CS180 Homework 3

Zhehao Wang 404380075 (Dis 1B)

Apr 16, 2016

1 Strongly connected components in a directed graph

(a) Prove SCC graph is a DAG.

Proof by contradiction: assume that a path $s_i, ..., s_j, s_i$ exists in the SCC graph, where s_k $(i \leq k \leq j)$ are the newly created distinct SCC nodes.

There exists a path from any node p_i in SCC s_i to any node p_{i+1} in SCC s_{i+1}, since there exists a node q_i in SCC s_i that has a directed edge to a node q_{i+i} in SCC s_{i+1}, and there exists a path from p_i to q_i, q_{i+1} to p_{i+1}.

Applying the above conclusion repeatedly till we reach s_j, we have the conclusion there exists a path from any node p_i in SCC s_i to any node p_j in SCC s_j. Similarly, we have there exists a path from any node p_j in SCC s_j to any node p_i in s_i. Thus the nodes p_i and p_j should belong to the same SCC node which is the combination of SCC s_i and s_j, and contradicts with the nodes being distinct in the assumption. And we have the directed SCC graph is acyclic, which makes it a DAG by definition.

(b) The algorithm is given in alg 1.

Time complexity:

Correctness:

2 Longest path in DAG

(a)

3 Optimal order of files

4 Sorting from SC

Algorithm 1 SCC building algorithm

```
1: function DFS(v, do_label, smallest_node, node_remaining, node_removed, SCC_graph)
2:     v.visited ← true
3:     if do_label = false then
4:         node_remaining.remove(v)
5:         node_removed.add(v)
6:         if v = smallest_node then
7:             smallest_node ← node_remaining.nextSmallest()
8:     for {i|(v, i) ∈ E, i.visited = false} do
9:         if do_label = true then
10:            DFS(i, do_label, smallest_node, node_remaining, node_removed, SCC_graph)
11:        else
12:            if i ∈ SCC_graph then
13:                SCC_graph.addEdge(v, i)
14:            else
15:                DFS(i, do_label, smallest_node, node_remaining, node_removed, SCC_graph)
16:    if do_label then
17:        id ← label(v)
18:        if id < smallest_node then
19:            smallest_node ← v
20: function GETSCC(G)
21:    smallest_node ← nil
22:    DFS(G.firstNode(), true, smallest_node, G.nodes, [])
23:    smallest_node ← smallest_node.copy()
24:    G.resetVisited()
25:    SCC_graph ← nil
26:    while G.node_count > 0 do
27:        G.resetVisited()
28:        node_removed ← []
29:        DFS(smallest_node, false, smallest_node_copy, G.nodes, node_removed, SCC_graph)
30:        smallest_node ← smallest_node_copy
31:        SCC_graph.addNode(node_removed)
32:    return SCC_graph
```