3 Algorithms for Network Resilience

Henry Zhang

January 2019

Algorithm 1: ComputeDistance

Input: Graph $g = (V, E)$; source node $i \in V$
Output: $d_j \in \mathbb{R}, \forall j \in V$

```
foreach j ∈ V do
    d_j ← ∞;
d_i ← 0;                                   // Start by visiting the source node i
Initialize Q to an empty queue;;
enqueue(Q, i);
while Q is not empty do
    j ← dequeue(Q);
    foreach neighbor h of j do
        if d_h = ∞ then
            d_h ← d_j + 1;
            enqueue(Q, h);
return d;
```

Discussion 1:The algorithm performs n+m operations. The first loop runs n times. The second loop runs for a multiple of m operations. Each edge is traversed exactly twice in this algorithm.

Algorithm 2: IsBipartite

Input: Graph $g = (v, E)$
Output: a boolean number

```
foreach j ∈ V do
    Initialize c to an empty mapping;
    c_j ← 0 ;
c_i ← 1 ;                                    // start with any node i ∈ V
Initialize Q to an empty queue;
enqueue(Q, i) ;
while Q is not empty do
    j ← dequeue(Q) ;
    foreach neighbor h of j do
        if c_h = c_j then
            return False ;
        else if c_h = 0 then
            c_h = −c_j;
            enqueue(Q, h)
return True
```

Discussion 2: The first loop runs n operations. The second loop checks the neighbors of each node dequeued, and traverses each edge twice, so it is a multiple of m operations.

Algorithm 3: ComputeLargestCCSize

Input: Graph $g = (v, E)$
Output: an integer

```
Initialize P to an empty queue;
Initialize v to an empty mapping;
foreach j ∈ V do
    v_j ← False;
    enqueue(P, j);
s ← 0 ;
while P is not empty do
    m ← dequeue(P) ;
    if v_m = False then
        v_m ← True ;
        k ← 1 ;
        Initialize Q to an empty queue;
        enqueue(Q, m);
        while Q is not empty do
            j ← dequeue(Q);
            foreach neighbor h of j do
                if v_h = False then
                    v_h ← True;
                    enqueue(Q, h);
                    k ← k + 1 ;
        if k > s then
            s=k ;
return s;
```

Discussion 3: The first loop in the algorithm runs n operations. In the second loop, the algorithm runs BFS first on a node, and again on nodes that are not previously explored. So when all the nodes and edges in the graph are explored, each edge is traversed twice so it is still a multiple of m operations. Therefore, the algorithm runs m+n operations.