Lecture 10: Graph Algorithms (IV)

Dr. Tsung-Wei Huang
Department of Electrical and Computer Engineering
University of Utah, Salt Lake City, UT

HW2 Due Today through Email

❑ Sample result

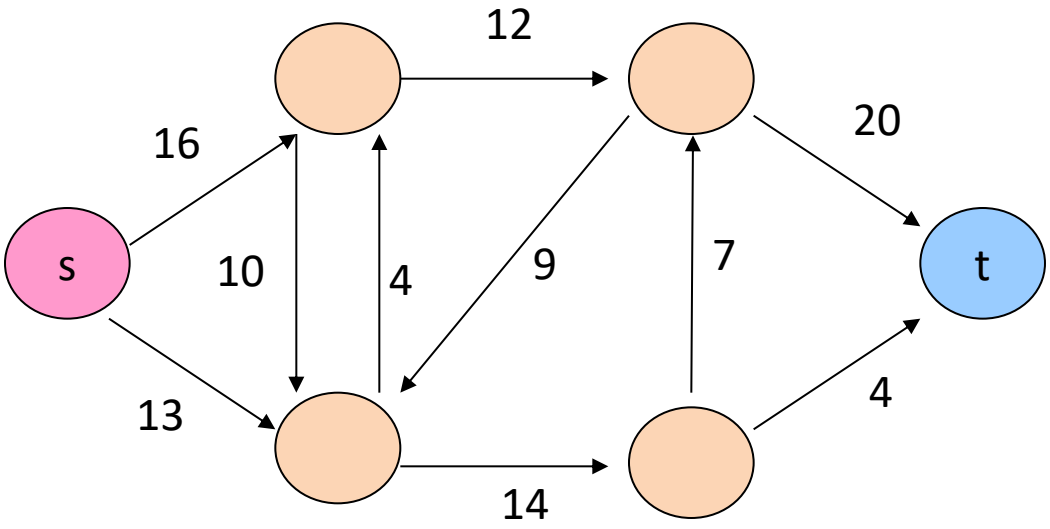
	Graph1	Graph2	Graph3	Graph4	Graph5	Graph6	Graph7	Graph8	Graph9	Graph10
Dijkstra Priority Queue	52ms	104ms	37ms	157ms	178ms	187ms	149ms	226ms	212ms	289ms
Dijkstra Min Heap	53ms	88ms	36ms	185ms	170ms	174ms	130ms	227ms	197ms	256ms
SPFA Queue	39ms	56ms	16ms	85ms	113ms	115ms	93ms	129ms	108ms	160ms
SPFA Stack	1005ms	5359ms	2238ms	27218ms	4600ms	10107ms	3579ms	24683ms	23578ms	30028ms

❑ What is the lesson we learned?

- ❑ BFS-styled propagation works for problems with:
  - Optimal substructure
  - Wavefront computing patterns
- ❑ In shortest path, we maintain wavefront
  - Relaxation: $d[u] + w(u, v) < d[v]$

Recap: Maximum Flow Problem

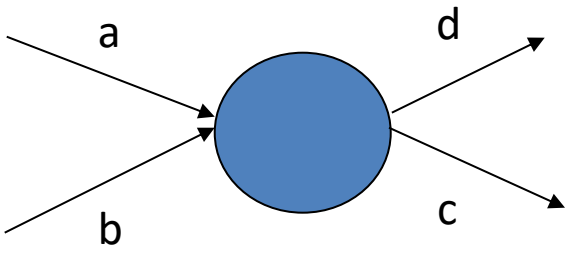
- **Network flow problem**
  - A flow network G=(V,E): a directed graph, where each edge (u,v)∈E has a nonnegative capacity c(u,v)>=0.
  - If (u,v)∉E, we can assume that c(u,v)=0.
  - two distinct vertices :**a source *s* and a sink *t*.**



Flow Constraint

- G=(V,E): a flow network with capacity function c.
- *s* -- the source and *t* -- the sink.
- A flow f(u, v) in G must satisfy
  1. Capacity constraint
     - For all u,v $\in$V, we require f(u, v) $\leq$ c(u, v).
  2. Flow conservation
     - For all u $\in$V-{s, t}, we require

$$\sum_{e.in.v} f(e) = \sum_{e.out.v} f(e)$$



a+b = d+c

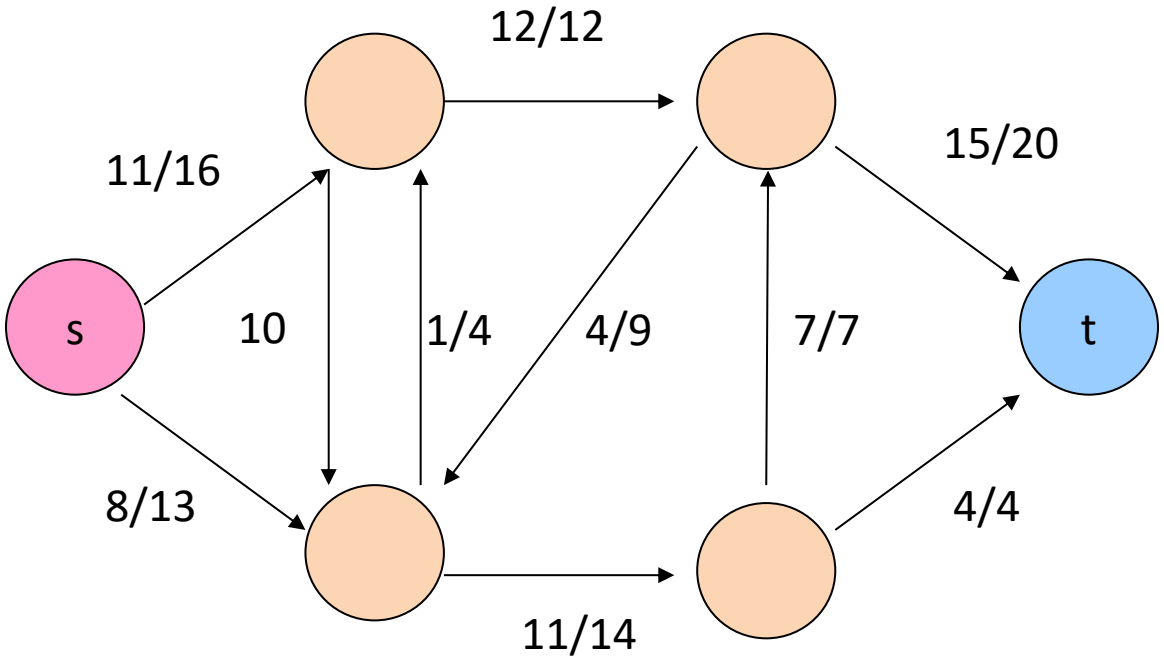
Objective

- **The quantity f (u, v) is called the net flow from vertex u to vertex v.**
- **The value of a flow is defined as**

$$|f| = \sum_{v \in V} f(s, v)$$

  - The total flow from source to any other vertices.
  - The same as the total flow from any vertices to **the sink.**

Example



A flow f in G with value $|f| = 19$

Method to Compute Maximum Flow

FORD-FULKERSON-FRAMEWORK(G, s, t)

initialize flow *f* to *0*

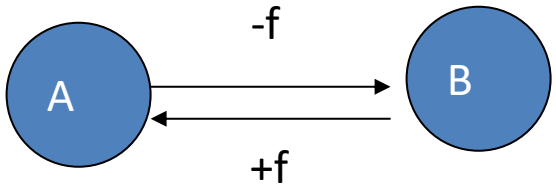
while there exists an *augmenting* path *p*

do *augment* flow *f* along *p*

return *f*

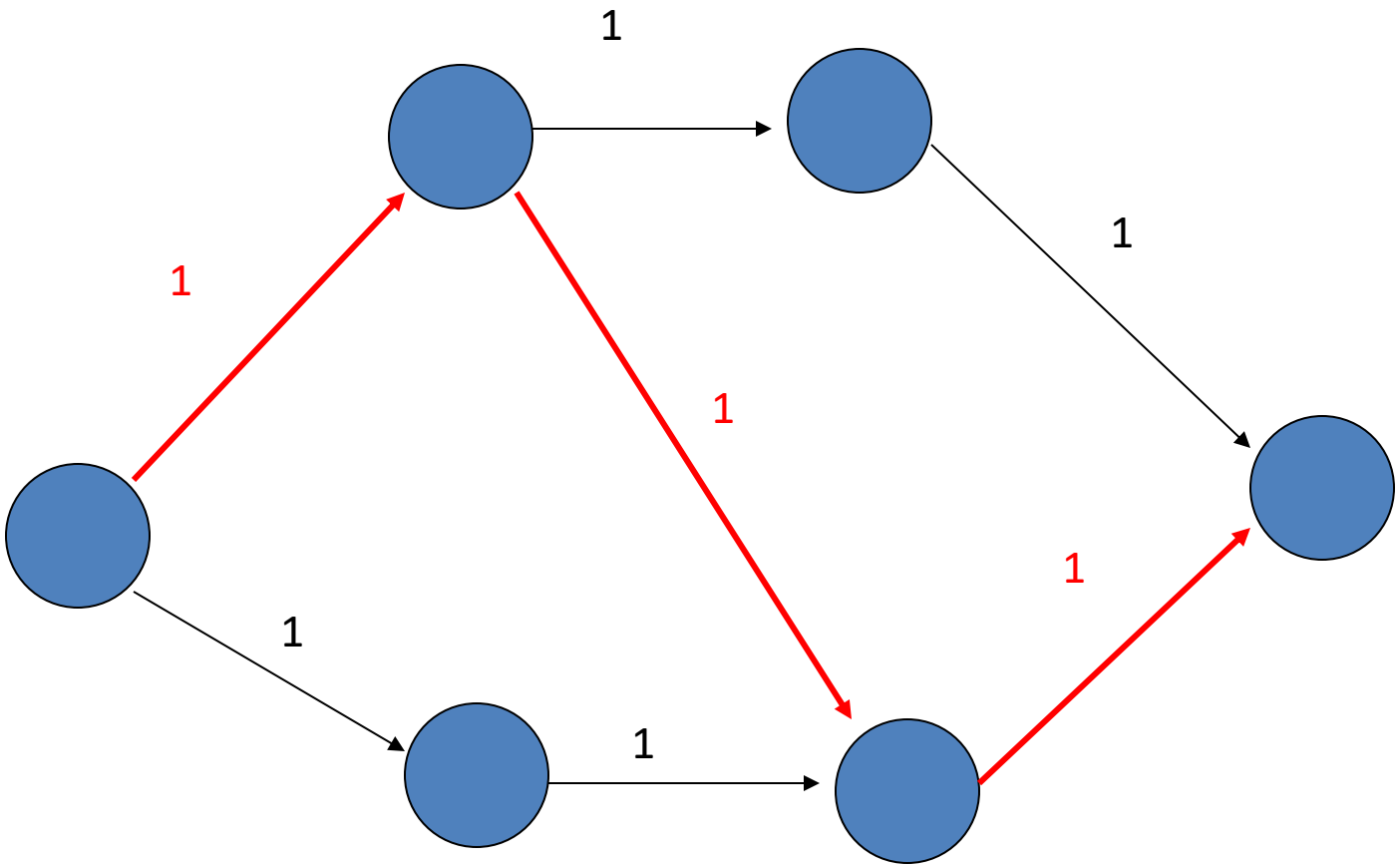
Residual Network

- **Residual network defines edges to admit net flow**
  - The amount of additional net flow from u to v before exceeding the capacity c(u,v) is the residual capacity of (u,v), given by:
    - In the regular direction: $c_f(u,v)=c(u,v)-f(u,v)$
    - In the opposite direction: $c_f(v,u)=c(v,u)+f(u,v)$.
- **If you flow f from A to B**
  - Subtract the regular direction capacity from f
  - Add f to the opposite direction capacity



