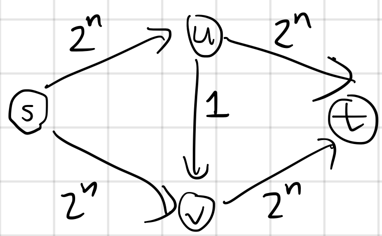
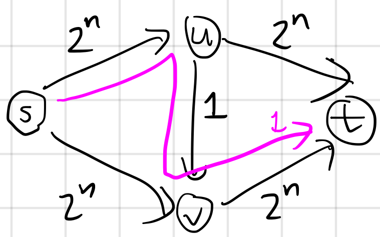
16 March 2018: The Edmonds-Karp Max-Flow Algorithms

Ford-Fulkerson $O(mC)$ running time bound is pseudopolynomial, not polynomial.

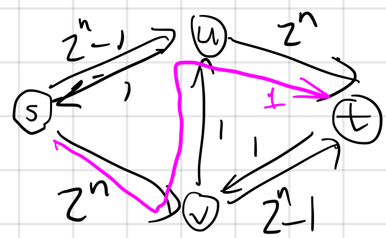
This can actually lead to exponential running time if you make deliberately weird choices of augmenting paths.



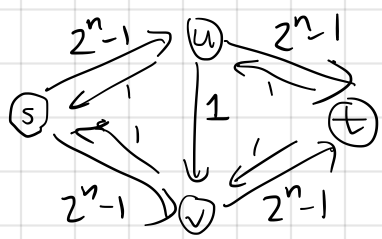
Takes only $O(n)$ bits to describe this network.



RESIDUAL

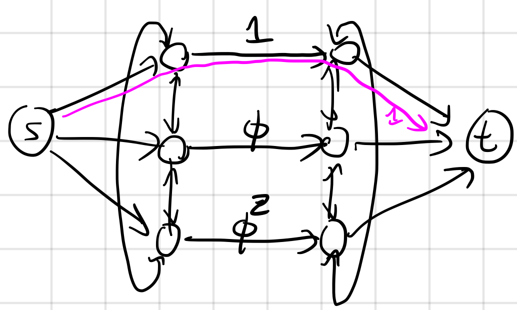


RESIDUAL #2



This can iterate 2^n-1 more times until completion. (i.e., $2\cdot(2^n-1)$ additional augmenting paths)

If edges have irrational capacities, could run for ∞ iterations.

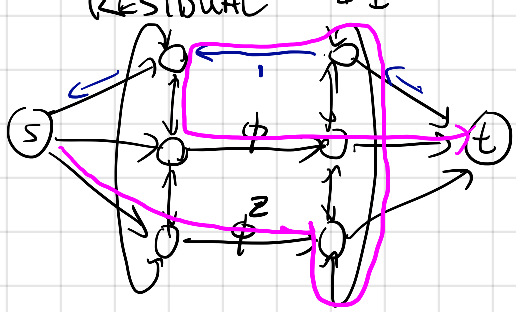


Let $\phi = \frac{\sqrt{5}-1}{2} = 0.618\ldots$

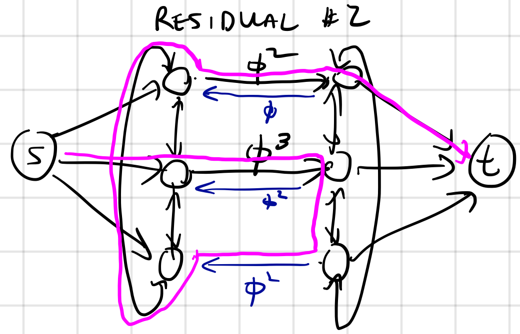
$\phi + \phi^2 = 1.$

All unlabeled edges have capacity > 2.

RESIDUAL #1



RESIDUAL #2



$$\phi^2 + \phi^3 = \phi$$
$$\Downarrow$$
$$\phi - \phi^2 = \phi^3$$

As we iterate this process, the residual capacities of the forward edges get scaled down by ϕ each time, so the process runs for infinitely many iterations.

Two polynomial-time algorithms for max flow:

* Edmonds-Karp Heuristic #1: always choose augmenting path P that maximizes $\text{bottleneck}(f, P)$. (Similar to Dijkstra)
  $O(m \log n)$ per iteration
* E.-K. Heuristic #2: always choose augmenting path P with fewest edges. (BFS) $O(m)$ per iteration.

Bounding # of iterations.

For heuristic #1, keep track of min-cut capacity of G_f as a measure of progress. Initially $G_f = G$ and $\text{min-cut}(G_f) \leq C$.

In each iteration let $b := \text{bottleneck}(f, P)$. I claim that G_f has a cut with capacity $\leq m \cdot b$.

Let $A := \{$ vertices reachable from s in G_f using any path made up of edges with resid. cap $> b \}$

$B := \{$all other vertices$\}$.

Observe $s \in A$, $t \in B$, edge set $E(A, B)$ in G_f is made up of $\leq m$ edges each of residual capacity $\leq b$, hence

$$c_f(A, B) \leq m \cdot b$$

as claimed.

In any iteration of EK#1 algorithm, at start of iteration

$$\text{min-cut}(G_f) \leq m \cdot b$$

and we manage to send b additional units of flow, so at end of iteration

$$\text{min-cut after} \leq (\text{min-cut before}) - b \leq \left(1 - \frac{1}{m}\right) \cdot (\text{min-cut before})$$

After k iterations, by induction,

$$\text{min-cut}(G_f) \leq \left(1 - \frac{1}{m}\right)^k \cdot C$$

So in particular, after m iterations, because $\left(1-\frac{1}{m}\right)^m < \frac{1}{e}$,

$$\text{min-cut}(G_f) < \frac{1}{e} \cdot C$$

After $m \ln(C)$ iterations,

$$\text{min-cut}(G_f) < \left(\frac{1}{e}\right)^{\ln C} \cdot C = \frac{1}{C} \cdot C = 1.$$

But min-cut(G_f) is also a non-negative integer. So it equals zero. So the algorithm terminates!

$$\text{Running time} \leq (\text{Running time per iter}) \cdot (\#\text{ iters})$$

$$\leq O(m \log n) \cdot O(m \ln C)$$

$$= O(m^2 \log n \log C)$$

This is polynomial in the input size.

Remark. EK#2 heuristic runs in $O(m^2 n)$. "strongly poly"
Proof in lecture notes online.