Explanation of runtimes of BFS and DFS

Why are the running times of BFS and DFS O(V+E), especially when there is a node that has a directed edge to a node that can be reached from the vertex, like in this example in the following site

http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/depthSearch.htm

data-structures graph time-complexity depth-first-search breadth-first-search

edited Nov 2 '11 at 16:17 dsolimano 5,706 3 23 39

asked Jul 27 '11 at 19:50 miss24 21 1 1 2

4 Answers

E is the set of all edges in the graph, as G={V,E}. So, |E| is count of all edges in the graph.

This alone should be enough to convince you that the overall complexity can't be |V| times |E|, since we are not iterating over all the edges in the graph for each vertex.

In the adjacency list representation, for each vertex v, we only traverse those nodes which are adjacent to it.

The |V| factor of the |V|+|E| seems to come from the number of queue operations done.

Note that the complexity of the algorithm depends on the data structure used. effectively we are visiting each piece of information present in the representation of the graph, which is why for matrix representation of the graph, complexity becomes V squared.

Quoting from Cormen,

"The operations of enqueuing and dequeuing take O(1) time, so the total time devoted to queue operations is O(V). Because the adjacency list of each vertex is scanned only when the vertex is dequeued, each adjacency list is scanned at most once. Since the sum of the lengths of all the adjacency lists is Θ(E), the total time spent in scanning adjacency lists is O(E). The overhead for initialization is O(V), and thus the total running time of BFS is O(V + E)."

answered Mar 4 '12 at 10:23 Abhijeet Kashnia 2,693 5 22 38

This issue consume like 4 hours of my time, but finally I think I have an easy way to get the picture, at the beginning I was tempted to say O (V * E).

Summarizing the algorithm that you find in Cormen, that is the same on http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/GraphAlgor/breadthSearch.htm you have something like this :

for (vi in V) Some O(1) instructions for (e in Adj (vi)) { Some O(1) instructions }

The question is how many instructions are executed here ? that will be the Sigma-Sum (Adj(vi)), and this value is upper-bounded by 2|E|.

At the beginning we automatically think about multiply the number of iterations of the inner and outer loops, but in this case the total number of iterations on the inner loop is a function of the outer iterator, so no multiplication is possible.

answered Jun 16 '12 at 10:51
Jose Francisco
71 1 2

You iterate over the |V| nodes, for at most |V| times. Since we have an upper bound of |E| edges in total in the graph, we will check at most |E| edges. Different vertices will have varying number of adjacent nodes, so we **cannot** just multiply |V|*|E| (it means that for each vertex, we traverse |E| edges, which is not true, |E| is the total number of edges over all nodes), rather, we check over V nodes, and we check over a total of E edges. At the end, we have O(|V|+|E|)

For DFS, it's something similar, we loop through all of a vertices adjacency lists, calling DFS(v) if it's not been visited, meaning that we incur |V| time steps, plus the time incurred to visit adjacent nodes (essentially, these form an edge, and we have a total of |E| edges, hence, O(V+E) time.

```
private static void visitUsingDFS(Node startNode) {
    if(startNode.visited){
        return;
    }
    startNode.visited = true;
    System.out.println("Visited "+startNode.data);
    for(Node node: startNode.AdjacentNodes){
        if(!node.visited){
            visitUsingDFS(node);
        }
    }
}
```

answered Nov 9 '14 at 4:17
Bavinho
1 1

You visit every edge at most twice. There are E edges. So there will be 2*E edge visit operations. Plus the nodes those have no edges or in other words, with degree 0. There can be at most V such nodes. So the complexity turns out to be, O(2*E + V) = O(E + V)

answered Aug 2 at 15:06
Fallen
2,237 1 7 24