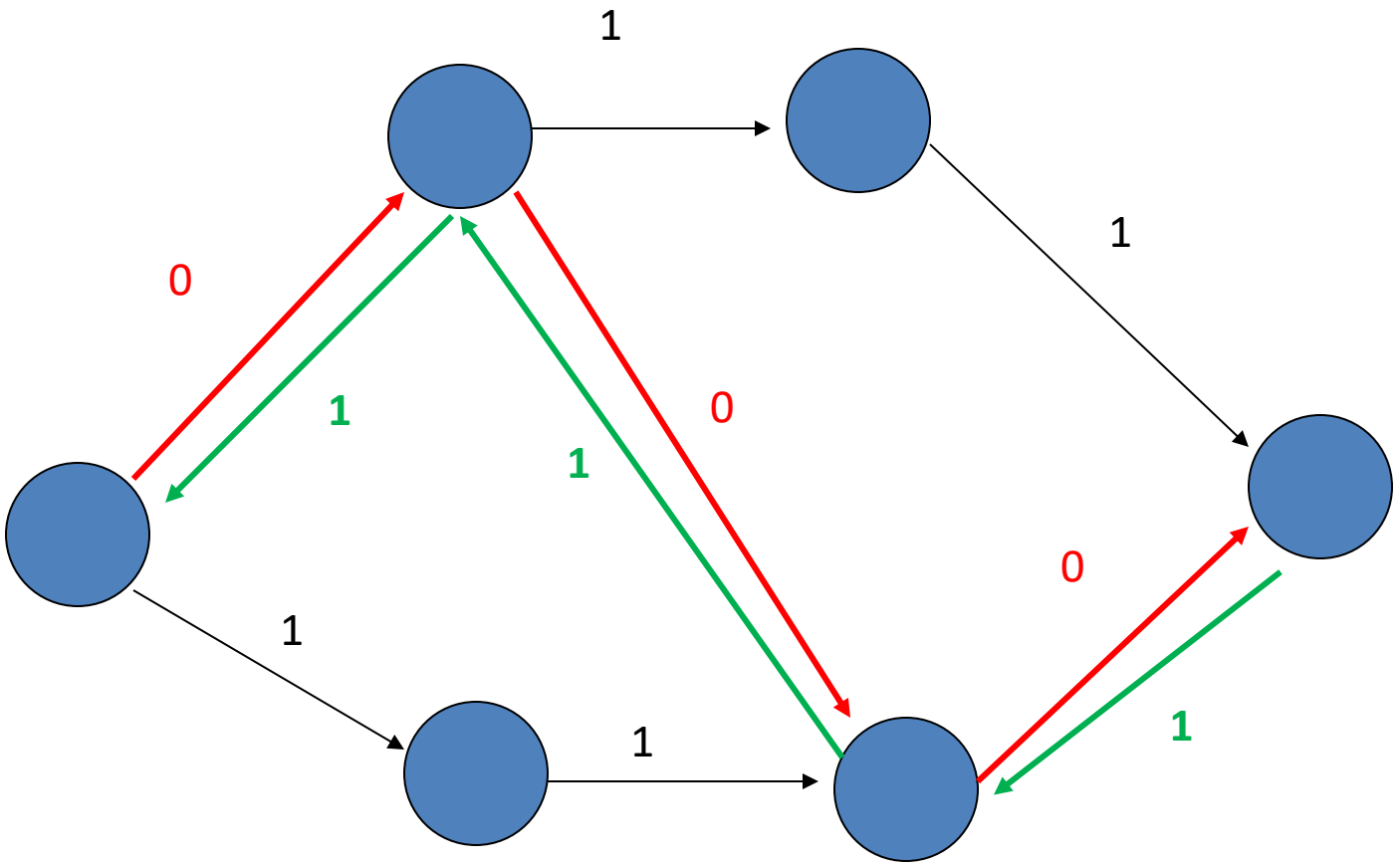
Example

- DFS augments a unit flow in the first iteration



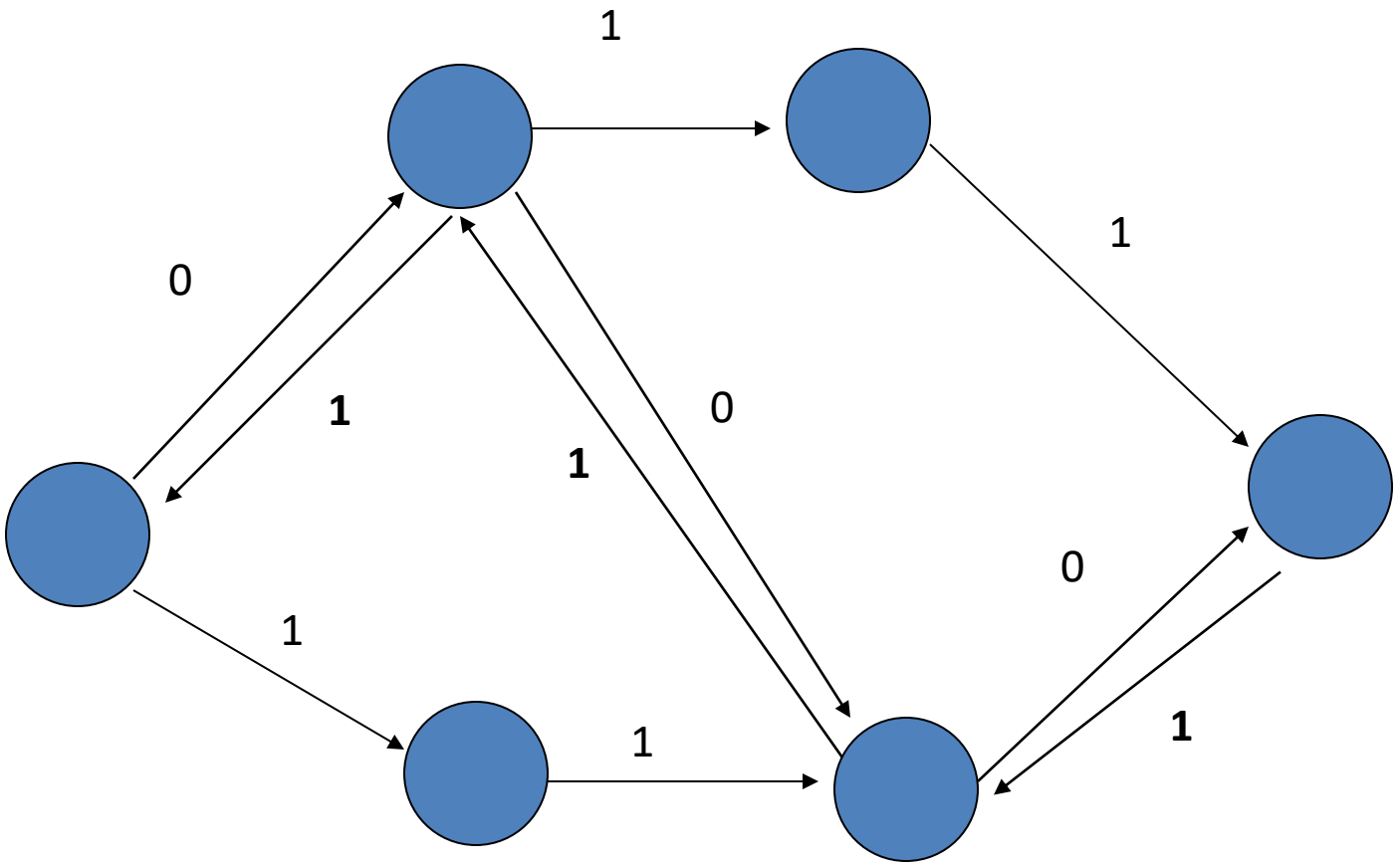
Example

- Update the residual network



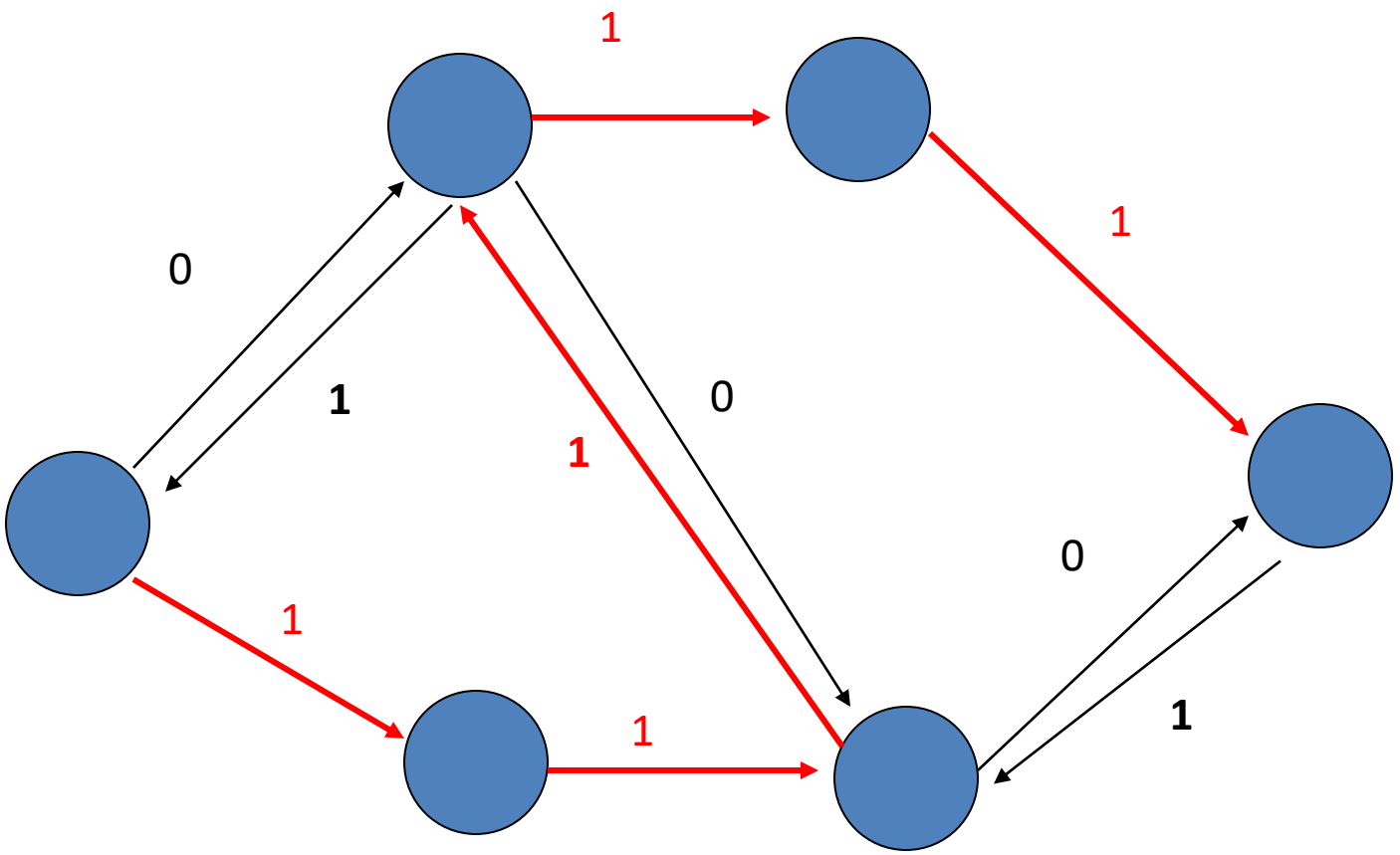
Example

- Residual network gives a chance to “circle back”



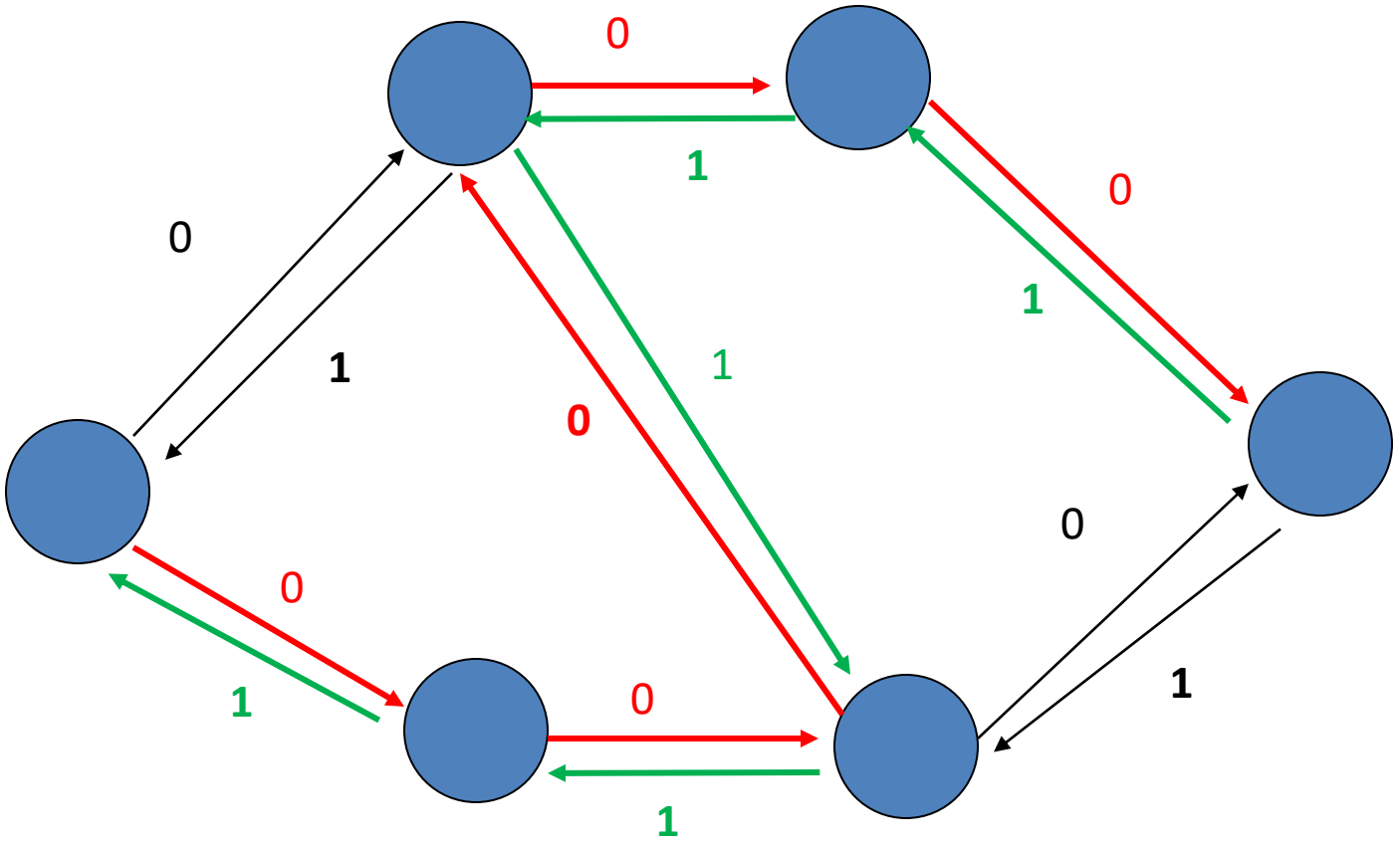
Example

- DFS augments another unit flow in the second iter



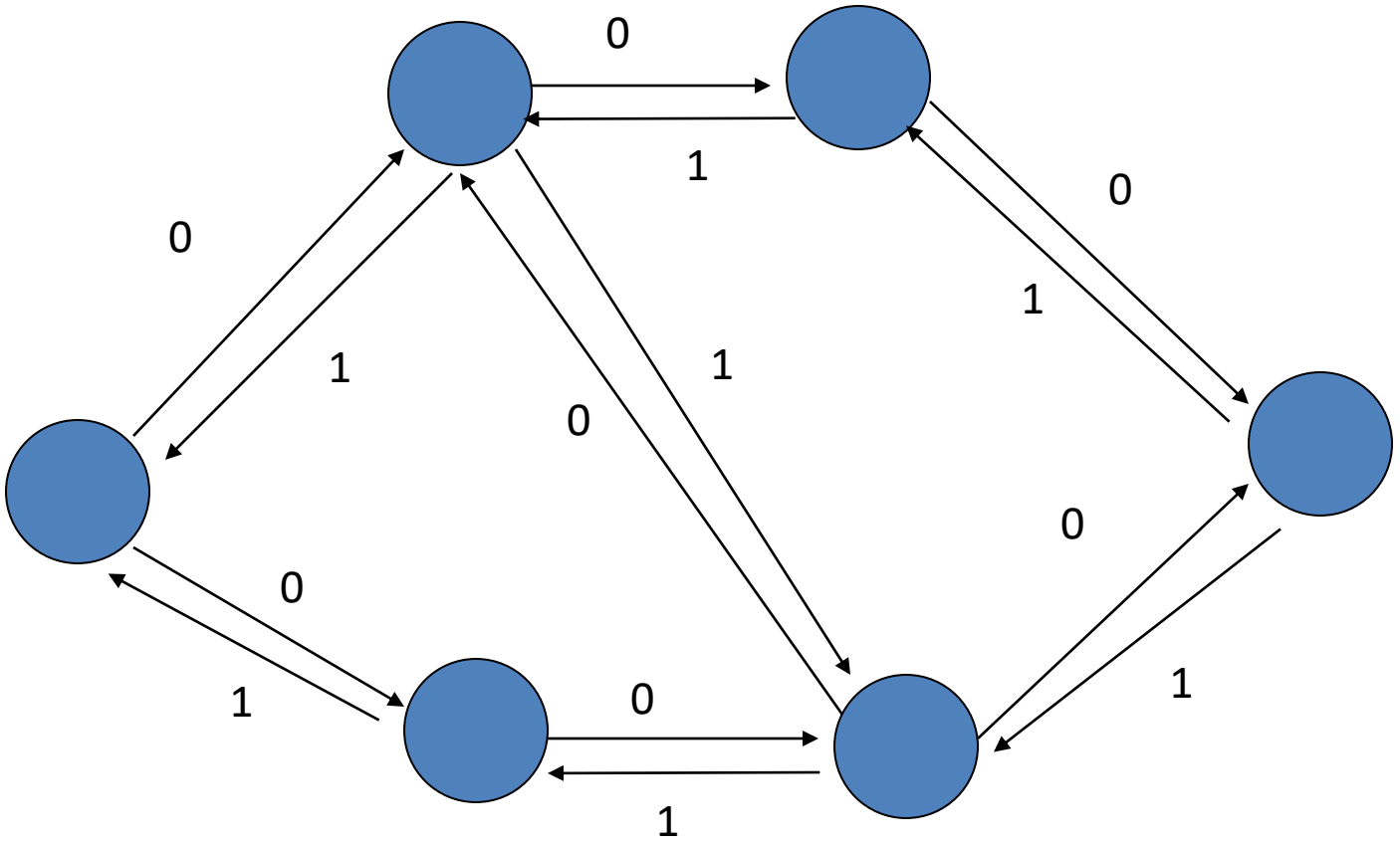
Example

- Update residual network



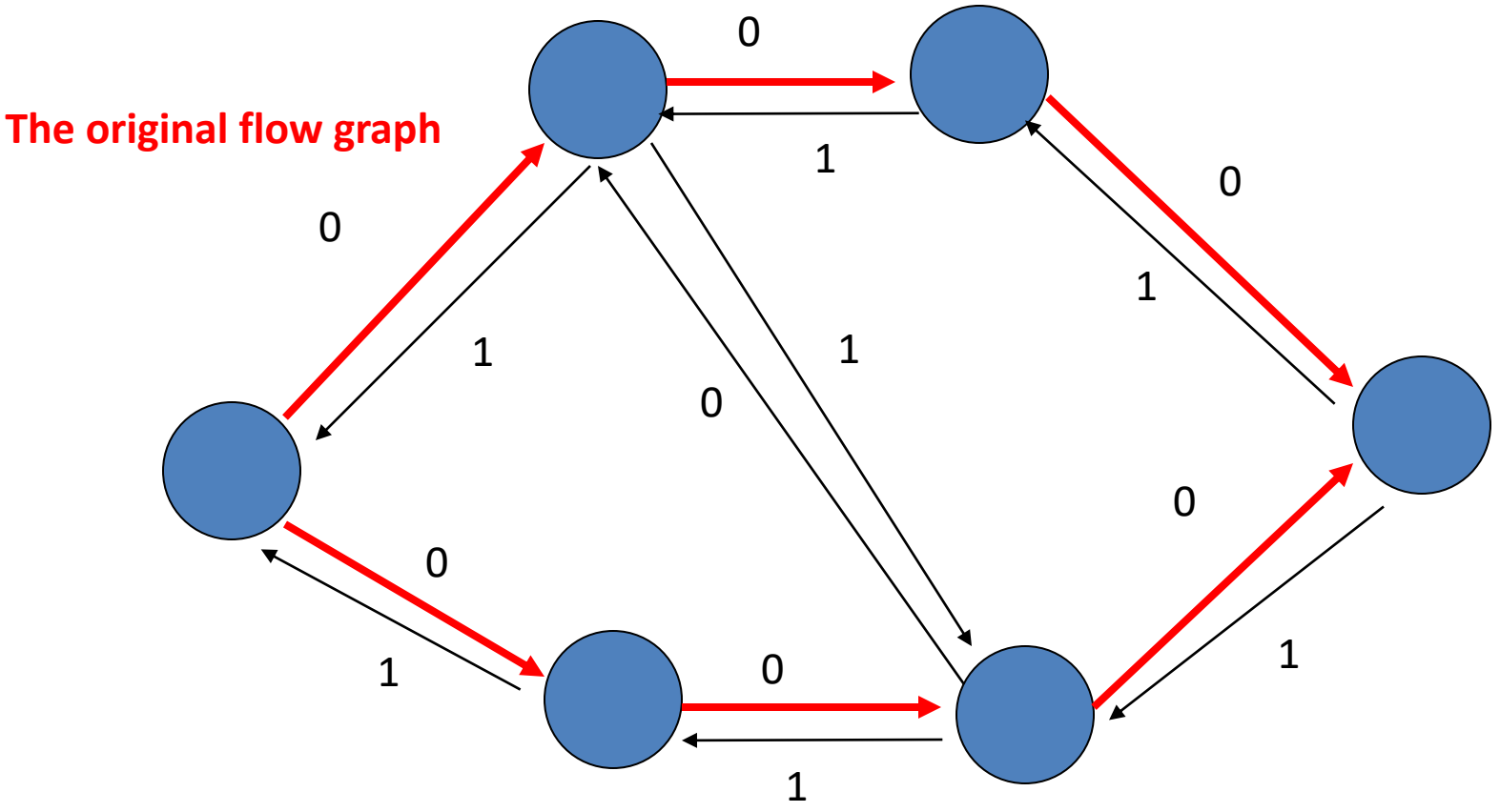
Example

- **Maximum flow: 2**



Example

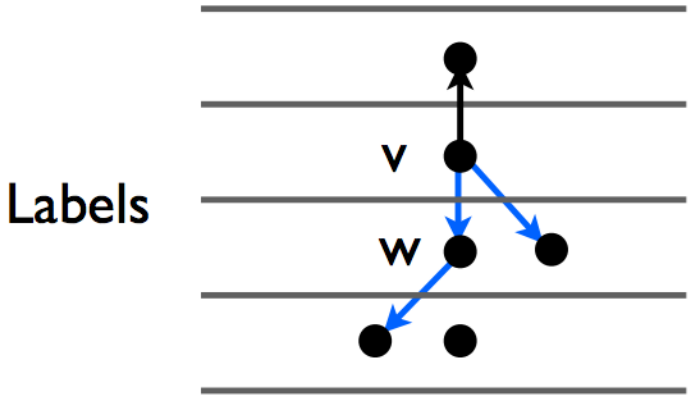
- Residual network gives us a way to circle flow back



Push-Relabel Algorithm

FORD-FULKERSON-FRAMEWORK(G, s, t)
initialize flow *f* to *0*
while there exists an *augmenting* path *p*
do *augment* flow *f* along *p*
return *f*

