ALGORITHMS AND DATA STRUCTURES · LECTURE 10

The Edmonds–Karp algorithm

Maximum flow in polynomial time

Heng Guo

School of Informatics, University of Edinburgh

Thursday 22 October 2026 · 16:10–17:00

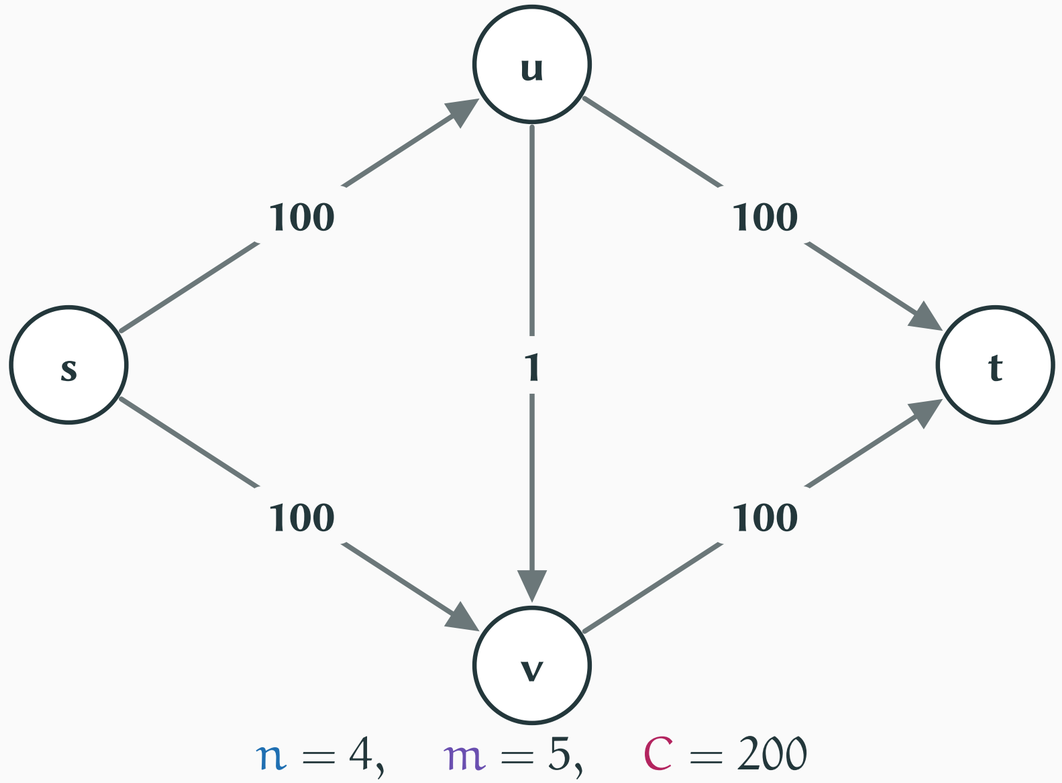


Can we avoid the dependence on C?

Last lecture: Ford–Fulkerson makes at most C iterations of $O(m)$ time each: $O(mC)$.

But C is a **number**, not the size of the input: five edges can have $C = 200$ or $C = 10^9$.

? *Can the running time depend on n and m only?*



$n = 4, \quad m = 5, \quad C = 200$

Recap: Ford–Fulkerson

Ford–Fulkerson(G, s, t)

1 $f(e) \leftarrow 0$ for every edge e

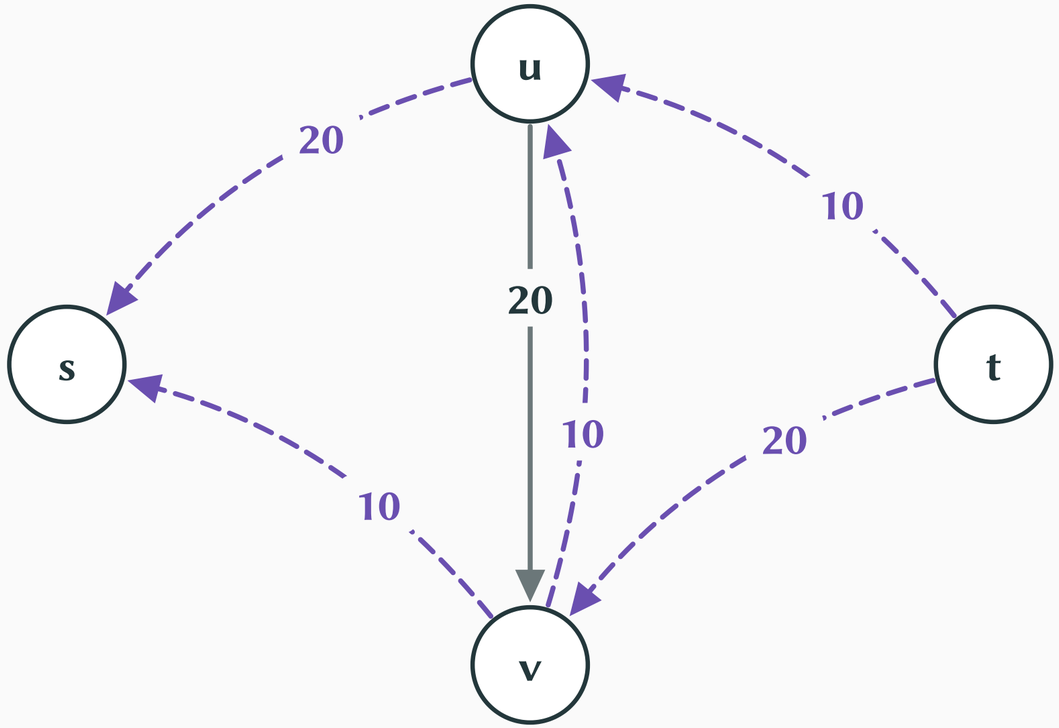
2 **while** G_f has a path from s to t

3 choose such a path P *any path will do*

4 augment f along P by $b = \text{bottleneck}(P, f)$

5 **return** f

Cost. One iteration takes $O(m)$ time. The value grows by at least 1 each time, so there are at most C iterations: $O(mC)$.

