The maximum flow algorithms of Goldberg and Rao

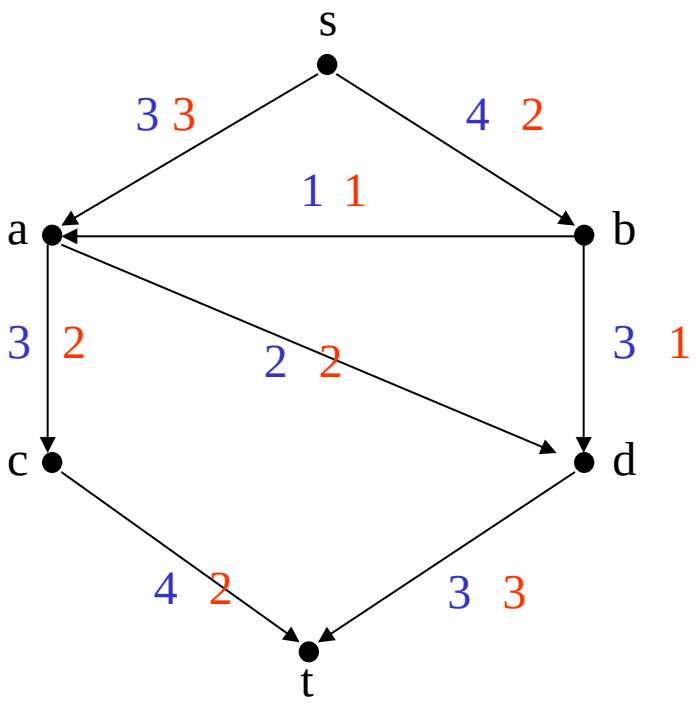
Lets get back to Dinic's algorithm

Dinic

Def: A flow f is blocking if every path from s to t contains a saturated edge.

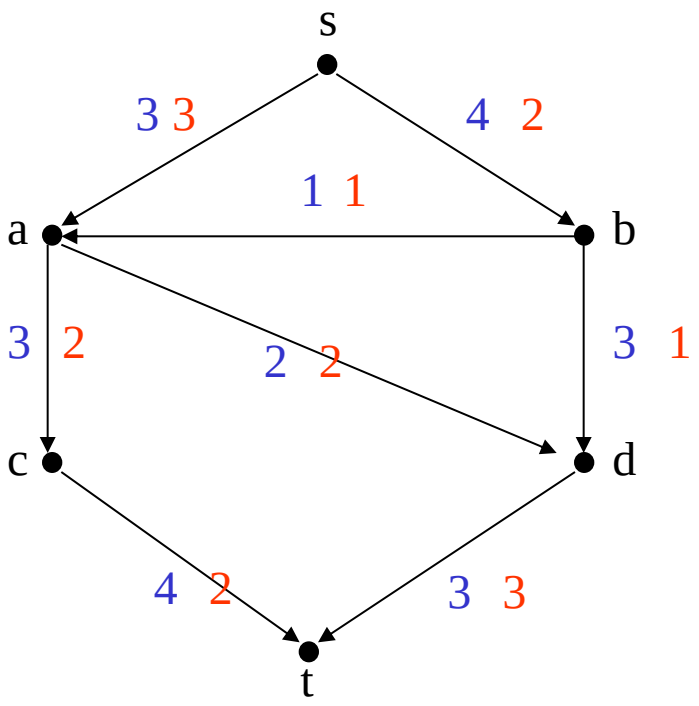
Note that a blocking flow need not be a maximum flow!

Example



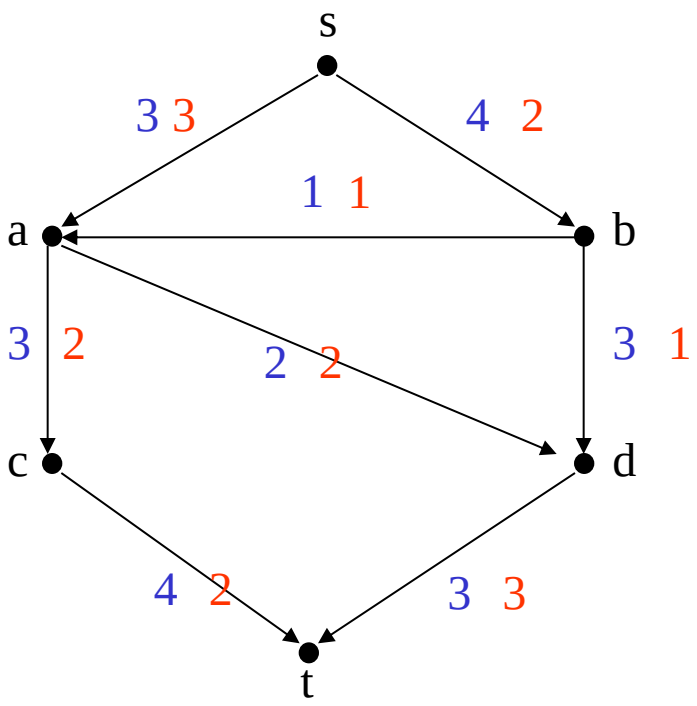
Is this a blocking flow ?

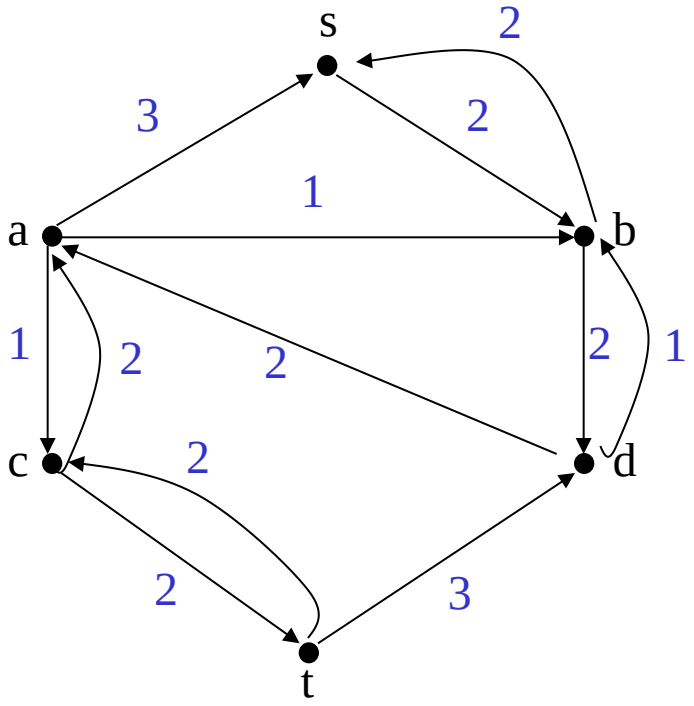
Example



Is this a maximum flow ?

Example (cont)





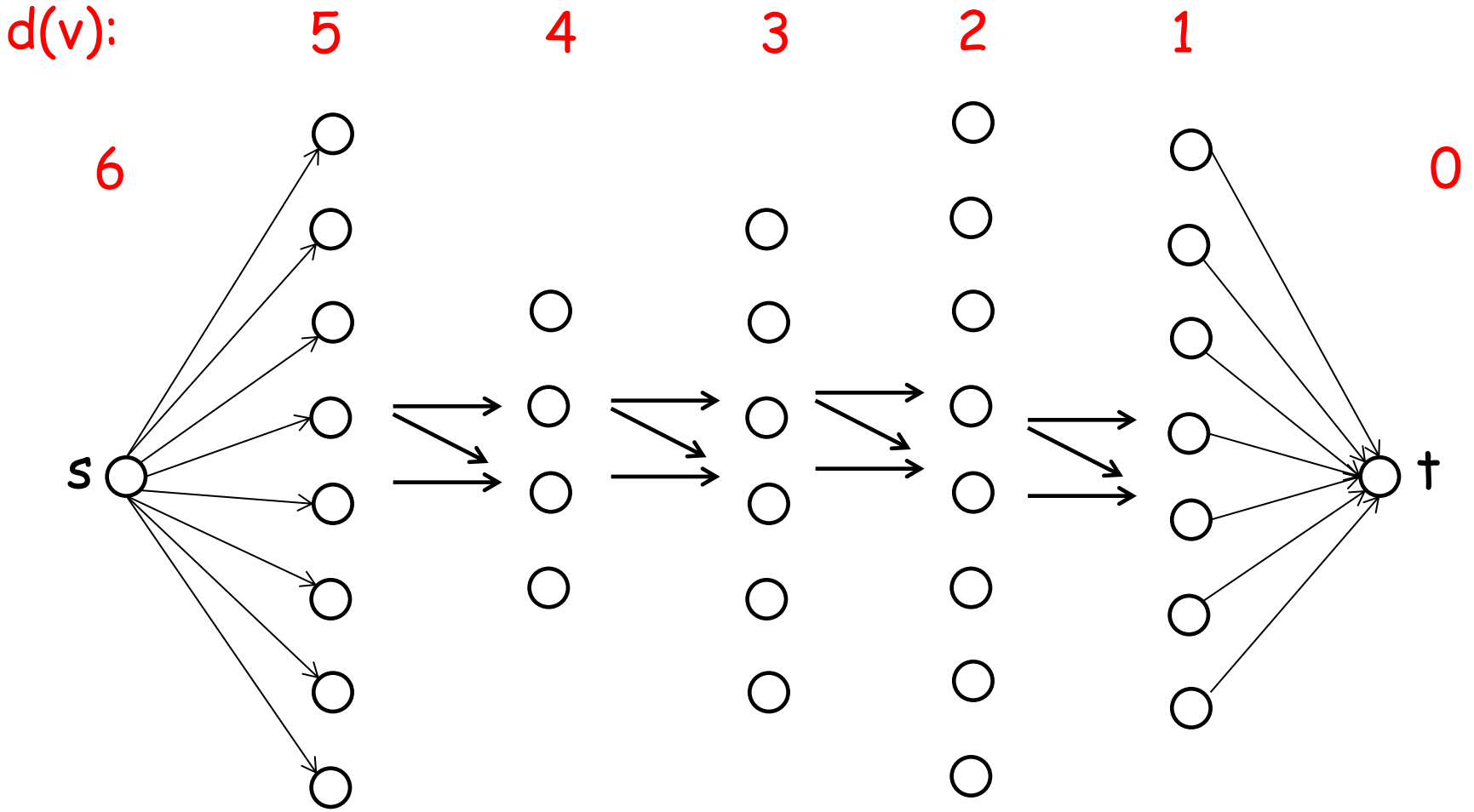
Dinic's algorithm

Let d(v) be the length of the shortest path from v to t in R.

Let L be the subgraph of R containing only edges (v,w) such that d(v) = d(w) + 1

Dinic: find a blocking flow f' in L let f := f + f' and repeat.

The subgraph L

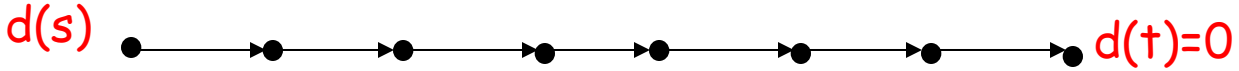


Dinic's algorithm (analysis)

Theorem: d(s) increases by at least 1 after each blocking flow computation ➔ f is a maximum flow after at most n-1 blocking flow computations.

Proof. Each edge in R' is either an edge in R or the reverse of an edge in R.

So for each edge $(v,w) \in R'$ we have $d(v) \leq d(w) + 1$



So d(v) decreases by at most one as we go from t to s along the shortest path in R', but it cannot do that on every step (since if so f' was not blocking) □

Dinic's algorithm (analysis)

How fast can we find a blocking flow ?

Using DFS in a straightforward manner would give us $O(nm)$.

➔ $O(n^2m)$ for the whole max flow algorithm.

One can do it using dynamic trees in $O(m\log(n))$

➔ $O(mn\log(n))$ for the max flow algorithm

When all capacities are 1

When all capacities are 1

How fast can we find a blocking flow ?

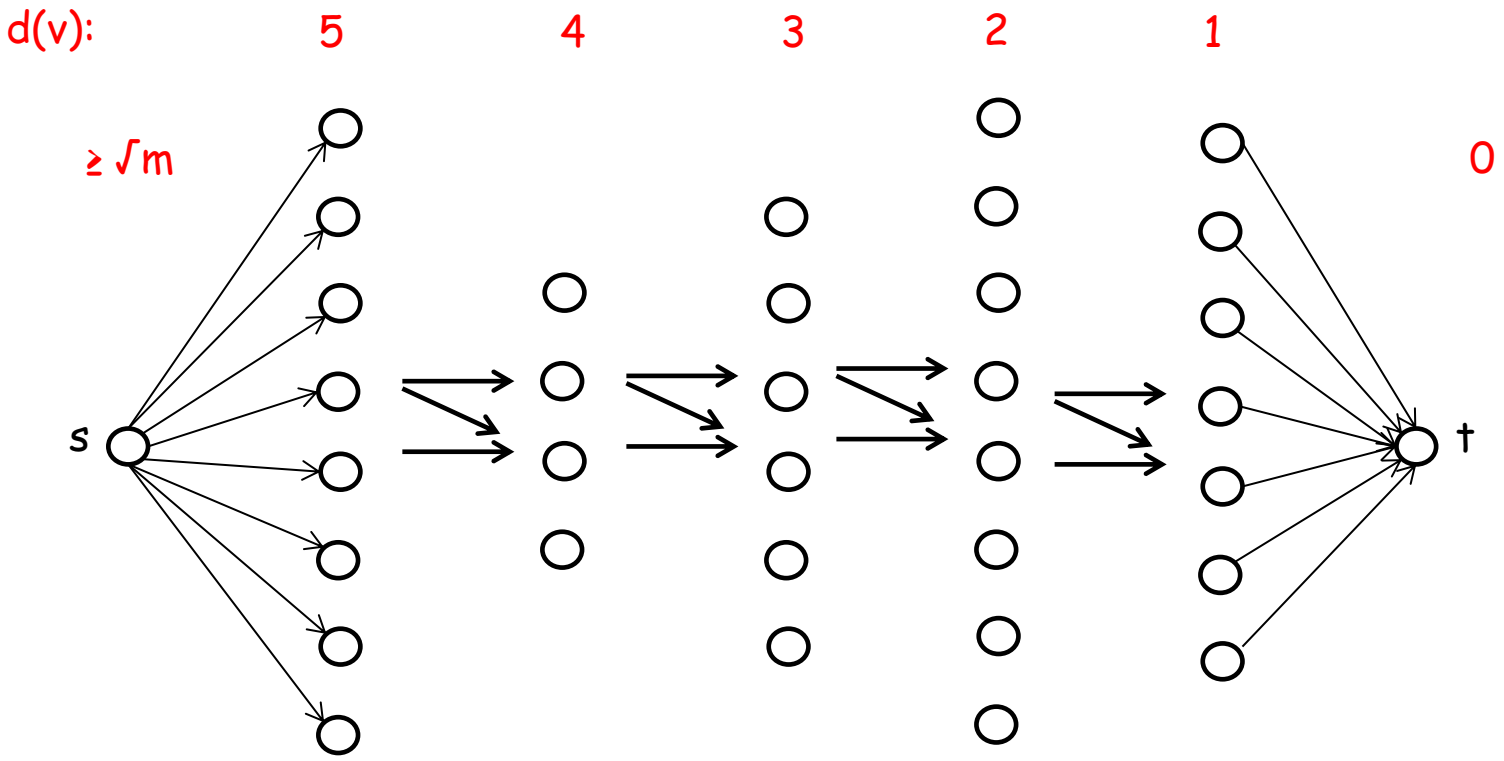
When you augment along a path all edges are saturated.

➔ We find a blocking flow in $O(m)$ time

How many blocking flow computations do we have ?

When all capacities are 1

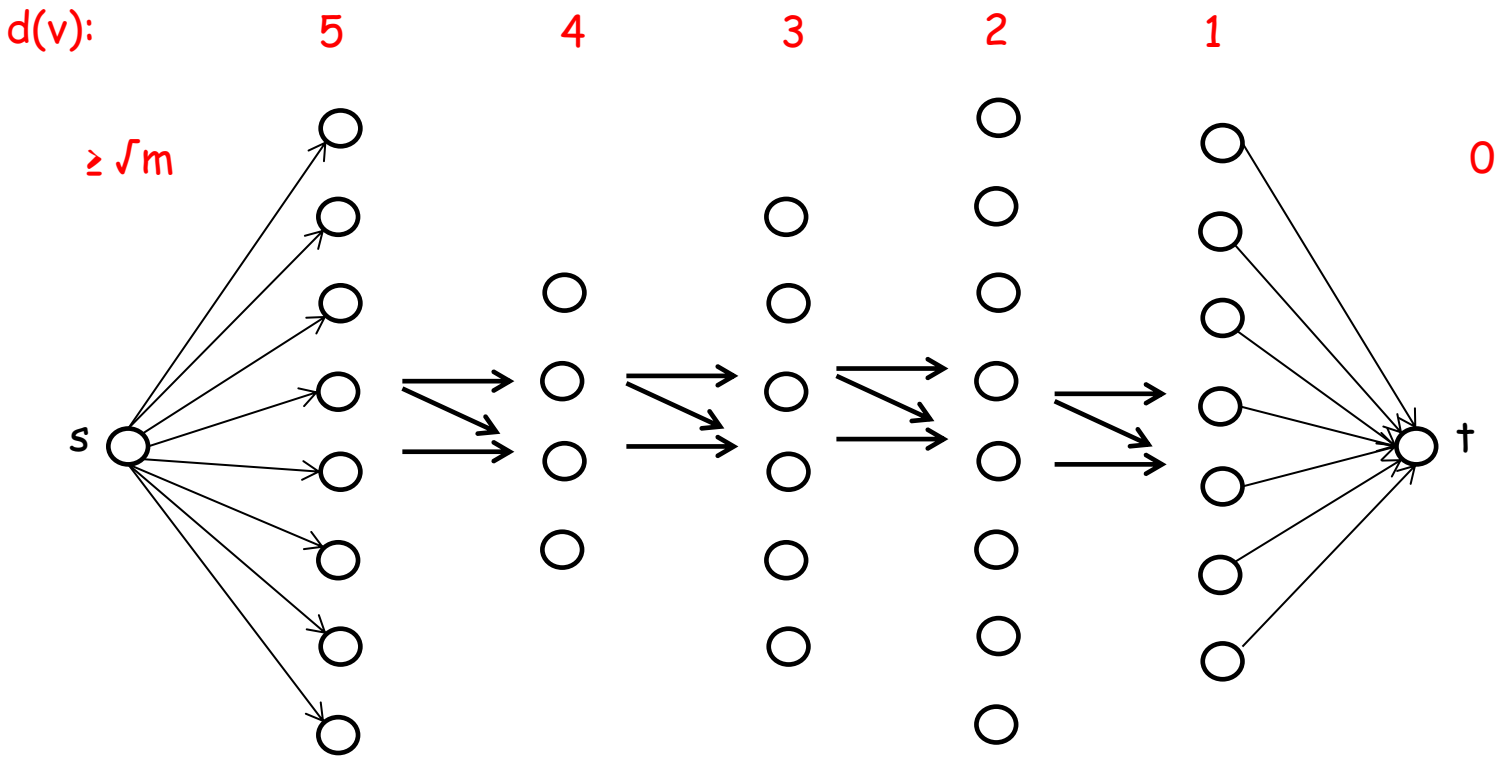
After $\sqrt{m}$ blocking flows:



We are $\sqrt{m}$ – close to optimal

When all capacities are 1

After $\sqrt{m}$ blocking flows:



So we will finish after at most $\sqrt{m}$ additional blocking flow computations ➔ $O(m\sqrt{m})$ time

When all capacities are small integers in {1,2,3,…,U} ?

Outline (Goldberg-Rao 98)

Maintain an upper bound F on the gap between our current flow f and the maximum flow f*

In a phase we will reduce F to F/2

We want to do this in $O(m\sqrt{m})$ time

This will lead to a bound of $O(m\sqrt{m} \log(nU))$

Outline (cont.)

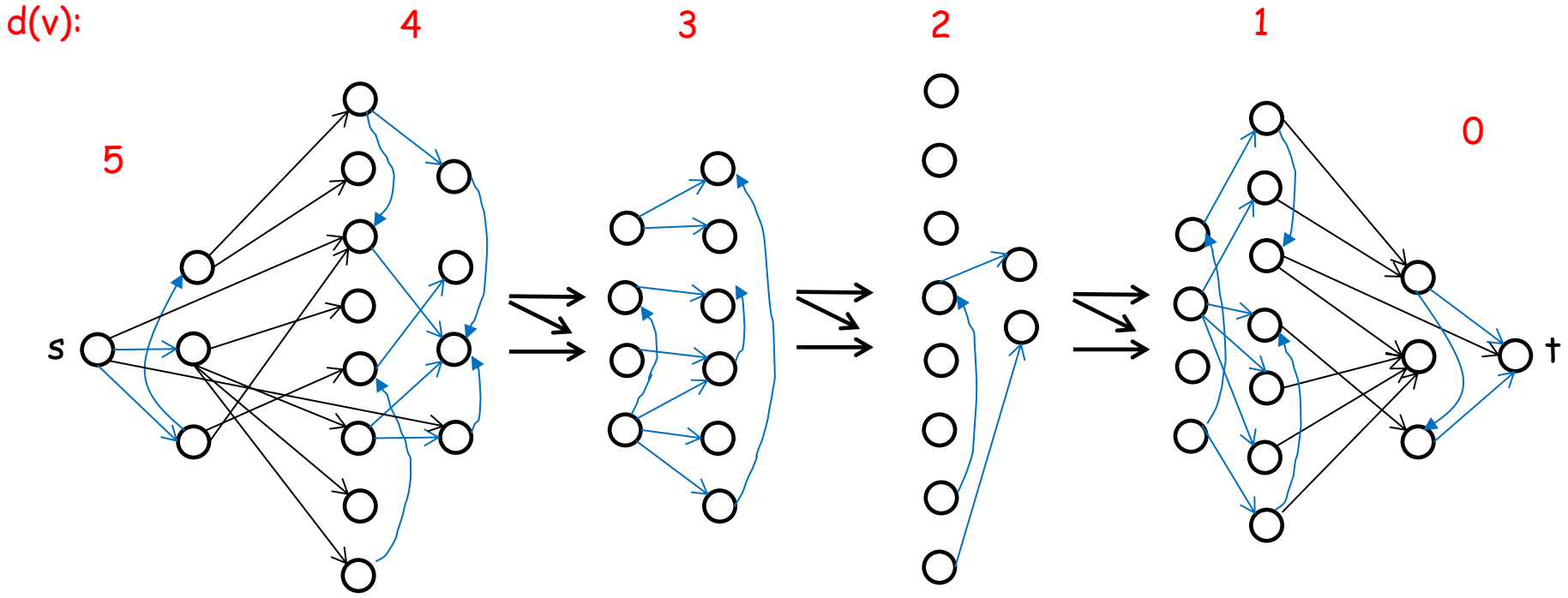
Set $\Delta = F/\sqrt{m}$

Lets push Δ flow in $O(m)$ time..

If we can do this, a phase would take $(\sqrt{m}/2)O(m)$ time

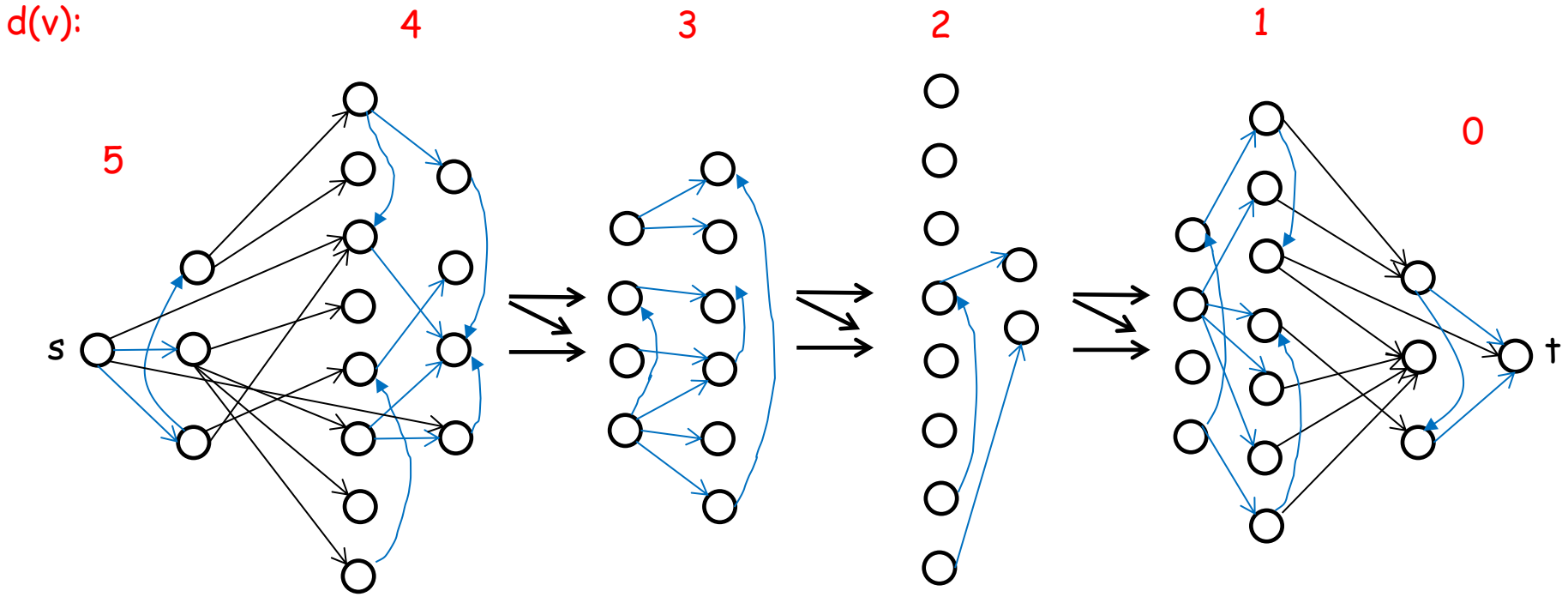
But we will not always be able to do this…

What will happen when we cannot push Δ flow ?