- **Main idea**
  - Don't want complexity to depend on the max flow value
  - Each node has a label h[v] of a potential (or height)
  - Route flow from high to low potential
- **We augment a flow "downhill" (blue edge)**
  - h[u] → h[v] if there is "flowable" units from u



Push-Relabel Algorithm Terminology

Preflow: A function f: E $\rightarrow$ R is a preflow if:

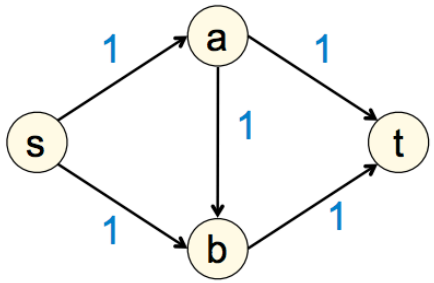
1. **Capacity Constraints:** 0 <= f(e) <= c(e)
2. Instead of conservation constraints:

$$\sum_{e \text{ into } v} f(e) - \sum_{e \text{ out of } v} f(e) \geq 0$$

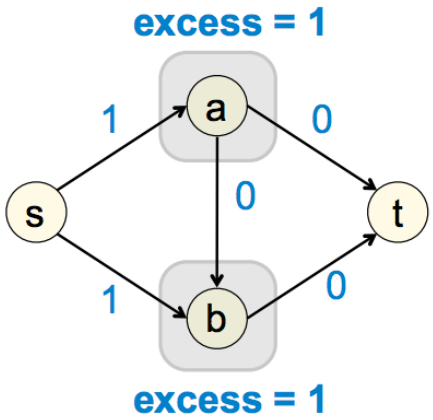
Excess(v) = $\sum_{e \text{ into } v} f(e) - \sum_{e \text{ out of } v} f(e)$

Example

G

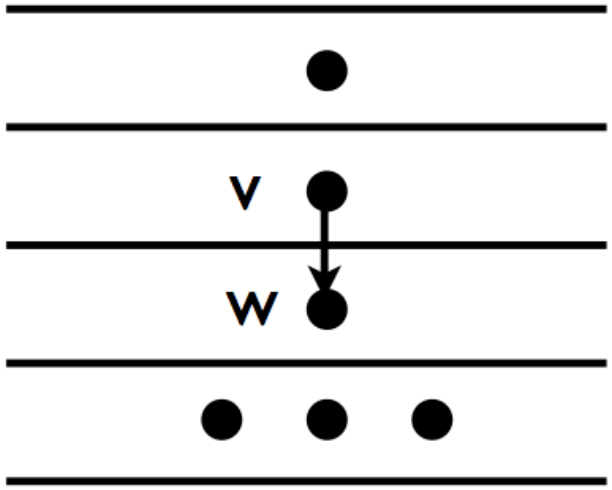


f



Iterate Two Operations

- **Push(v, w): applies if excess(v)>0, h(w)<h(v)**
  - f = min(excess(v), c_f(v, w))
  - Add f to f(v, w) and update residual network
- **Relabel(v): applies if excess(v)>0 and no h(w)<h(v)**
  - Increase h(v) by 1



Algorithm (V. Goldberg and R. Tarjan)

Start with labeling: $h(s) = n, h(t) = 0, h(v) = 0$, for all other v
Start with preflow f: $f(e) = c(e)$ for $e = (s, v)$, $f(e) = 0$, for all other edges e

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f such that push(v, w) can be applied
    Push(v, w)
  Else
    Relabel(v)

Push(v, w): Applies if excess(v) > 0, $h(w) < h(v)$, (v, w) in E_f
  $q = \min(\text{excess}(v), c_f(v,w))$
  Add q to f(v, w)

Relabel(v): Applies if excess(v) > 0, for all w s.t (v, w) in E_f, $h(w) >= h(v)$
  Increase h(v) by 1

Two Giants …

Andrew V. Goldberg

From Wikipedia, the free encyclopedia

Andrew Vladislav Goldberg (born 1960) is an American computer scientist working primarily on design, analysis, and experimental evaluation of algorithms. He also worked on mechanism design, computer systems, and complexity theory.[2] Currently he is a Senior Principal Scientist at Amazon.com.

Contents [hide]

1 Education and career
2 Career and research
  2.1 Selected publications
  2.2 Awards and honors
3 References

Education and career [edit]

Goldberg did his undergraduate studies at the Massachusetts Institute of Technology, graduating in 1982. After earning a master's degree at the University of California, Berkeley, he returned to MIT, finishing his doctorate there in 1987 with a thesis on the *Efficient graph algorithms for sequential and parallel computers*[3] supervised by Charles E. Leiserson.[G87][1]

‹ The template *Infobox scientist* is being considered for merging. ›

Andrew Goldberg

Robert Tarjan

From Wikipedia, the free encyclopedia

Robert Endre Tarjan (born April 30, 1948) is an American computer scientist and mathematician. He is the discoverer of several graph algorithms, including Tarjan's off-line lowest common ancestors algorithm, and co-inventor of both splay trees and Fibonacci heaps. Tarjan is currently the James S. McDonnell Distinguished University Professor of Computer Science at Princeton University, and the Chief Scientist at Intertrust Technologies Corporation.[1]

Contents [hide]

1 Early life and education
2 Computer science career
  2.1 Algorithms and data structures
3 Awards
4 Patents
5 Notes
6 References
7 External links

Early life and education [edit]

He was born in Pomona, California. His father was a child psychiatrist specializing in mental retardation, and ran a state hospital.[2] As a child, Tarjan read a lot of science fiction, and wanted to be an astronomer. He became interested in mathematics after reading Martin Gardner's mathematical games column in Scientific American. He became seriously interested in math in the eighth grade, thanks to a "very stimulating" teacher.[3]

While he was in high school, Tarjan got a job, where he worked IBM punch card collators. He first worked with real computers while studying astronomy at the Summer Science Program in 1964.[2]

Tarjan obtained a Bachelor's degree in mathematics from the California Institute of Technology in 1969. At Stanford University, he received his master's degree in computer science in 1971 and a Ph.D. in computer science (with a minor in mathematics) in 1972. At Stanford, he was supervised by

‹ The template *Infobox scientist* is being considered for merging. ›

Robert Endre Tarjan

Born	April 30, 1948 (age 71) Pomona, California
Citizenship	American
Alma mater	California Institute of Technology (BS) Stanford University (MS, PhD)
Known for	Algorithms and data structures
Awards	Paris Kanellakis Award (1999) Turing Award (1986)

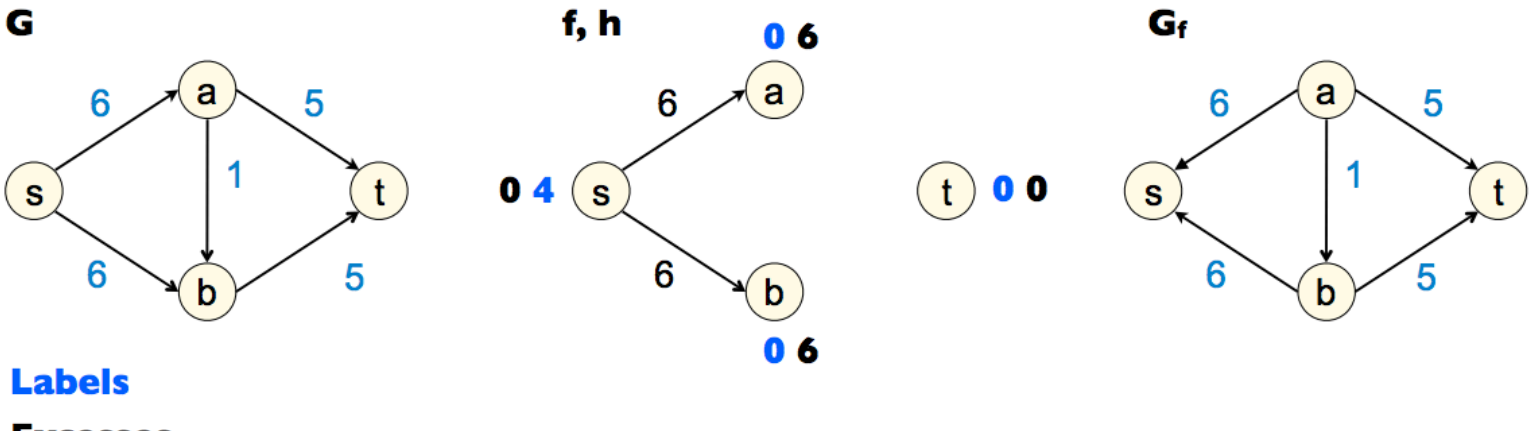
Example

Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1



Preflow from source and select a in iteration 1

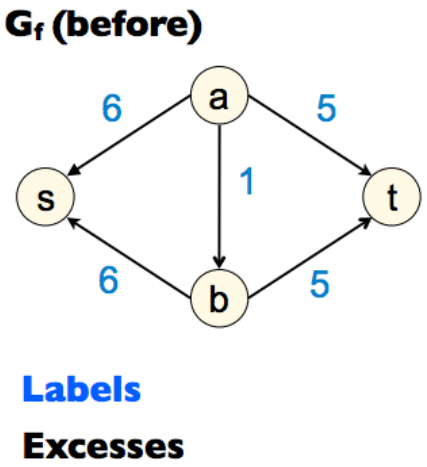
Example

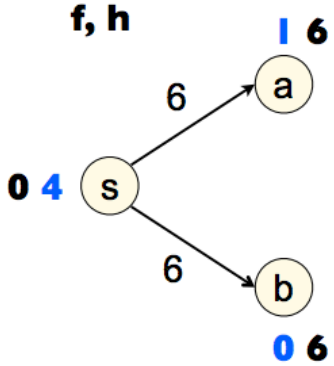
Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

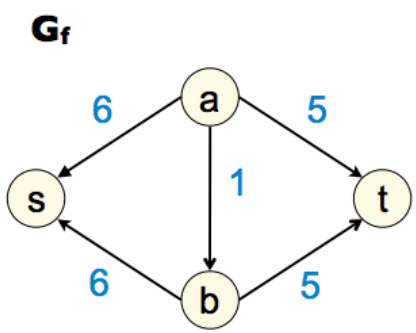
While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1







Relabel(a) to increase h[a] by one

Example

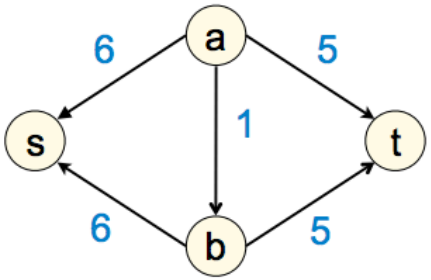
Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

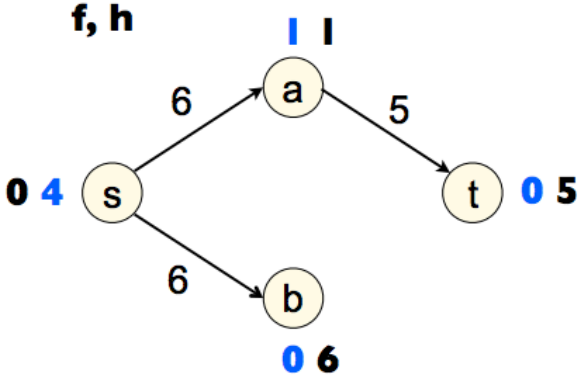
G_f (before)



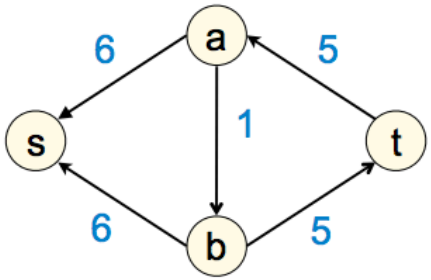
Labels

Excesses

f, h



G_f



Push(a, t) to add 5 flow along a→t

Example

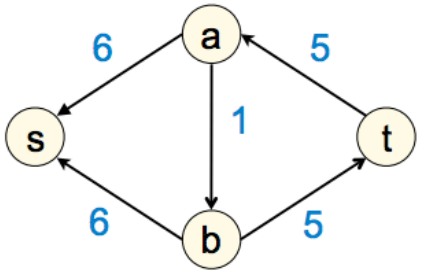
Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

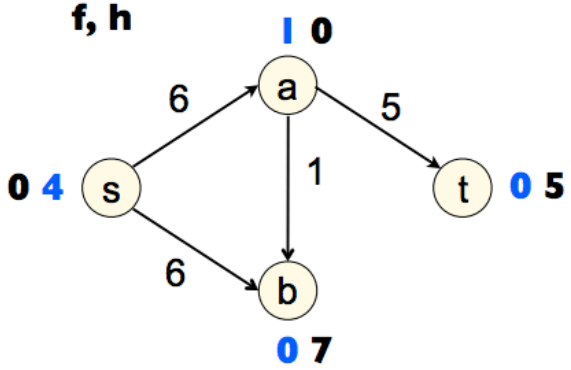
Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

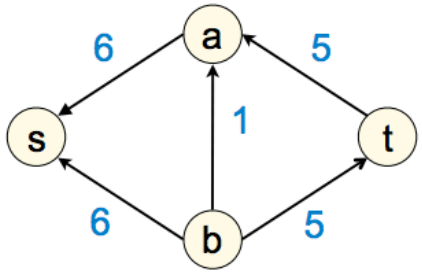
G_f (before)



f, h



G_f



Labels

Excesses

Push(a, b) to add 1 flow along a→b

Example

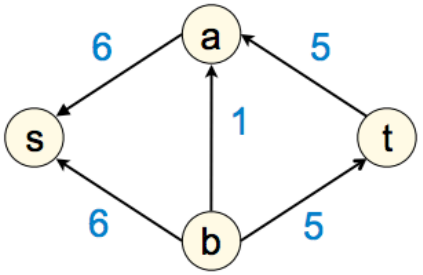
Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

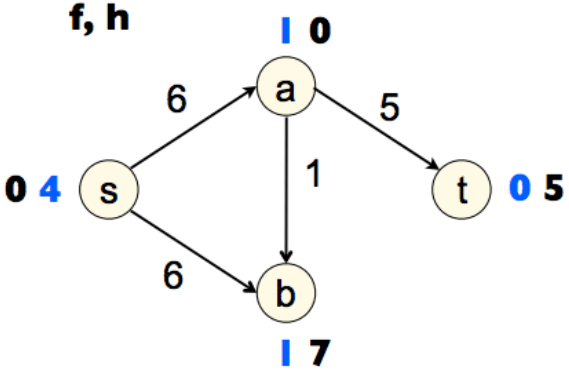
G_f (before)



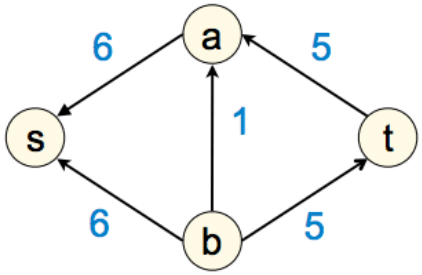
Labels

Excesses

f, h



G_f



Relabel(b) to increase h[b] by 1

Example

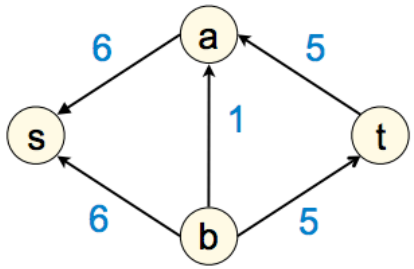
Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

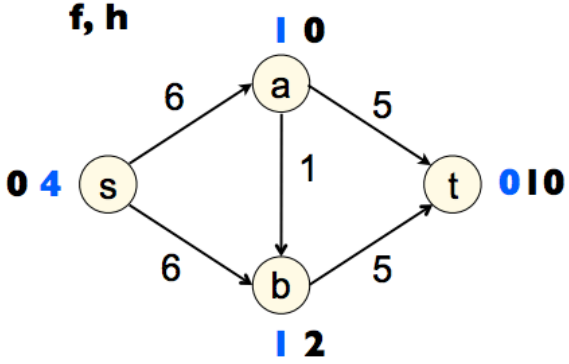
G_f (before)



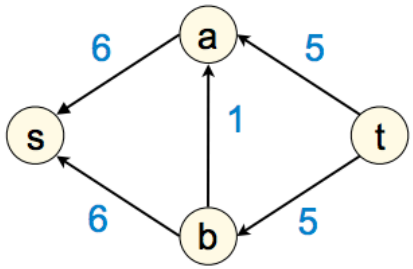
Labels

Excesses

f, h



G_f



Push(b, t) to increase flow by 5 along b→t

Example

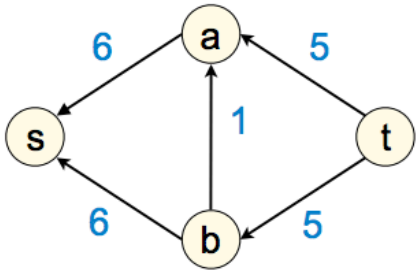
Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

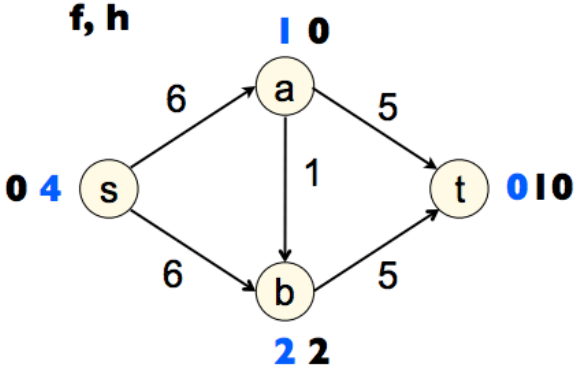
G_f (before)



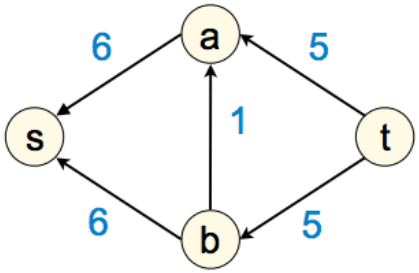
Labels

Excesses

f, h



G_f



Relabel(b) to increase h[b] by 1

Example

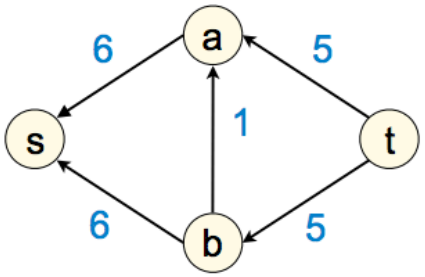
Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

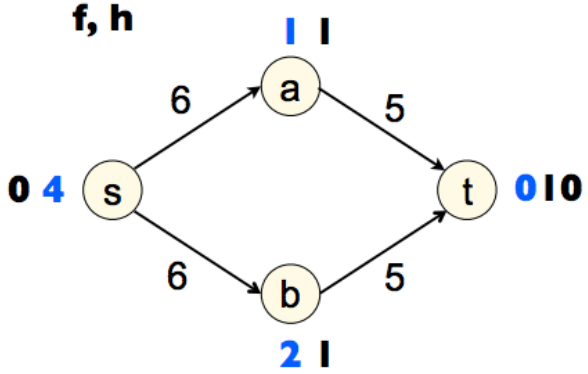
G_f (before)



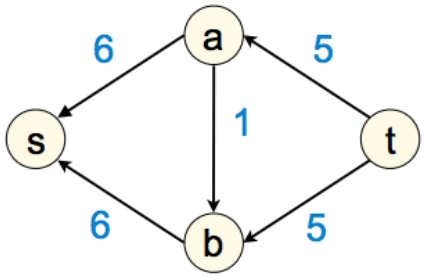
Labels

Excesses

f, h



G_f



Push(b, a) to increase flow along b→a by 1

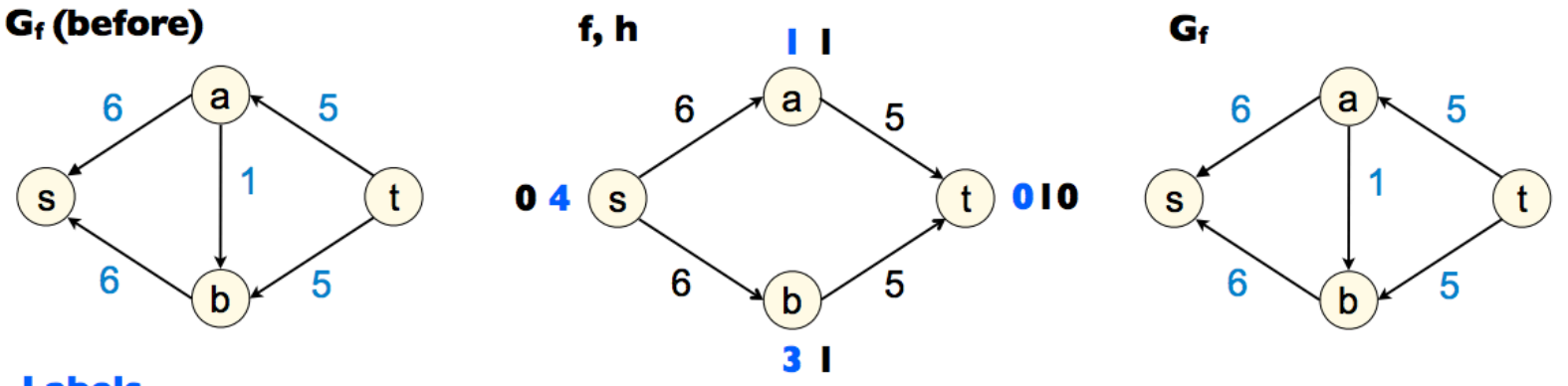
Example

Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1



Relabel(b) to increase h[b] by 1

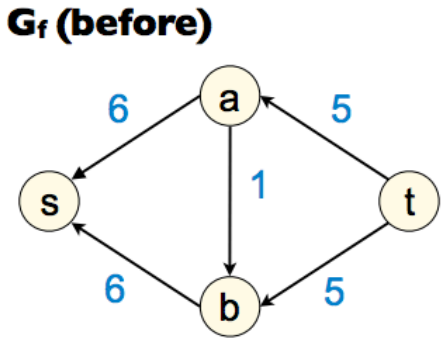
Example

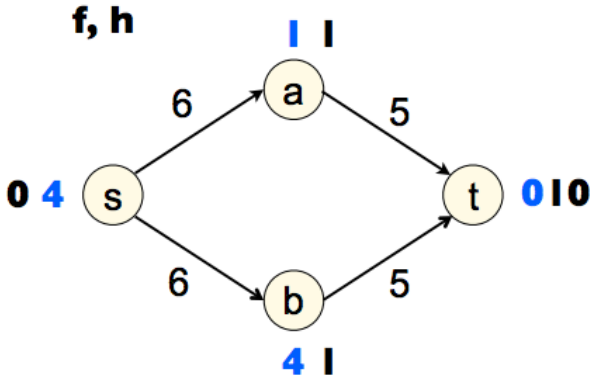
Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

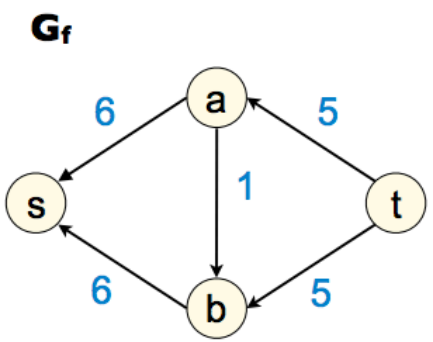
While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1







Labels

Excesses

Relabel(b) to increase h[b] by 1

Example

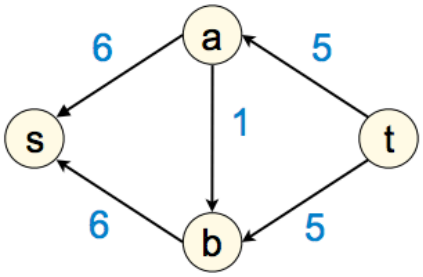
Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

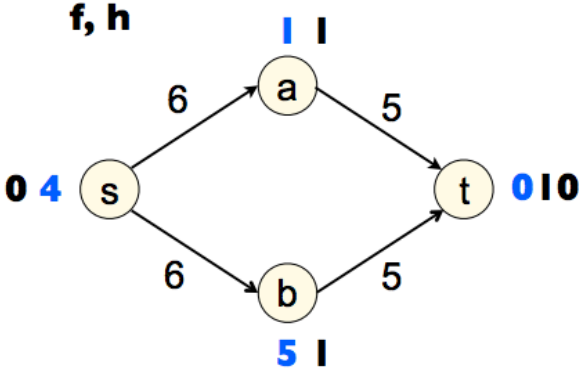
G_f (before)



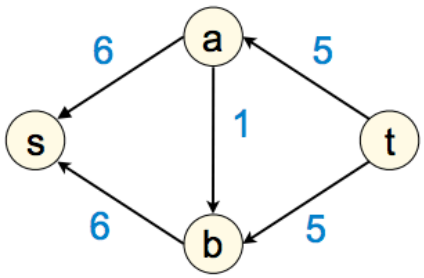
Labels

Excesses

f, h



G_f



Relabel(b) to increase h[b] by 1

Example

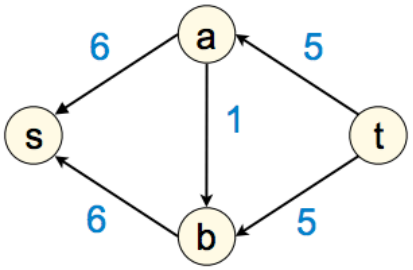
Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all
w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

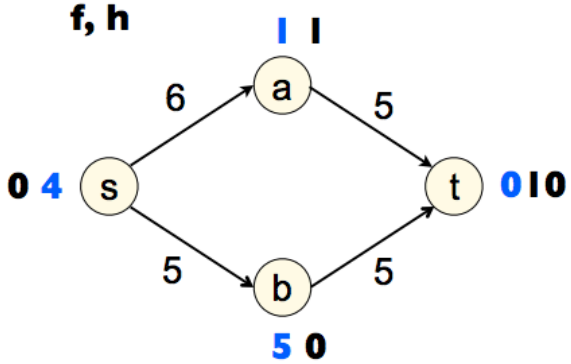
G_f (before)



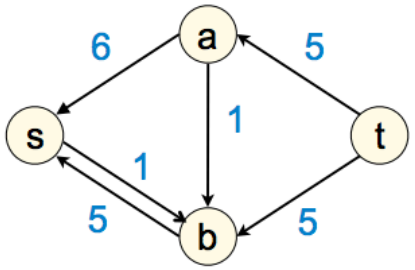
Labels

Excesses

f, h



G_f



Push(b, s) to increase the flow along b→s by 1

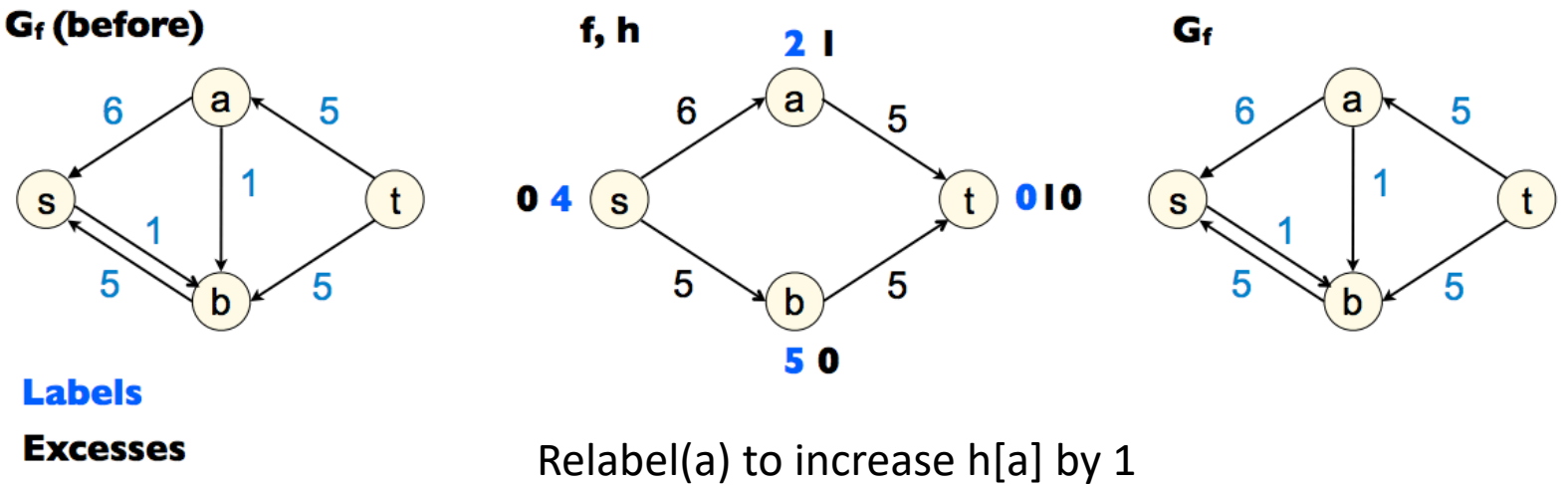
Example

Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1



Relabel(a) to increase h[a] by 1

Example

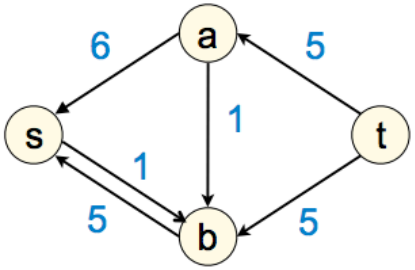
Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

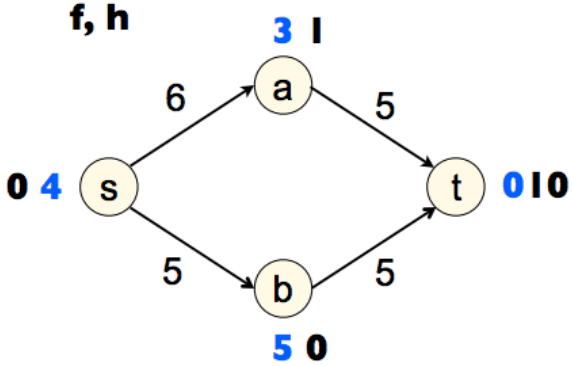
G_f (before)



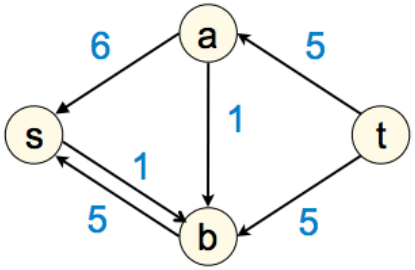
Labels

Excesses

f, h



G_f



Relabel(a) to increase h[a] by 1

Example

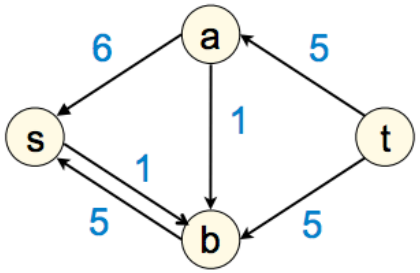
Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

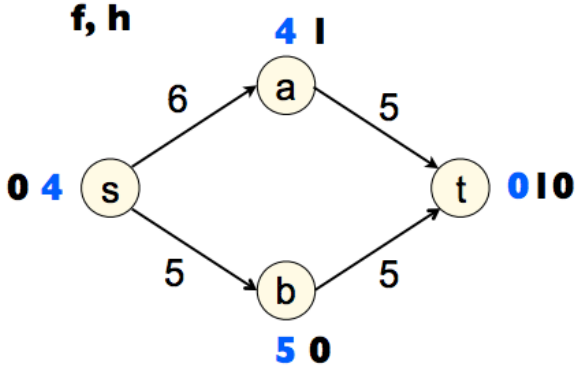
G_f (before)



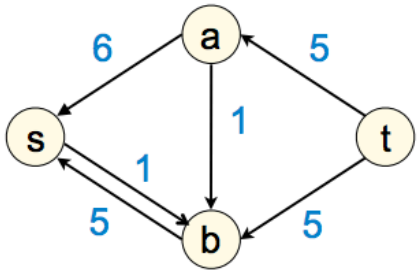
Labels

Excesses

f, h



G_f



Relabel(a) to increase h[a] by 1

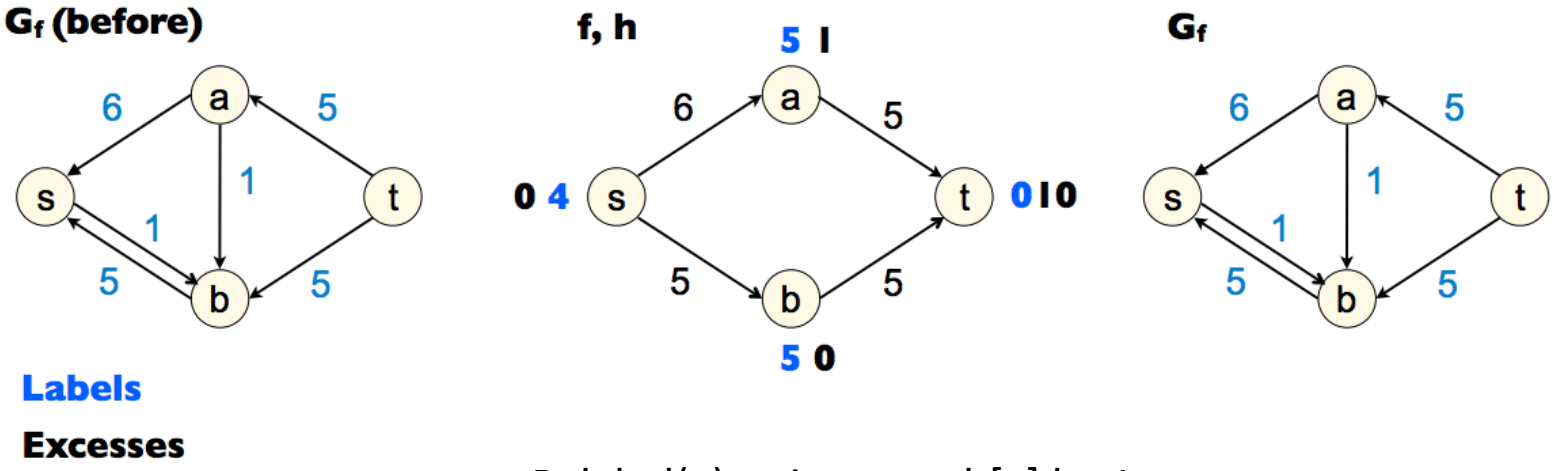
Example

Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1



Relabel(a) to increase h[a] by 1

Example

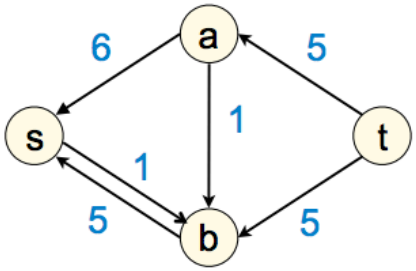
Start with labeling:h(s) = n,h(t) = 0,h(v) = 0, for other v
Start with preflow f: f(e) = c(e) for e = (s, v), f(e) = 0, ow

While there is a node (other than t) with positive excess
  Pick a node v with excess(v) > 0
  If there is an edge (v, w) in E_f s. t. push(v, w) applies
    Push(v, w)
  Else
    Relabel(v)

Push(v, w):
Applies if excess(v) > 0, h(w) < h(v)
  q = min(excess(v), c_f(v,w))
  Add q to f(v, w)

Relabel(v):
Applies if excess(v) > 0 and for all w s.t (v, w) in E_f, h(w) >= h(v)
  Increase h(v) by 1

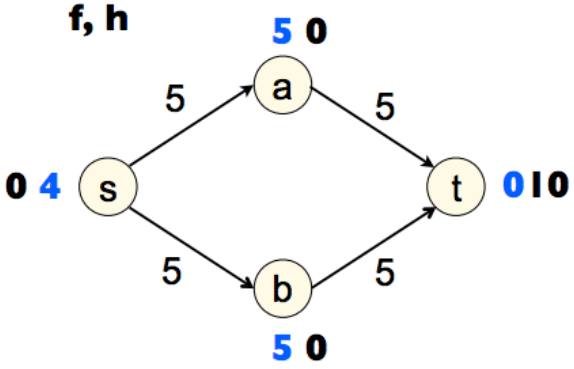
G_f (before)



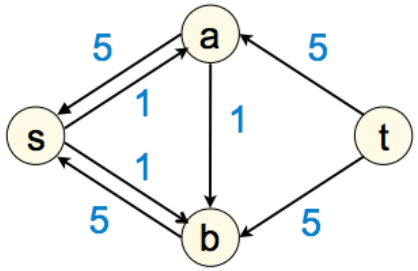
Labels

Excesses

f, h



G_f



Push(a, s) to increase the flow along a→s by 1

Time Complexity

- **How many push/relabel operations?**
  - Bound the maximum value of h(v) for any node v
  - Bound the number of relabel operations

Min-cost Flow

Goal: Build a cheapest network that satisfies the flow constraint

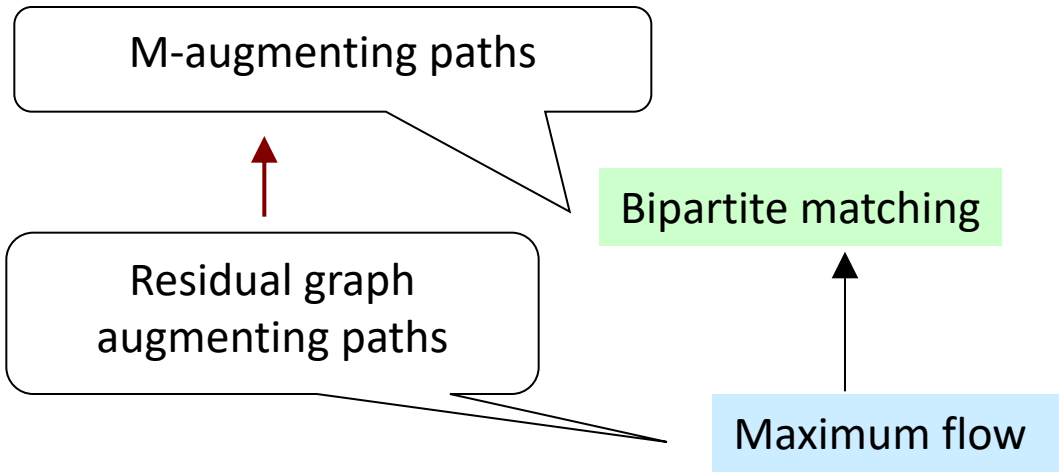
Input:

- A directed graph *G*
- A source vertex *s*
- A sink vertex *t*
- A capacity function *c* on the edges, i.e. *c: E->R*
- A cost function *w* on the edges, i.e. *w: E->R*

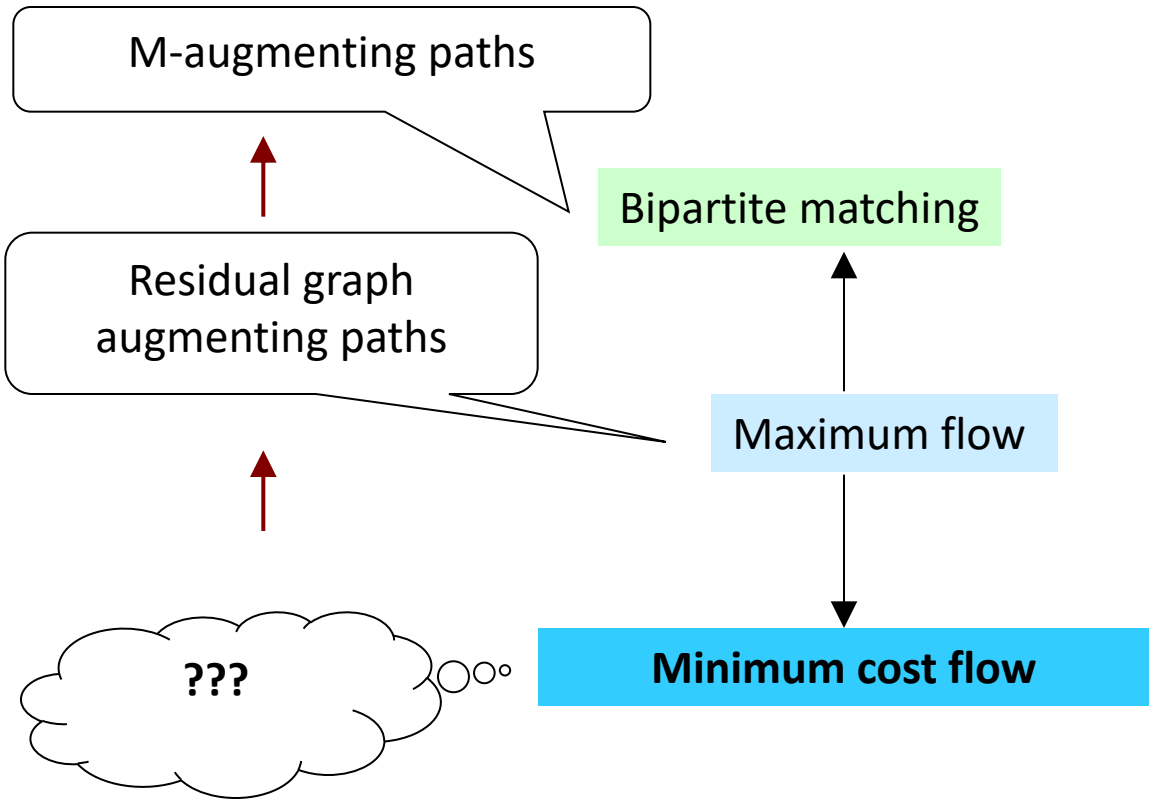
Output: a **maximum** *s-t* flow *f* which **minimizes** $\Sigma f(e)\, w(e)$

Concept: There can exist multiple maxflow solutions and we want to find the one with the minimum cost

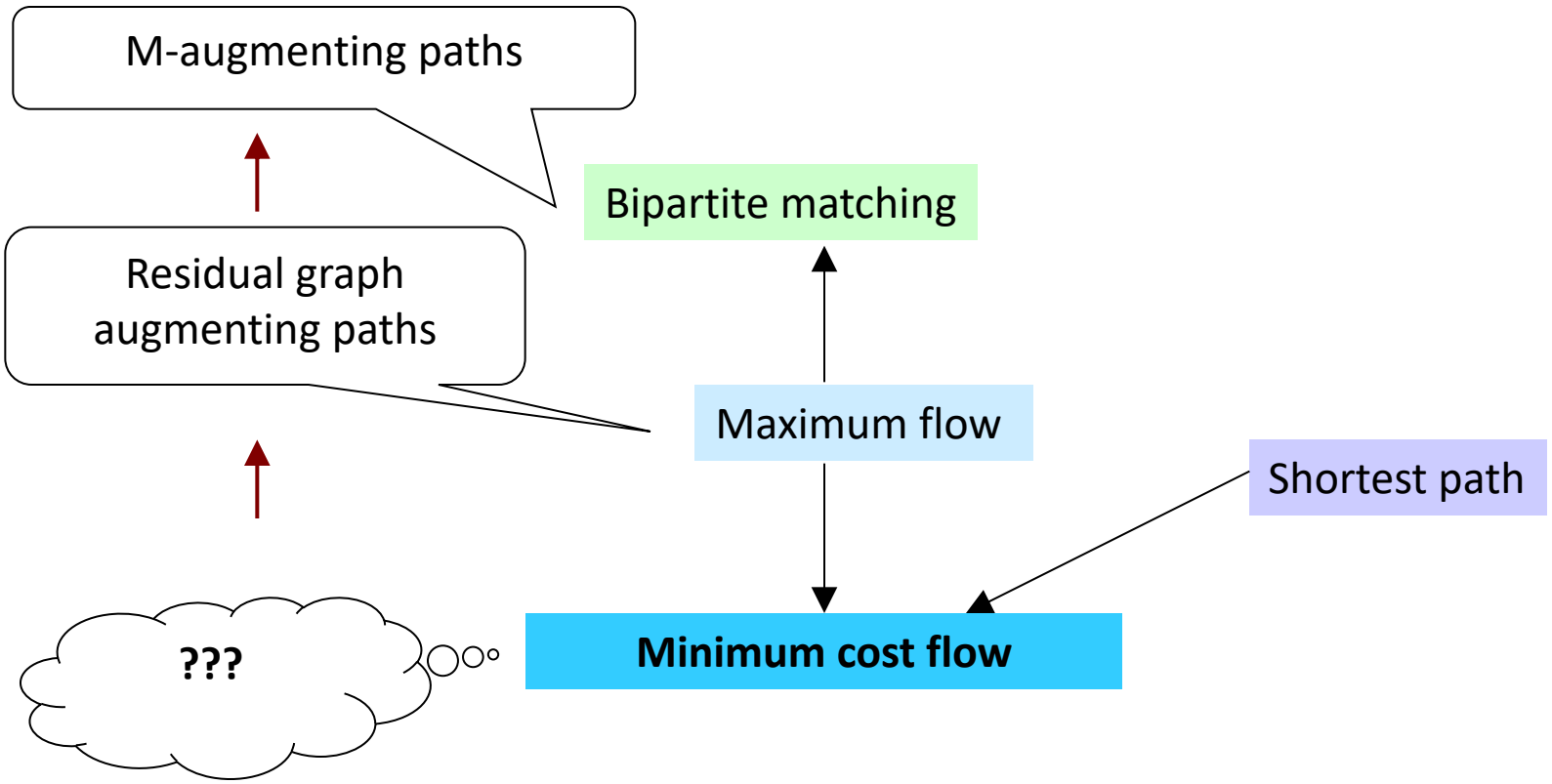
Problem Structure



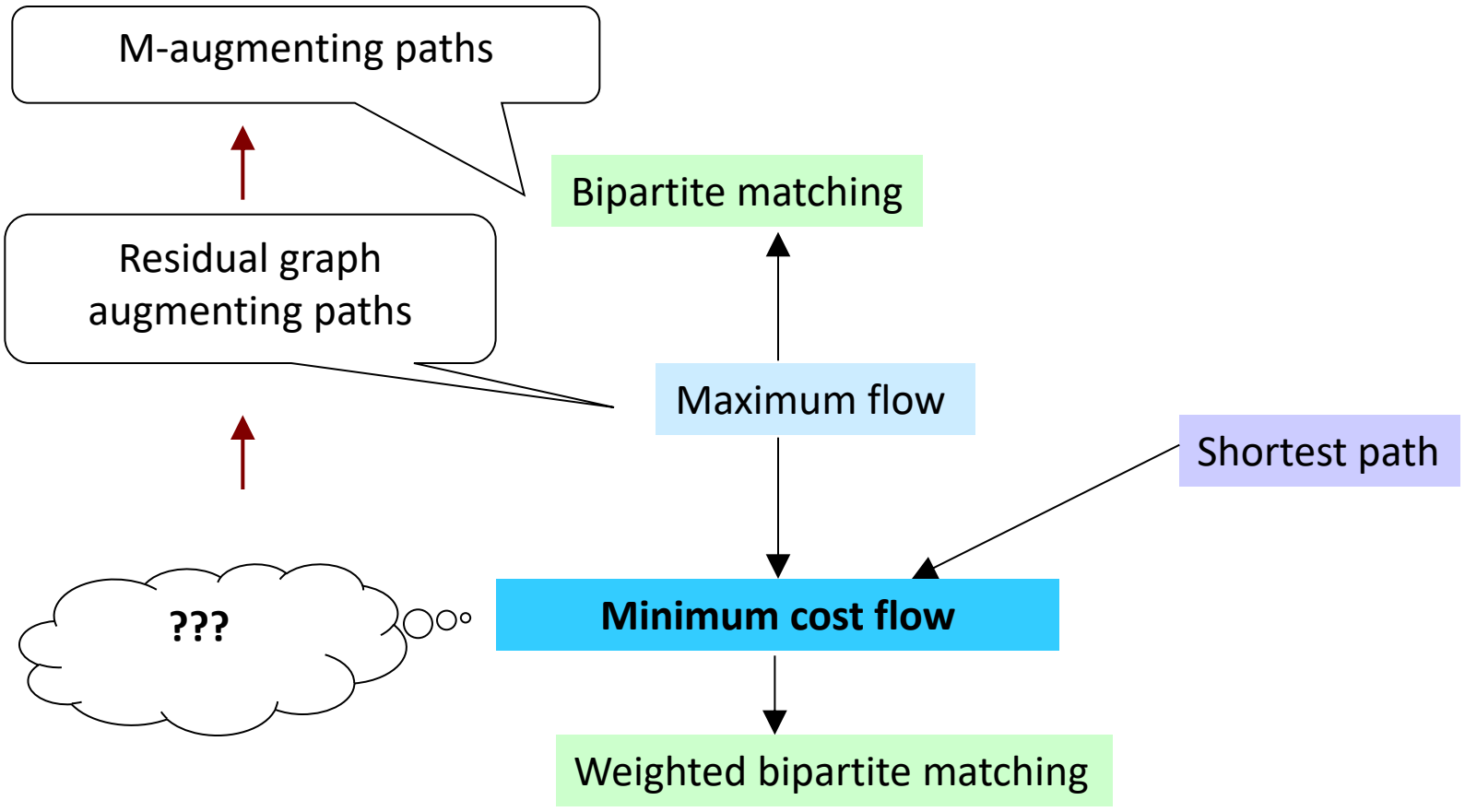
Problem Structure



Problem Structure



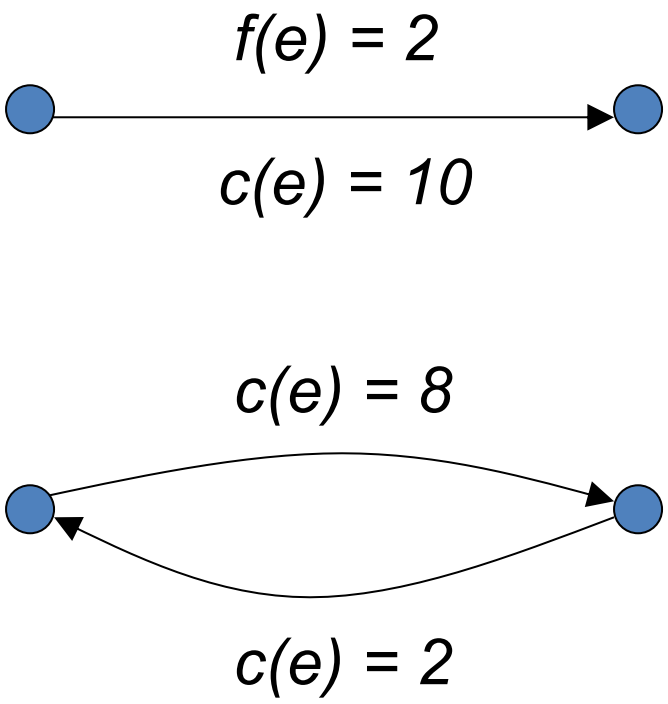
Problem Structure



Algorithm?

