Advanced Graph

Homer Lee

2013/10/31

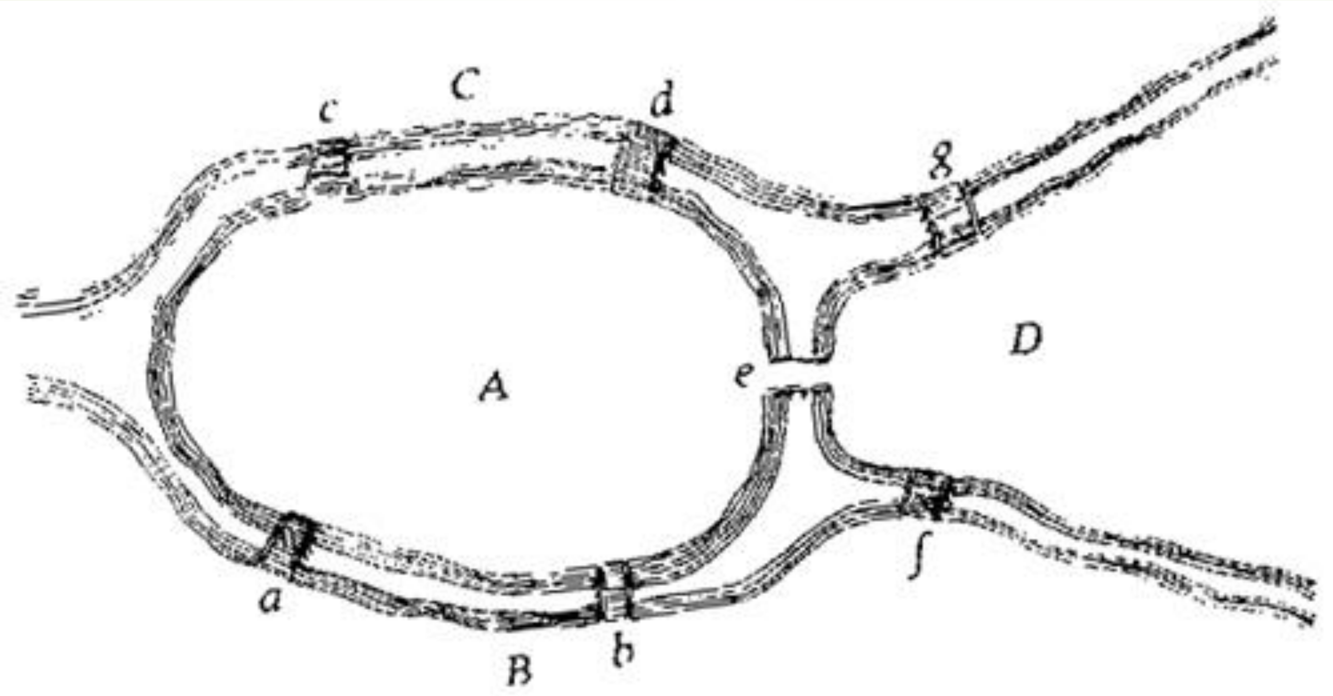
Reference

- Slides from Prof. Ya-Yunn Su's and Prof. Hsueh-I Lu's course

Today’s goal

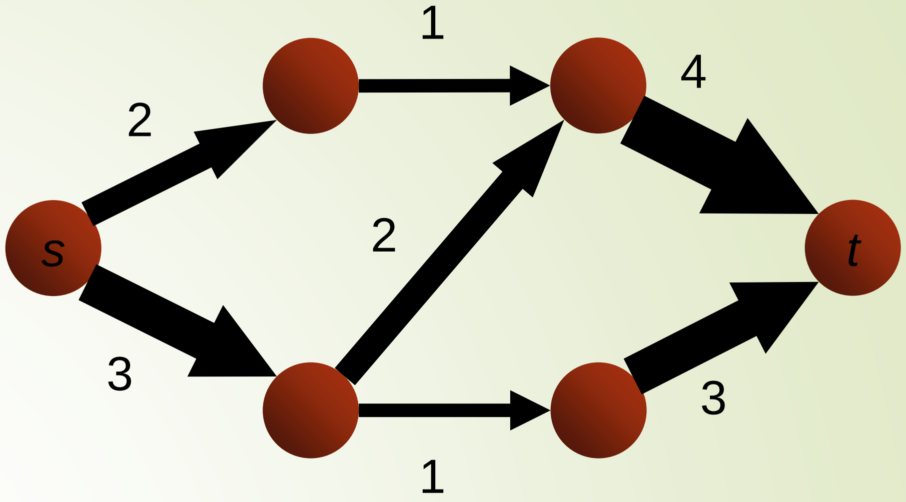
- Flow networks
- Ford-Fulkerson (and Edmonds-Karp)
- Bipartite matching

