The algorithm is stated as follow:

Algorithm 1 Algorithm for problem 1

Draw the residual graph $G_f = (V, E_f)$ based on the pre-computed flow f
$(v, w) = [\ ]$
Run BFS from the sink t (reversed BFS):
While paths **is not** empty:
Check the edges (v_i, w_i) (v is on s side and w is on t side) at each level of paths
    If v **is** reachable from s, and w **is not**:
        (v, w).add((v_i, w_i))
        Remove the path containing (v_i, w_i) from paths
    Endif
Endwhile
EndBFS
Cut the graph into (A, B) on the edge set (v, w) with $s, v \in A$ and $w, t \in B$
Return (A, B)

Proof of Correctness:

From the lecture we already proved that for the residual graph G_f generated from the maximum flow f, there exists a cut $C(A, B)$ and the set of edges that links A and B is (v, w) where v is reachable from s and w is not. Such cut is a minimum capacity cut and the capacity equals to the value of maximum flow f. Therefore we guaranteed that the cut in our algorithm is the minimum capacity cut. Now we need to prove this cut has as many node in A as possible, which is equivalent to proving as few nodes in B as possible. We use contradictory: If there is another minimum cut (A', B') with fewer nodes in B', at least one node w' in B should be in A'. However, running BFS on G_f from node t, we get (A, B) if and only if we get set (v, w) where all edges satisfies v is reachable from s and w is not for the **first time**. Hence for any node w'' in B, (w', w'') (i.e., any node pair in B) would not satisfy the condition. Therefore (A', B') is not a minimum cut. Hence we proved that (A, B) is the minimum cut with as many nodes in A as possible.

Time Complexity Analysis::

Since f has already been pre-computed, the only time cost in this algorithm is BFS. We know in a graph the time complexity for a BFS algorithm is $O(m' + n)$. Given that $n \leq m$ and we know $m' \leq 2m$, the time complexity is $O(m' + n) = O(3m) = O(m)$.