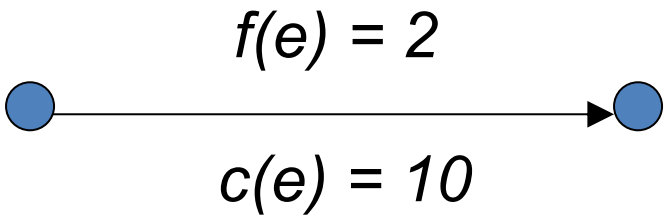
- **Two parameters: value and cost of the flow**
  - Value: maxflow value
  - Cost: the cost of the maxflow value
- **Each edge is associated with two values c and w**
  - c is the capacity and w is the cost per unit flow
- **What is the residual network structure?**

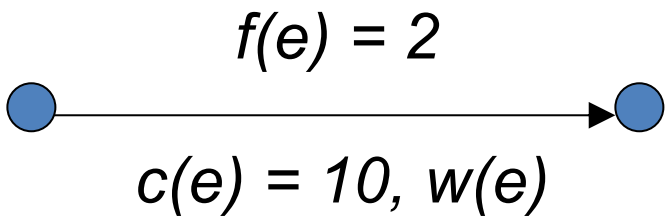
Residual Network

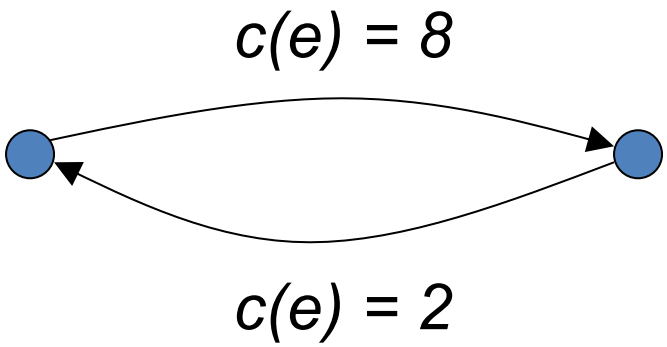


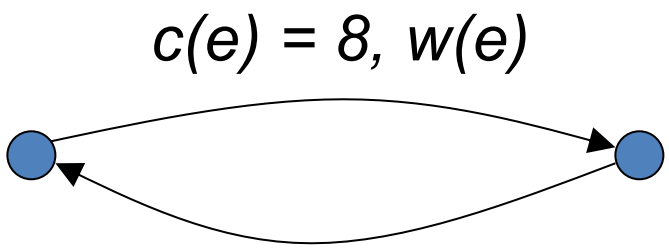
Max Flow

Residual Network



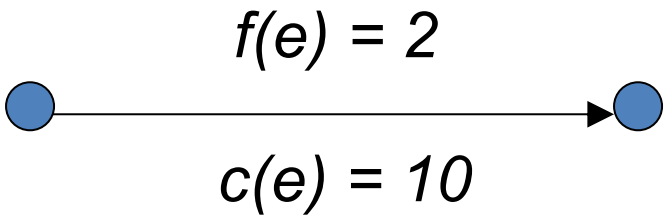


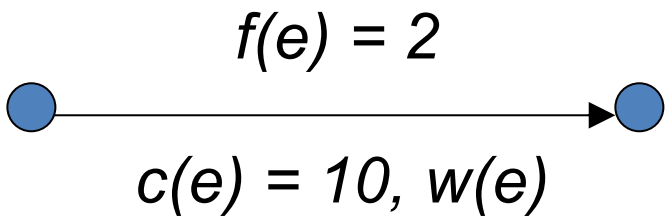


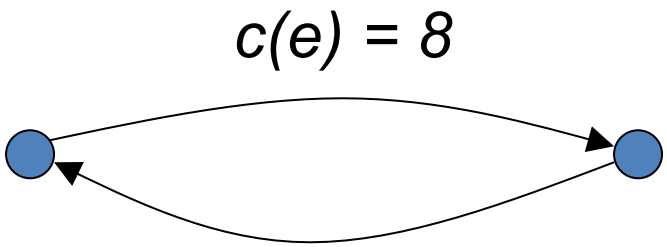


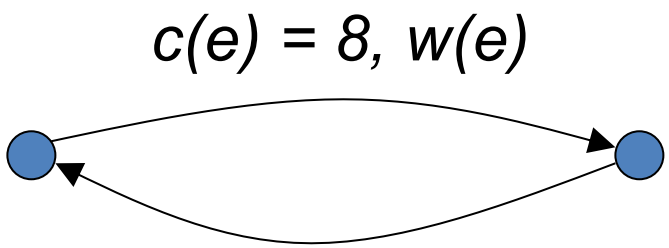
Max Flow

Residual Network







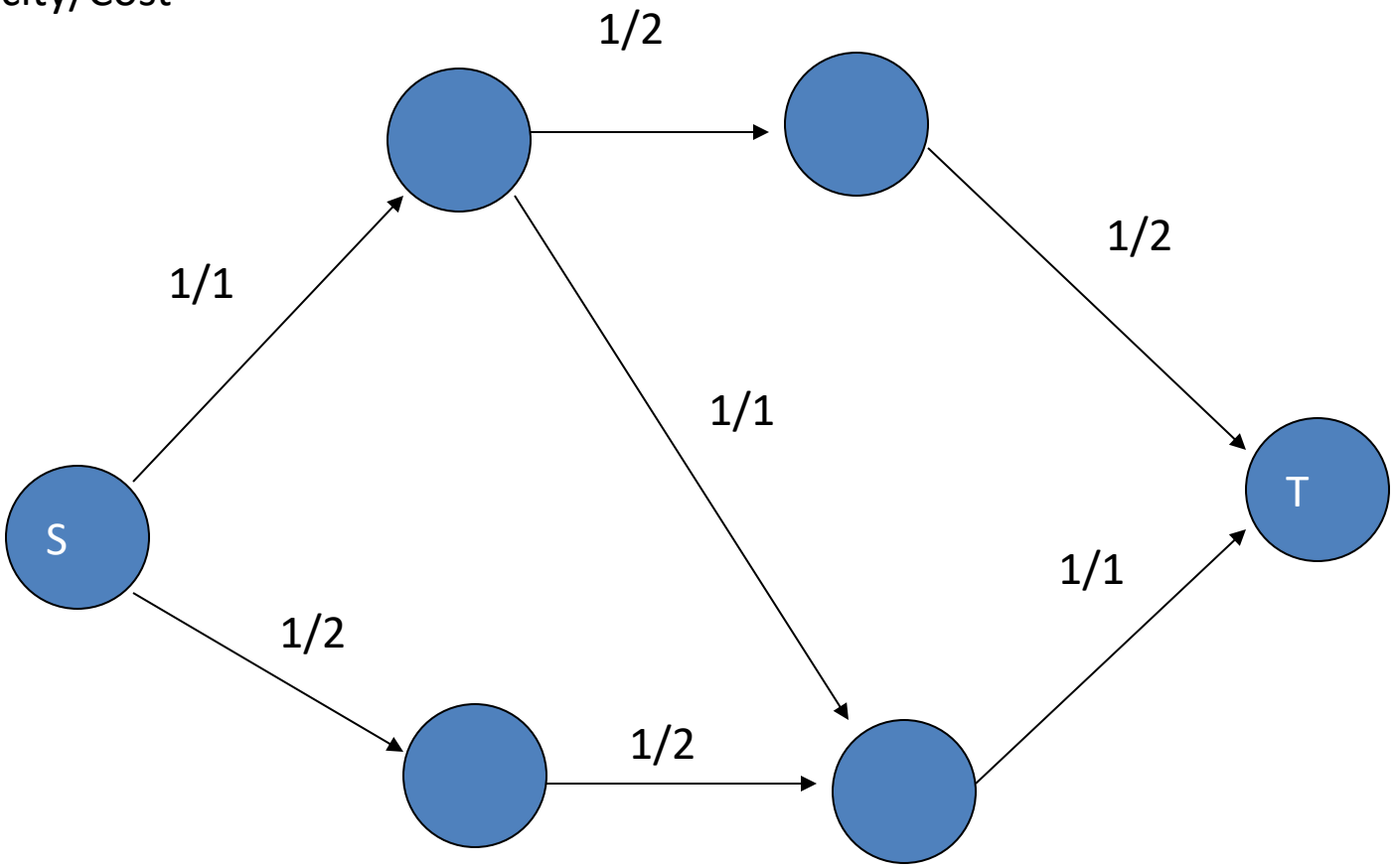


Max Flow

Min-cost Flow

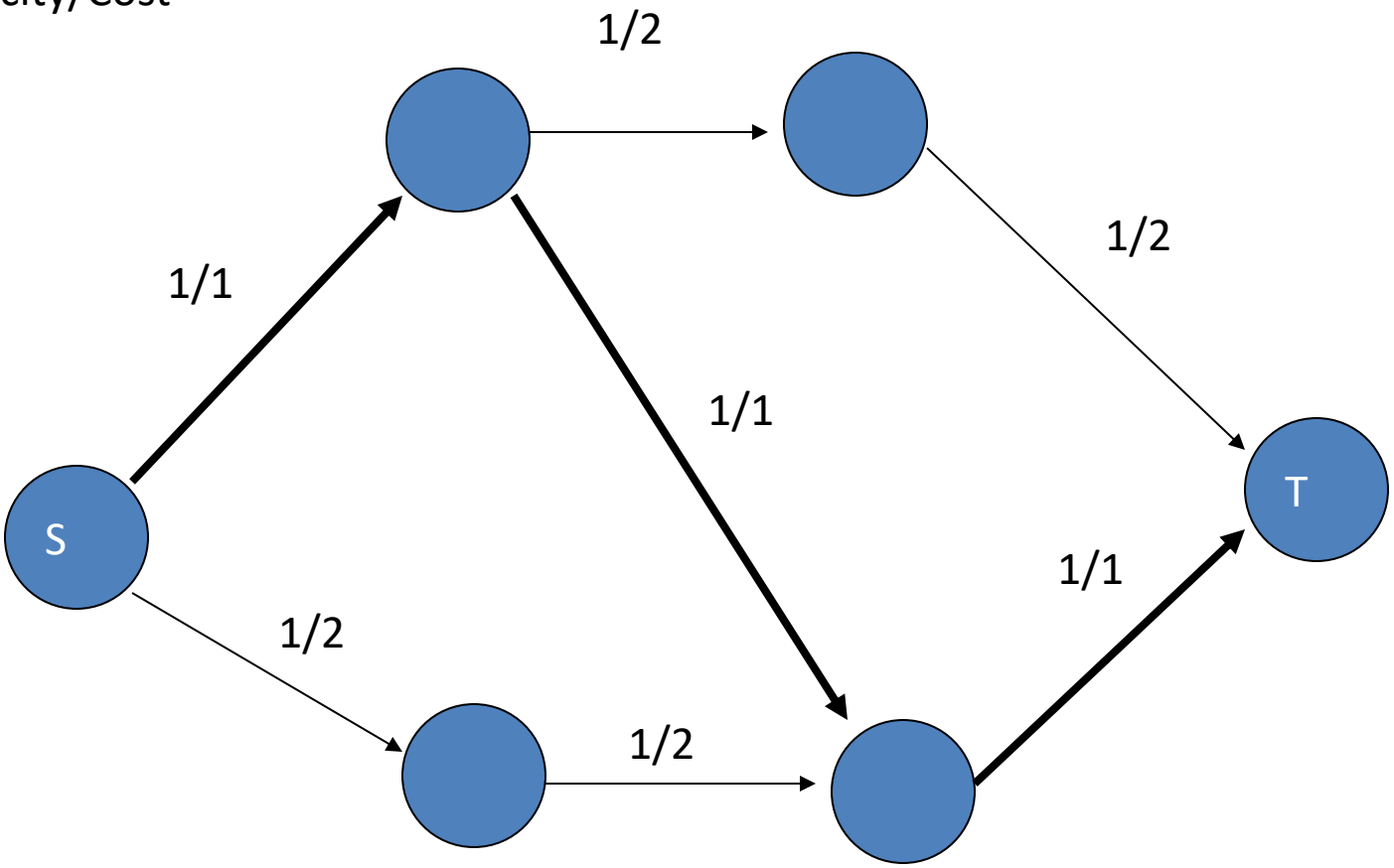
Example

Capacity/Cost



