3/15/2019

A truly polynomial time Max-Flow algorithm.

Ford-Fulkerson (G)

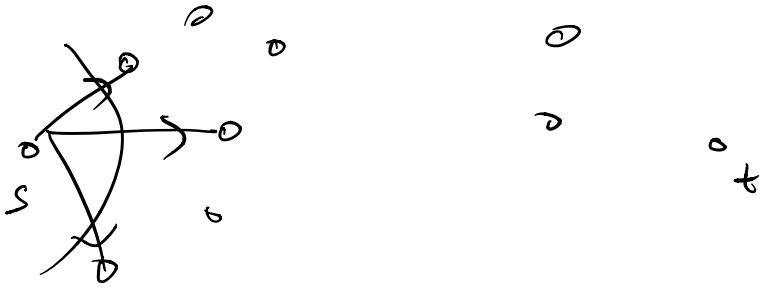
$f \leftarrow \vec{0}$; $G_f \leftarrow G$

while (G_f has an augmenting path P)

augment (f, P)

recompute G_f.

endwhile.



$$C = \sum_{u \in V} c(s,u)$$

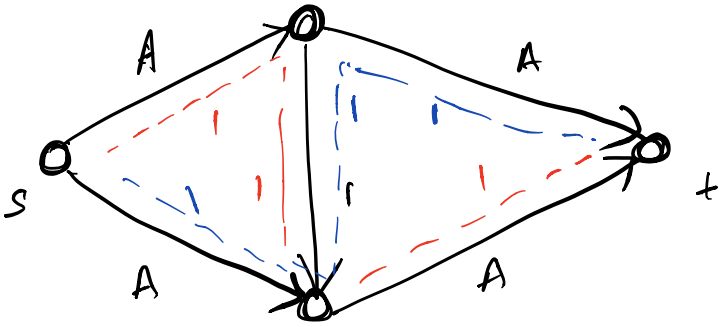
Running time of F-F : $O(C \cdot m)$;

m is the number of edges in G.

Concern: F-F is pseudopolynomial : to write C takes just $(\log C)$ bits. So, if C is large, F-F can be exponential time.

Is our analysis tight?

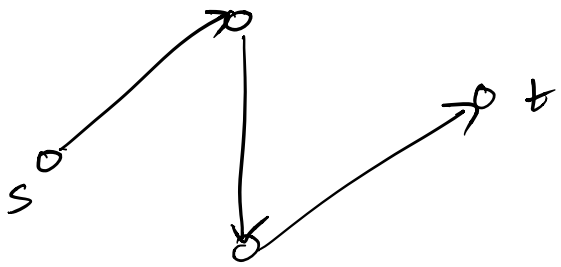
Pathological examples:



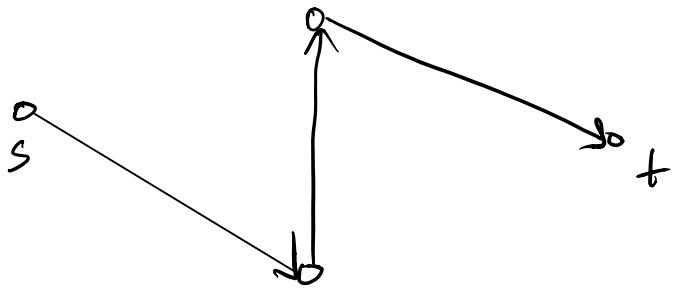
A is a large number.

Augmenting paths:

P_1 :



P_2 :



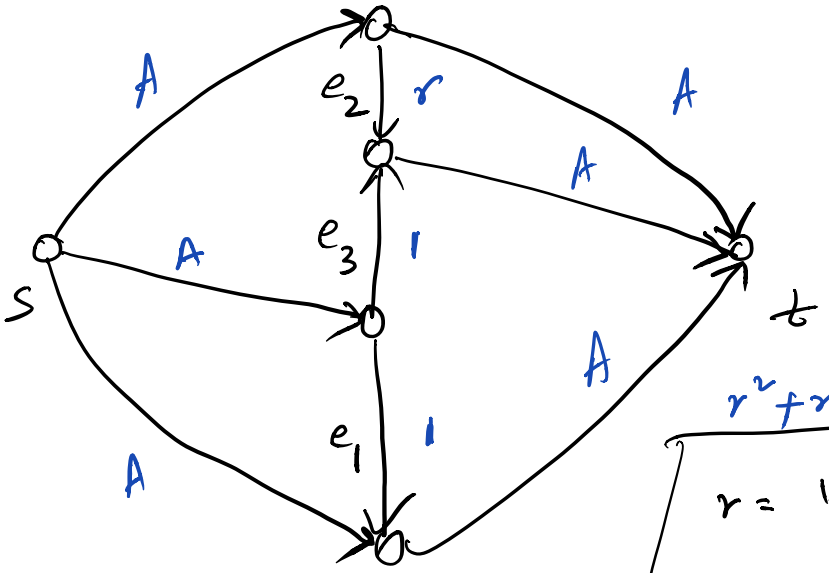
Consider a sequence of augmenting paths:

$P_1, P_2, P_1, P_2, \ldots$

Takes $2A$ iterations. Thus running time is $\Omega(A)$.