The "distance" d(s) from s to t will increase when we consider only edges of capacity ≤ Δ

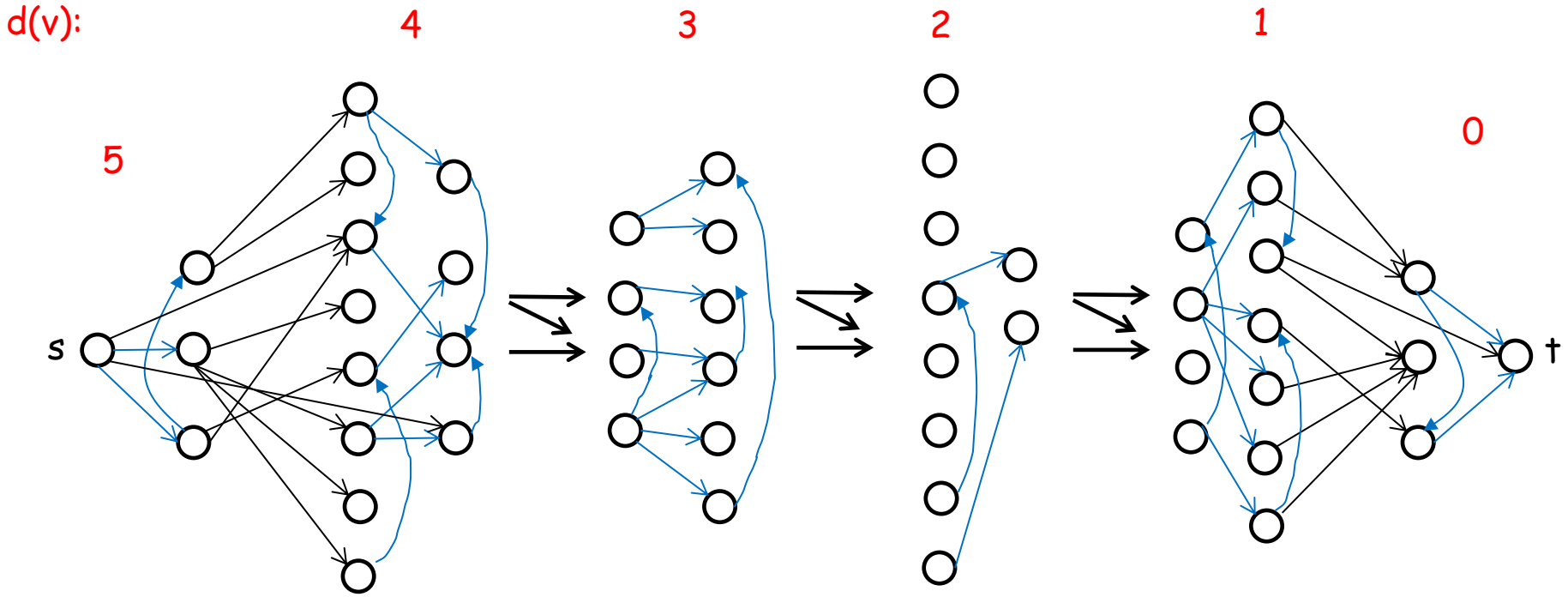
Outline (cont.)



After $2\sqrt{m}$ such iterations the distance will be $\geq 2\sqrt{m}$ and we will have a cut of capacity $m\Delta/(2\sqrt{m}) \leq F/2$

Now the details..

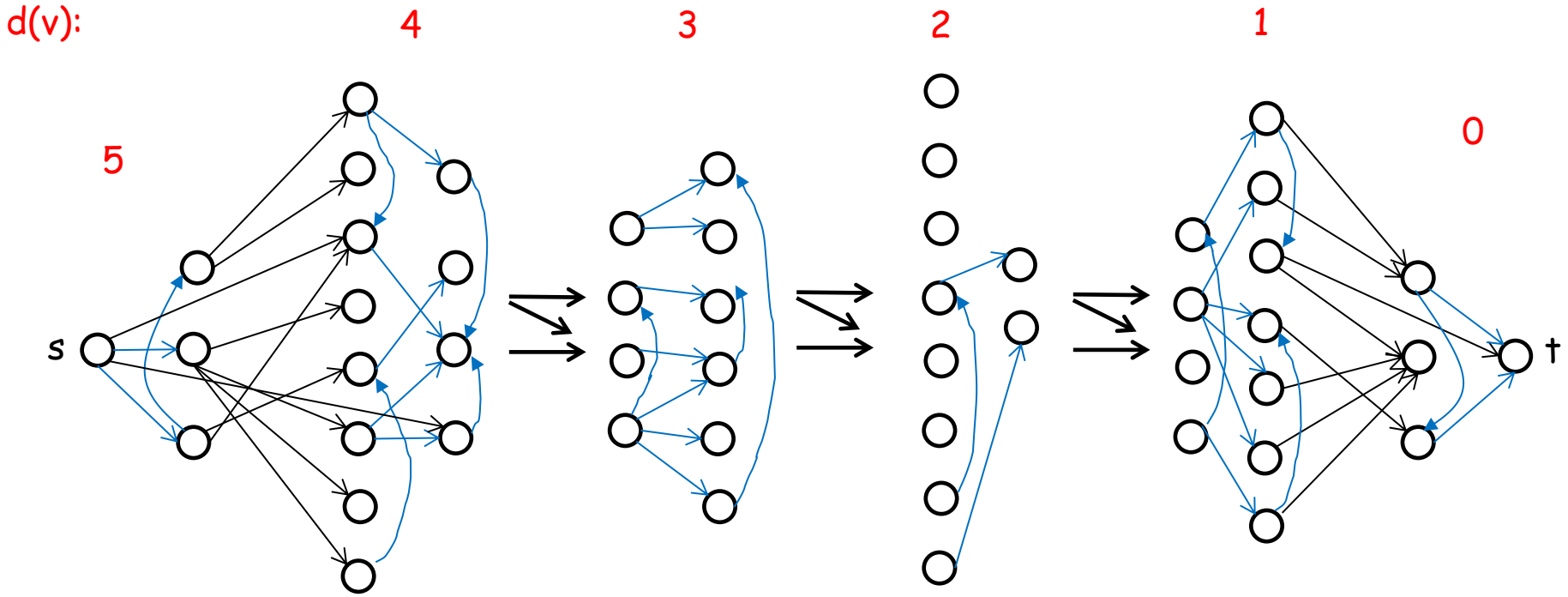
Binary length functions



ℓ: Arcs e of large residual capacity ($\geq 3\Delta$) have $\ell(e)=0$
other arcs e have $\ell(e)=1$

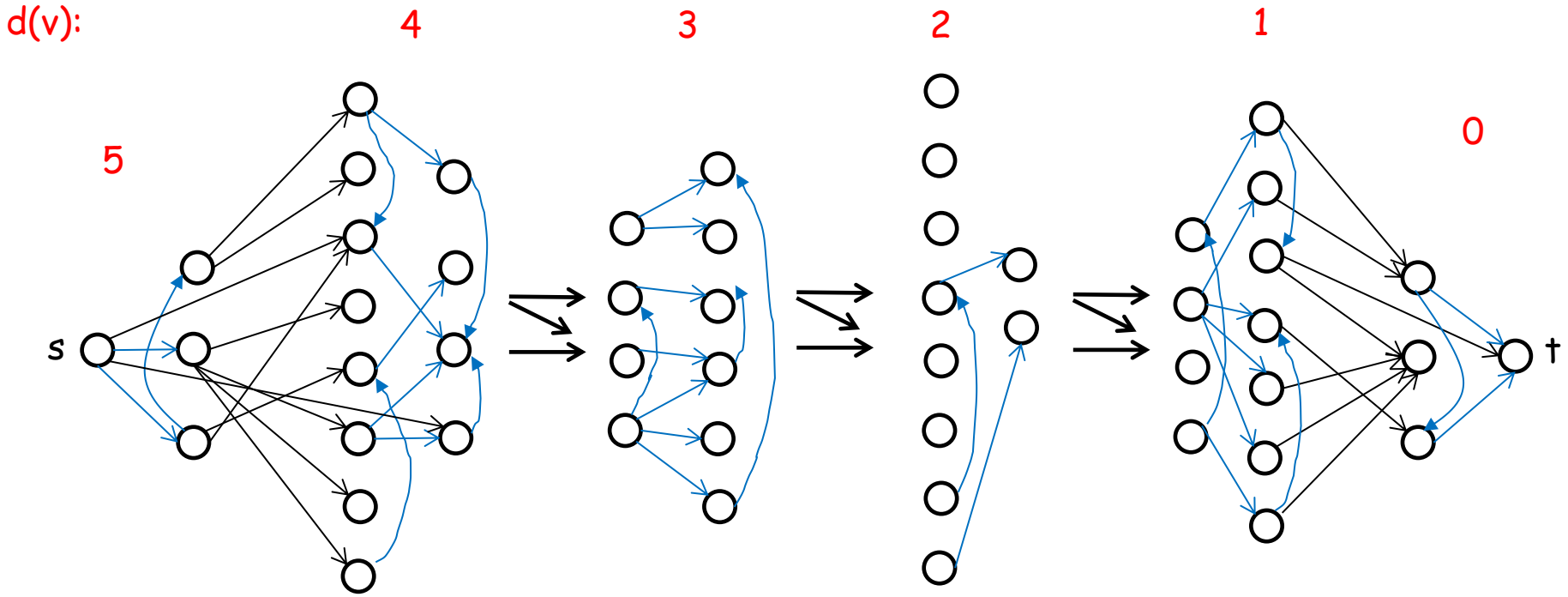
(v,w) admissible iff $d_\ell(v) = d_\ell(w) + \ell(v,w)$

