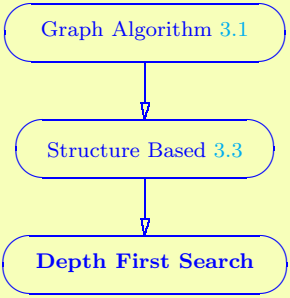
3.5.1. Depth First Search (DFS)

Section authors: Jill C. Magsam, Jeffrey P. Norris, Stephen E. Zingelewicz



Refinement of: Structure Based Graph Algorithm (§3.3), therefore of Graph Algorithm (§3.1).

Prototype:

```
template <class VertexListGraph,
          class DFSVisitor,
          class ColorMap>
void depth_first_search
     (const VertexListGraph& g,
      DFSVisitor vis, ColorMap color)
```

Input: The input to the depth-first search algorithm is a graph, which may be directed or undirected.

Output: The output of the depth-first-search algorithm is a predecessor subgraph. This subgraph may be a forest of several trees because the search may be repeated from multiple sources. However, each vertex appears in only one tree so each of the trees in the set is disjoint.

Effects: The structure of the graph is not altered by the algorithm. However, as a result of running depth-first-search on a graph, every vertex is assigned a discovery time and a finishing time. These times may vary slightly depending on how the order of vertices is chosen.

Asymptotic complexity: Let V = number of vertices. Let E = number of edges.

- Average case (random data): $O(V + E)$
- Worst case: $O(V + E)$

Complexity in terms of operation counts:

- Average case:

Value comparisons:	$6V + 2.9E$
Value assignments:	$10V + 2E$
Integer operations:	$4V + E$

Experimental Data: In order to experimentally obtain the operation counts, several types of graphs were examined. All of these were generated using the BGL random graph generation function. The two which gave the clearest results are presented here. To isolate the impact of the number of edges and vertices on the overall asymptotic order, the number of edges was held constant while the number of vertices was varied. Then, the number of edges was varied while the number of vertices were held constant. The results are presented in the tables below:

This table shows the results when the number of edges was held constant at 2 and the number of vertices was varied:

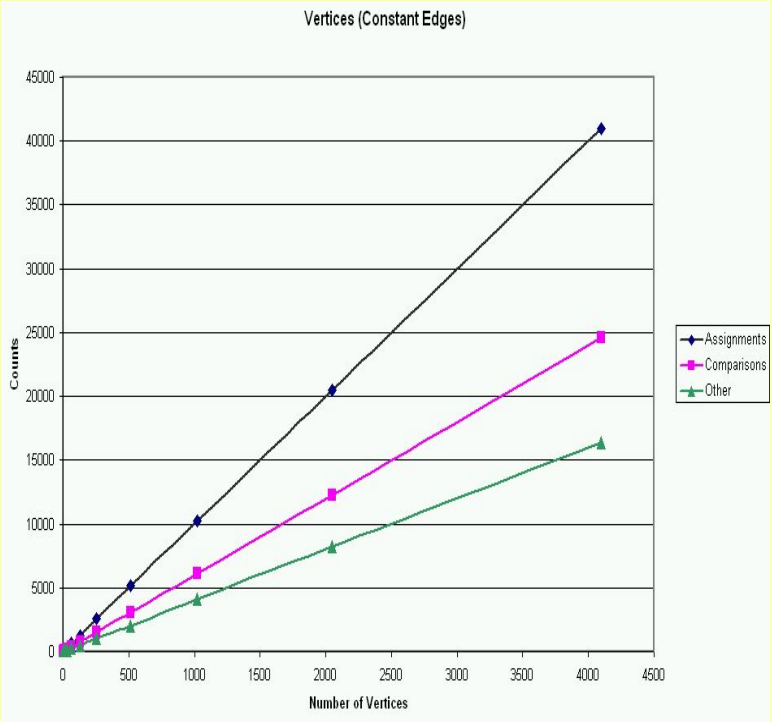
V	Assign	Compare	Integer
2	24	16	9
4	44	28	17
8	84	52	33
16	164	100	65
32	324	196	129
64	644	388	257
128	1284	772	513
256	2564	1540	1025
512	5124	3076	2049
1024	10244	6148	4097

This table shows the counts when the number of vertices were held constant at 128 while the number of edges were varied:

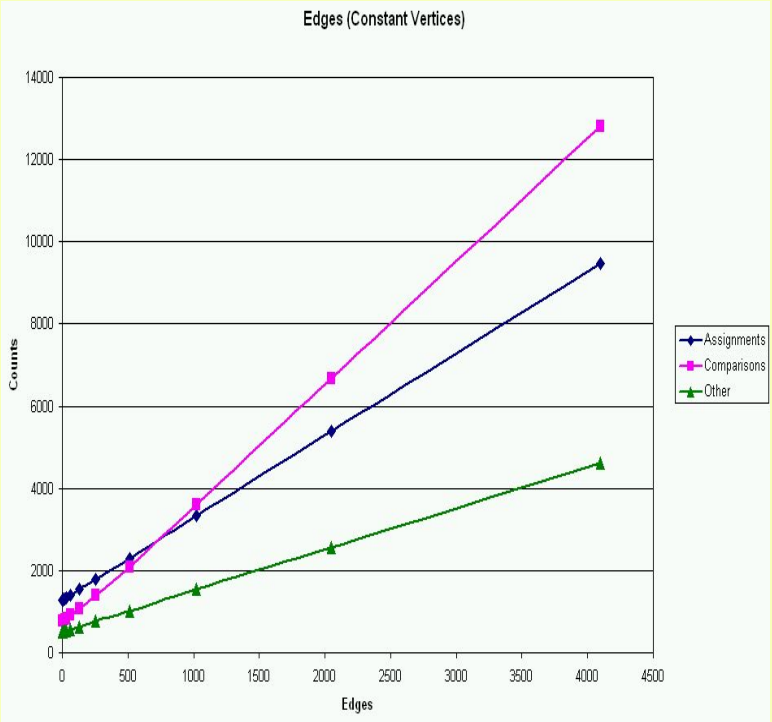
E	Assign	Compare	Integer
2	1286	775	514
4	1290	779	516
8	1298	789	520
16	1314	808	528
32	1346	848	544
64	1410	926	576
128	1538	1084	640
256	1794	1410	768
512	2306	2086	1024
1024	3330	3592	1536

Experimental Plots: The following charts were created by counting operations within the depth first search code in the BGL. The aim was to characterize the number and types of operations being performed on various graphs. The order calculation for DFS relies on two input variables, the number of vertices, V, and the number of edges, E. For these two experiments, the variables were adjusted independent of each other, with the results given below.

Experiments with Vertex counts:



Experiments with Edge counts:



Pseudocode: Pseudocode of depth first search as defined in the Boost Graph Library:

```
DFS(G)
  >>> initialize vertex u
  for each vertex u in V
    color[u] := WHITE
    p[u] = u
  end for
  time := 0
  >>> start vertex s
  if there is a starting vertex s
    call DFS-VISIT(G, s)
  for each vertex u in V
    >>> start vertex u
    if color[u] = WHITE
      call DFS-VISIT(G, u)
  end for
  return (p,d_time,f_time)

>>> discover vertex u
DFS-VISIT(G, u)
  color[u] := GRAY
  d_time[u] := time := time + 1
  >>> examine edge (u,v)
  for each v in Adj[u]
    >>> (u,v) is a tree edge
```

```
    if (color[v] = WHITE)
      p[v] = u
      call DFS-VISIT(G, v)
    >>> (u,v) is a back edge
    else if (color[v] = GRAY)
      ...
    >>> (u,v) is a cross or forward edge
    else if (color[v] = BLACK)
      ...
  end for
  >>> finish vertex u
  color[u] := BLACK
  f_time[u] := time := time + 1
```

References: .

- An animation of an operating depth first search is given at:
  http://www.cs.duke.edu/csed/jawaa/DFSanim.html
  The animation shows a depth first search of a simple, directed graph. You can start, pause or stop the animation at any point. You can also choose to step through it. The scroll bar at the bottom of the animation

controls the speed that it runs. To slow the animation down, scroll to the left.

As it runs, you can watch which vertices are pushed and popped onto the stack. The animation also creates two lists. The first is called Preorder. Vertices get added to the list as they are discovered, or visited the first time. The other list is called Postorder. Vertices get added to this list as they are finished, in other words when their adjacency list has been examined completely.

As the animation progresses, the vertices are colored to indicate their state. All vertices are initially grey. When a vertex is discovered, it is colored green and added to the preorder list. When a vertex is finished, it is colored blue and added to the postorder list.

- The BGL reference material for depth first search can be found at:
  http://www.boost.org/libs/graph/doc/depth_first_se