Flow networks



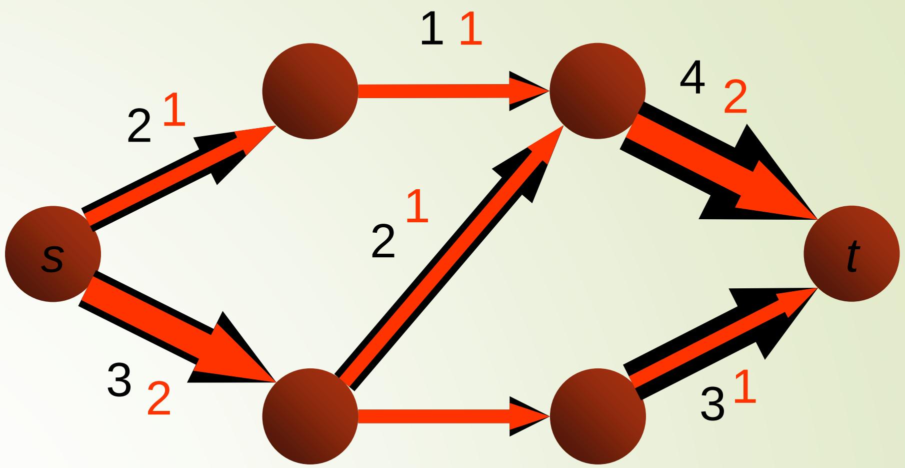
*

Network



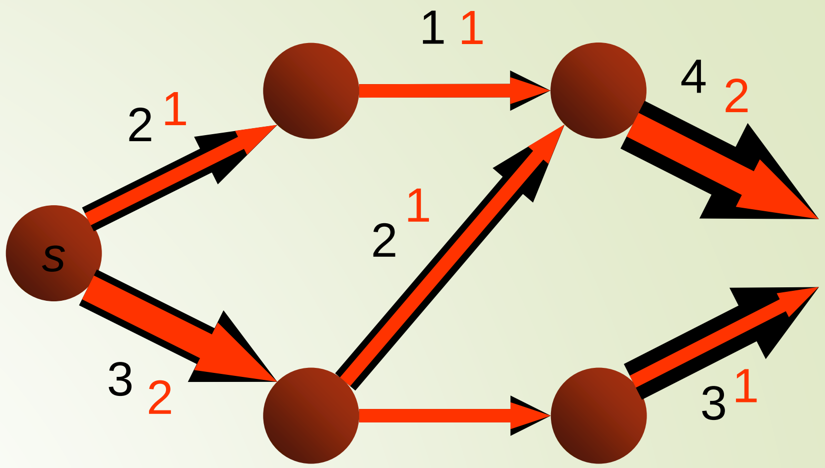
- A directed graph G, each of whose edges has a capacity
- Two nodes s and t of G
- Denoted as (G,s,t)

Flow



- A flow of network (G,s,t) is (weighted) **subgraph** of G satisfying the capacity constraint and the conservation law

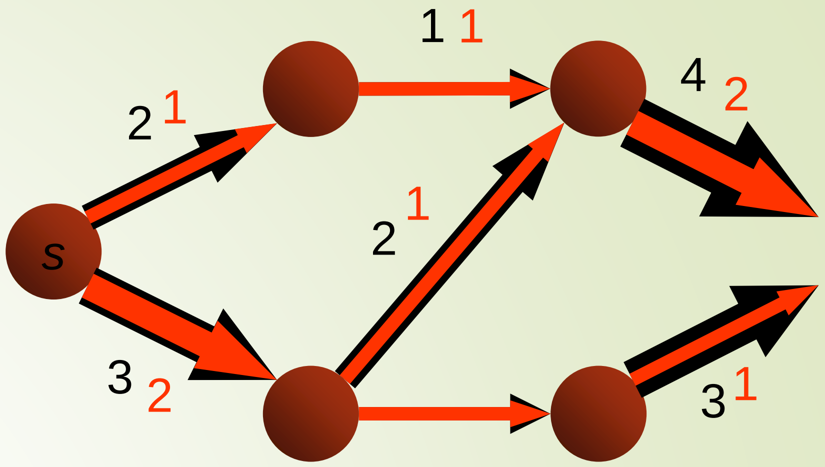
Capacity constraint



- Given capacity constraint function c, for all $u, v \in V$, $0 \le f(u,v) \le c(u,v)$