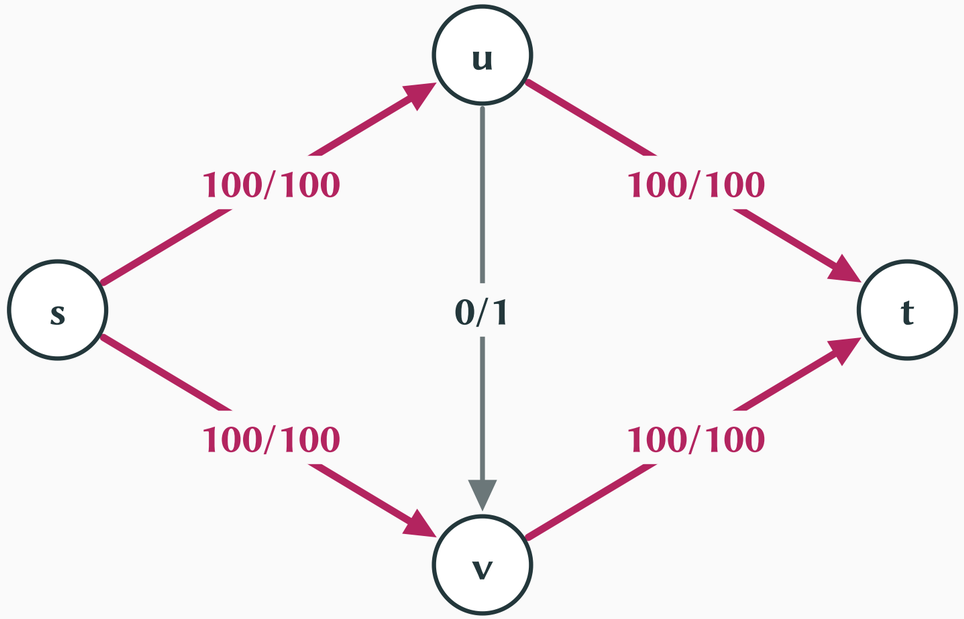


$\text{val}(f') = 30$

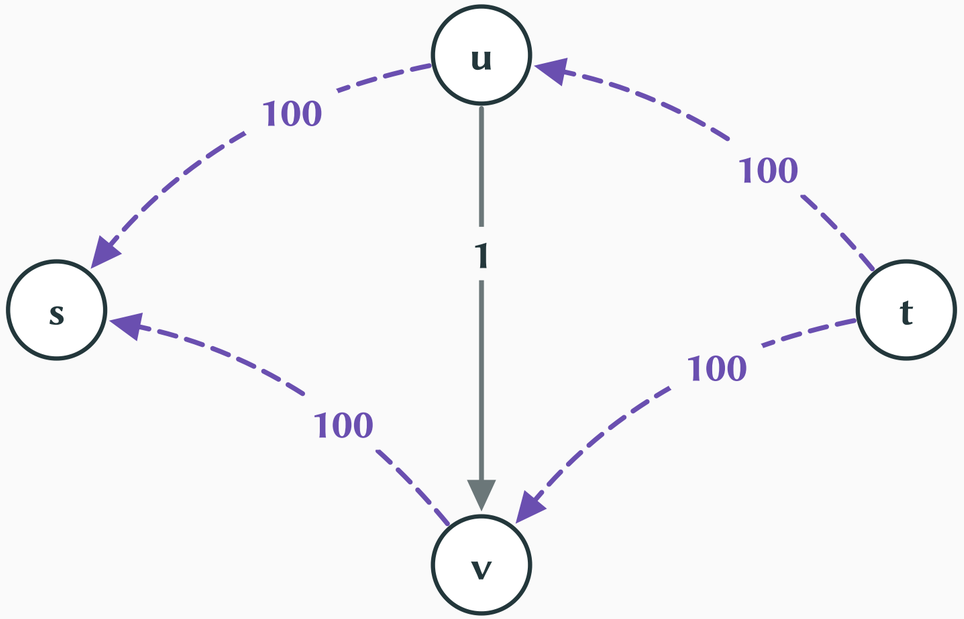
No path from s to t is left: return f.

Ford–Fulkerson with unlucky paths

G with the flow f



G_f and the next path



after 200 iterations: $\mathrm{val}(f) = 200 = C$

no augmenting path is left; each iteration added a single unit

Larger capacities, more iterations

The same five edges, with capacities $M, M, 1, M, M$.

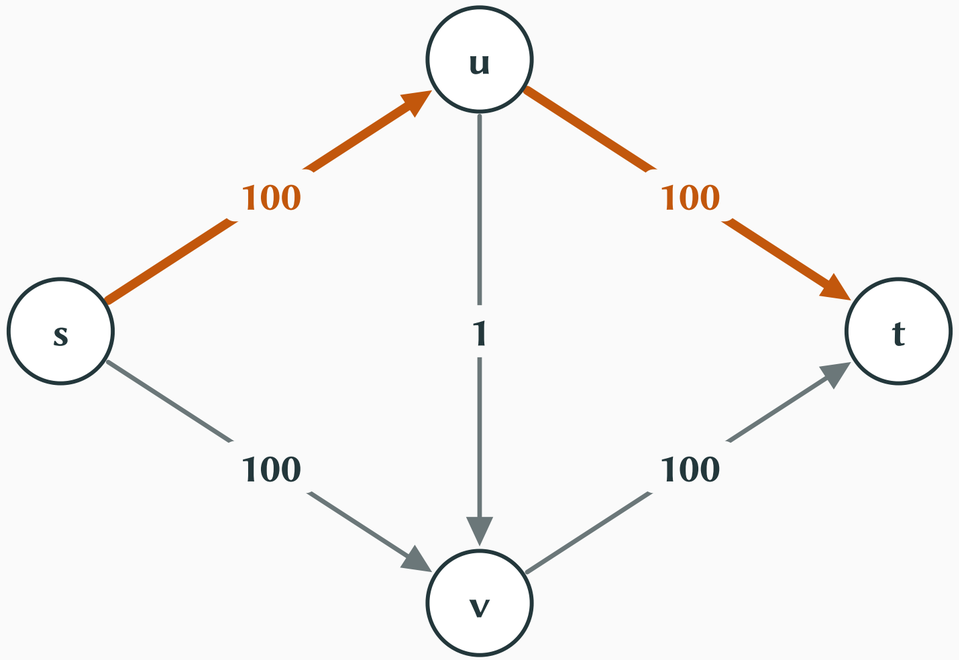
> The unlucky run takes $2M = C$ iterations: the bound $O(mC)$ is **tight**.

? *Is there a rule for choosing the path that avoids this?*

M	bits of M	unlucky paths
10	4	20
100	7	200
1 000	10	2 000
10^6	20	2 000 000

number of iterations of Ford–Fulkerson

Choose the best path?



Largest bottleneck (the fattest path):
s, u, t, with bottleneck 100.

> Two iterations here, but $O(m \log C)$ in general: still growing with C.
>
> Edmonds and Karp, 1972.
>
> Via a modified Dijkstra's algorithm.

Fewest edges (a shortest path):
s, u, t, with 2 edges, found by BFS.

The unlucky path s, u, v, t has 3.

Larger capacities, more iterations

The same five edges, with capacities $M, M, 1, M, M$.

The unlucky run takes $2M = C$ iterations: the bound $O(mC)$ is **tight**.