Example

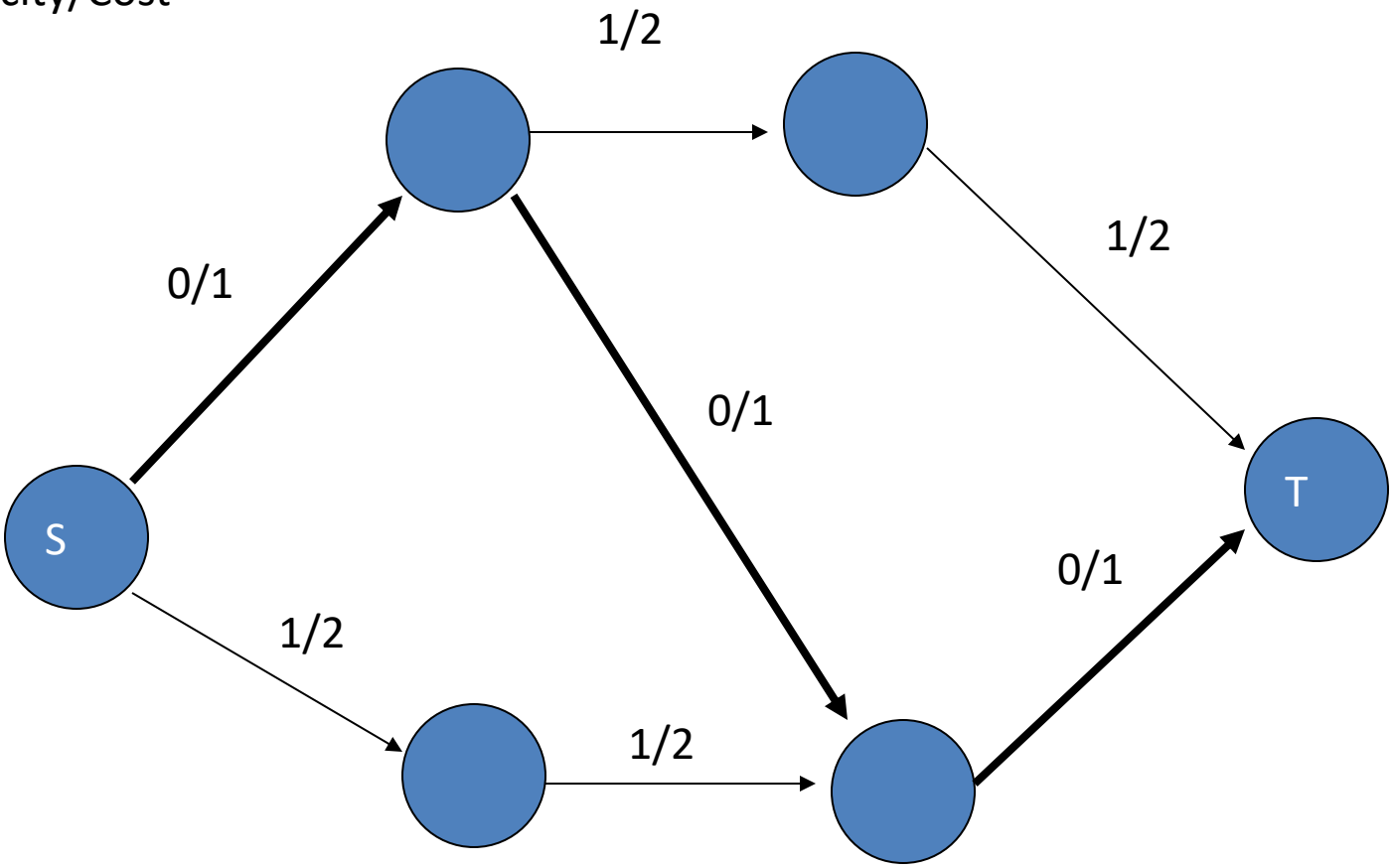
Capacity/Cost



Find the shortest path

Example

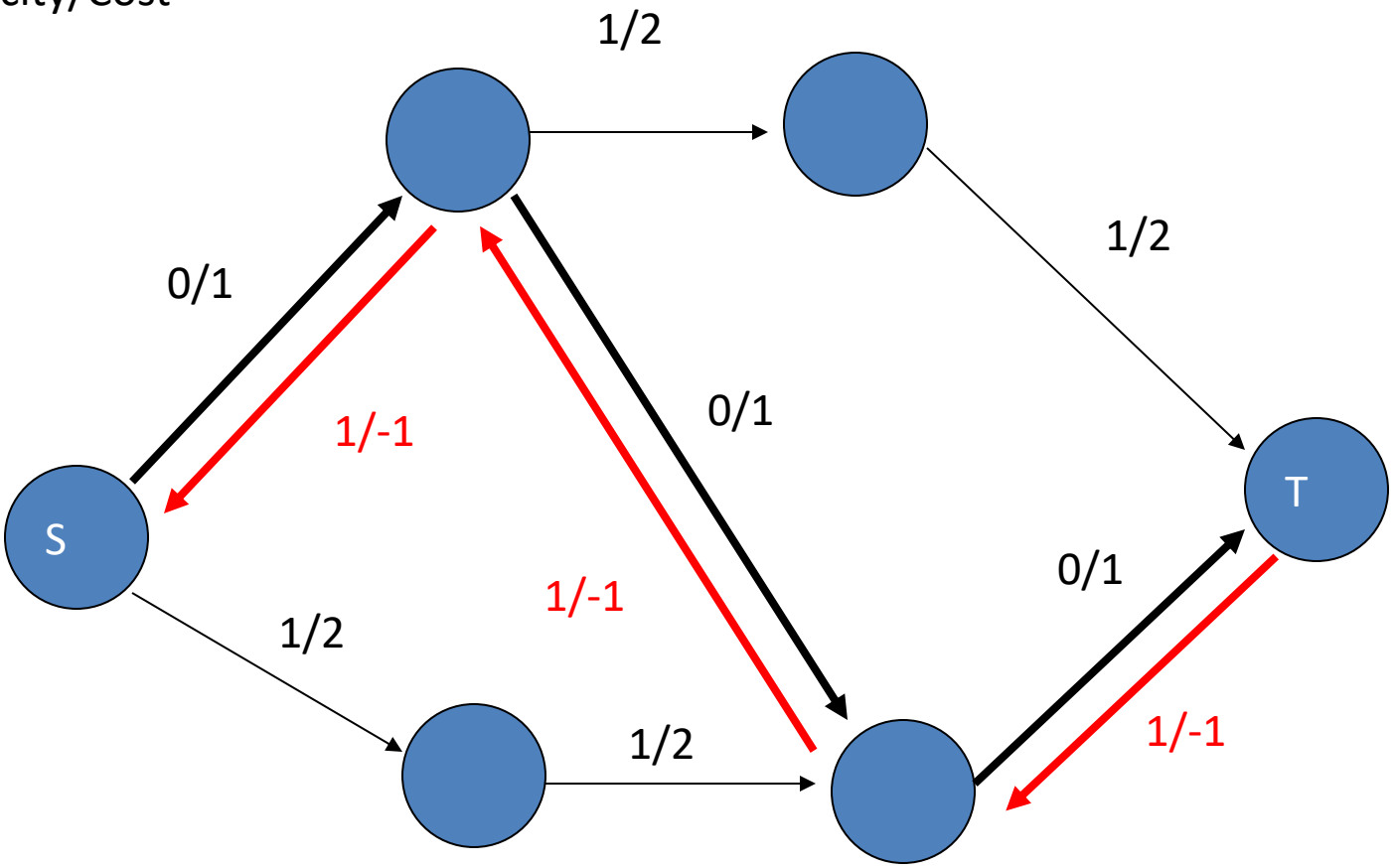
Capacity/Cost



Flow=1, cost=3

Example

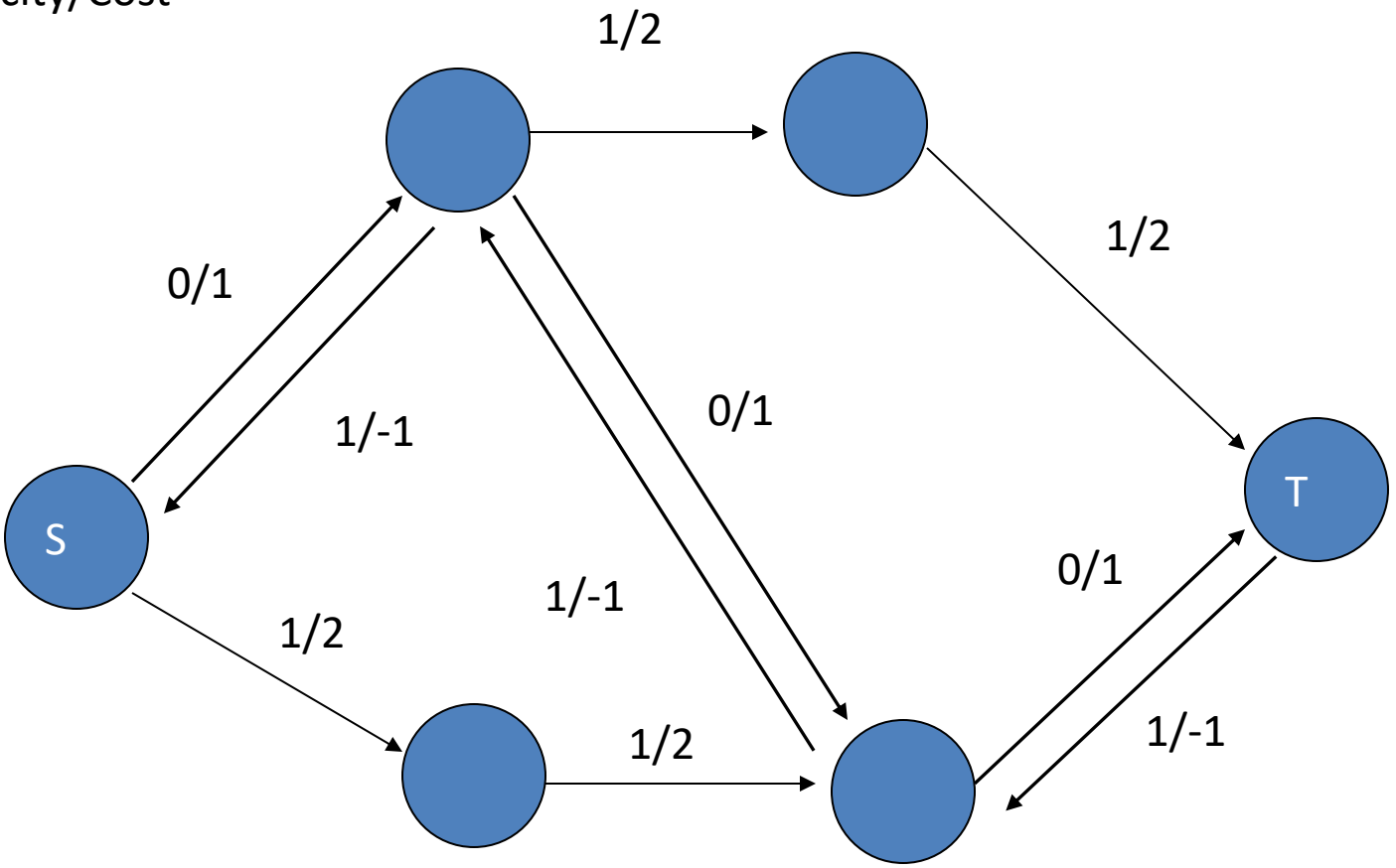
Capacity/Cost



Flow=1, cost=3

Example

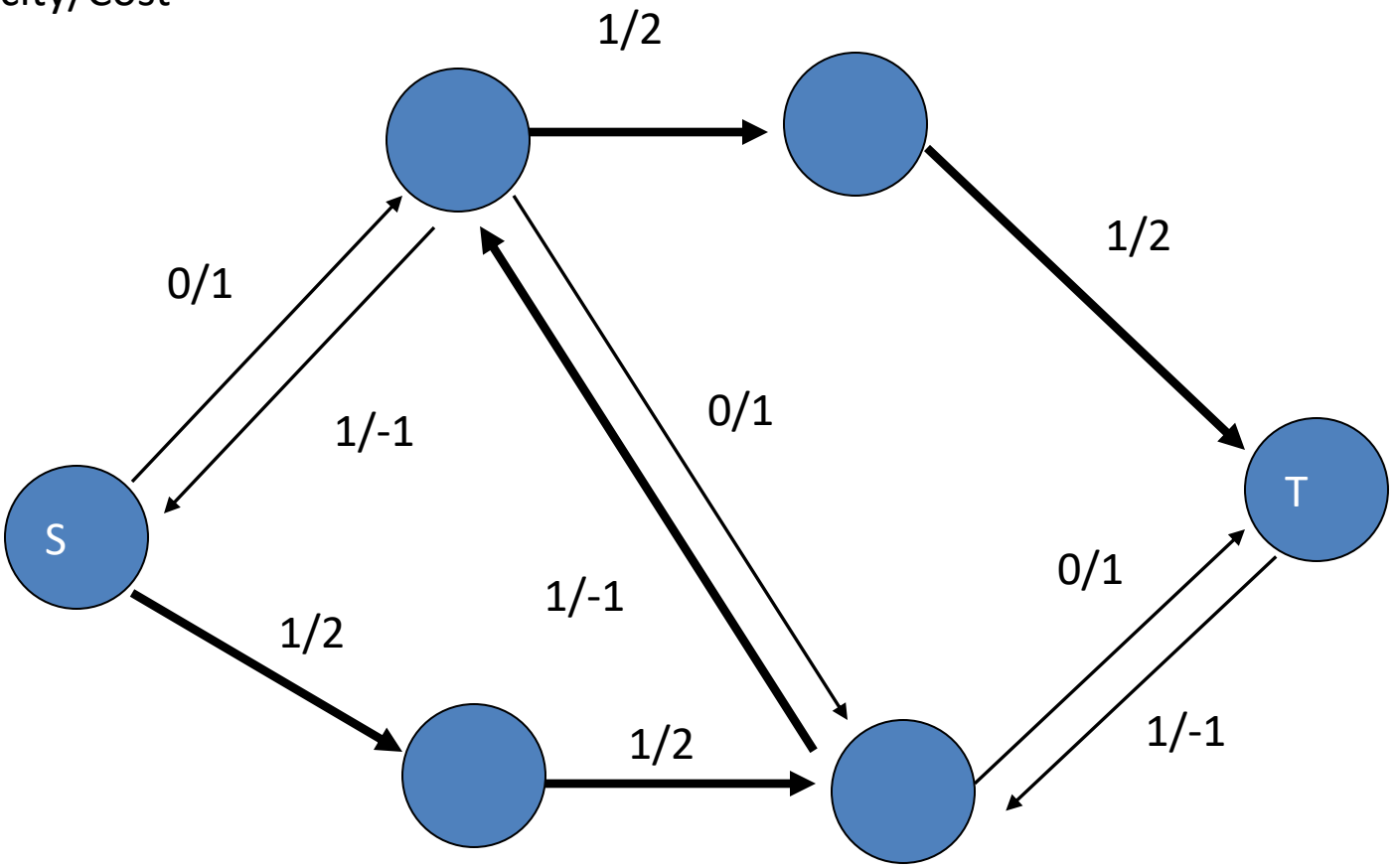
Capacity/Cost



Find another shortest path

Example

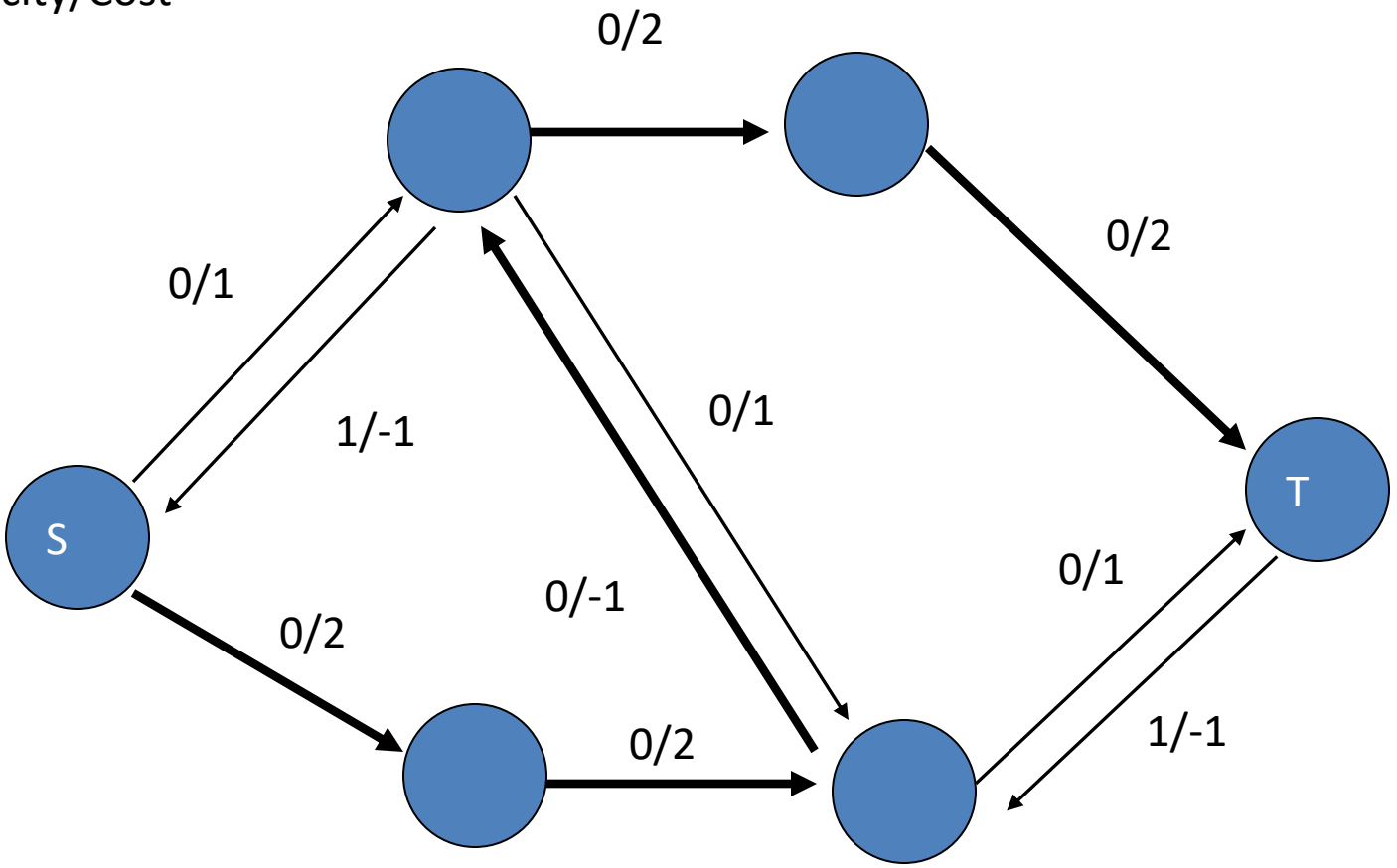
Capacity/Cost



Flow=1, cost = 2+2-1+2+2=7

Example

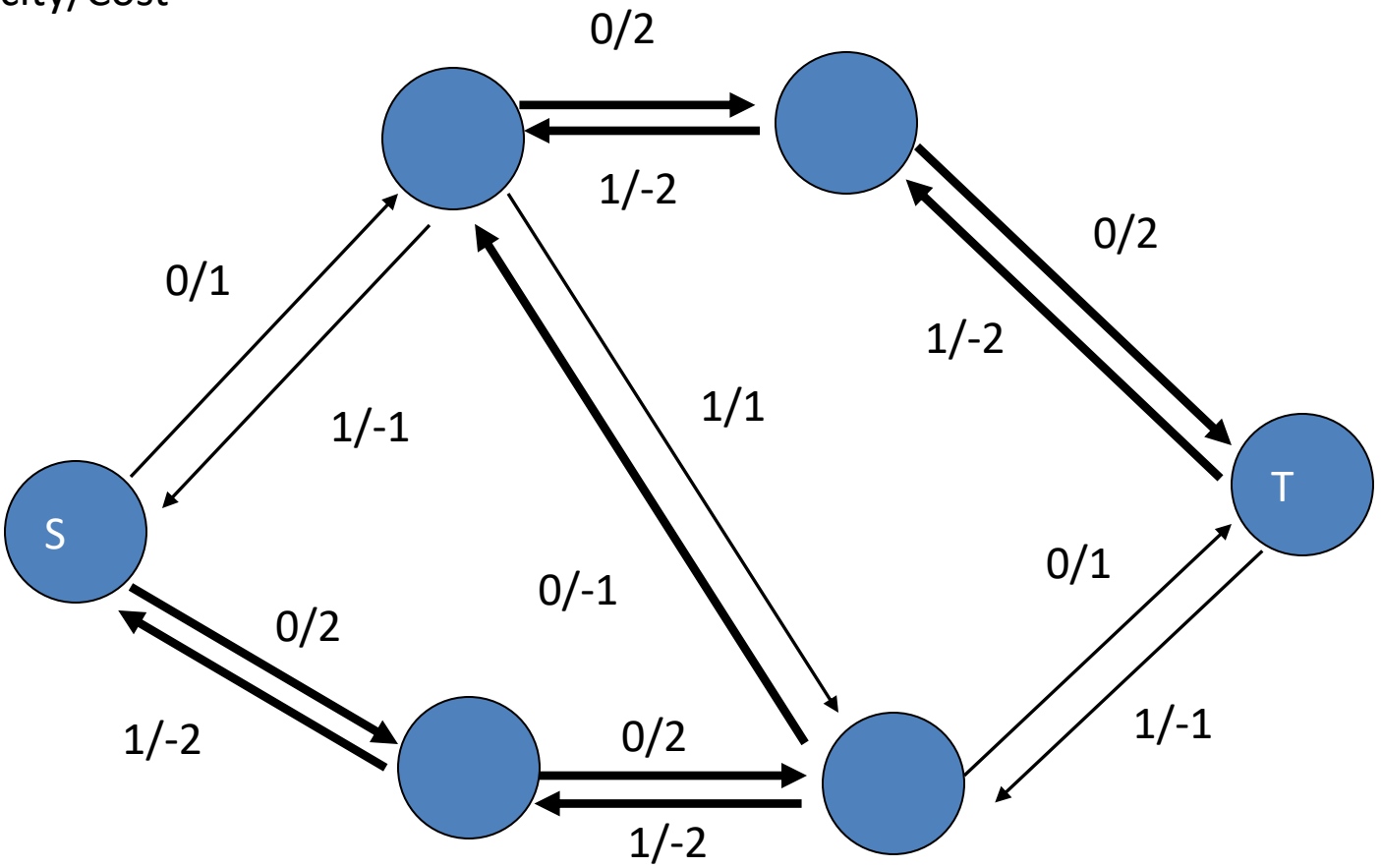
Capacity/Cost



Flow=1, cost = 2+2-1+2+2=7

Example

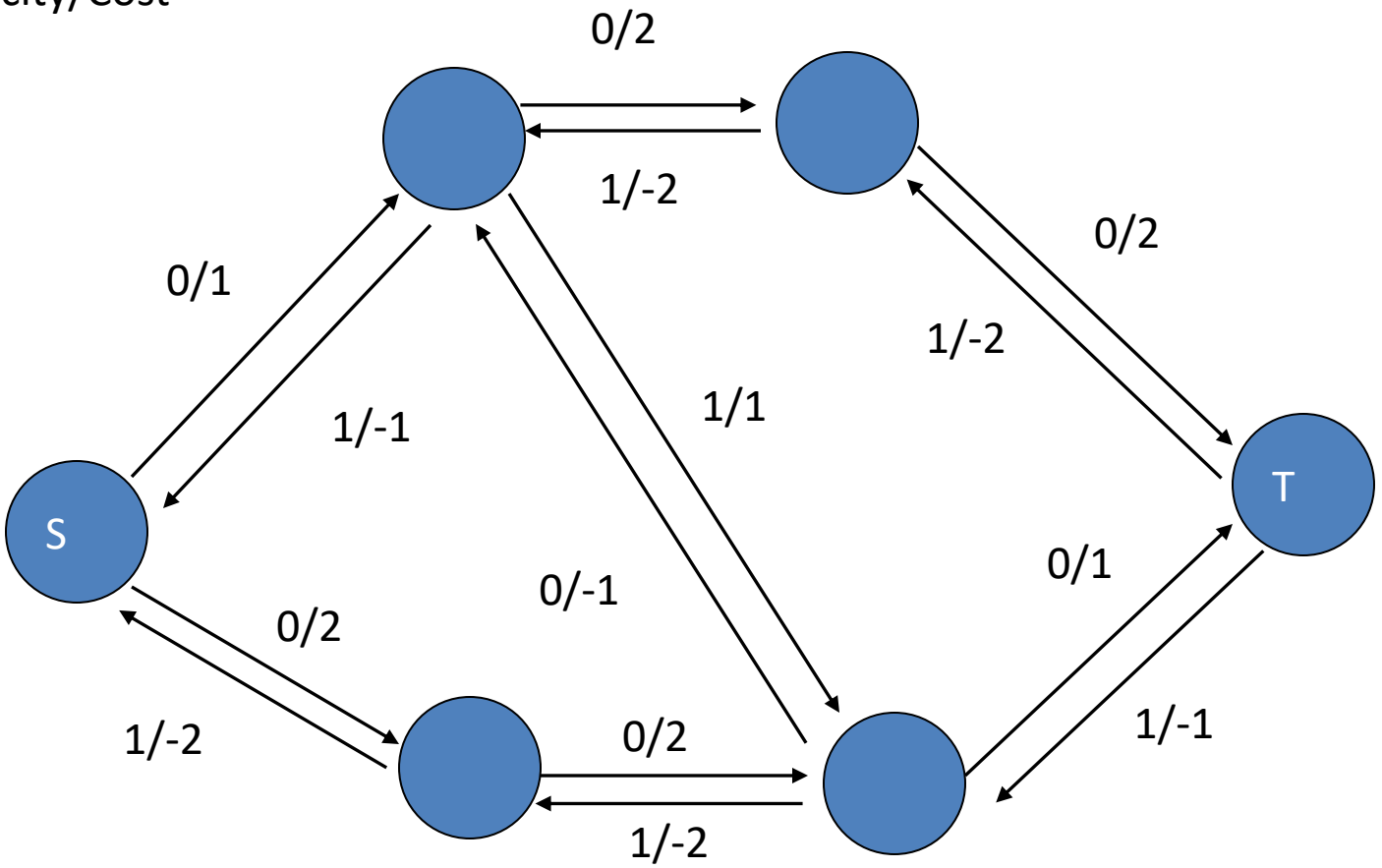
Capacity/Cost



Flow=1, cost = 2+2-1+2+2=7

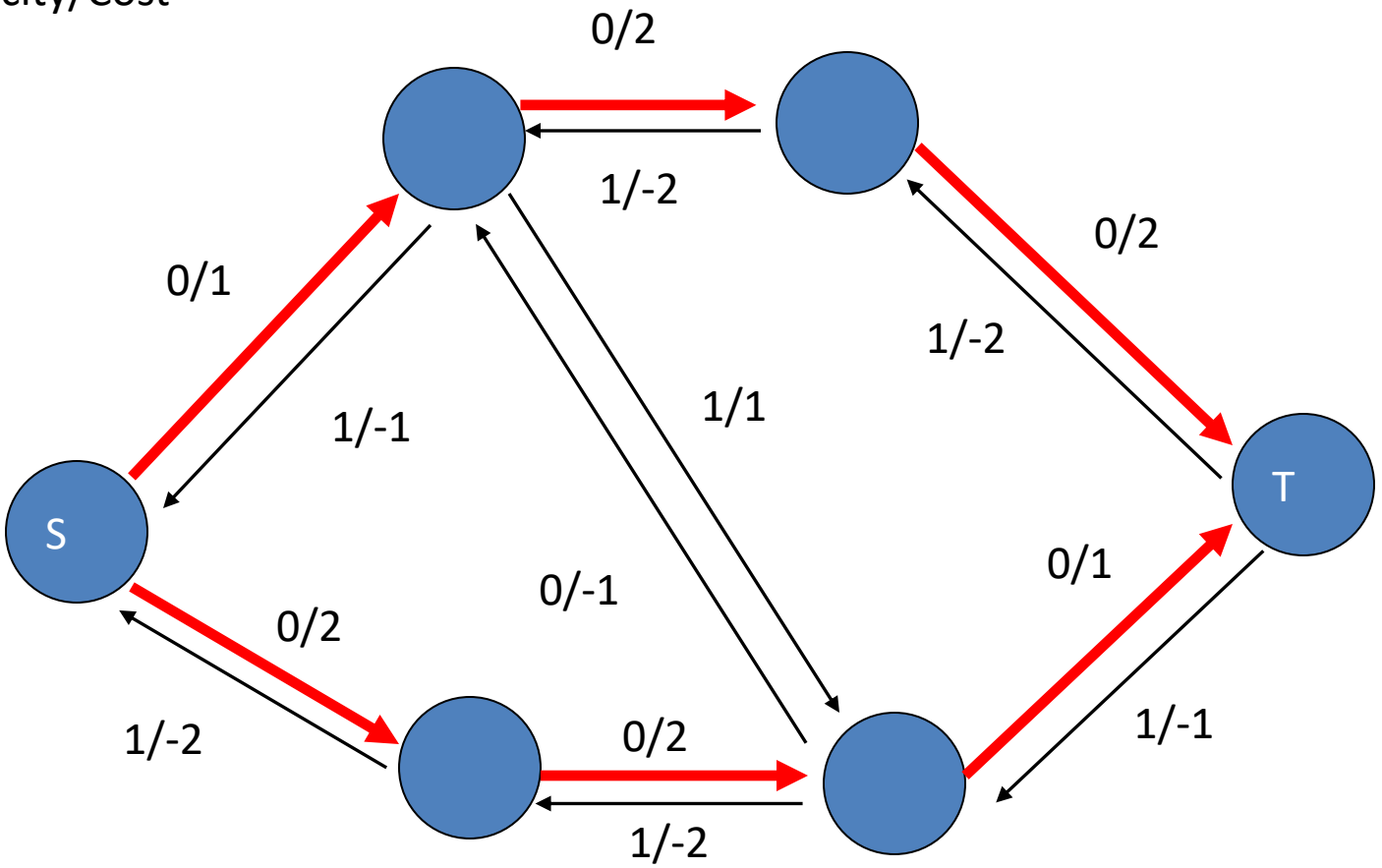
Example

Capacity/Cost



Example

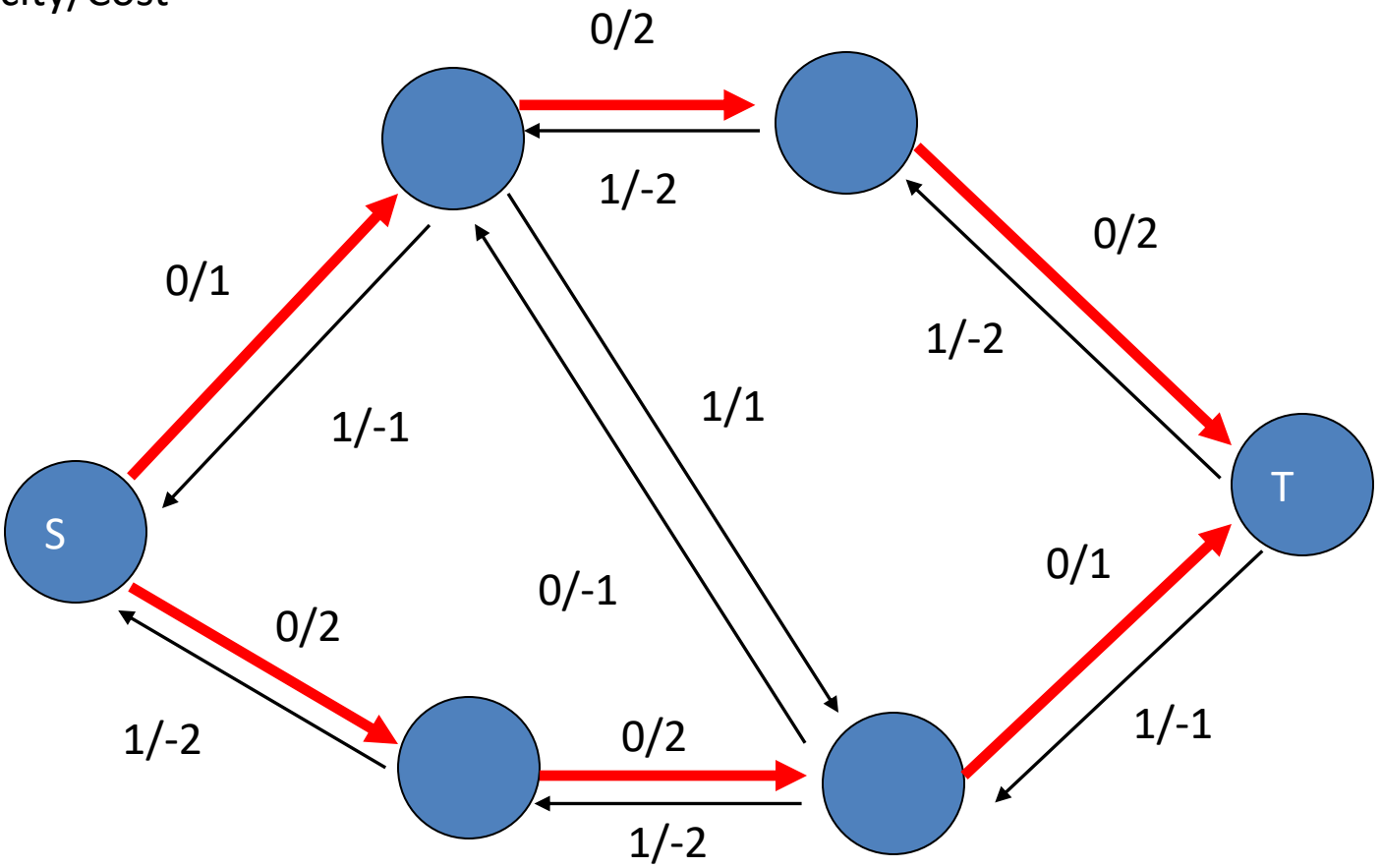
Capacity/Cost



Flow=1, cost=3 and Flow=1 cost=7

Example

