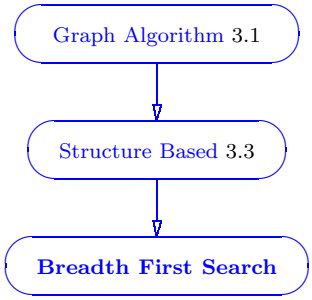
3.19 Breadth First Search

Section Authors: Derek Carey, Martina Davis, Terrell Holmes.



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

Prototype:
```
template <class VertexListGraph, class P,
          class T, class R>
void breadth_first_search(VertexListGraph& G,
typename graph_traits<VertexListGraph>
          ::vertex_descriptor s,
const bgl_named_params<P, T, R>& params);.
```

Inputs: Inputs are a graph(directed or undirected) $G = (V, E)$ and a source vertex s, where s is an element of V. The adjacency list representation of a graph is used in this analysis.

Outputs: The outputs are a predecessor graph, which represents the paths traveled in the BFS traversal, and a collection of distances, which represent the distance of each of the vertices from the source vertex.

Effects: The color property of each vertex in the graph will be set to black to symbolize that the vertex and all of its adjacent vertices have been visited. A distance from the source vertex and a predecessor vertex are assigned to each vertex in the graph.

Asymptotic complexity: Let $V = |V(G)|$ Let $E = |E(G)|$.

Average case (random graph): $O(V + E)$

Worst case (complete graph): $O(V * V)$

Random Number Generator Specifications:

Random Number Generator 1

C standard library rand() function

Random Number Generator 2

Random Number Generator functor from the Boost library

Testing Strategy

Testing was performed on the Sun Microsystems Ultra 10 workstations.

Random graphs were generated using the two random number generators specified above. Time tests were run on directed and undirected graphs of $|V| = 100$, 500, and 1000, with edge sizes ranging from $|E| = |V|$ to $|E| = |V^2|$. Each test was performed three times and the average of the BFS running time tests are displayed in the following tables.

Pseudocode: `BFS(G, s)`

```
for each vertex u in V[G]
  color[u] := WHITE
  d[u] := infinity
  p[u] := u
end for
color[s] := GRAY
d[s] := 0
ENQUEUE(Q, s)
while (Q != EMPTY)
  u := DEQUEUE(Q)
  for each vertex v in Adj[u]
    if (color[v] = WHITE)
      color[v] := GRAY
      d[v] := d[u] + 1
```

Table 1: Performance of BFS using Random Number Generator 1 on large directed and undirected graphs. (Time is represented in seconds)

Random Number Generator 1

Vertices $\|V\|$	Edges $\|E\|$	Directed Graphs	Undirected Graphs
100	100	0.00033	0.00177
100	1000	0.00746	0.01467
100	10000	0.06516	0.13467
500	500	0.00165	0.00855
500	5000	0.03640	0.07155
500	50000	0.33218	0.67370
500	250000	1.60266	3.35799
1000	1000	0.00296	0.01699
1000	10000	0.07283	0.14277
1000	100000	0.64975	1.35201
1000	1000000	6.50440	13.2974

```
        p[v] := u
        ENQUEUE(Q, v)
      endif
    endfor
    color[u] := BLACK
end while
return (d, p)
```

Table 2: Performance of BFS using Random Number Generator 2 on large directed and undirected graphs. (Time is represented in seconds)

Random Number Generator 2

Vertices $\|V\|$	Edges $\|E\|$	Directed Graphs	Undirected Graphs
100	100	0.00056	0.00170
100	1000	0.00743	0.01426
100	10000	0.06856	0.13490
500	500	0.00166	0.00832
500	5000	0.03671	0.07134
500	50000	0.32464	0.67446
500	250000	1.61271	3.36002
1000	1000	0.00303	0.01693
1000	10000	0.07272	0.14272
1000	100000	0.67229	1.36000
1000	1000000	6.42800	13.4317

Algorithm animation 1: Contains animations for a few AI and Search Algorithms

Algorithm animation 2: Animation of BFS being performed on an undirected graph of 7 vertices. This graph shows the state of the queue, the distances being assigned to the vertices and the state of the predecessor graph.

Algorithm animation 3: Animation of BFS being performed on a directed graph of 7 vertices. This graph shows the state of the queue, the distances being assigned to the vertices and the state of the predecessor graph.