- 流過edge的flow大小要小於流量限制
- 常表示為圖的edge的weight

Conservation Law



- For all $u \in V - \{s,t\}$,$\sum_{v\in V} f(u,v) = \sum_{v\in V} f(v,u)$
- 換句話說，流進來的等於流出去的
- 上面的sigma是對所有的v，那假如u,v沒有接在一起怎麼辦?
- => If $(u,v) \notin E$, $f(u,v) = 0$

What's maximum flow problem?

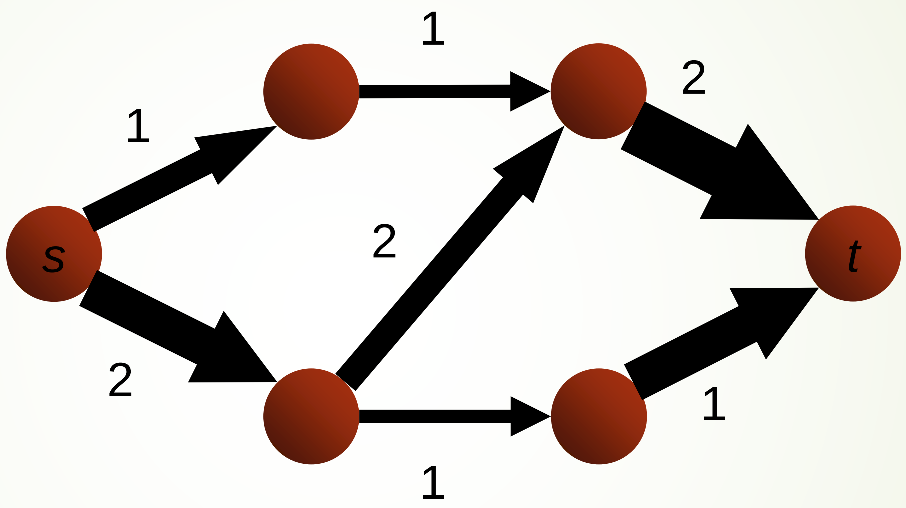
- The value of a flow is denoted as |f|
- $|f| = \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s)$

- Given flow network (G,s,t)
- => Find a flow of maximum value

A famous theorem

- Maximum flow ←→ Minimum cut

How to solve the problem?



Ford-Fulkerson

- Idea: Iteratively increase the value of the flow

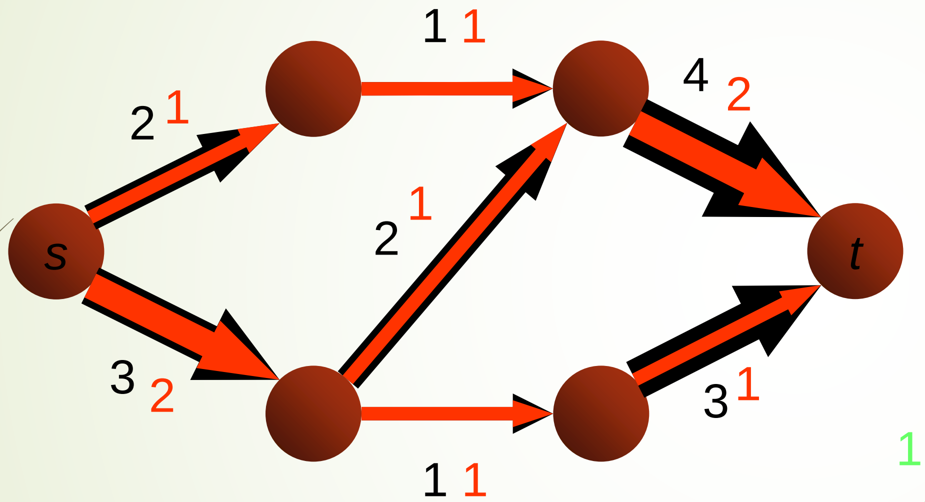
- Start with f(u,v) = 0
- In each iteration
  - Find an **augmenting path** in the **residual network** G_f
  - Until no more augmenting paths exist