? *Is there a rule for choosing the path that avoids this?*

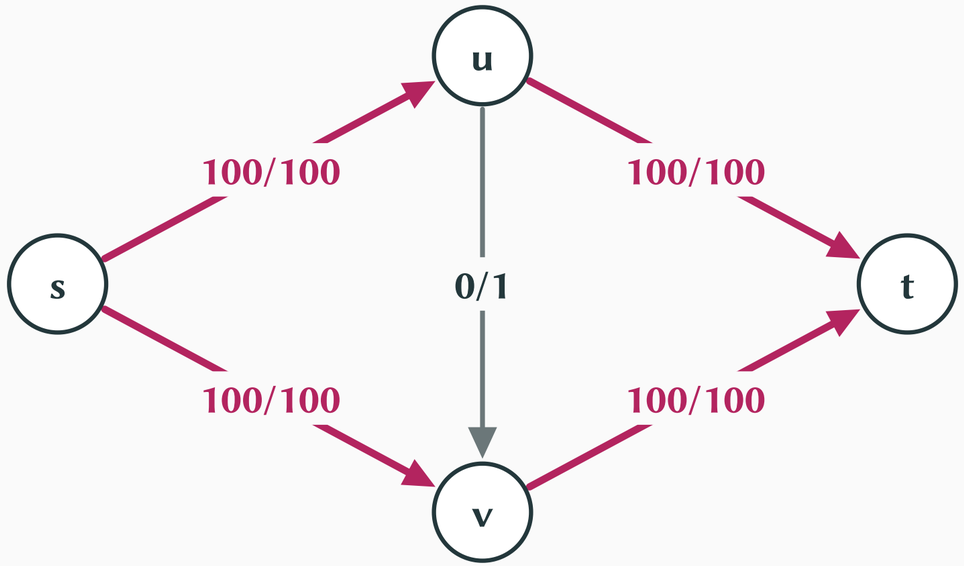
Yes: take a **shortest** path.
Two iterations, whatever M.

M	bits of M	unlucky paths	shortest paths
10	4	20	2
100	7	200	2
1 000	10	2 000	2
10^6	20	2 000 000	2

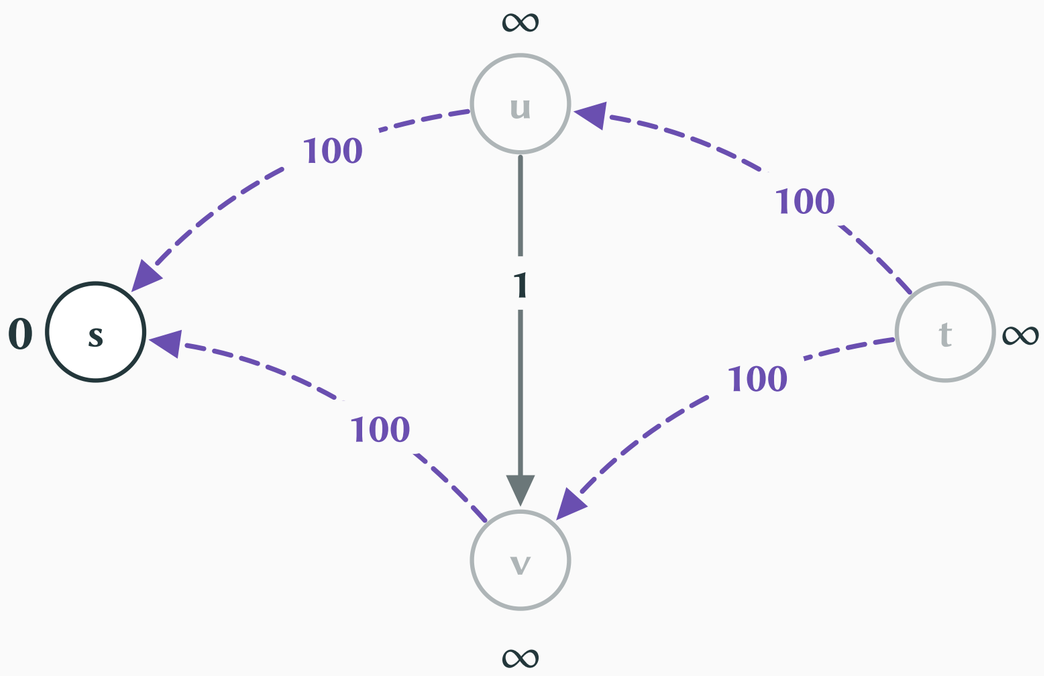
number of iterations of Ford–Fulkerson

Idea: always take a shortest path

G with the flow f



G_f, with the distance $d_f(s, v)$ at each node



after 2 iterations: $\mathrm{val}(f) = 200 = C$

no edge leaves s: all distances are infinite, done

The Edmonds–Karp algorithm

Edmonds–Karp(G, s, t)

1 $f(e) \leftarrow 0$ for every edge e
2 **while** G_f has a path from s to t
3 $\quad P \leftarrow$ a **shortest** such path
4 $\quad$ augment f along P by $b = \text{bottleneck}(P, f)$
5 **return** f

Shortest means the fewest edges, whatever the capacities.

Breadth-first search from s finds it in $O(m)$ time.

It is still Ford–Fulkerson: the result is a maximum flow (Lecture 11).

Theorem. Edmonds–Karp makes at most mn augmentations, so it runs in time $O(m^2 n)$, whatever the capacities.

Pseudo-polynomial, weakly and strongly polynomial

	pseudo-polynomial	weakly polynomial	strongly polynomial
time polynomial in	n, m and C	n, m and $\log C$	n and m only
example	Ford–Fulkerson $O(mC)$	capacity scaling (KT 7.3) $O(m^2 \log C)$	Edmonds–Karp $O(m^2 n)$

Fastest known for exact maximum flow:

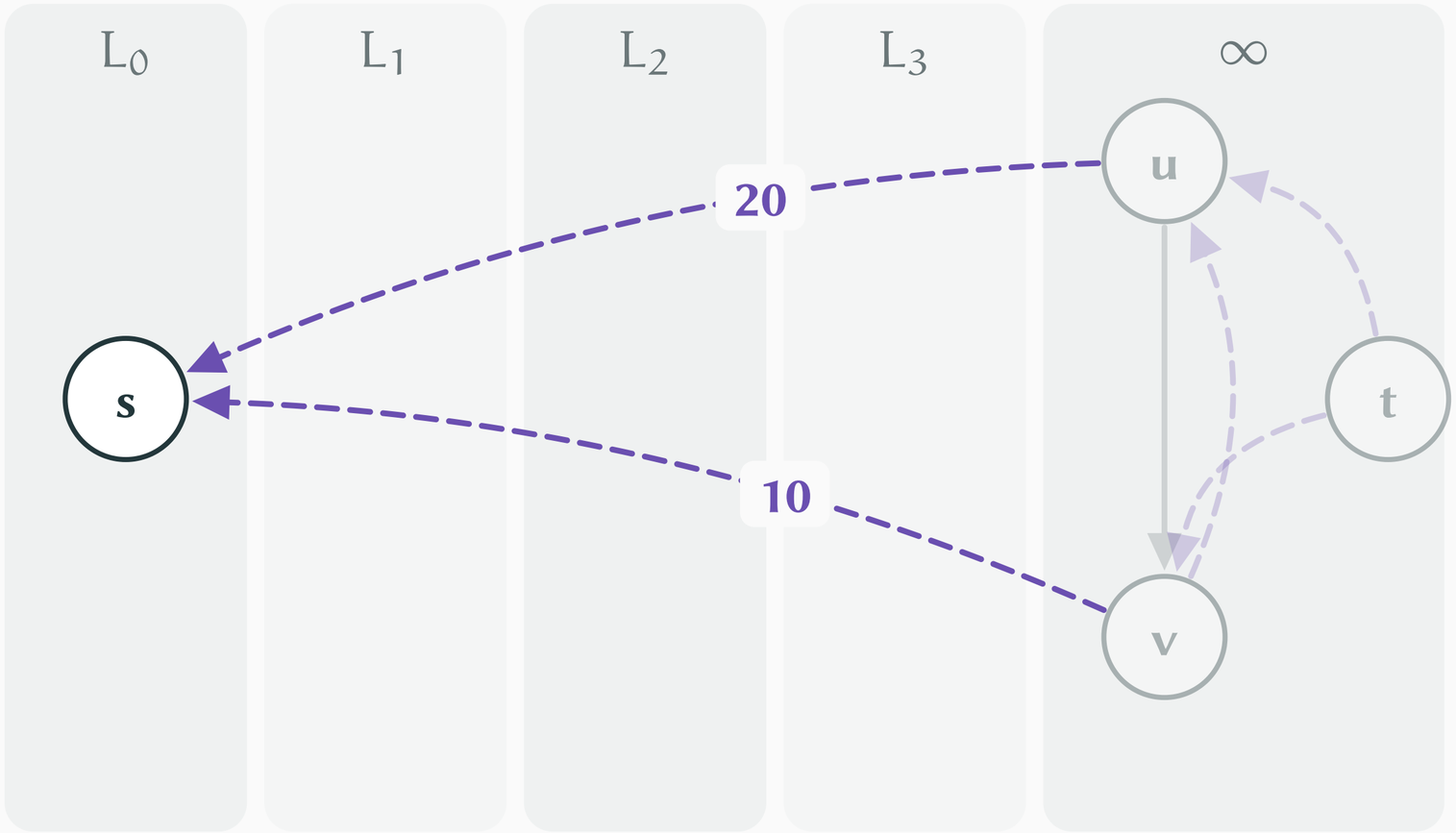
weakly polynomial:
$m^{1+o(1)} \log C$

Chen, Kyng, Liu, Peng, Probst Gutenberg and Sachdeva, 2022 (randomised)
van den Brand, Chen, Kyng, Liu, Peng, Probst Gutenberg, Sachdeva and Sidford, 2023 (deterministic)

strongly polynomial:
$O(nm)$

Orlin, 2013
King, Rao and Tarjan, 1994

Edmonds–Karp on the running network



distance $d_f(s, v)$ from s in G_f

	f_0	f_1	f_2	f_3
u	1	1	1	∞
v	1	1	2	∞
t	2	2	3	∞

? *Is that an accident?*

no path from s to t: $\mathrm{val}(f_3) = 30$

every distance only grew: the last column is ∞

Why is it fast? Three steps

1

Distances never decrease

augmenting along a shortest path creates no shortcuts

$$d_f(s, v) \nearrow$$

→

2

Each arc is critical at most $n/2$ times

a full arc returns only after its tail has moved away by 2

$$\leqslant n/2$$

→

3

At most mn augmentations

every augmentation fills some arc; $O(m)$ time each

$$O(m^2 n)$$

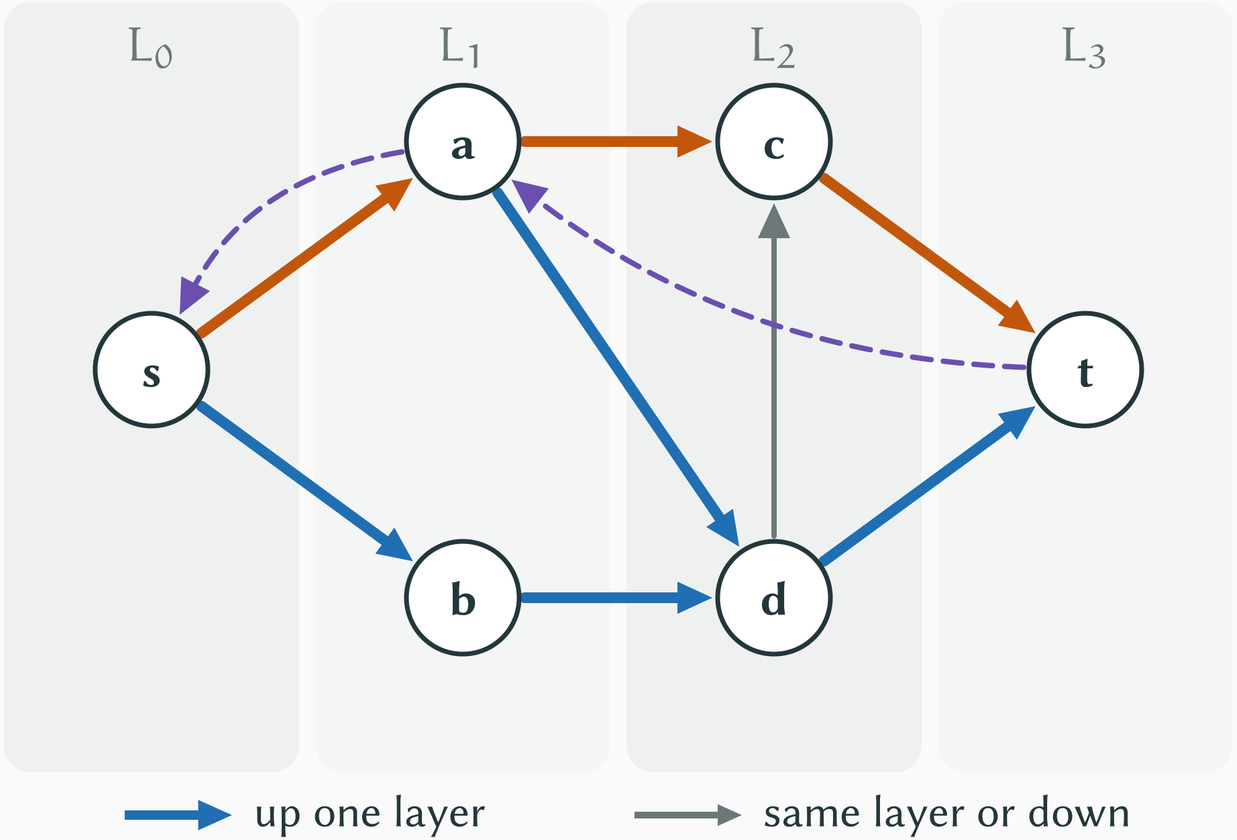
Distances and layers

Distance $d_f(s, v)$: the fewest arcs on a path from s to v in G_f (∞ if there is none).

Layer $L_i = \{v : d_f(s, v) = i\}$: the nodes BFS reaches in round i.

> **Fact 1.** Every arc of G_f goes up **at most one** layer.
> Else there is a shorter path to its head.

> **Fact 2.** A shortest s–t path goes up **exactly one** layer per arc.



Lemma: distances never decrease

Lemma. Let f' arise from f by augmenting along a **shortest** s–t path of G_f. Then

$$d_{f'}(s, v) \geqslant d_f(s, v) \quad \text{for every node } v.$$

Idea. The arcs that $G_{f'}$ has and G_f has not point **backwards**: no shortcuts.