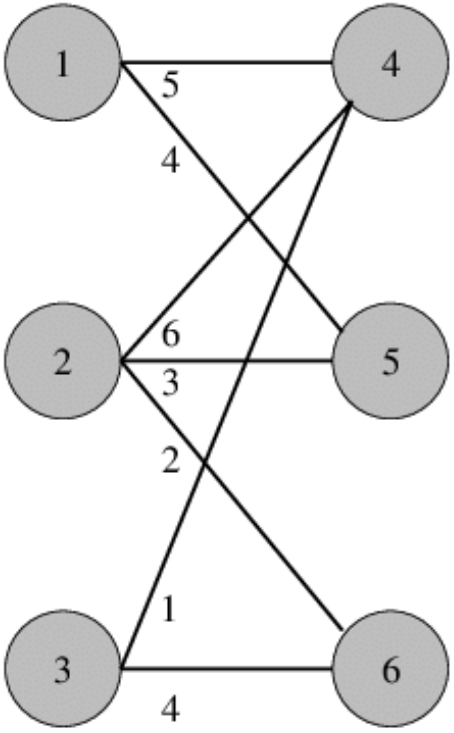
Capacity/Cost



Flow = 2, cost=10

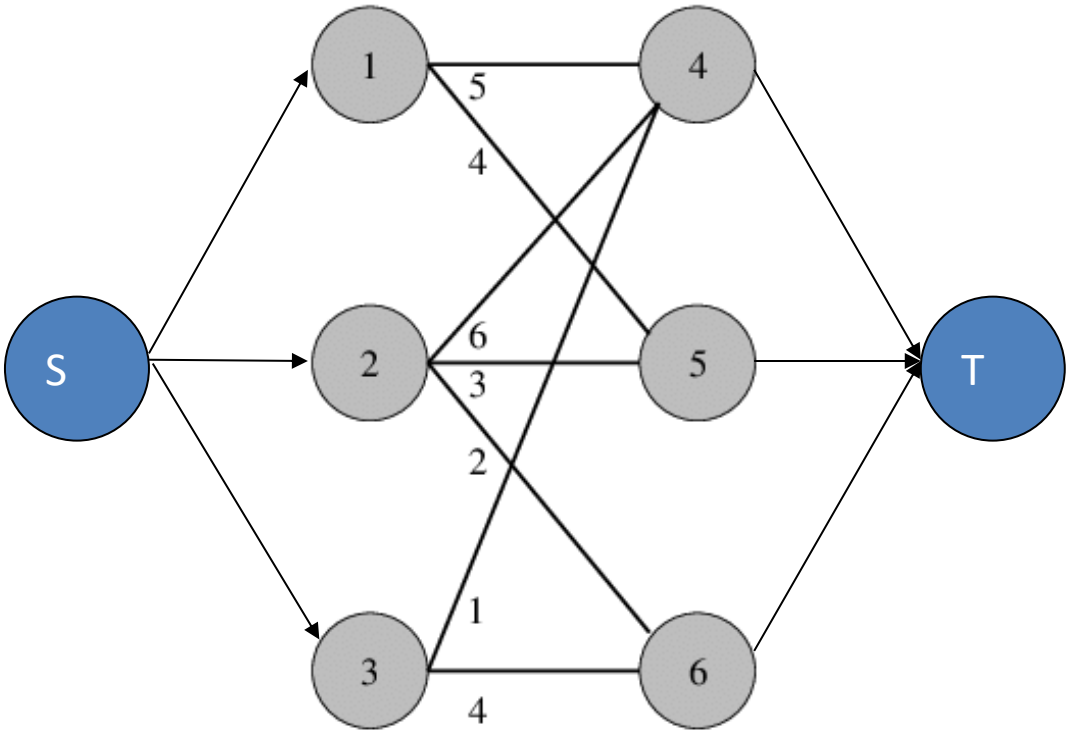
Minimum Weight Matching (MWM)

- Similar to bipartite matching but with weight



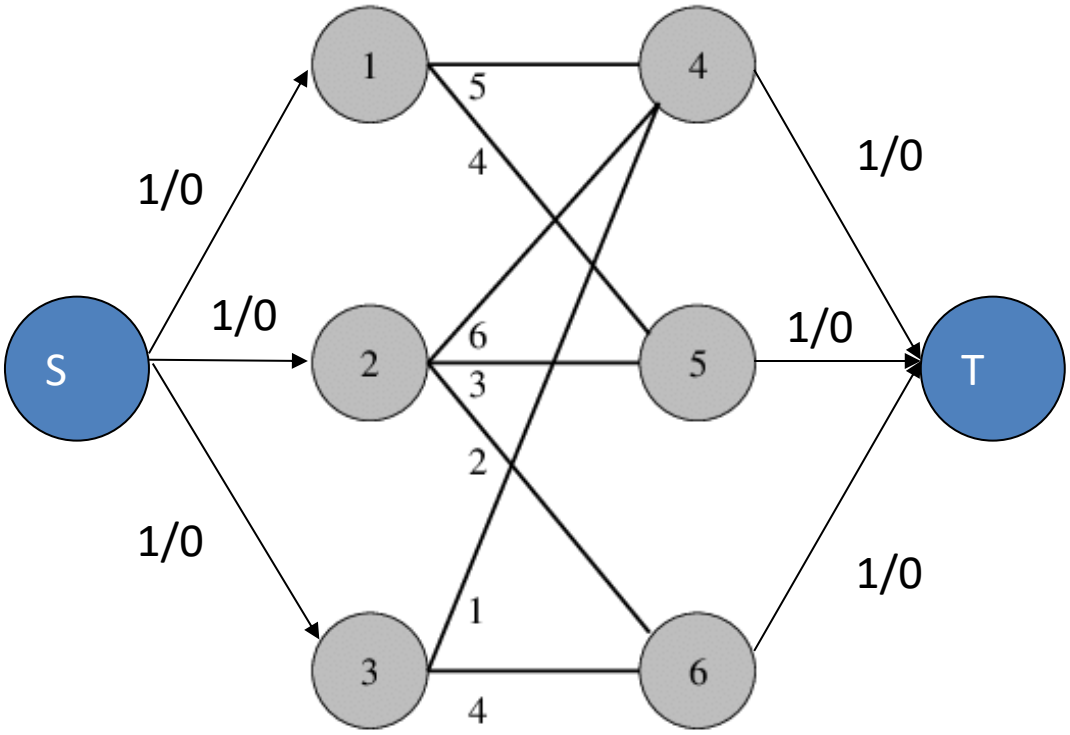
Minimum Weight Matching (MWM)

- Transform to a flow network with cost



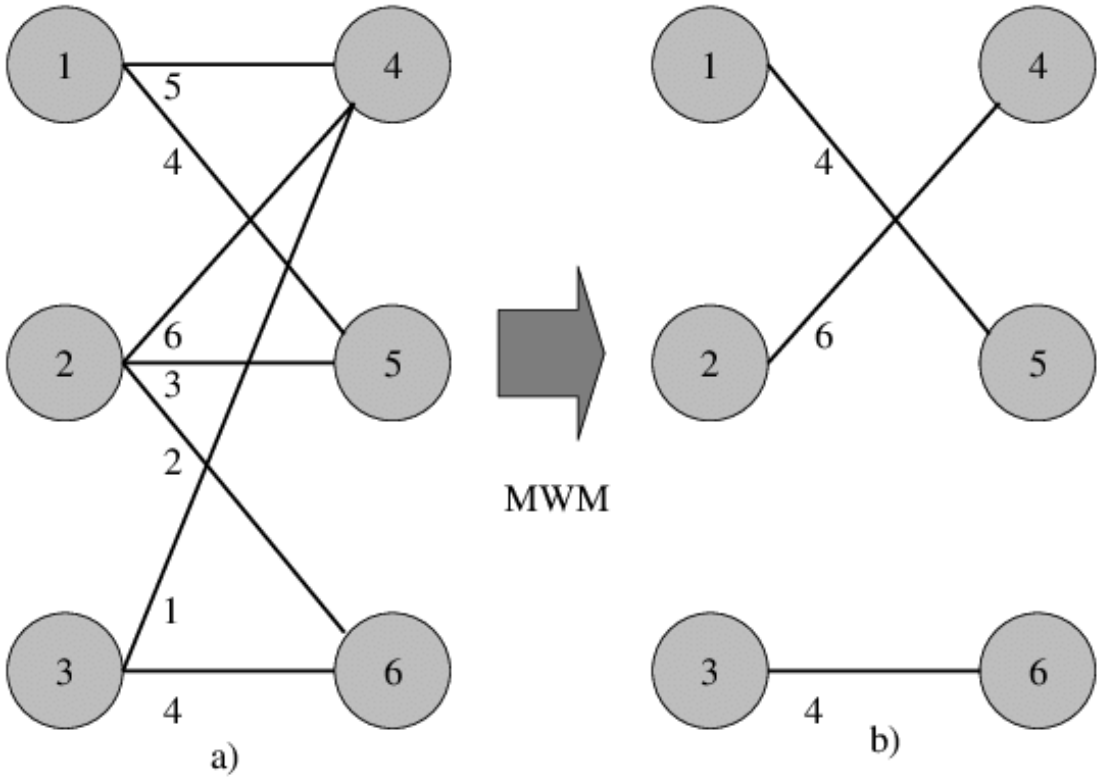
Minimum Weight Matching (MWM)

- Attach capacity and weight



Minimum Weight Matching (MWM)

- Min-cost flow equals MWM



Summary

- **Debrief on HW2**
- **Maximum flow Ford-Fulkerson algorithm recap**
  - Iterative, simple to implement
  - Complexity depends on the unknown “maxflow” value
- **Maximum flow push-relabel algorithm**
  - Iterative with relaxed conservation constraints
  - Time complexity is $O(N^3)$
- **Minimum cost maximum flow algorithm**