Table of Contents

Question 1.

Algorithm

Find_Critical_Edge (G = {V, E}: a network, s: start vertex, t: end vertex, c: capacities)

```
1: min_cut = Ford_Fulkerson(G, s, t, c)
2: edge = find an edge from min_cut to outside the min_cut using BFS
3: return edge
```

The complexity of this algorithm is simply the complexity of `Ford_Fulkerson`, which is $\mathcal{O}(mC)$ where m is the number of edges and C is the capacity.

Proof of correctness:

Let `e` be the edge returned by `Find_Critical_Edge()`

WTS: `e` is a critical edge

We know `e` is an edge across the min cut since since we run `Ford_Fulkerson()` in `Find_Critical_Edge()` and return an edge from the min cut to outside the min cut.
By the Max-flow Min-cut theorem, we know that the maximum value of the `s-t` flow is equal to the sum of the capacities of an `s-t` cut.

So if we reduce the capacity of any edge leaving a min cut, then the value of the min cut decreases. Which in turn, reduces the max flow by the Max-flow Min-cut theorem. ■

Question 2.

To show that finding the minimum vertex cover reduces to finding the min cut, we will show that we can create an algorithm that takes the minimum vertex cover and gives the min cut of the corresponding graph.

Algorithm
Reduce_Vertex_Cover (`G = {L, E, R}`: bipartite graph, V: minimum vertex cover)

```
1: let $G' = G \cup \{s,\ t\} \cup \{(s,\ u) \mid \forall u \in L\} \cup \{(u,\ t) \mid \forall u \in R\}$
2:
3: set a weight of 1 to all edges going out of $s$, and all edges going into $t$
4: set all other edges to $\infty$
5:
6: let $V_1 = \{V \cap L\}$
7: let $V_2 = \{V \cap R\}$
8:
9: **return** $\{s\} \cup \{L - V_1\} \cup V_2$
```

Proof of correctness:

This algorithm takes in a minimum vertex cover and returns a min cut of the corresponding network flow.

This is because we set all edges in the bipartite graph to ∞ and our cut doesn't include any of those edges. The cut includes all L vertices that are not part of the vertex cover, and all the R vertices that are part of the vertex cover, therefore, any edge going from $L - V_1$ to $R - V_2$ will not be covered by the vertex cover. This goes against our against our restriction that v must be a vertex cover in the original graph. Therefore, the only edges leaving the cut are from s to V_1 and from V_2 to t.

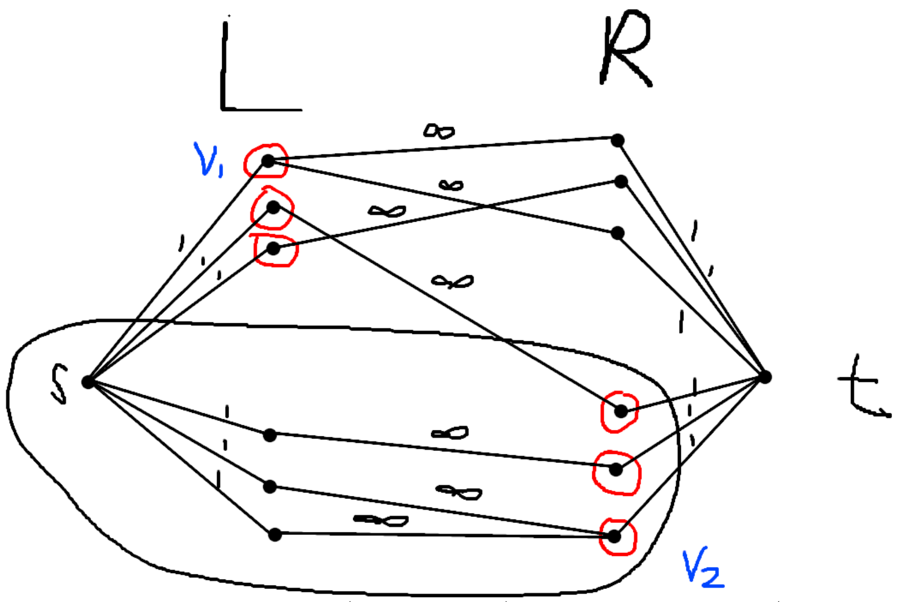

Figure 1: a minimum vertex cover (circled in red) with a minimum cut (circled in black)

This is the minimum cut because the only way to disconnect the flow network is to disconnect s from L, L from R, or R from t. We have shown above that L and R are disconnected. Therefore we must include all edges from S to V_1 and from V_2 to t in this cut as well. Otherwise, a path could be made from s to V_1 to $R - V_2$ to t or s to $L - V_1$ to V_2 to t. Therefore this cut includes all the necessary edges to disconnect the graph while not including any ∞ weight edges. We know it's the minimum cut since each edge has weight 1 and the cut has the only necessary edges.

This is a reduction from vertex cover to max flow min cut because we have shown that an algorithm exists that takes a bipartite graph and turns it into a flow graph, where the vertex cover can give us the min cut and a false vertex cover will give us a false min cut.

Question 3.

Algorithm
Classify_Vertices (`G = {V, E}`: a flow network, `s`: start vertex, `t`: end vertex, `c`: capacities)

```
1: let $(f,\ G_r)$ = Ford-Fulkerson $(G,\ s,\ t,\ c)$
2: let $(f',\ G_r')$ = Ford-Fulkerson $(\bar{G},\ t,\ s,\ c)$
3:
4: let $u$ be all vertices reachable by $s$ in $G_r$
5: let $d$ be all vertices reachable by $t$ in $G_r'$
6: let $c$ be $E - u - d$
7:
8: return $(u,\ d,\ c)$
```

Proof of complexity:

We run Ford-Fulkerson twice, and do 2 BFS's from the 2 start nodes in each residual graph. So in total, we have $2O(m^2 \log c) + 2O(m) \in O(m^2 \log c)$. Therefore, we are still in a constant factor of the complexity of Ford-Fulkerson.

Proof of correctness:

Lemma 1: All vertices which are upstream must be reachable from the source of a flow residual graph G_r returned by Ford-Fulkerson

Proof of **Lemma 1** by contradiction

Suppose to the contrary. Then there exist a vertex u that is not reachable from the source s but is on the source side of a min cut. Therefore in any min cut, we can find a path in the residual graph from an arbitrary vertex v reachable from s, to u, to t to push flow to create the min cut where u is reachable from the source. However, this path connects s to t in the original min cut in the residual graph, which is a contradiction. Therefore our assumption is wrong and u must always be reachable from the source in any min cut.

By **Lemma 1**, we can run Ford-Fulkerson and run BFS from s on G_r to get the upstream vertices. Also by **Lemma 1**, we can do the same thing with the edges reversed and switching t and s to get the downstream vertices.

Since we have the upstream and downstream vertices, we can simply subtract u and d from V to get all the central vertices. We know that they are central because by the definition of a cut, there are vertices reachable by the source, and everything else. Therefore every vertex must be in one of the 2 partitions. And since the upstream ones are always on one partition, and the downstream are always in the other, the remaining must lie in one of the partitions in some of the cuts, and in the other one in the other cuts.