graph_connectivity

Robin Deits

February 13, 2014

In [1]:
```
module Con
export Graph, Node, Edge

type Node
    label::ASCIIString
end

type Edge
    a::Node
    b::Node
end

type Graph
    nodes::Set{Node}
    neighbors::Dict{Node,Set{Node}}
end

function Graph(nodes::Set{Node}, edges::Array{Edge})
    neighbors = Dict{Node,Set{Node}}();
    for e in edges
        for (n1, n2) in ((e.a, e.b), (e.b, e.a));
            for e in edges
                if haskey(neighbors, n1)
                    push!(neighbors[n1], n2);
                else
                    neighbors[n1] = Set(n2);
                end
            end
        end
    end
    Graph(nodes, neighbors)
end

type Path
    nodes::Array{Node}
end

function BFS_path(graph::Graph, s::Node, t::Node)
    visited = Set(s)
    active_set = [Path([s])]
    while true
        new_active_set = Path[]
        for p in active_set
            for u in graph.neighbors[p.nodes[end]]
                n = copy(p.nodes)
                push!(n, u)
                new_path = Path(n)
                if u == t
                    return new_path
                end
                if !(u in visited)
```

```
                    push!(visited, u)
                    push!(new_active_set, new_path)
                end
            end
        end
        if length(new_active_set) == 0
            return Path(Node[])
        end
        active_set = new_active_set
    end
end

function is_fully_connected(graph::Graph)
    s, state = next(graph.nodes, start(graph.nodes))
    visited = Set(s)
    active_set = [s]
    while true
        new_active_set = Node[]
        for n in active_set
            for u in graph.neighbors[n]
                if !(u in visited)
                    push!(visited, u)
                    push!(new_active_set, u)
                    if length(visited) == length(graph.nodes)
                        return true
                    end
                end
            end
        end
        if length(new_active_set) == 0
            return false
        end
        active_set = new_active_set
    end
end

end
```

In [2]:
```
import Con
```

In [3]:
```
nodes = [Con.Node("a"), Con.Node("b"), Con.Node("c"), Con.Node("d")]
edges = [Con.Edge(nodes[1], nodes[2]),
    Con.Edge(nodes[2], nodes[3]),
    Con.Edge(nodes[1], nodes[4])]
graph = Con.Graph(Set(nodes...), edges)
```

Out [3]:
```
Graph(Set{Node}(Node("a"),Node("c"),Node("d"),Node("b")),[Node("a")=>S
et{Node}(Node("d"),Node("b")),Node("c")=>Set{Node}(Node("b")),Node("d"
)=>Set{Node}(Node("a")),Node("b")=>Set{Node}(Node("a"),Node("c"))])
```

In [4]:
```
path = Con.BFS_path(graph, nodes[1], nodes[3])
```

Out [4]:
```
Path([Node("a"),Node("b"),Node("c")])
```

In [5]:
```
path = Con.BFS_path(graph, nodes[1], nodes[4])
```

Out [5]:

```
Path([Node("a"),Node("d")])
```

In [6]:

```
Con.is_fully_connected(graph)
```

Out [6]:

```
true
```

In [7]:

```
graph2 = Con.Graph(Set(nodes...), edges[1:2])
Con.is_fully_connected(graph2)
```

Out [7]:

```
false
```

In [8]: