CS/ECE 473 Fall 2020	Ryan Prendergast (ryanp4)
Homework 7 Problem 2	Noah Watson (nwatson3)
	Lawson Probasco (lawsonp2)

Let $G = (V, E)$ be an undirected graph. Let V be partitioned into two sets of nodes R and N where R is the set of reliable nodes and N is the set of non-reliable nodes. The reliable nodes never fail but non-reliable nodes and edges can fail. Call NUE the elements. Given two reliable nodes s, t describe an algorithm that computes the minimum number of elements whose failure results in s and t being disconnected from each other.

Solution: First, define a flow network $G' = (V', E')$. Include every vertex V in V', and for each undirected edge in G, create two directed edges in G', one for each direction.

Once this is done, modify $V' \in G'$ such that every non-reliable node $n \in N$ becomes two nodes, here called n_a and n_b. Create an edge (n_a, n_b). Redirect all of the incoming edges of n to n_a and create edges for all of the outgoing edges of n which go from n_b to the destination of the corresponding edge. Every edge in G should receive a capacity of 1 unit.

We construct this flow network using brute force. We know that the number of nodes in G' is at most twice that of G (if every node is non-reliable). G' also has every edge E plus one for each additional node (V). This requires $O(V' + E') = O(2V + V + E) = O(3V + E)$ time.

The purpose of this graph modification is to recast the problem as a minimum cut problem. For every edge $v_a \rightarrow v_b$ which is cut, it represents removing the vertex v in N. For every other edge which is cut, it represents simply removing the edge in E. Since all edge weights are 1, removing an edge counts the same as removing a non-reliable vertex. The algorithm will not be able to "remove" reliable edges because they are represented as a single vertex in G'.

We run a max-flow algorithm on G' (equivalent to a min-cut capacity) to find the solution to the problem.

Correctness We prove that a bijection exists between a number of elements failing in G and a cut capacity in G'. Thus, the minimum cut capacity of G' is equivalent to the minimum number of elements whose failure disconnects s and t, and solves the problem.

Assume we have a cut in G' with capacity c. Since every edge has a capacity of 1, this means there are c edges cut through. We define a one-to-one mapping f from each of the c edges in G' to an element failing in G as follows. For each of the c edges $e \in E'$, if $e \in E$, we map to the edge in G. If $e \notin E'$, then e must be one of the edges we added in G'. These edges all have the form $v_a -> v_b$, so we map to the node $v \in V$.

Now, assume we have a collection of c elements in G. We define a one-to-one mapping from each element to an edge in a cut of G'. If an element x is a node v, then we know G' has nodes v_a and v_b with exactly one edge between. So, we put v_a in S and v_b in T, increasing the cut capacity by exactly one. If the element x is an edge $v -> u$, we put v in S and u in T in the s,t cut of G', cutting through the edge and adding one to the capacity.

Time Complexity Graph size: The new graph will have $O(|V|)$ vertices because each vertex is turned into at most 2 vertices. The graph will have $O(|N| + |E|)$ edges, where N is $O(V)$. Summed together, the graph construction takes $O(2V + E)$ time.

Graph algorithm: Maximum flows can be computed in $O(VE)$ time. For our flow network, this translates to $O(V(N + E)) = O(V(V + E))$ time.

■

CS/ECE 473 Fall 2020
Homework 6 Problem 3

Ryan Prendergast (ryanp4)
Noah Watson (nwatson3)
Lawson Probasco (lawsonp2)

Given G and s, t and an integer k describe an algorithm that checks whether G has a minimum cut with at most k edges.

Solution: We used as inspiration for our solution the modifications described here: https://cs.stackexchange.com/questions/115159/minimum-cut-with-minimum-number-of-edges.

We break up this problem into three parts.

- First, we modify the flow network G into G' such that the minimum cut in G' is equivalent to the minimum cut in G with the minimum number of edges. We define $G' = (V', E')$ as a flow network with the same vertices as G, $V' = V$, and the same edges $E' = E$. We then modify the edge capacities $C'(e)$ as follows:

  $$C'(e) = C(e) * (|E| + 1) + 1$$

  We note that a cut in G of capacity m corrpesponds to a cut in G' with capacity $m*(|E|+1)+l$ where l is the number of edges in the cut.

  Any min cut in G' will be a min cut in G. Assume we have a min cut in G of size m, with corrpesponding size in G' of $m * (|E| + 1) + l$. If you take any cut capacity in G of size $m + 1$ (and all larger capacities must be at least this large since all capacities are integers), the cut capacity in G' will be $(m + 1) * (E + 1) + l'$. $l' \leq |E|$, so this new capacity will be larger than the cut capacity corrpesonding to the min cut in G, $m*(|E|+1)+l$.

  We note further that any min cut in G' will be a min cut in G with the fewest possible number of cut edges. We defined our capacity function such that the cut capacity in G' increases by 1 for every edge it cuts, thus a cut with fewer edges will have a smaller capacity.

- Second, we use our black box algorithm to find a min cut in G'. The cut it provides gives us a minimum cut in G with the minimum number of edges in it. Call this number of edges l.

- Finally, we check to see if G has a min cut with at most k edges. If $l \leq k$, then we return True. Otherwise, we return False.

Analysis We construct the graph G' in $O(V + E)$ time. We call the runtime of the black box algorithm BB, where BB is a O function according to V and E. We run the black box algorithm once, and our third step runs in constant time, so we say the whole routine runs in $O(V + E + BB)$ time.

■