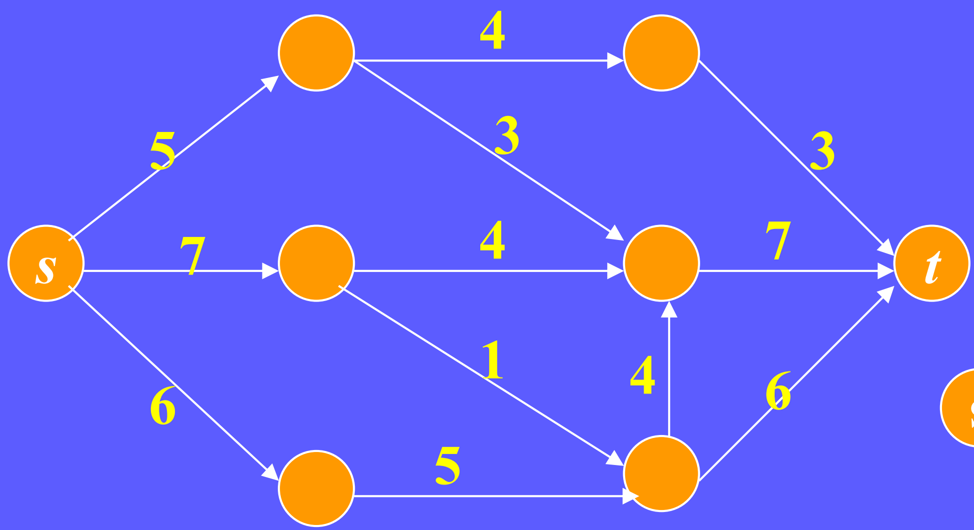
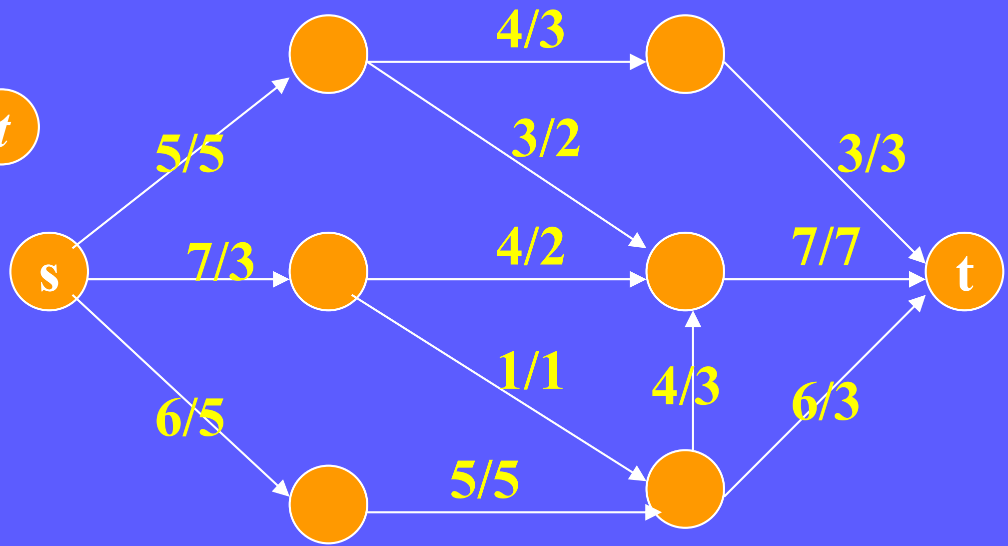
Network Flow

(pp. 238~243)