An even more pathological example:

Suppose we are allowed to have real valued capacities.



Max-Flow: $2A+1$

$r^2 + r - 1 = 0;\ r > 0$

$r = \frac{\sqrt{5}-1}{2} \approx 0.62.$

$a_n = r^n.$

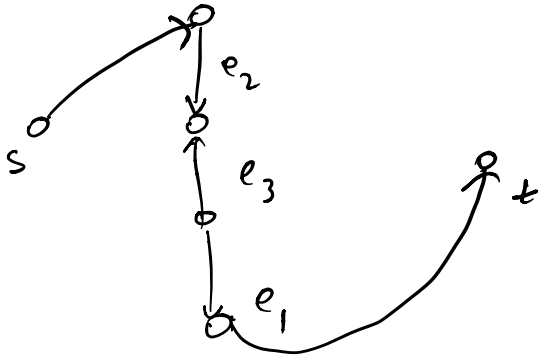
$c(e_1) = 1 \qquad c(e_2) = r \qquad c(e_3) = 1$

$a_0 = 1\ ;\quad a_1 = r\ ;\quad a_{n+2} = a_n - a_{n+1}.$

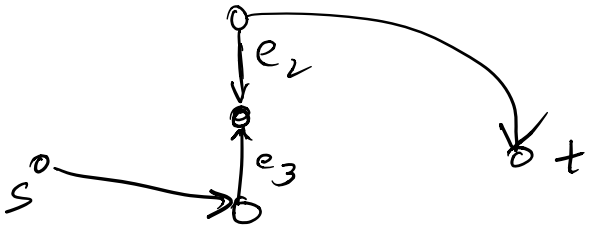
Capacity of all other edges is $A \geq 5$.

Consider the augmenting paths;

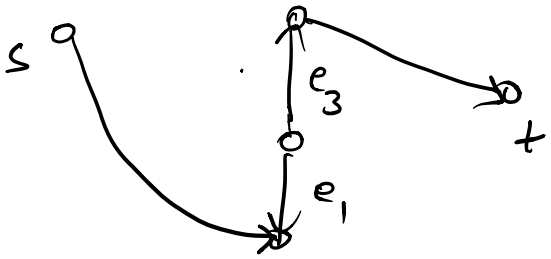
P_1:



P_2:



P_3:

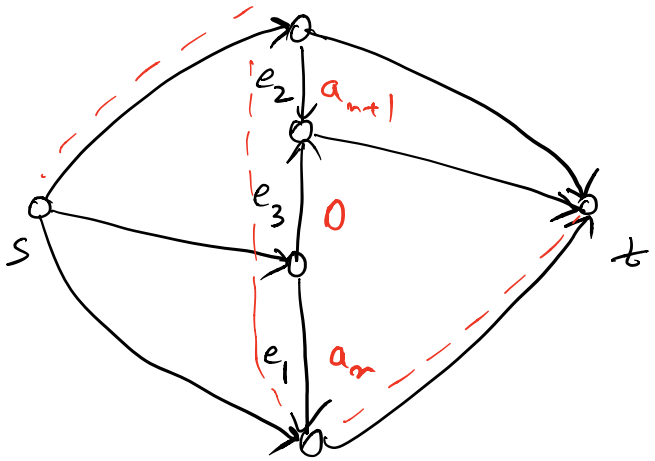


Suppose residual capacities of e_1, e_2, e_3 are $a_n, a_{n+1}, 0$. [Note initially, $n = 0$]

$$(a_n, a_{n+1}, 0) \xrightarrow{P_1} (a_{n+2}, 0, a_{n+1}) \xrightarrow{P_2}$$

$$(a_{n+2}, a_{n+1}, 0) \xrightarrow{P_1} (0, a_{n+3}, a_{n+2})$$

$$\xrightarrow{P_3} (a_{n+2}, a_{n+3}, 0).$$



In path P_1, e_3 is backward and has capacity 1.