Binary length functions



In the admissible subgraph find a flow of value Δ or a blocking flow

Binary length functions



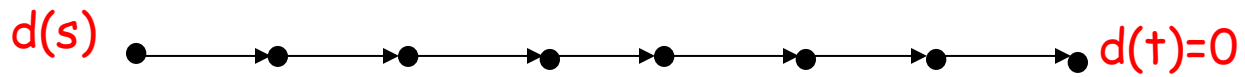
After adding a (blocking) flow lengths changes, and the residual graph changes…

Consider the situation after adding a (blocking) flow

Lemma 1: Let ℓ and d_ℓ be the length and distances before the update and let ℓ' and $d_{\ell'}$ be the length and distances after the update then

for (v,w) residual: $d_\ell(v) \leq d_\ell(w) + \ell'(v,w)$

Corollary 1:

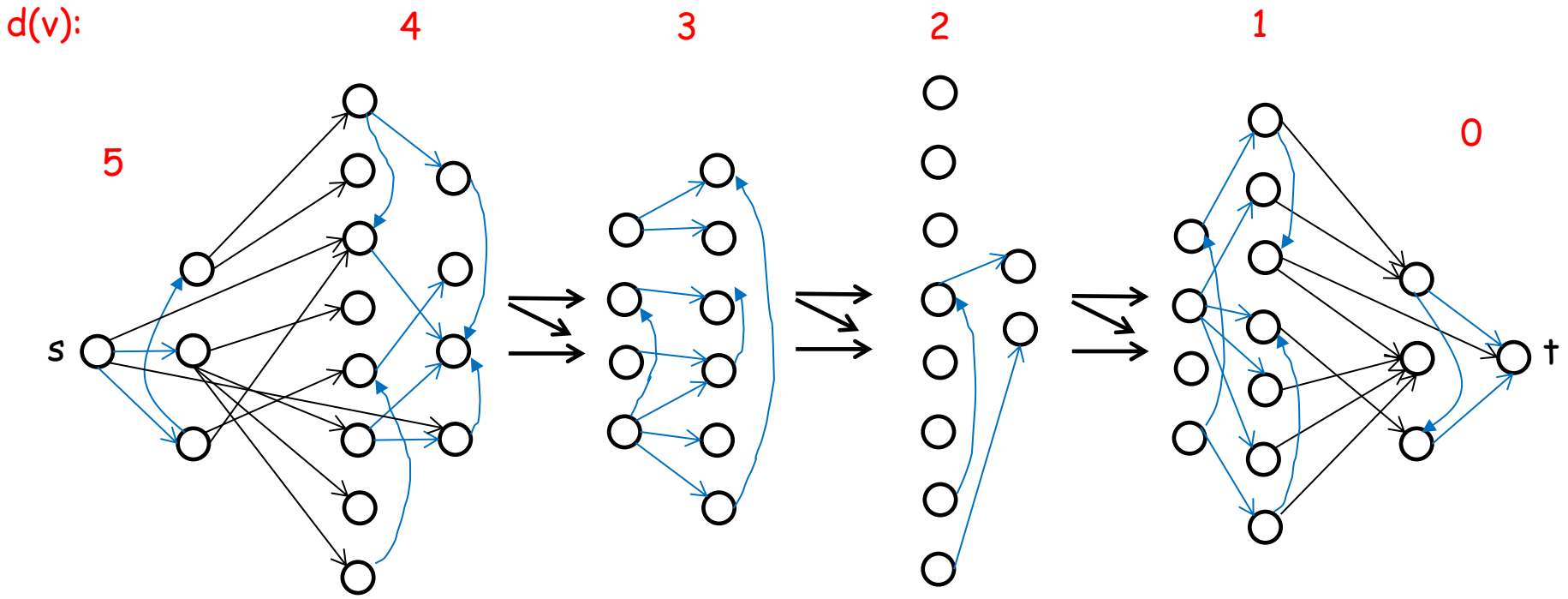


$d_\ell(s) \leq d_\ell(v_1) + \ell'(v,v_1) \leq d_\ell(v_2) + \ell'(v_1,v_2) + \ell'(v,v_1)$

$\ldots\ldots \leq d(t) + d_{\ell'}(s)$

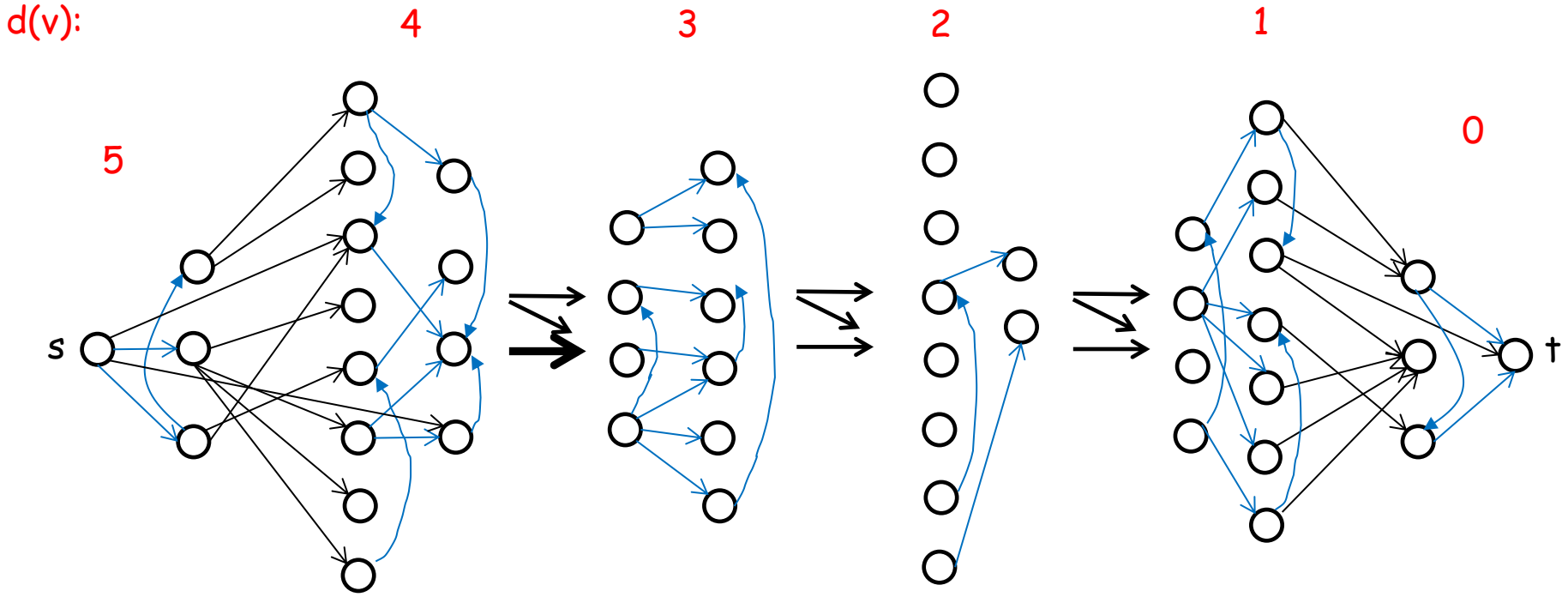
➔ Distances do not decrease

Proof of Lemma 1



Backward arcs and arcs inside a level are ok: $d_\ell(v) \le d_\ell(w)$

Proof of Lemma 1



$d_\ell(v) = d_\ell(w) + \ell(v,w)$ (=1). Can $\ell'(v,w)$ be 0 ?