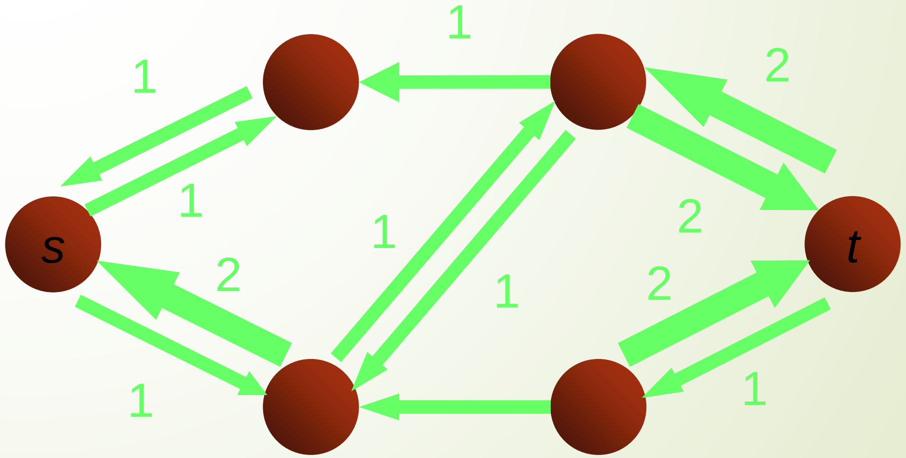
Residual Network

- What if we choose the wrong path?
- Just give it a second chance!!

- For each edge (u,v) in G,construct G_f
  - If f(u,v) > 0, G_f has an edge **(v,u)** with weight f(u,v)
  - If c(u,v) ≥ f(u,v) G_f has an edge **(u,v)** with weight c(u,v) – f(u,v)

illustration





Some notes

- Why adding an edge if c(u,v) ≥ f(u,v) ?
  - 讓他有回頭的機會
- $|E_f| \leq 2|E|$,Why?
  - Residual Networks上的edges會是原本Flow network有的edge (with different weight) 或反方向
  - You can try to prove it by simply using case analysis.

How can residual network help us?

- Lemma: Let G = (V,E) be a flow network, and let f be a flow in G. Let G_f be the residual network of G induced by f. Let g be a flow in G_f. Then

  f+g = is a valid flow in G.
- Idea: capacity constraint and flow conservation still holds.

Augmenting paths

- Given a flow network G and a flow *f*, an augmenting path is a simple path from *s* to *t* in the residual network G_f
- 簡單的來說，就是在residual network上找一條可以走的路

How can Augmenting paths help us?

- There exists an augmenting path
- => there exist some potential flow in the path
- => By the capacity constraint, trivially the maximum flow in the path

  = min{C_f (u,v) | (u,v) is on augmenting path}

How do we know when we have found maximum flow?

- From the maximum-flow-minimum-cut theorem, we stop when its residual graph contains no augmenting graph

- 2 equivalent things:
- 1. f is a maximum flow in G
- 2. The residual network G_f contains no augmenting path

Let’s prove it!

- f is a maximum flow in G => The residual network G_f contains no augmenting path
- => is simple, use contradiction.
- If G_f still contains augmenting paths, then we can still find f_p to add to f. Then result in bigger flow |f| + |f_p| > |f|

Let's prove it!!

- Goal: f is a maximum flow in G <= The residual network G_f contains no augmenting path
- Equivalent statement: f is ***not*** a maximum flow in G => The residual network G_f contains ***some*** augmenting path
- If *h* is a flow whose value larger than that of *f*, then g = h – f has to be a positive flow in G_f

Let's prove it!!!

- Goal: f is ***not*** a maximum flow in G <= The residual network G_f contains ***some*** augmenting path
- Here provides a **sketch** of the proof
- If *h* is a flow whose value larger than that of *f*, then g = h – f has to be a positive flow in G_f
  - Then why g has to be a positive flow in G_f?
  - Do the remaining job by yourself!

