# Notes 11-20

Cameron Chalk

## 1 Max Flow Algorithm

Consider the algorithm in the slide titled "Towards a Max Flow Algorithm". To augment the flow along a path $P$ is to increase the flow as much as possible on that path; i.e., increase the flow on each edge by the minimum of the remaining capacities along the path. This can be seen as a Greedy algorithm. In some cases, this algorithm seems to give a max flow correctly; however, as shown in class, it does not always return the max flow. Instead, we will see a different type of algorithm.

## 2 Ford-Fulkerson algorithm

First, we will consider paths which can take edges "backwards". I.e., we will ignore the directionality of edges when considering $s-t$ paths. When considering such a path, when we augment the path, edges which are "forward" are increased and edges which are "backward" are decreased.

More formally, given a directed graph $G$ and a flow $f$ for $G$, we will construct the *residual graph* for $f$, $G_f$. The residual graph will include "backwards" edges. We construct $G_f$ in the following way: for each edge $e = (u, v)$ in $G$, $G_f$ contains two edges: $(u, v)$ with weight $c(e)-f(e)$ (the *remaining capacity edge*) and $(v, u)$ with weight $f(e)$ (the *flow edge*). If either the value $c(e) - f(e) = 0$ we do not include the corresponding edge $(u, v)$, and if $f(e) = 0$ then we do not include the corresponding edge $(v, u)$.

Now we use the residual graph $G_f$ to extend our previous technique. See the slide titled "The Ford-Fulkerson Algorithm". Intuitively, we generate $G_f$, choose a path, augment it, update the flow $f$ based on our changes to $G_f$, then update $G_f$ based on our changes to $f$, and repeat.

The algorithm terminates when there is no $s-t$ path in $G_f$. On termination, the flow $f$ is optimal. Next, we will find the runtime (and thus prove that the algorithm does terminate), and we will prove that the flow is optimal upon termination.

Running time: We will assume the graph is connected an thus $m \geq n - 1$ ($m$ is the number of edges and $n$ is the number of vertices). In each loop, we must (1) find $s-t$ path, (2) we must augment the flow $f$ to $f'$, and (3) we must construct $G_{f'}$ (given $G_f$). (1) requires a run of BFS/DFS which takes $\mathcal{O}(m)$ time. (2) takes $\mathcal{O}(m)$ time, since an $s - t$ path may be of length approximately

$m$. (3) requires $\mathcal{O}(m)$ time to update the $G_f$ to $G_{f'}$, since we must change each edge on the $s-t$ path which could be length $m$. Next, we must find the number of iterations of the loop. Note that each iteration increases the value of the flow by at least one. This gives an easy upper bound on the number of iterations; and thus we know that the number of iterations is finite. Let $C$ be the sum of the capacities out of $s$; this upper bounds the value of the flow, and thus $C$ also bounds the number of iterations. Then the runtime is $\mathcal{O}(C \cdot m)$.

Now we prove correctness of the algorithm.

**Claim:** if no $s-t$ path in $G_f$, then there is $s-t$ cut $(A, B)$ s.t. the value of the flow is equal to the capacity of the cut; i.e., $v(f) = \text{cap}(A, B)$.

Recall that $v(f)$ can be defined as the net flow of edges crossing any $s-t$ cut $(X, Y)$, where if the edge crosses from $X$ to $Y$ the flow is added, and if the edge crosses from $Y$ to $X$, the flow is subtracted. Also recall $\text{cap}(A, B)$ can be defined as the sum of capacities of edges which cross from $A$ to $B$ (not counting those crossing from $B$ to $A$). Recall from the previous class that if $v(f) = \text{cap}(A, B)$, then $f$ is the max flow and $(A, B)$ is the min-cut—the lemma we used to see that is true is the following: $\forall$ flows $f$ and $\forall$ $s-t$ cuts $(A, B)$, $v(f) \leq \text{cap}(A, B)$.

Now we prove the claim. Assume that there is no $s-t$ path in $G_f$. Let $A$ be the set of all nodes reachable from $s$ in $G_f$. Recall,

$$v(f) = \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e).$$

By definition of our set $A$, we know that in $G_f$ there are no edges which cross from $A$ to $B$. Because there is no edge $e$ out of $A$ in $G_f$, we know that the flow on each of these edges in $G$ is $f(e) = c(e)$, and we can rewrite $\sum_{e \text{ out of } A} f(e)$ to $\sum_{e \text{ out of } A} c(e)$. Similarly, because there is no edge $e$ out of $A$ in $G_f$, we know that each edge in $G$ in to $A$ has no backward edge in $G_f$. For that reason, we know that $\sum_{e \text{ in to } A} f(e)$ as 0. Thus we have $v_f = \sum_{e \text{ out of } A} c(e) = \text{cap}(A, B)$, which is the claim to prove.

# 3 Towards extending Ford-Fulkerson algorithm

Note that Ford-Fulkerson gives integer-value flows. It is not obvious whether or not allowing non-integer flow values can lead to different solutions. Next class we will see how Ford-Fulkerson can be used to solve many other problems which are not apparently related to max flow.