So during a run of Edmonds–Karp
every distance only grows,
and the augmenting paths never get shorter.

$d_f(s, v)$ in the run on the running network

	f_0	f_1	f_2	f_3
u	1	1	1	∞
v	1	1	2	∞
t	2	2	3	∞

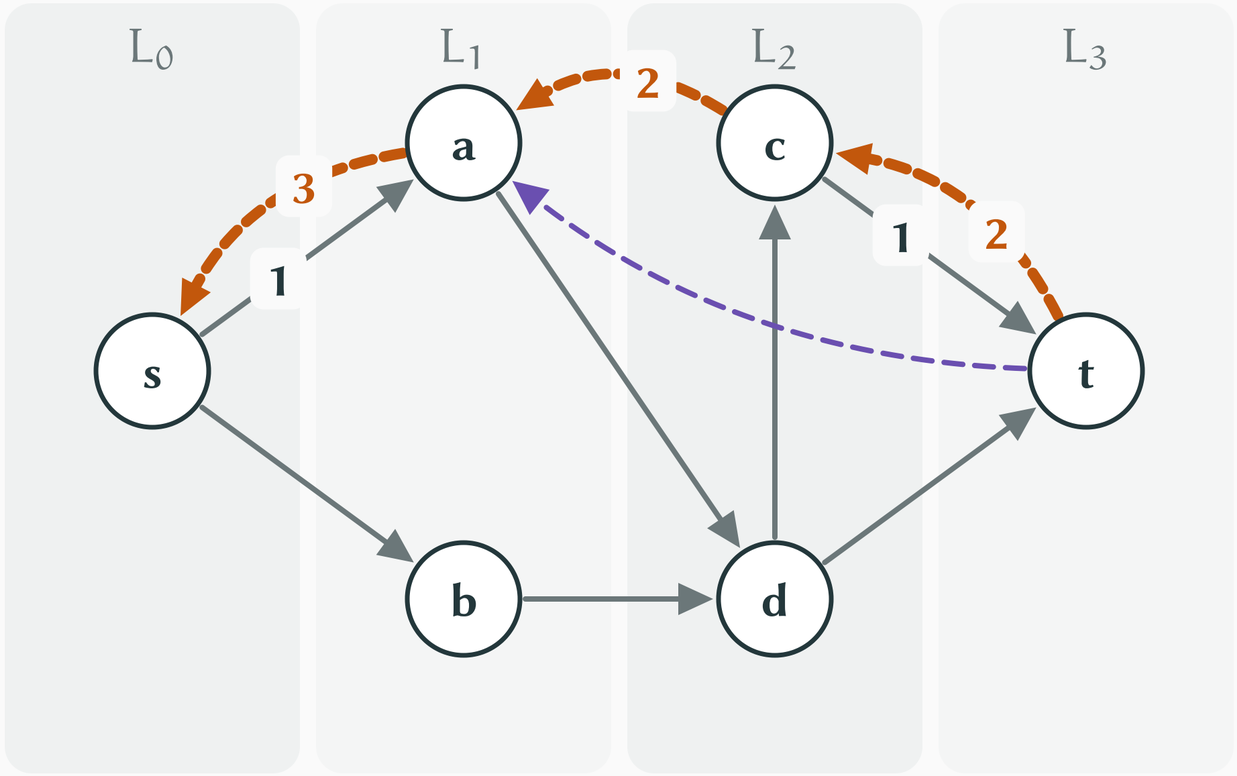
every row grows from left to right

Proof, step 1: what augmenting changes

Augment along the shortest path P
by its bottleneck b.

- Every arc of P goes up **exactly one** layer.
- The **bottleneck arc** is full:
  it **disappears**.
- Each arc (x, y) of P gives a
  reverse arc (y, x), new or bigger:
  it goes **down** one layer.

Nothing else changes.

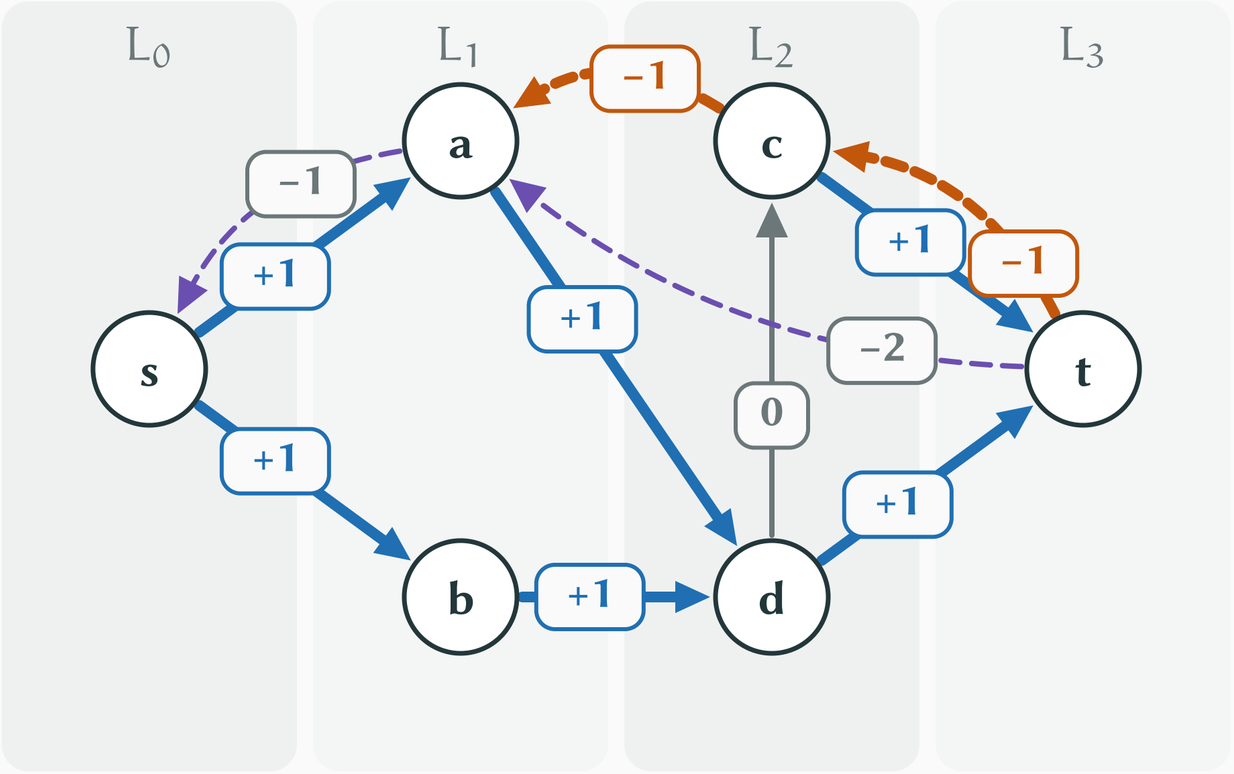


Proof, step 2: no arc goes up by two layers

$G_{f'}$, drawn in the layers of G_f.
Two kinds of arcs:

- **Old** arcs (already in G_f):
  up at most one layer, by Fact 1.
- **New** arcs: reverses of arcs of P,
  so one layer **down**.

