Report 4 – AEDs III

Willian de Souza Soares – 2014.1.08.044

March 16, 2020

Abstract

The Depth First Search, DFS, is an algorithm to identify if a value is allocated in the graph and, if it is, determine it's position.

1 The algorithm

In short, the DFS can be described as an algorithm that looks for the deepest vertex v in a graph and then, turning back to an parent vertex of v and going for the deepest sibling of v, repeating this until all the vertices are visited or the search is completed.

For this algorithm, we can 'colour' the vertices one of three colors:

- White – The vertex hasn't been visited yet;
- Gray – The vertex itself has been visited, but not all it's adjacents;
- Black – The vertex and all it's adjacents has been visited.

We can markdown the time when the vertex was discovered (turned gray) and when it was closed (turned black).

1.1 DFS – Recursive implementation

This implementation has a time complexity of $\theta(V)$ as V the numbers of Vertices on the Graph.

```
function DFS(Graph G){
   //All vertices must start on white
    foreach(Vertex u in Vertices(G))
        color[u] = WHITE
    time = 0;

    //Visiting all unvisited vertices of the graph G
    foreach(Vertex u in Vertices(G))
        if color[u] == WHITE
            VISIT(u)
}

function VISIT(Vertex u){
   //Marking this vertex as visited and its discovery time
    color[u] = GREY
    time += 1
    discoveryTime[u] = time

    //Visiting all unvisited adjacents of the vertex u
    foreach(vertex v in Adjacent(u))
        if color[v] == WHITE
            VISIT(v)

    //When the code reach here, the vertex u and all it's adjacents are visited.
    //Marking u as finished.
    color[u] = BLACK
    time += 1
    closeTime[u] = time
}
```

1.2 DFS – Iterative implementation

This implementation has a time complexity of $\theta(E)$ as E the numbers of Edges on the Graph.

```
function DFS(Graph G, Vertex v){
    //All vertices must start on white
    foreach(Vertex u in G)
        color[u] = WHITE

    //Putting on the stack the first vertex to be visited
    time = 0
    stack.push(v)
    while(stack.hasNext())
        //Grab the first vertex on the stack.
        u = stack.pop()
        //If not visited
        if(color[u] == WHITE)
            //Mark as visited
            time +=1
            color[u] = GREY
            discoveryTime[u] = time

            //Put all adjacents on stack to be visited
            foreach(Vertex v in adjacentEdges(u))
                S.push(v)

            //Mark u as closed
            time += 1
            color[u] = BLACK
            closeTime[u] = time
}
```