# CSCC73H3 - A6

Jan Garong & Leo Wang

March 13, 2023

## Question 1

### Algorithm

**Algorithm 1** Critical_Edge($G$)

**return** an edge cut in the minimum cut algorithm on $G$.

### Complexity

Since the min cut algorithm just runs Ford Fulkerson, then a BFS/DFS on the residual (which is linear), which keeps the complexity at $O(mC)$, with $C$ being the flow out of the sink and $m$ being the number of edges.

### Correctness

*Proof.* Since the algorithm just returns the edges that are cut in the min cut algorithm, we will need to prove that those edges are critical edges.

We know that by the Weak Duality theorem for the maximum flow + min cut problem states that $v(f) \leq cap(A, B)$, such that $v(f)$ is an arbitrary flow in the graph $G$ and that $A \cup B = G$. Since we know running the min cut algorithm will produce a cut such that $cap(A, B) = v(f)$, with $v(f)$ being the max flow. Since the edges joining $A$ and $B$ is equal to the maximum flow, decreasing its capacity would in turn decrease the maximum flow. This by definition is what a critical edge does, which therefore concludes the edges that are cut are critical edges. □

# Question 2

*Proof.* **Note:** We will assume that all nodes in the bipartite graph has an edge. If that is not the case, we should remove them since it won't change the edges we need to cover in the bipartite graph.

Lets use the fact that the minimum cut problem can be reduced to the maximum flow problem. If we can show the problem of minimum vertex cover can be reduced to a minimum cut problem, we will be able to prove that minimum vertex cover can be reduced to the maximum flow problem.

When we structure the bipartite graph similar to how we structure the graph for max matching, with the bipartite section being composed of $L$, the nodes on the left, and $R$, the nodes on the right, we get a cut of $A$ and $B$, when running the minimum cut algorithm. Since the edges between the bipartite graph are set with edges of infinity, the minimum cut will only go through edges weighted 1 (that being between the sink or source), which means the number of edges cut in minimum cut must be $|A \cap R| + |B \cap L|$.

Due to this, we know that a sufficient vertex cover for the bipartite graph would be all nodes in $Z = (A \cap R) \cup (B \cap L)$. To prove this is a vertex cover, we will use proof by contradiction. Let's assume that $Z$ does not create a vertex cover, which means there exists an edge that is not covered by the nodes in $Z$. Since $Z$ covers all edges that are part of the min cut, this edge must be connected by two nodes in either $A$ or $B$. For now, let's assume the 2 nodes are both in $A$. If that is the case, then one of the nodes must be in $A \cap L$, and the other in $A \cap R$, which guarantees that one of these nodes must be in $Z$ which should cover this edge. Since the same applies to $B$, we can say that the set $Z$ is a valid vertex cover.

To prove that this produces a minimum cover, we need to show that removing a node in the set $Z$ will not produce a vertex cover, which is by showing how it will miss an edge. The edge it will end up missing will be any that joins $A$ and $B$ together in the min flow diagram, which we know has to exist since the existence of $Z$ is based on how they are the nodes that share an edge that is cut by the min cut algorithm.

Therefore the minimum vertex cover problem can be reduced to a maximum flow problem.

□

# Question 3

## Idea

There are two important min cuts we need to find: the one closest to the source and the one closest to the sink. We can define this notation of "closest" as a min cut $(A, B)$ such that there does not exist another cut $(A', B')$ such that $A' \subset A$ (for being closest to the source, $B' \subset B$ for sink).

The Ford-Fulkerson algorithm should find the min-cut closest to the source, and we if reverse the direction of all the edges in the flow network graph and run Ford-Fulkerson on this reversed graph, we should get the min-cut closest to the sink of the original graph.

## Algorithm

**Algorithm 2** Classify_Nodes$(G, s, t)$

1: Label all nodes in G as "central"
2: $G_r \leftarrow$ residual of $G$
3: $A \leftarrow$ All nodes reachable using DFS/BFS on $G_r$ starting at node $s$
4: Label all nodes in $A$ as "upstream"
5: $G' \leftarrow$ G with all edges direction reversed
6: $G'_r \leftarrow$ residual of $G'$
7: $A' \leftarrow$ All nodes reachable using DFS/BFS on $G'_r$ starting at node $t$
8: Label all nodes in $A'$ as "downstream"
9: **return** $G$ with labels

## Justification & Complexity

### Show Ford-Fulkerson gets "Closest" min cut

For the "closest" min-cut $(A, B)$, suppose there exists a min cut $(A', B')$ such that $A' \subset A$.
This means there must exist at least 1 node $a \in A$ which is not in $A'$ and an edge of the form $(a', a)$ where $a' \in A'$ which would have weight 0 in the residual graph since it would not be optimal flow if you did not use all possible capacity between $(A', B')$.
However, the FF algorithm would not have been able to choose $(A, B)$ as the min cut as the graph search would not have traversed $(a', a)$ because the weight is 0.
Thus FF algo chooses the "closest" min cut to the source.

### Show reversing $G$ and running FF will get right-most min cut

It's fairly easy to see reversing the graph should yield the same max flow since if $x$ amount can flow from s to t, it should be able to go backwards (also hence the use of a residual graph in Ford-Fulkerson).
Thus reversing the graph and running FF will get the min cut closest to the sink.

### Correctness

Given we have the min cuts closest to the source $(A_s, B_s)$ and the source $(A_t, B_t)$, this means all other min cuts $(A', B')$ must have properties $A_s \subset A'$ and $B_t \subset B'$. Thus all nodes in $A_s$ are upstream, $B_t$ are downstream, and the rest are central.

### Complexity

We only run Ford-Fulkerson 2 times, so the complexity is constant with respect to that running time, $O(mC)$ at worst as shown in lecture. We know BFS/DFS is $O(n + m) \in O(m)$ for finding the sets for a min cut given the residual graph.
This total time complexity is $O(mC)$