Acknowledgements

This document and the reported performance measurements are an extension of those previously presented by Bob Bachman, Kadriye Vargun, Jeff Richards ($Adopt - an - Algorithm2002$)

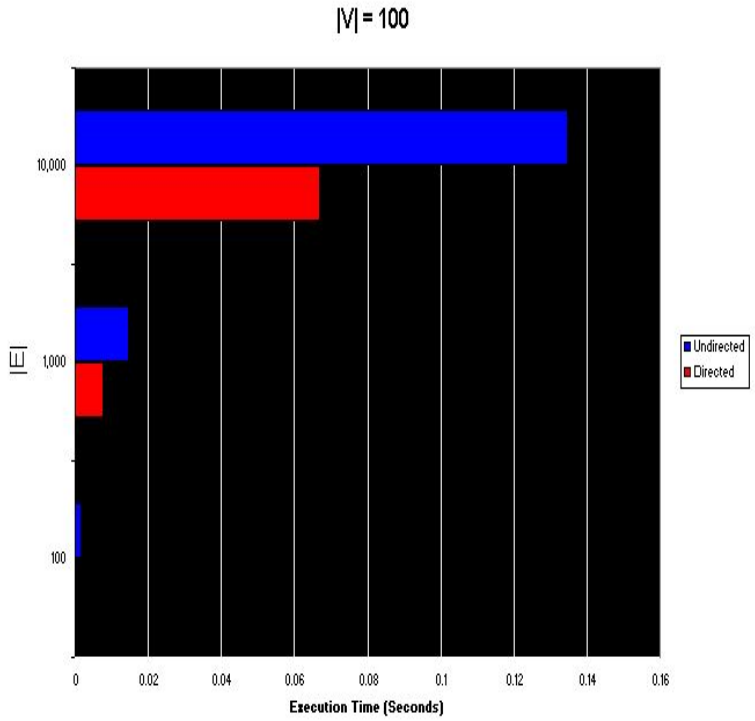


Figure 1: This chart is an average of the two tables based upon $|V| = 100$

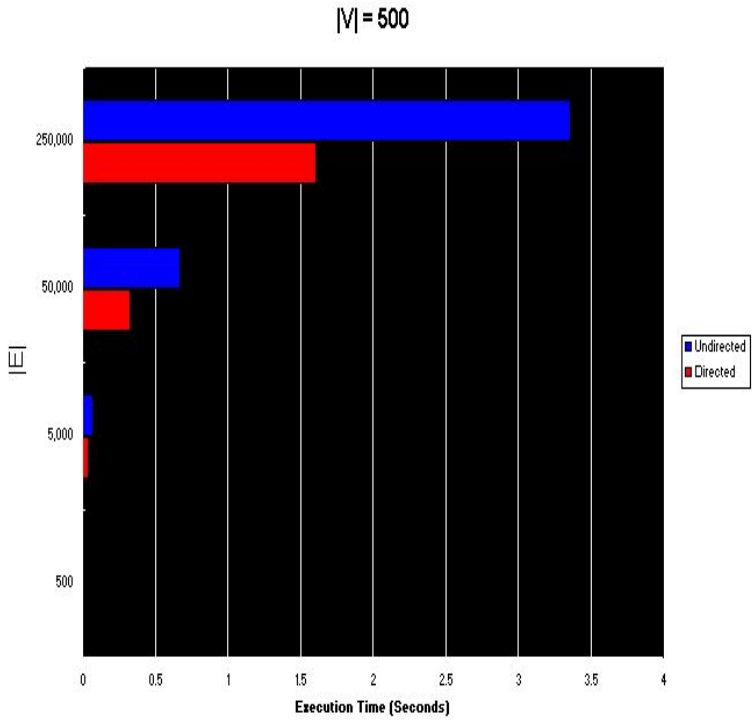


Figure 2: This chart is an average of the two tables with $|V| = 500$

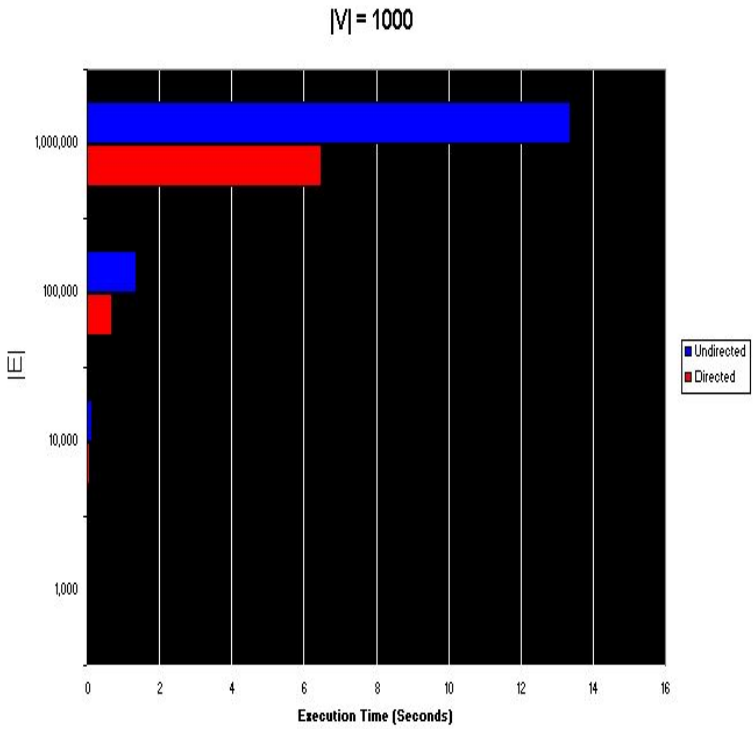


Figure 3: This chart is an average of the two tables with $|V| = 1000$