> Every arc (x, y) of $G_{f'}$ has
> $d_f(s, y) \leqslant d_f(s, x) + 1$.



Proof, step 3: a path climbs one layer per arc

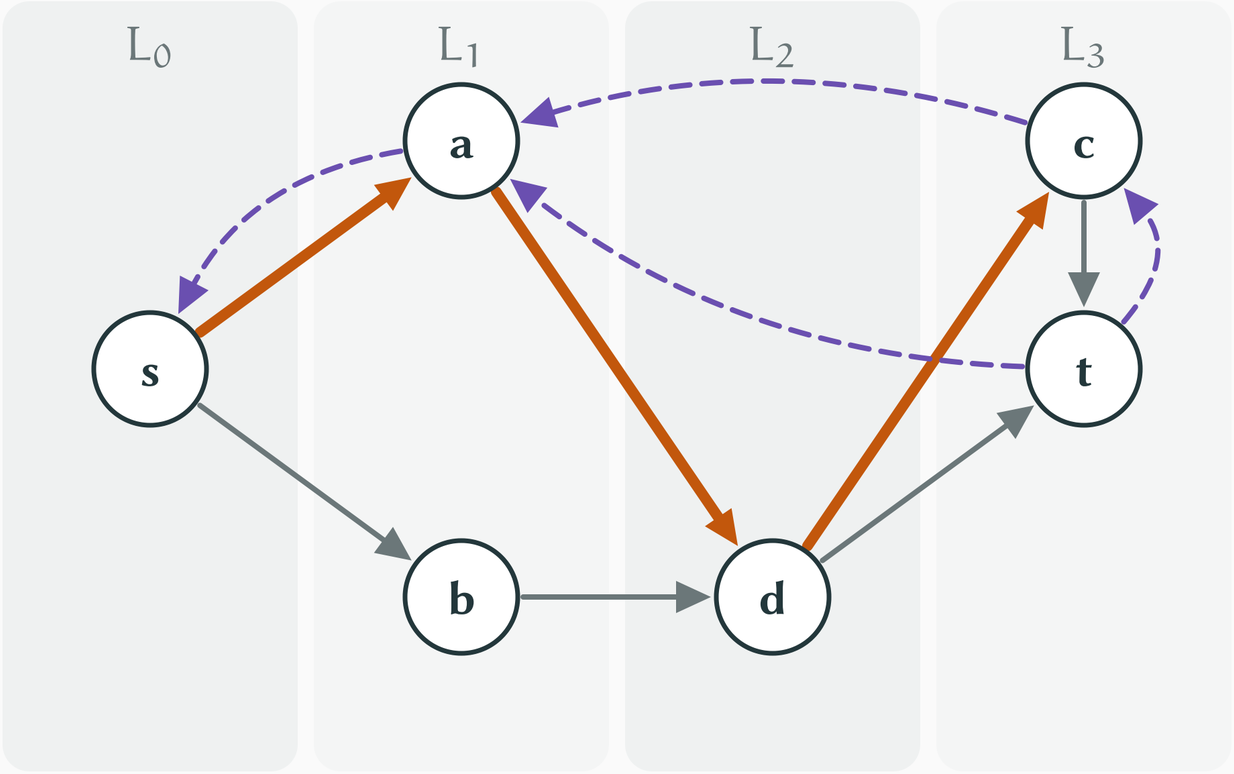
A shortest path $s = x_0, x_1, \dots, x_k = v$ in $G_{f'}$: $k = d_{f'}(s, v)$.

arcs used j	0	1	2	3
node x_j	**s**	**a**	**d**	**c**
layer of x_j in G_f	0	1	2	2

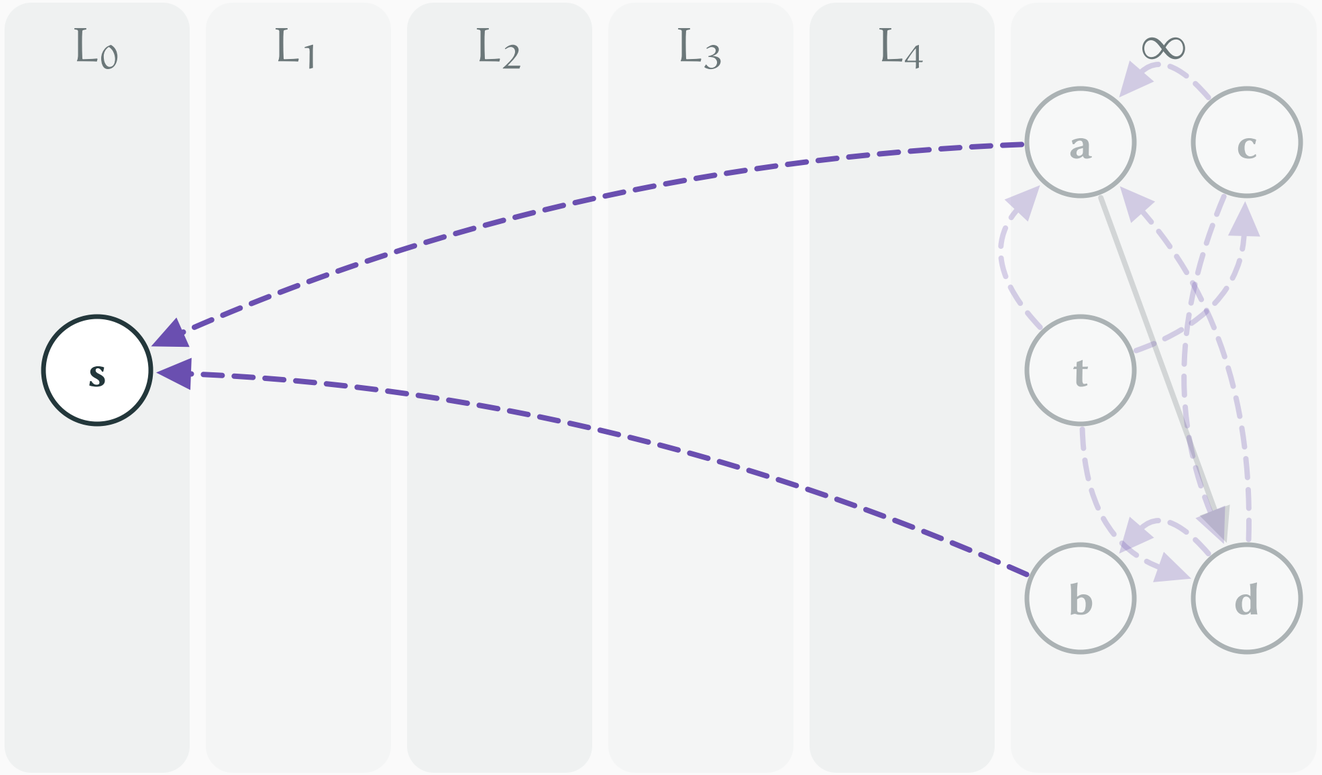
Each arc goes up at most one layer of G_f: after j arcs we are in a layer $\leqslant j$.

