CSCI 3104
Spring 2018
Problem Set 8

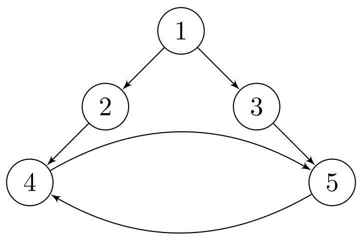
Allison, George
02/17

March 22, 2018

1. The number of pathways to the destination point can be found by compiling the cumulative amount of paths for all of the preceding nodes. Each node has some number of possible paths to reach it, referred here as "paths". A node's number of paths is the sum of paths of the preceding nodes that are connected to it. In the given network, node 1 is the starting point, and thus has 1 path. Next, node 2 is connected to node 1, which has 1 path, so node 2 also has 1 path. The number of paths for each node is calculated in this fashon until the destination node is reached. Below is a table consisting of each node and its number of paths; it can be shown that there are 30 different paths that can be used to traverse from node 1 to 14.

Node	Paths
1	1
2	1
3	1
4	1
5	1
6	1
7	2
8	4
9	5
10	7
11	9
12	5
13	21
14	30

2. Because of the loop between node 4 and 5 of the graph below, a depth first search will not be able to produce the proper set of edges, while a breadth first search will not be hindered by those connections since they are on the same row.



3. (a)In an adjacency matrix the out-degrees of node i would be stored in the row i. Each in-degree for that node would be stored in colummn i. Therefore, to find the in-degrees and out-degrees you would need to traverse a row and column of which both have length V. The asymptotic notation for this search is $O(V^2)$.

(b)In an edge list representation the start and end points of each node can be checked to measure the number of in-degrees and out-degrees for a given node. If an edge in the list is of the form $[x, y]$ then the number of out-degrees would correspond with the number of times a node i is found in position x for an edge. The number of in-degrees corresponds with the number of times i is in position y. Searching through the list of edges is linear, with an asymptotic complexity of $O(E)$.

(c) In an adjacency list representation the number of out-degrees corresponds with the number of its adjacent verticies in the list. To find in-degrees for node i you would need to search for i in the out-degrees for all other nodes. Checking for in and out degrees in an adjacency list yeilds $\Theta(V+E)$, however it has a possibility of $O(V^2)$ in the case that all nodes are connected to each other.

4. The algorithm to simplify the graph would need to index through all nodes and check each of their outbound connections for duplicates and self-loops. The search through all V nodes results in V operations, and the checks on outbound connections would amount to, at most, an additional E operations, in total resulting in $O(V + E)$ operations. Below is the psuedocode for the algorithm in Java:

```
public static void GraphConvert(LinkedList[] adjLst) //assuming use of LinkedList object
    with add() and remove() functions
{
  int[] temp = new int[V]; //going to be used while checking for duplicates
  int c = 0;
   for(int i = 1; i <= V; ++i) //looping through each node in the graph [V ops]
   {
     while(adjLst[V].hasNext()) //looping through each element in the node's adjacency
         list [E ops]
       {
        Node myNode = adjLst[V].next();
        if(myNode == V) //self loop
           adjLst[V].remove(myNode); //remove self loops
        for(int x : temp){
           if(x == myNode);
```

```
                adjLst[V].remove(myNode); //remove redundencies
                break;
          }
          temp[c] = myNode;
          c++;
         }
    }
}
```

---

5. For this problem each battery can be represented as a node in a graph. In order to split up the morts and get a number (12) that is around half of the capacity of a battery an algorithm should be used that fills up the largest empty battery and then distributes the morts back into the others. The order of swaps to get 12 morts in a battery are as follows:

42	27	16
0	27	16
27	0	16
42	0	1
15	27	1
15	12	16