Pseudo code

- Ford-Folkerson (G,s,t){

  for each edge (**u,v**) in G.E

  (**u,v**).f = 0

  while there exists an augmenting path **p** in residual network G_f

  $\boldsymbol{c_f}(\mathbf{p})$ **=** min($\boldsymbol{c_f}(\mathrm{u,v})$ | (u,v) is on **p**)

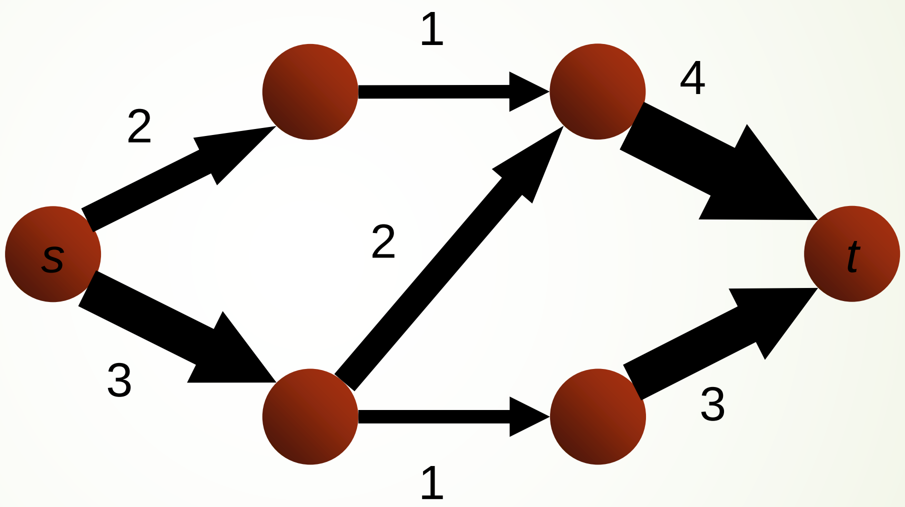
  for each edge (u,v) on **p** //雙向都要考慮

  if (u,v) ∈ E, (u,v).f += $\boldsymbol{c_f}(\mathbf{p})$

  else, (v,u).f -= $\boldsymbol{c_f}(\mathbf{p})$

}

Example:



Running time analysis

- Initializing part: O(E)
- How to find a path in residual network?
  - BFS or DFS
  - What time complexity does it take?
    - $|E_f| \leq 2|E|$
    - It takes $O(V+E_f) = O(E)$ times

Running time analysis

- If the edge capacity are integer, and f be the maximum flow of the network
- The for-loop may be executed at most f times(increment by 1 unit at a time)
- Each time takes O(E) times
- Totally O(E)+O(E)*O(f) = O(Ef)
- => This depends on f, not a good idea

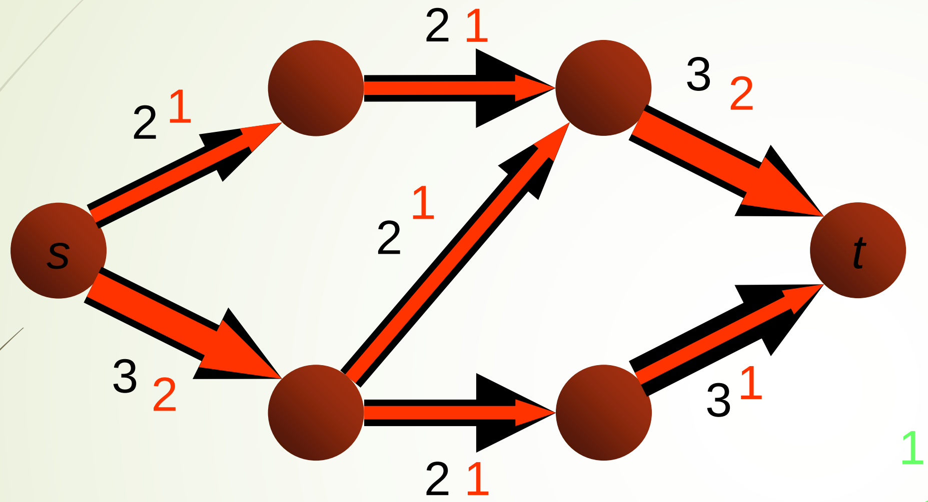
Issues

- What is C is not an integer?(each time the amount that the augmenting path adding has no lower-bound)
- How can we improve this bound?