So $d_f(s, v) \leqslant k = d_{f'}(s, v)$. ■

The new layers: a node can only move to the right.



The lemma in action: a whole run



distance $d_f(s, v)$

	f_0	f_1	f_2	f_3	f_4	f_5
a	1	1	1	3	3	∞
b	1	1	1	1	1	∞
c	2	2	3	3	3	∞
d	2	2	2	2	2	∞
t	2	3	3	3	4	∞

no entry ever decreases

no path from s to t: $\text{val}(f_5) = 7$

every other node is at distance ∞

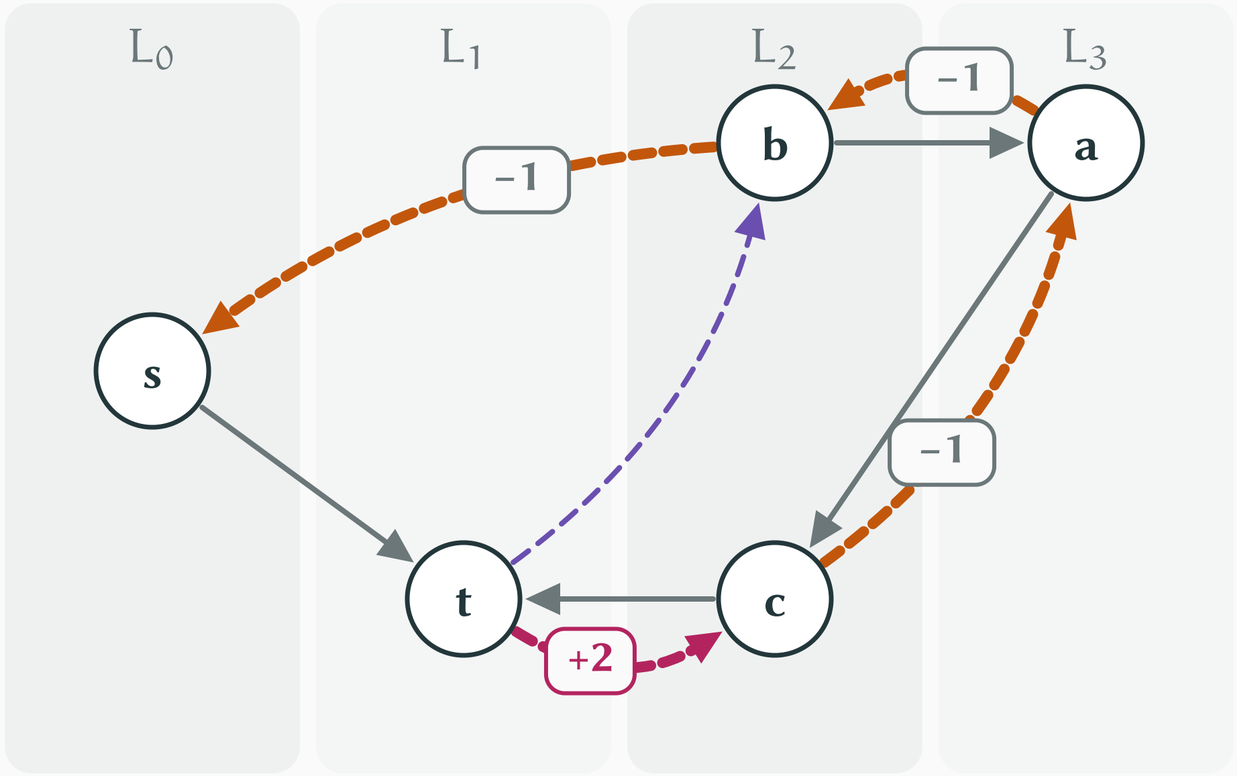
Where did we use that P is shortest?

$P = s, b, a, c, t$ is an augmenting path, but not a shortest one: its last arc goes **down two** layers.

Augmenting adds the reverse arc (t, c): it goes **up two** layers, a shortcut.

The distance of c drops from 3 to 2.

> On a shortest path every arc goes up **exactly one**, so every reverse arc goes down **exactly one**.



Critical arcs

Definition. An arc of the augmenting path P is **critical** if its residual capacity equals the bottleneck b.

- After augmenting it has capacity 0: it **leaves** G_f.
- Every augmentation has at least one critical arc.

augmentations $\leqslant$ critical (arc, time) pairs

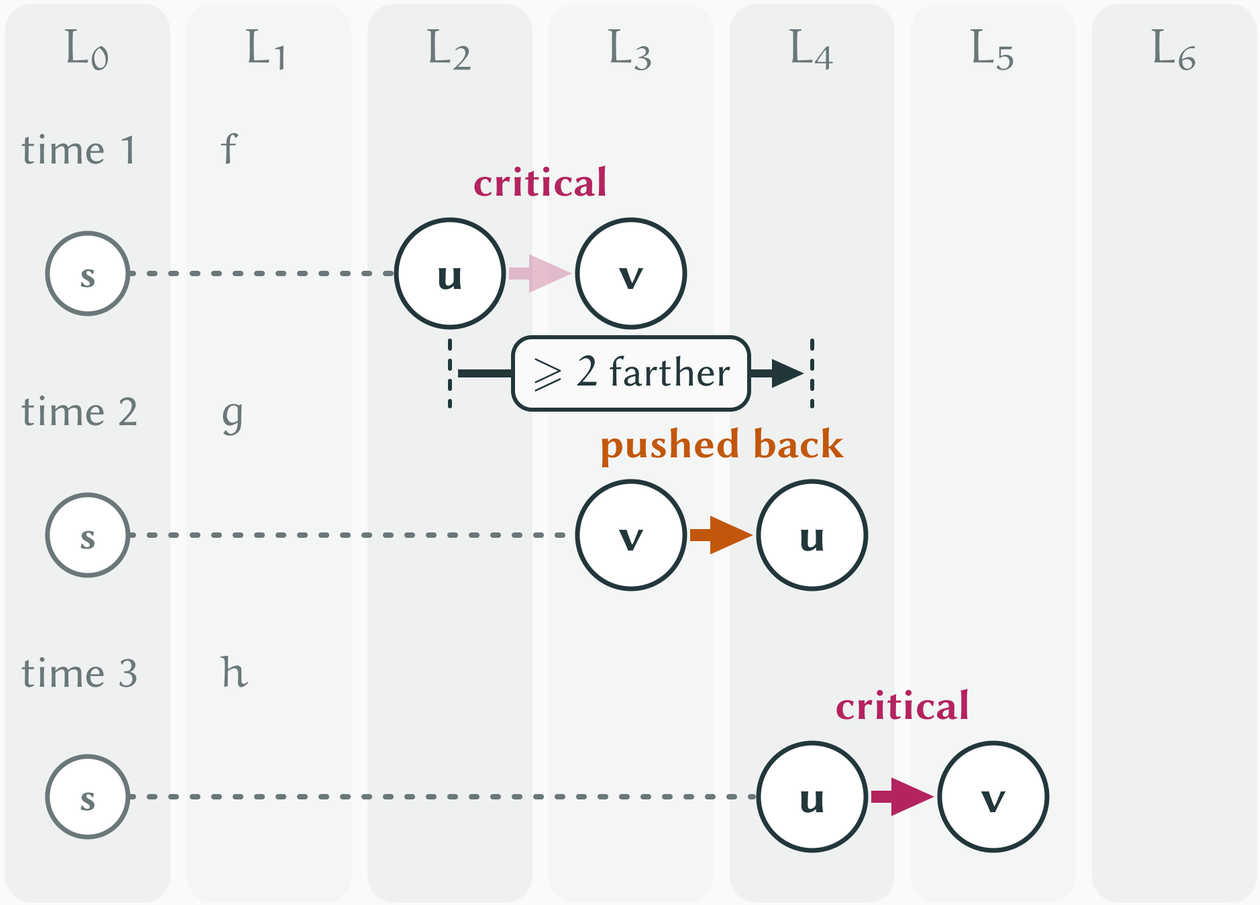
iteration	path P	b	critical arcs
1	s, u, t	10	(u, t)
2	s, v, t	10	(s, v)
3	s, u, v, t	10	$(s, u),\ (v, t)$