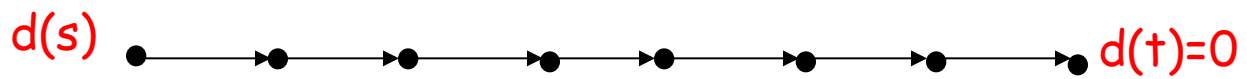
For that its residual capacity should increase but it is impossible since (w,v) is not admissible □

If we add a blocking flow we want the distance to increase

Lemma 2: Let ℓ and d_ℓ be the length and distances before the (blocking) update and let ℓ' and $d_{\ell'}$ be the length and distances after the update then

For some residual arc (v,w) along the shortest (by ℓ') s-t path $d_\ell(v) < d_\ell(w) + \ell'(v,w)$

Corollary 2:



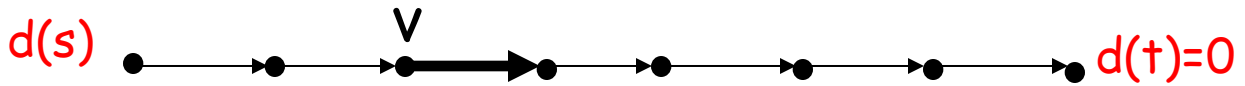
$d_\ell(s) \le d_\ell(v_1) + \ell'(v,v_1) \le d_\ell(v_2) + \ell'(v_1,v_2) + \ell'(v,v_1)$

$\ldots\ldots < d(t) + d_{\ell'}(s)$

➔ Distance **increases** after adding a blocking flow

Proof of Lemma 2

For some residual arc (v,w) along the shortest (by ℓ') s-t path $d_\ell(v) < d_\ell(w) + \ell'(v,w)$

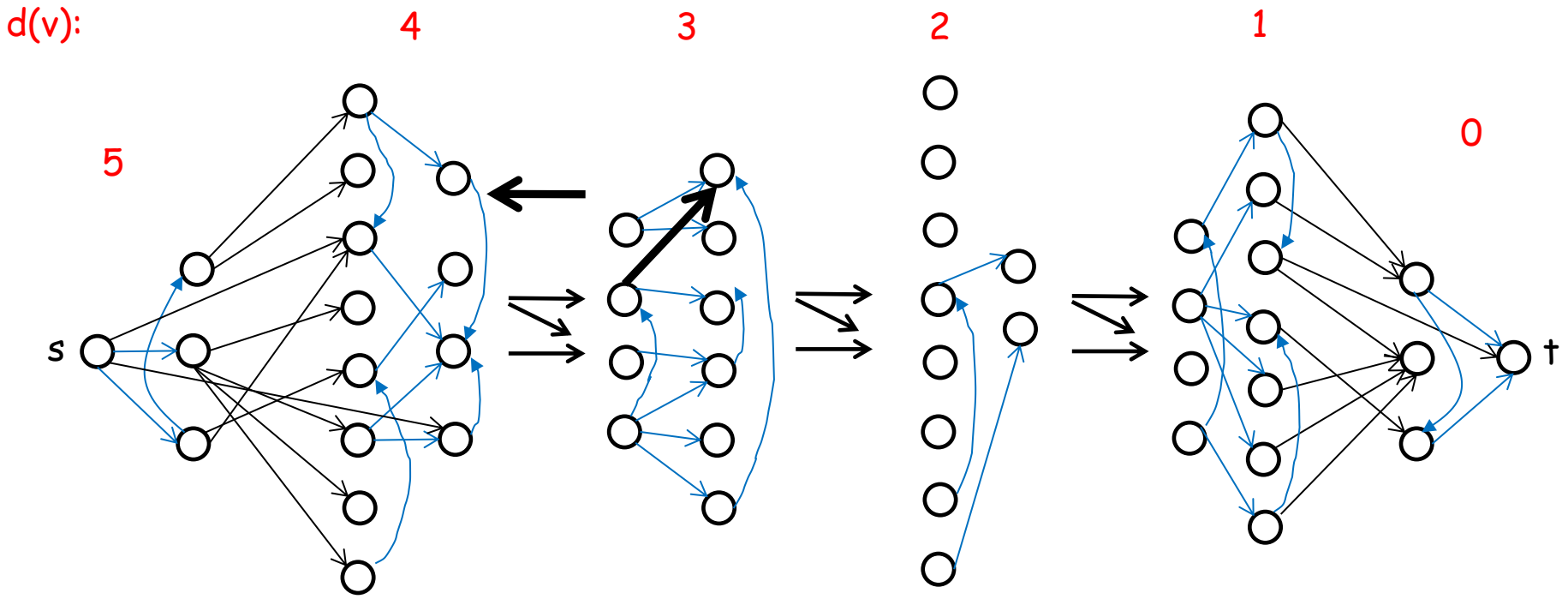


Since the flow is blocking at least one edge along this path was not admissible (for ℓ and d_ℓ)