  (Edmonds-Karp)

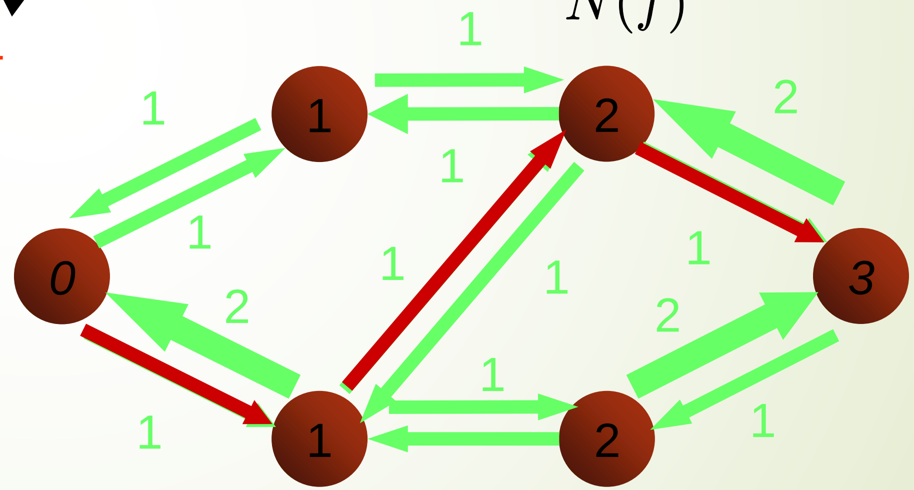
To improve the time complexity

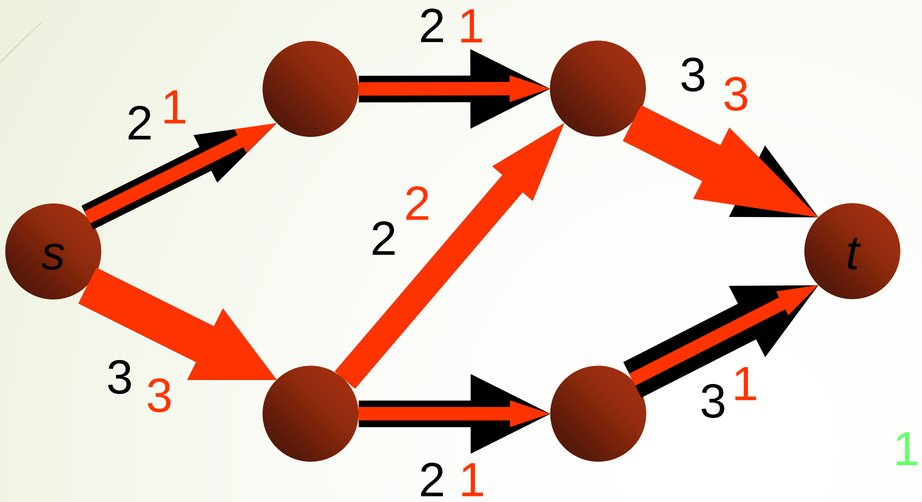
- A key observation: Let P be a shortest path from s to t in the residual network G_f. Let g be the flow corresponding to P.
- Then, the distance of any nodes v from s in G_{f+g} is no less than that in G_f
- **IDEA**: If the above thing holds, it seems that the update times will be bounded



