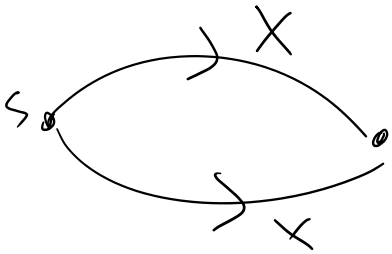
- Ford-Fulkerson takes X step



(Ideal)

There is related Weighted — Vertex Cover

Idea: Choose path more intelligently.

Edmonds-Karp Algo

- init $f=0$
- while there is path from s to t in G_f

$O(m)$ iter
- Let p be the shortest path from s to t in G_f
- Send flow along p (i.e. update f)
- Update G_f

Thm It takes $O(mn)$ iter, $O(m^2n)$ time

Claim Let $d_f(s,t)$ be the shortest dist (in terms of edges) from s to t in G_f.

a) $d_f(s,t)$ never decreases

b) $d_f(s,t)$ increase by at least 1 per m iter

Proof of Thm $d_f(s,t) \in \{0, \dots, n-1, +\infty\}$

After mn iter $d_f \geq n \Rightarrow d_f = +\infty$

After mn iter, $d_f \geq n \Rightarrow d_f = +\infty$.

So, algo terminates in mn iter.

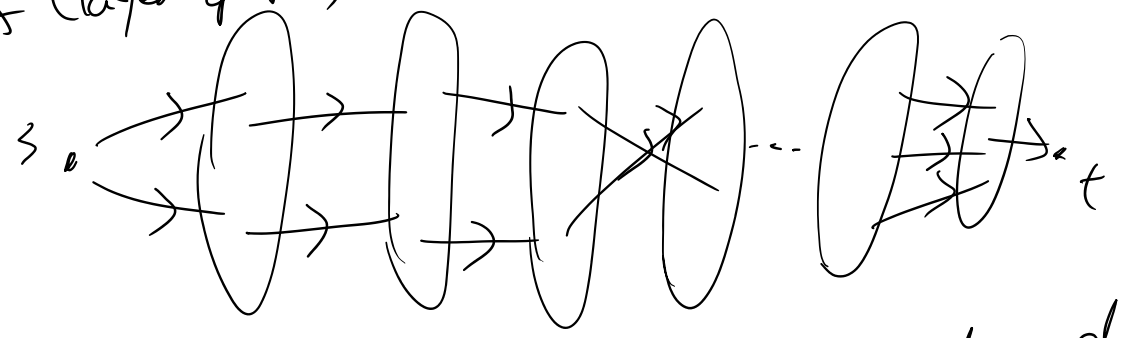
shortest path has $\leq n-1$ edges.

Understand shortest path s to t

Recall BFS discovers vertices in layer.

$v \in$ layer i if $d_f(s,v) = i$

L_f (layer graph)



layer 0 1 - - - - d_f-1 d_f

There are 3 types of edges

- forward: layer $i \to$ layer $i+1$
- sideways: layer $i \to$ layer i
- backward: layer $i \to$ layer j $(j < i)$ (because directed)

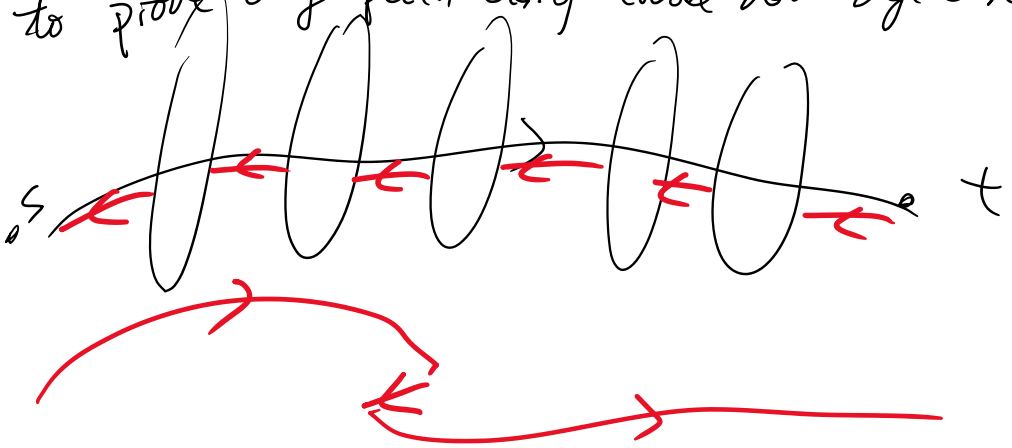
Def L_f is subgraph of G_f consists only forward edges.

Yet L_f is subgraph of G_f consists only forward edges.

Obs: all shortest path from s to t is on L_f.

Proof of claim

Worry: augmentation introduce new edge.
We need to prove any path using those new edge are longer



Any path from s to t using backward edges has dist $\geq d_f(s,t)+2$.

So, $d_f(s,t)$ didn't decrease.

Proof of claim b)

Consider t iterations of algo st $d_f(s,t)$ stays same.

Every augmentation ~~use~~ use at least t forward edges.

So, after m iter, L_f becomes empty.

So, $d_f(s,t)$ must increase after m iter.

Karzanov's E-K algo

Key idea: E-K algo

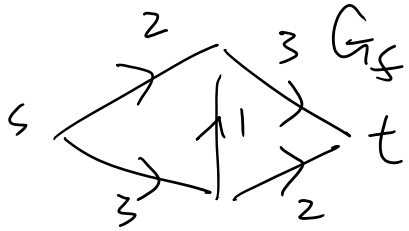
Instead of sending flow,
it's about disconnecting s and t.

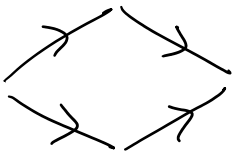
Dinic's algorithm

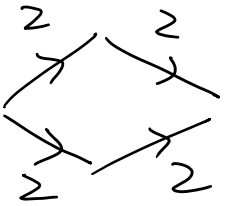
Key idea: Flood the layer graph.

Instead of sending 1 path from s to t,
send multiple one.

Def A flow g is a blocking flow in L_f if
for every s-t path in L_f, there is edge $e \in P$ s.t
it saturated by g.







(example of blocking)

How to find blocking flow?

Naive : $O(mn)$ time

Better : $O(m \log n)$ time (link-cut tree)

Algo:

Algo:

- init $f = 0$
- While there path s to t in G_f
  - Find blocking flow g in $L_f \subset G_f$
  - Send g
  - Update G_f and L_f

Lem $d_f(s,t)$ increase by 1 every iter.

Thm $O(mn \log n)$ [Sleator, Tarjar 1982]

Thm $O(mnC)$ (last lecture)

Best algo doesn't depends on C $O(mn)$ [Orlin 2012]

Homework Dinic takes $O(m\sqrt{m})$ if $C = 1$.

In general $O(m\sqrt{m} \log C \log n)$ [Goldberg, Rao 1998]

Now $O(m\sqrt{n} \log^{O(1)} nC)$ [Lee, Sidford 2014]