➔ $d_\ell(v) < d_\ell(w) + \ell(v,w)$

We want to prove that $d_\ell(v) < d_\ell(w) + \ell'(v,w)$

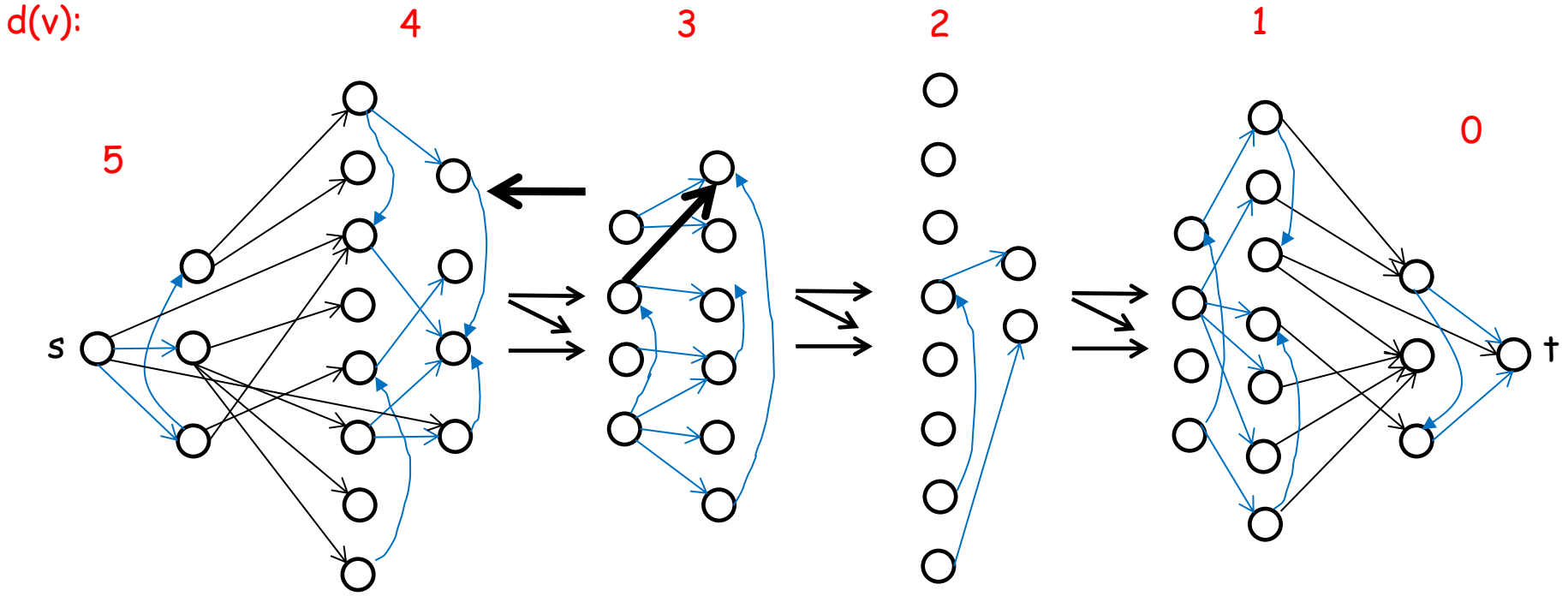
Proof of Lemma 2



$d_\ell(v) < d_\ell(w) + \ell(v,w)$

This arc can go backward, or can have $\ell(v,w) = 1$ and be inside a level

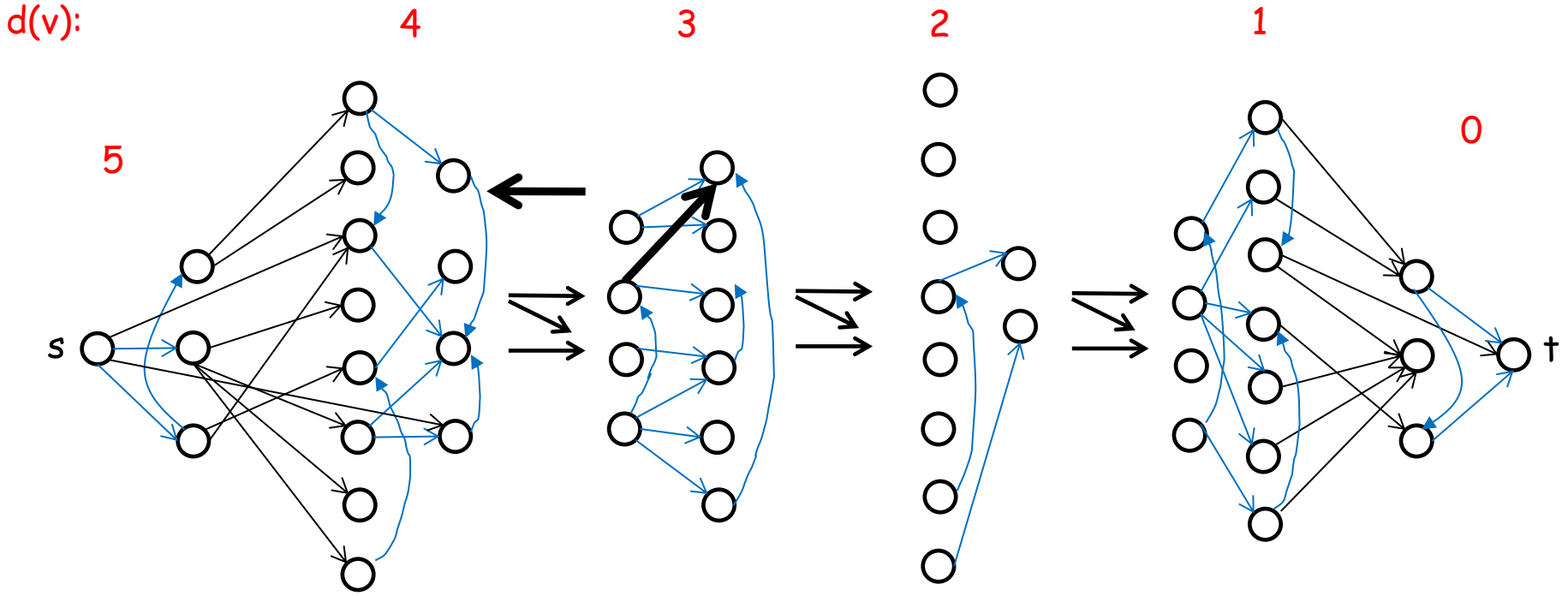
Proof of Lemma 2



$d_\ell(v) < d_\ell(w) + \ell(v,w)$

If it goes backward then $d_\ell(v) < d_\ell(w)$ and then $d_\ell(v) < d_\ell(w) + \ell'(v,w)$

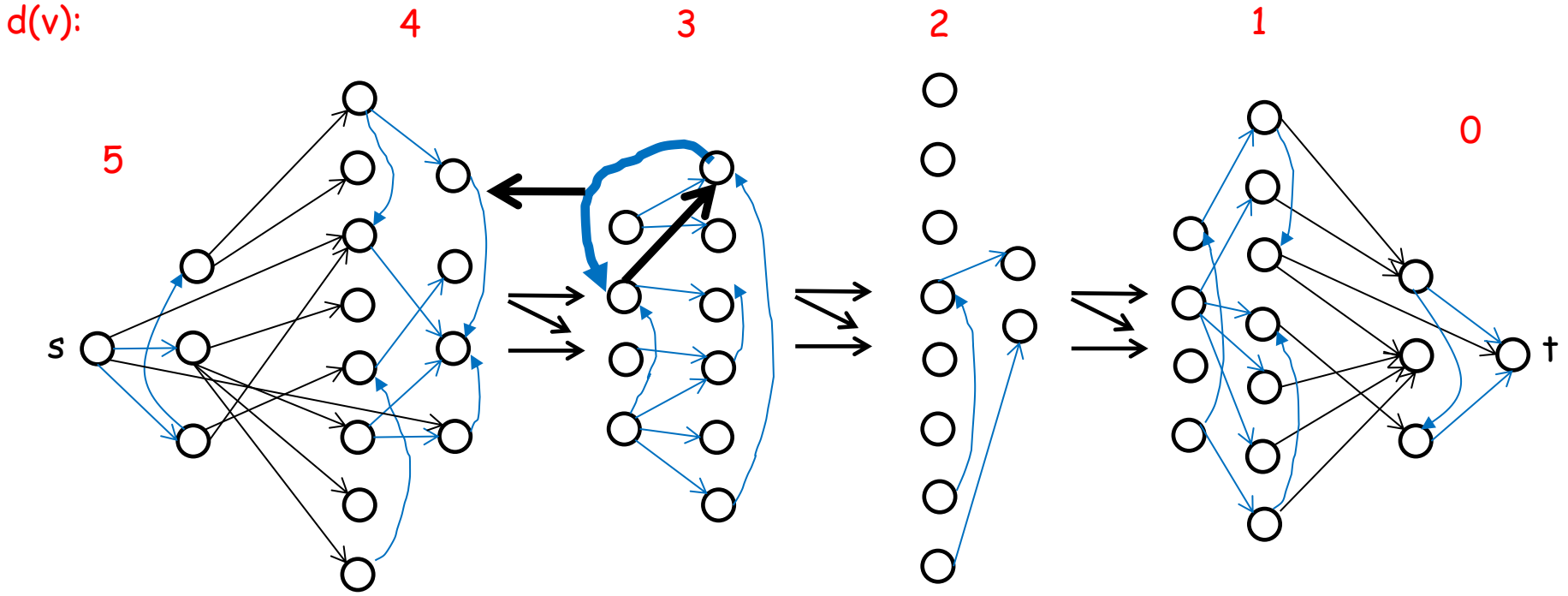
Proof of Lemma 2



$d_\ell(v) < d_\ell(w) + \ell(v,w)$

If $\ell(v,w) = 1$ and it goes inside a level then $d_\ell(v) = d_\ell(w)$ and we have a problem when $\ell'(v,w) = 0$

Proof of Lemma 2



$d_\ell(v) < d_\ell(w) + \ell(v,w)$

We should have pushed flow on (w,v) so (w,v) was admissible and $\ell(w,v) = 0$

Proof of Lemma 2

At the moment there is nothing which prevents this from happening

So we make such arcs admissible

Set $\ell(v,w) = 0$ if $d_\ell(v) = d_\ell(w)$, $\ell(w,v) = 0$, and $u(v,w) \geq 2\Delta$

We call such arcs special

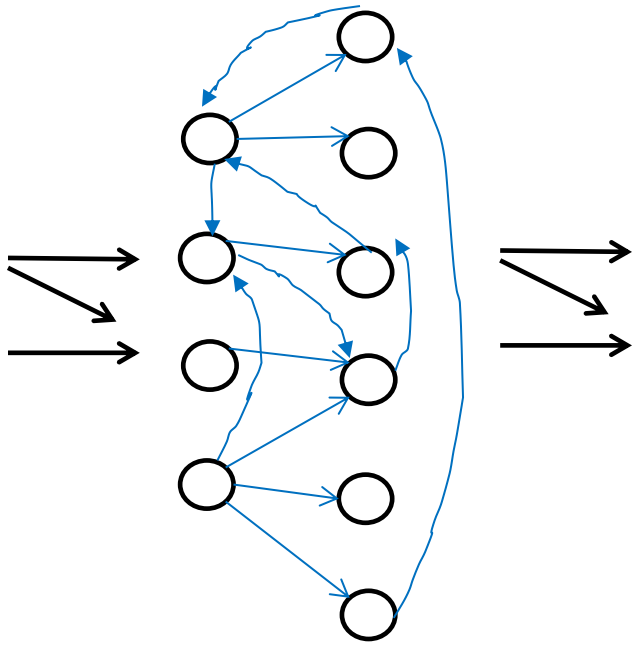
This modification fixes the proof of Lemma 2 and does not hurt Lemma 1

More details

- How do we find a flow of value Δ or a blocking flow ?
- Our admissible graph is cyclic…

Making the admissible graph acyclic

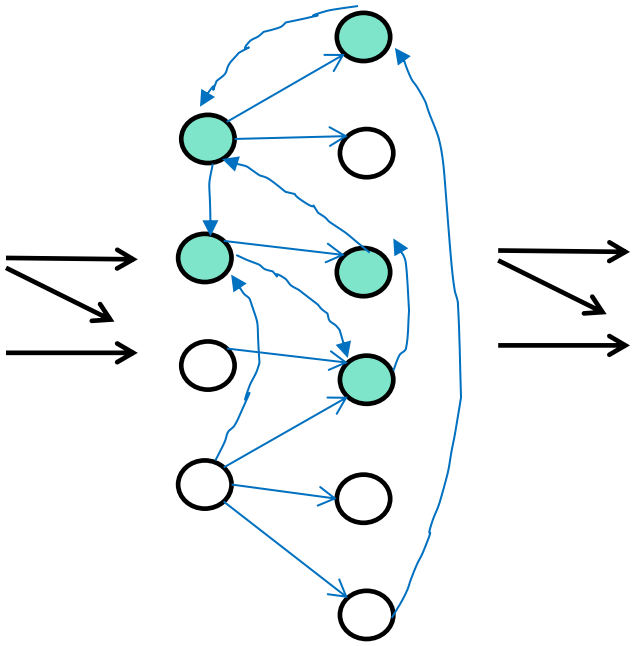
- Make it acyclic by contracting strongly connected components (of 0 length arcs)