the run on the running network

Can a critical arc come back?

1. Time 1, flow f: (u, v) is **critical**, so $d_f(s, v) = d_f(s, u) + 1$. It leaves G_f.
2. To return, flow must be pushed along (v, u). At time 2 (flow g) that arc is on a shortest path: $d_g(s, u) = d_g(s, v) + 1$.
3. By the lemma $d_g(s, v) \geqslant d_f(s, v)$, so $d_g(s, u) \geqslant d_f(s, u) + 2$.
4. Time 3 (flow h): critical again, and $d_h(s, u) \geqslant d_g(s, u) \geqslant d_f(s, u) + 2$.

> Between two critical times of (u, v) the distance of u from s grows by at least 2.

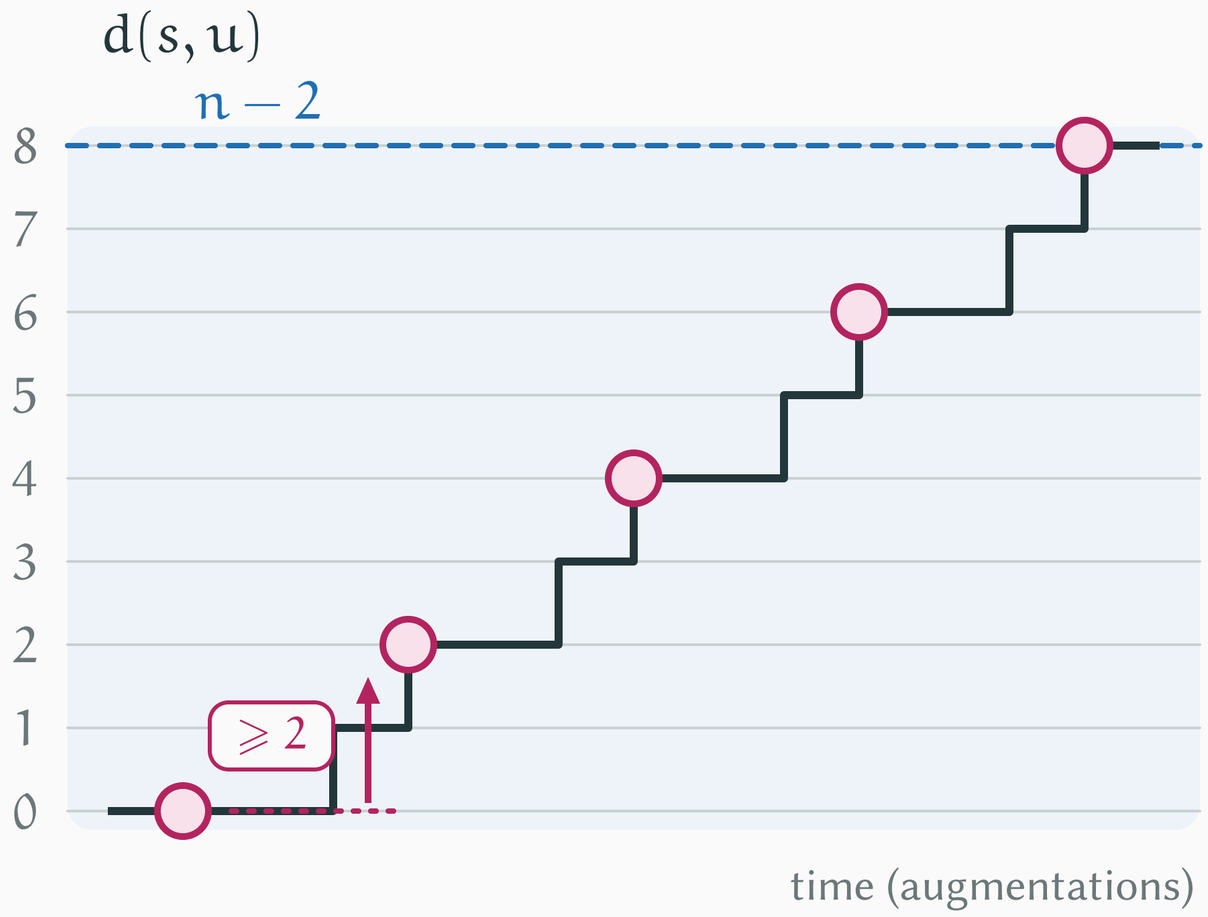


Each arc is critical at most $n/2$ times

(u, v) lies on a shortest s–t path:
at most $n - 1$ arcs,
and u is not its last node.
So $0 \leqslant d(s, u) \leqslant n - 2$.

At each critical time,
$d(s, u)$ is at least 2 more
than at the previous one.

> **Lemma.** An arc is critical at most $n/2$ times.



$\leqslant n/2 = 5$ critical times

Counting the augmentations

arcs: a forward and a backward arc per edge	$\leqslant 2m$
critical times of each arc (previous slide)	$\leqslant n/2$
critical (arc, time) pairs	$\leqslant 2m \cdot \frac{n}{2} = mn$
every augmentation has at least one: augmentations	$\leqslant mn$
time per augmentation: BFS, augment, new G_f	$O(m)$

Theorem. Edmonds–Karp runs in time $mn \cdot O(m) = O(m^2 n)$.

Polynomial, whatever the capacities

	Ford–Fulkerson	Edmonds–Karp
choice of P	any path	a shortest path
augmentations	$\leqslant C$	$\leqslant mn$
running time	$O(mC)$	$O(m^2n)$
depends on the capacities	yes	no
real capacities	may never stop	always stops

Summary

1. **The choice of path matters:** unlucky paths need C iterations even on five edges, so $O(mC)$ is tight.
2. **Edmonds–Karp** augments along a **shortest** path (fewest edges), found by BFS.
3. **Distances never decrease:** augmenting along a shortest path adds only arcs that point backwards.
4. **Critical arcs:** each arc is a bottleneck at most $n/2$ times, so at most mn augmentations, $O(m)$ each.

> Maximum flow in time $O(m^2 n)$, whatever the capacities.

Reference: CLRS, Chapter 24.2