f

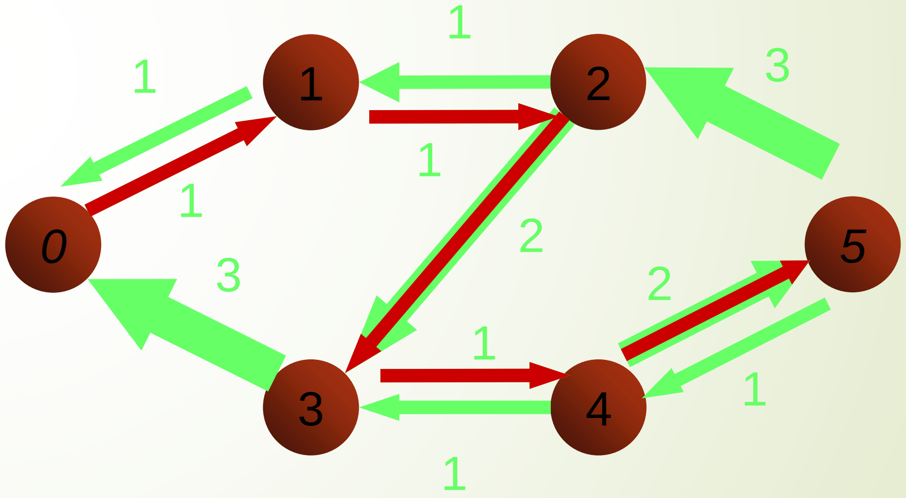
$N(f)$

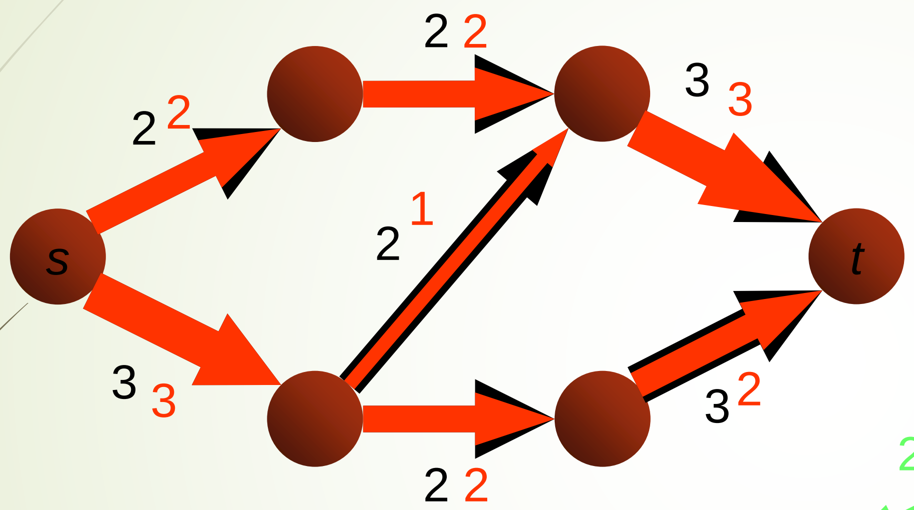




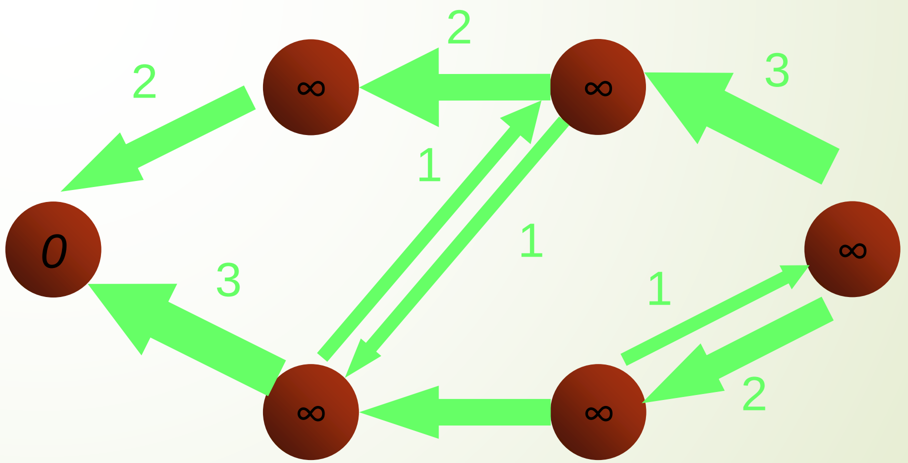
$f+g$

$N(f+g)$



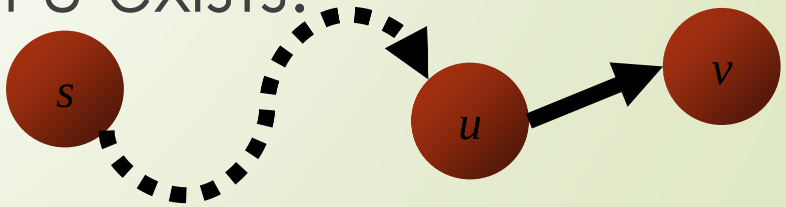


$N(f+g+h)$



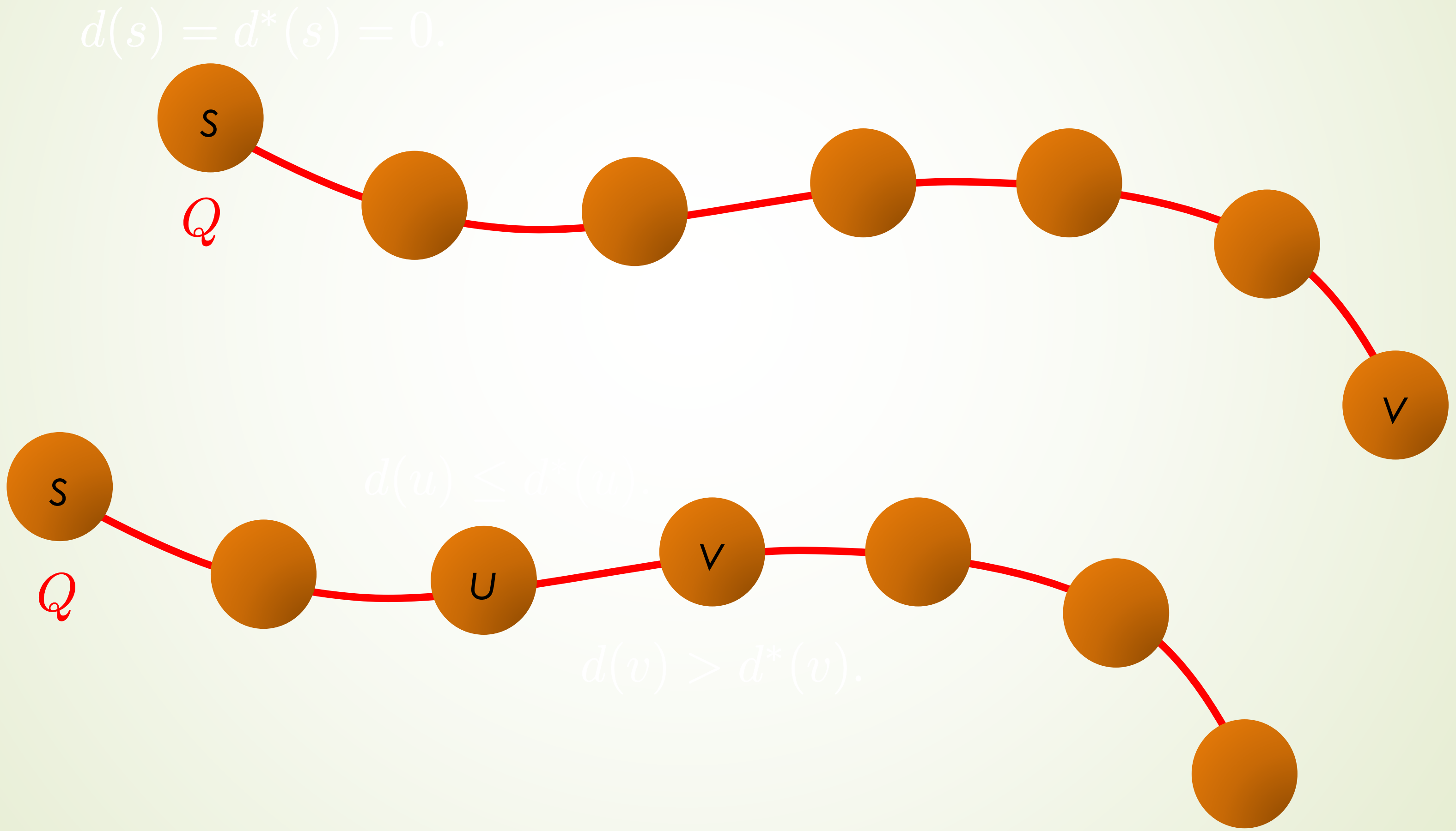
Proof of the monotonically increased distance

- Assume for contradiction that there is **a** node v whose distance d*(v) in G_{f+g} is less than its distance d(v) in G_f
- Let Q be a shortest path from s to v in G_{f+g}
- There has to be some node u on Q such that d*(u) ≥ d(u).(s is such a u.)
- So we can assume Q = s~u->v, so d*(v) = d*(u)+1 and (u,v)∈ G_{f+g} . such u exists.



That's what we said last page

Note: d*(u) ≤ d(u)



Proof-contd.

- Claim: (u,v) cannot be an edge in G_f since the following contradiction:
  - d(v) ≤ d(u)+1 ≤ d*(u)+1 = d*(v)
- Since (u,v) belongs to G_{f+g} but not G_f, we know g goes from v to u, but still reach the following contradiction:
  - d(v) = d(u)-1 ≤ d*(u)-1 = d*(v)-2