from a vertex s in linear time, by a depth-first search.

7.Discovery Order

7.1.Definitions

Discovery order is a numbering of the vertices from 1 to n in the order they are discovered. This is also called the preorder of the dfs tree.

In finish time order the vertices are numbered from 1 to n by increasing finish time. This is the postorder of the dfs tree.

7.2.Inference

Most algorithms based on the depth-first search tree use discovery order. These algorithms identify each vertex v with its discovery number, also called v. This is how the vertices are named in Figure 1.3.

In discovery order, the descendants of a vertex v are numbered consecutively, with v first, followed by all its proper descendants. This gives a quick way to test if a given vertex w descends from another given vertex v: Let v have d descendants. w is a descendant of v exactly when v <=w < v + d. This method can be implemented to run in 0(1) (i.e., constant) time.

7.3.Observation

The power of depth-first search comes from its simplification of the edge structure the absence of cross edges in undirected graphs, and the absence of left-to-right edges in directed graphs. Depth-first search algorithms work by propagating information up or down the dfs trees.

Many simple properties of graphs can be analyzed without using the full power of depth-first search. The algorithm always works with a path in the dfs tree, rather than with the entire dfs tree. The algorithm propagates information along the path.

As a simple example of observation 4 we give a procedure that shows an undirected graph with minimum degree S has a path of length > S: execute DFS(s) (for any s), stopping at the first vertex t that becomes finished. The portion of tree T constructed by this procedure is a path from s to t of length > S. The reason is that all of its neighbors must be in the path for it to be finished.

Figure 3 deal with simpler graph properties that can be handled by the path view of depth-first search.

8.References

1. J. Hopcroft and R. E. Tarjan, Efficient algorithms for graph manipulation, Comm. ACM 16 (1973), 372-378.
2. J. Kleinberg, The small-world phenomenon: An algorithmic perspective, Proc. 32nd Annual ACM Symp. on Th. Comput. (2000), 163-170.
3. S. Milgram, The small world problem, Psychology Today 1 (1967), 60-67.
4. I. Stewart, "Cows in the maze pp.116-118 in Mathematical Recreations, Sci. American Dec. 1996.
5. R. E. Tarjan, Depth-first search and linear graph algorithms, SIAM J. Comput. 1 (1972), 146-16