# CS 401 Computer Algorithms I, UIC, Fall 2012
# End of Lecture on Scaling Max-Flow Algorithm (Nov 2, 2012)

At this point in the lecture, we had discussed a proof of Lemma 7.18, which stated the following.

**Lemma 1.** *Let $f$ be the flow after the $\Delta$-scaling phase. Then, the true max-flow $f^*$ satisfies $v(f^*) \leq v(f) + m\Delta$.*

Recall that our Scaling Max-Flow algorithm ran its outer loop $1 + \log \Delta \leq 1 + \lceil \log C \rceil$ times. Hence, it remains to bound the cost of any one run of the loop. Using the lemma above, we can bound this cost as follows.

**Claim 2.** *The number of augmentations (i.e. calls to augment(f,P)) in* any *scaling phase is at most $2m$.*

*Proof.* We begin with the first scaling phase, in which $\Delta$ is the largest power of 2 satisfying $\Delta \leq C$. Hence, there can only exist 1 edge out of $s$ in $G_f(\Delta)$. This implies we can have at most 1 augmentation, which satisfies our claim.

Consider now any later scaling phase $\Delta$, and let $f_p$ be the flow at the end of the scaling phase just before $\Delta$; this previous scaling phase had parameter $\Delta' = 2\Delta$. Thus, by the lemma above, we know that the true max-flow after the $\Delta'$-scaling phase satisfied the condition

$$v(f^*) \leq v(f_p) + m\Delta' = v(f_p) + 2m\Delta. \tag{1}$$

But now in the $\Delta$-scaling phase, each augmentation operation increases the flow by at least $\Delta$. Hence, by Eqn. (1), the number of augmentations possible in the $\Delta$-scaling phase is at most $2m$. This concludes the proof. □

In sum, we have that the Scaling Max-Flow algorithm computes at most $2m(1 + \lceil \log C \rceil)$ augmentations. Since each augment call costs $O(m)$, our total runtime is hence $O(m^2 \log(C))$. This is polynomial-time. Compare this to the pseudo-polynomial-time FF-algorithm, which required $O(mC)$ time. Note: In practice, if $m$ is large and $C$ is small, the FF-algorithm is actually faster!