Stephen Woodbury   6/5/17   CMPS 102
I have read and agree to the collaboration policy. Stephen Woodbury.
Collaborators: none

Assignment 4_1 : Zombie Infestation

a) Finding Maximum Flow Algorithm

Assumptions:

1) Every City's capacity is a positive integer.

Algorithm:

```
1 Set up G' from G by changing all nodes (except s and t) into two nodes
with an edge between them with capacity equal to the city's capacity. Find
the city with the max capacity, set all other edge capacities equal to this
max city capacity. Modify c into c' to do so.
2 Max-Flow-Find(G', s, t, c'){
3   foreach e that exists in E'
4     f(e) = 0
5   Gf = Residual Graph of G'
6   while (There exists a path from s to t){
7     f = Augment(f, c, p)
8     update Gf}
9   return f}
10Augment(f,c',p){
11  b = bottleneck(p) //finds the edge with the lowest capacity in path.
12  foreach e that exists in p{
13    if(e exists in E')
14      f(e) = f(e)+b
15    else
16      f(eR) = f(e)-b}
17  return f}
```

Notes: G' = (V',E'), G=(V,E), Gf = (V, Ef) where Ef = {e: f(e)<c(e), eR : c(e) > 0} (Note, capacity of eR changes every time f(e) changes), s = infestation source, t = cure sink, c' = weight of all edges, f(e) = flow through an edge [we're not using fin(v) as that's for node capacities], f= total flow so far calculated [initially set to 0], p = path found in while loop in Max-Flow-Find and passed to Augment.

Algorithm Description: The algorithm is basically the same as the Ford-Fulkerson Algorithm. We make the flow of all edges of G' equal to 0. We set up a residual Graph of G', call if Gf. While there is a path from s to to, we load that path into augment, find the least capacity edge, for each edge in path given, we adjust the flow through said edge. If the edge is a part of G', we add the bottleneck value to the flow. Else, we subtract the bottleneck from the flow. Then we return f to Max-Flow-Find. We update Gf according to the changes we made, then we determine if there is another path from s to t and repeat the process.

Proof of Correctness : Termination:

Claim: Algorithm terminates after at most [v(f*) <= Cn] iterations where V(f*) is the value of the max flow, C is the largest city capacity in the Graph, and n is the number of cities times two minus two.

Pf: Every iteration through the while loop, f increments by at least one. This is because the while loop only continues to iterate if it has found an augment path, if it has found an augment path, Augment is called, which increases f by at least 1 before returning. Since we know v(f*) is finite, the algorithm must terminate. v(f*) can't be greater than the number of cities * the capacity of the city with the highest capacity intuitively.

Proof of Correctness : Max Flow Found:
Claim: Our Algorithm yields the max flow after termination.
Pf: There is a corollary that states that if there is a s-t cut s.t v(f)=Cap(A,B), then f is the max flow of the Graph. If after termination of our algorithm, we took an s-t cut s.t A includes all nodes reachable from s (and s) and B includes all other nodes, the we'd find that our cap(A,B) does indeed equal the value of the flow returned from our Max-Flow-Find algorithm, which makes the flow returned the max flow.

RunTime Analysis
O(mnC) where C is the capacity of the city with the greatest capacity, m is the number of edges created for G', and n is the number of cities times two minus 2.

Space Analysis
Array for edge weights of size m. array of size n for flow through each city. Arrays mapping out the nodes and edges of G, G', and Gf. Array of size at most m for mapping our path. Overall, O(m).

b) S-T Cut Meaning and what Capacity is of this Cut
I didn't understand the question entirely, so this is what I thought was being asked to the best of my knowledge.
S-T cut meaning: With respect to this problem will be enacted on our modified version of G, G'. This is the Graph where our city nodes are separated into two city nodes with an edge between the two that carries a capacity equal to the city's capacity. The s-t cut will be made between these two city nodes (at least, with the cuts that matter, like after termination of our algorithm, the min-cut). The min cut in this instance represents which cities would be most effective in terms of cutting all transportation to in order to halt progression of the virus.
Capacity of Cut: The capacity of this cut for a min-cut after termination of our algorithm would be zero, but for any other cut, this would be the cost to cut all transportation to specific cities cut through by the s-t cut.

c) max-flow min-cut theorem holds
The max flow represents the spread of the virus, maybe say, the spread of infected individuals. We want to stop the flow of zombies by the greatest amount while spending the least amount of money to stop the flow. A maximum flow represents the total flow of infected individuals in the system and a cut's capacity represents the cost to shut down cities cut through by the s-t cut. Our max-flow min-cut theorem is relevant as it cuts off the greatest amount of virus spread while costing the least amount of money.