Thus, min-capacity is a_{n+1}.

e_2: $a_{n+1} \longrightarrow 0$

e_3: $0 \longrightarrow a_{n+1}$

e_1: $a_n \longrightarrow a_n - a_{n+1} = a_{n+2}$.

Work out the other transformations.

Thus, $(a_n, a_{n+1}, 0) \xrightarrow{P_1 P_2 P_1 P_3} (a_{n+2}, a_{n+3}, 0)$;

and flow increases by $2(a_{n+1} + a_{n+2})$.

Thus, repeating this k times, $k \to \infty$,

Total flow $\leq 2(a_1 + a_2) + 2(a_3 + a_4) + \cdots$

$$= 2\sum_{i=1}^{\infty} a_i = \frac{2r}{1-r} = \frac{2}{r} < 4.$$

$(1 - r = r^2)$

Edmonds - Karp Algorithm.

```
Ford-Fulkerson (G)
    f ← 0⃗ ;  G_f ← G
    while ( G_f has an augmenting path )
        Let P be the shortest s-t path in G_f.
        augment (f, P)
        recompute G_f.
    endwhile.
```

Running time complexity:

Use Breadth-first search (BFS) to compute the shortest s-t path in G_f. $O(m+n) = O(m)$ time.

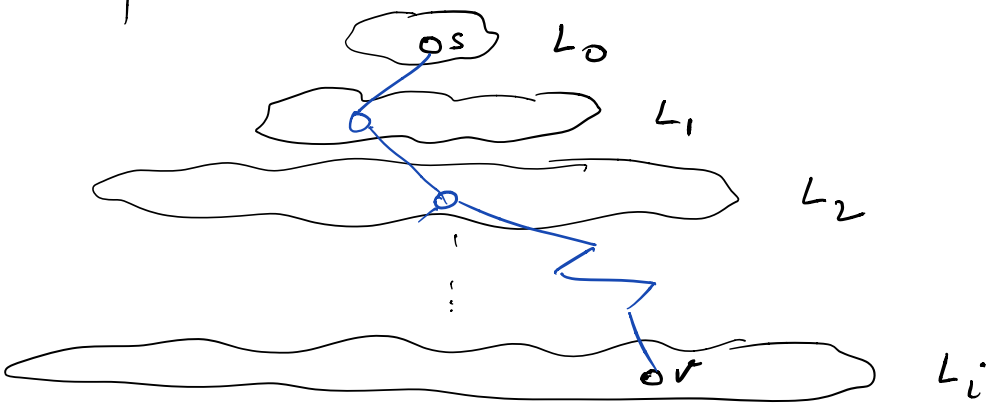
Each loop takes $O(m)$ time.

We prove: At most $O(mn)$ loop iterations.
Thus, running time is $O(m^2 n)$.

Bound on number of loop iterations:
Recall BFS: with source s.

We get Layers $L_0, L_1, \ldots, L_k$,

L_i: All vertices at distance exactly i from s.



Any shortest path from $s \rightsquigarrow v$, $v \in L_j$ uses exactly one vertex from each L_i, $i < j$.

Claim: Any new edge that appears in an iteration of the 'while loop' in Edmonds-Karp must be from L_j to L_{j-1}, for some $j \geq 1$.

Pf sketch: Observe

(a) If flow is pushed on a forward edge, a back edge may be added.

(b) If flow on a back edge is reduced, new forward edge may appear.

(or)

(c) No new edges may appear.

Thus if a new edge appears, it is in opposite direction of an edge in P.

Since P is a shortest path, if $e = (u, v) \in P$, then $u \in L_i$, $v \in L_{i+1}$ (for some $i \geq 0$).

Thus, a new added edge must be from L_j to L_{j-1} for some $j \geq 1$.

□.

Corollary: Distance of any $v \in V$ never decreases.

Now suppose $e = (u, v)$ appears in the shortest path from s to t, and is the bottleneck edge. It is removed from G_f.

Recall, $u \in L_i$, $v \in L_{i+1}$ for some $i \geq 0$.

At a later point, if e is added, it must be that $u \in L_j$, $v \in L_{j-1}$.

Further, $j-1 \geq i+1 \Rightarrow j \geq i+2$.

Thus, u goes from L_i to atleast L_{i+2}.

$\Rightarrow$ e can be added at most $\frac{n}{2}$ times.

$\therefore$ Any edge e can be a bottleneck edge at most $\frac{n}{2}$ times.

Since there are m edges, the max. iterations of the while loop is $m \cdot \left(\frac{n}{2}\right)$.

□.