Making the admissible graph acyclic

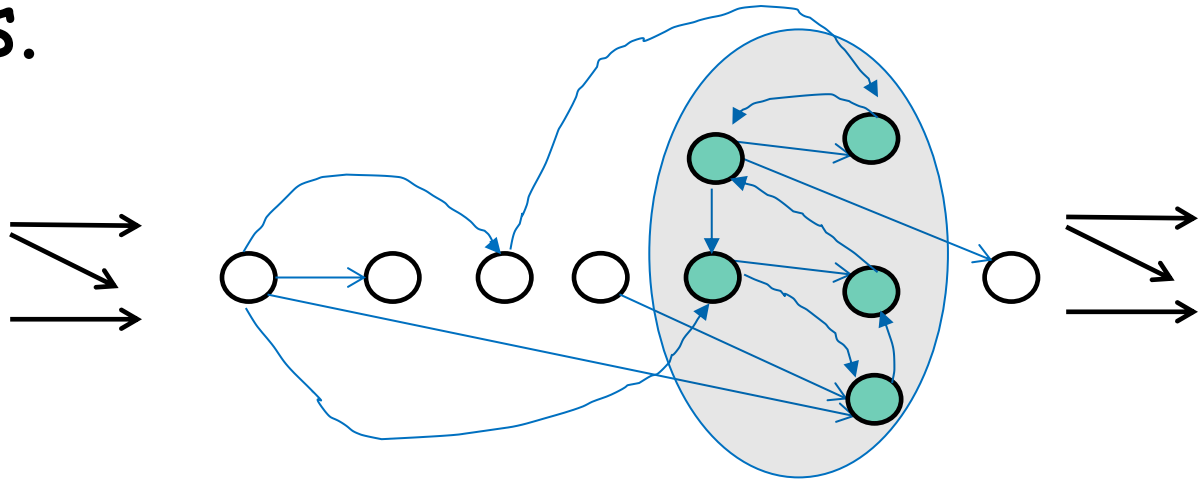
- Make it admissible by contracting strongly connected components (of 0 length arcs)

Construct a new node for each component that points to the vertices of the component, O(m) time.



Find a blocking flow in the acyclic graph

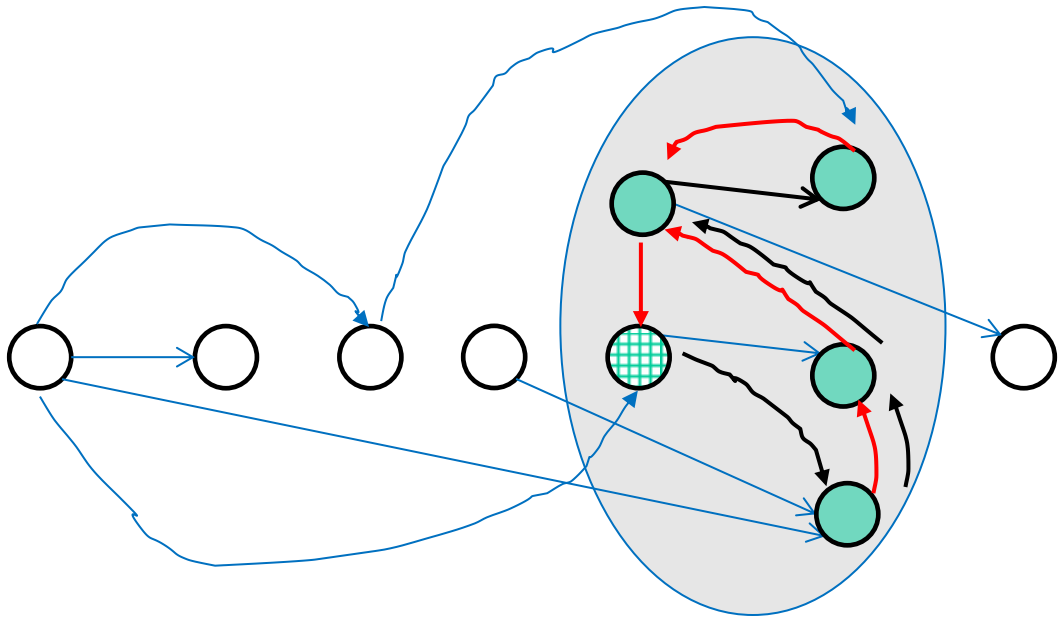
- If the flow value > Δ then return flow to s.



- Order the (contracted) vertices in topological order and return flow along this order

Route flow through contracted vertices

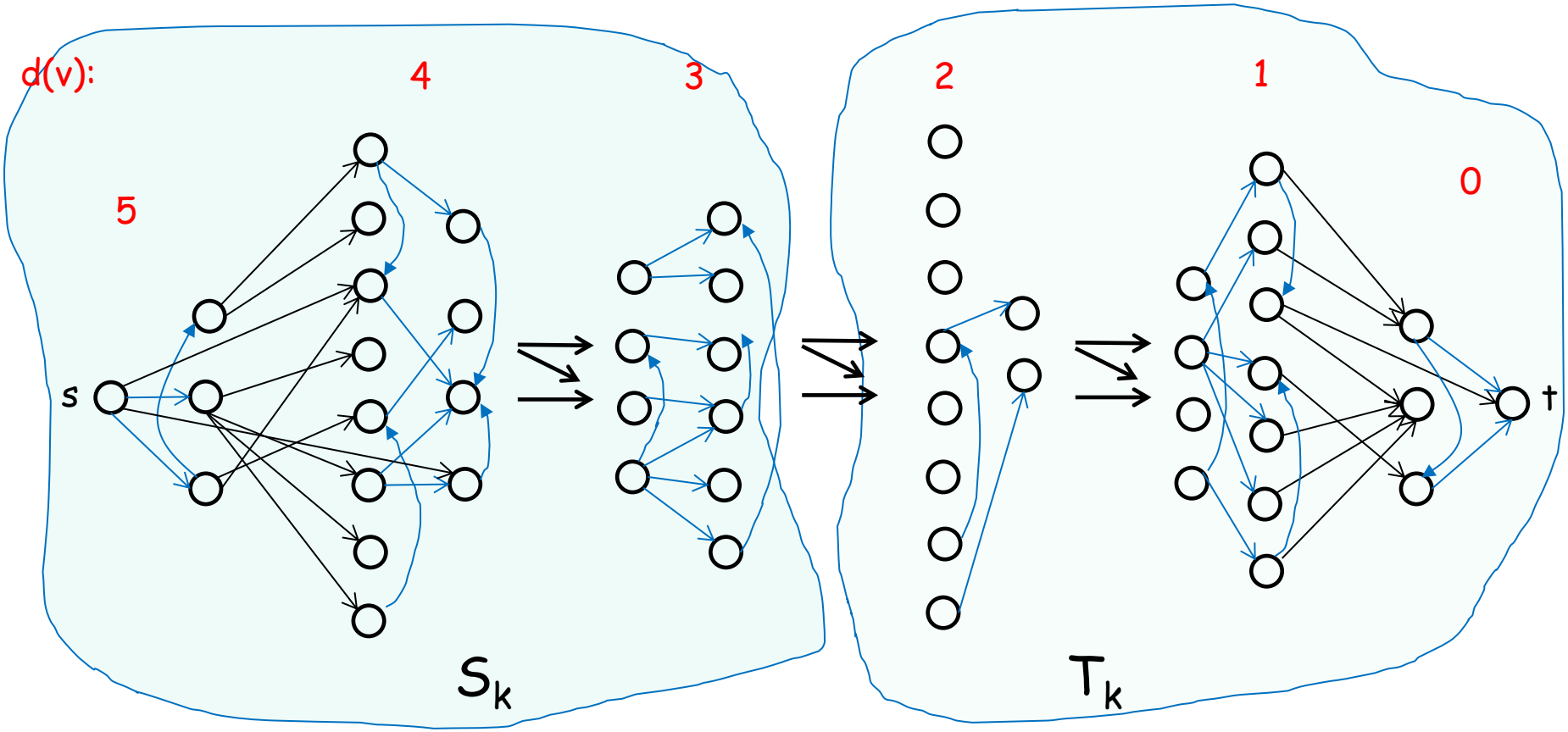
- Pick an arbitrary vertex z in the component



The capacity of all arcs is $\geq 2\Delta$, and no more than Δ flow enters and leaves the trees..

- Pick a spanning tree into z, and a spanning tree out of z; send all the incoming flow to z on the red tree and then out of z on the black tree

Maintaining the upper bound F



Compute the capacity of each canonical cut (S_k, T_k) and take the min, if smaller than F/2 start the next phase

Summary

- Compute the smallest canonical cut, if ≤ F/2 start a new phase
- Compute ℓ, modified on special arcs, and d_ℓ
- Contract strongly connected components of 0-length arcs
- Compute a blocking flow in the contracted admissible (acyclic) graph
- Add the blocking flow to the current flow

Improving the bound

- The bound we got is $O(m\sqrt{m}\log(n)\log(nU))$
- It is, in fact, $O(m\sqrt{m}\log(n)\log(U))$ by a more careful analysis

The open problem

- Can we generalize these idea to minimum cost flow ?

- Get an algorithm for min cost flow that runs in

  $O(m\sqrt{m}\log(n)\log(nU)\log(nC))$ time ??