CSCI 3104 Assignment 8

10:00 - 10:50 Wanshan

Jackson Chen

April 11 2016

1. (a) Topological ordering: 1, 2, 3, 4, 5
   Topological sorting takes $\Theta(m+n)$

   (b) Pseudocode for topological shortest path finding:

```
def findShortest(startNode):
  dist = [0, Infinity, Infinity, Infinity, ...]
  path = [nil, nil, ...]
  // length of path and dist is the # of vertices
  dist, path = find(startNode, dist, path)
  return dist, path

def find(node, dist, path):
  if (node.children.exist?):
    for (node.children):
      dist, path = relax(dist, path, node, weight)
      find(node.child, dist, path)
  return (dist, path)
```

   The algorithm first gives an infinite distances to all nodes except 2 (which has dist 0). Then the children of node 2 (nodes 3, 4, and 5) are assigned distances 2.4, 1.3, and -0.3 respectively. Then the edge between node 3 and its child (node 5) is relaxed, but that distance to 5 is not changed since 2.4 - 1 is still larger than -0.3. Then the edge between node 4 and its child (node 5) is relaxed, but that distance to 5 is not changed since 1.3 + 2 is still larger than -0.3. Thus the final distances for nodes 1-5, respectively, are infinity, 0, 2.4, 1.3, -0.3.

   (c) $\Theta(n+m)$

2. (a) Each edge can have a weight of 2. Then there are back edges from the last character of a known codeword to the first character, which would negate the sum of the weights that connected all the letters (but one less than that to allow codewords to contribute a cost of 1). Adding new edges isn't an issue since all edges are of weight 2 except for backedges. Thus if it's a codeword that is being added, then the backedge needs to be adjusted to reflect the addition of the new edge.

   (b) Since codewords are strongly connected components with a net weight of 1 upon traversal, it incentives the algorithm to traverse those characters that make up the codeword once. Then it follows the path of ciphered characters as normal.