# 3.5

Andrew Lee

March 10, 2017

The *reverse* of a directed graph *G = (V, E)* is another directed graph $G^R = (V, E^R)$on the same vertex set, but with all edges reversed; that is,$E^R = \{(v, u) : (u, v) \in E\}$. Give a linear-time algorithm for computing the reverse of a graph in adjacencylist format.

```
1  #!/usr/bin/python
2  def reverseGraph(arg):
3      edges = []
4      for count, data in enumerate(arg):
5          for count2, data2 in enumerate(data):
6              edges.append([data2, count])
7      print(edges)
8
9  representation = [[1, 2], [0, 2], [0, 1, 3], [2]]
10 reverseGraph(representation)
```

The above algorithm work because all it is doing is taking in the adjacency list and converting them all to an list with all the given edges, then flipping element one and two in the append([data2, count]) command. This runs in linear time relative to the total number of edges in the graph because the overall time complexity is in e (for number of edges) and it is iterating through all elements and with all the operationg of reversing running at a constant rate.

T(e) = O(c*e) where c is some constant and e is the total number of edges