Network & Network Flows

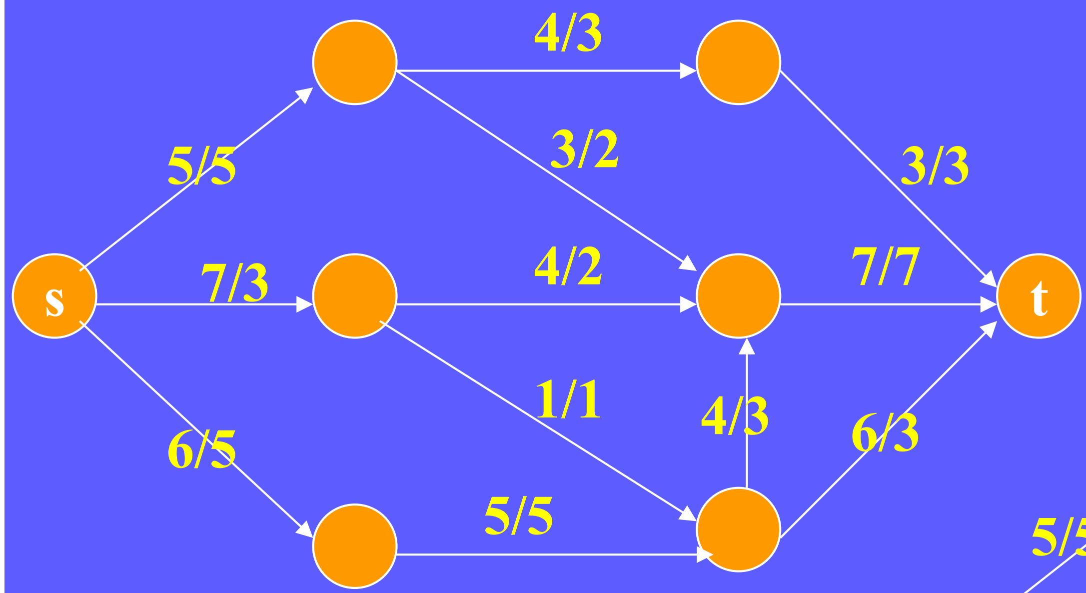
- Let G=(V, E) be a directed graph with 2 distinguished vertices, *s* (source, indegree = 0), *t* (sink, outdegree =0)
  - capacity: each edge *e* is associated with a positive weight c(*e*)
  - flow: a function *f* on the edges that satisfies
    - $0 <= f(e) <= c(e)$
    - $\forall\, v \in V - \{s, t\}, \sum f(u, v) = \sum f(v, w)$

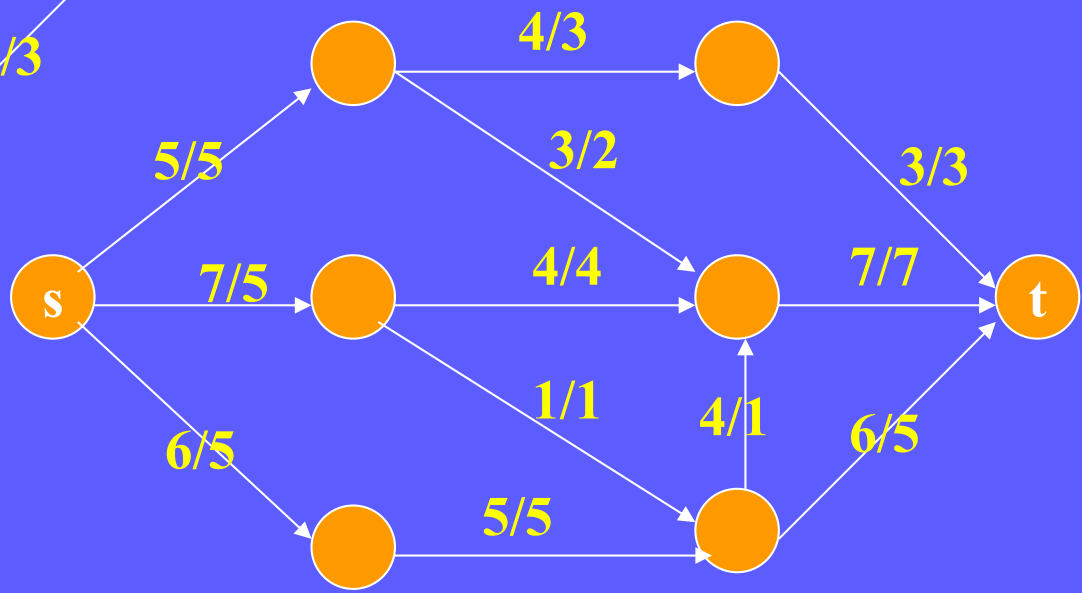




Maximum Flows

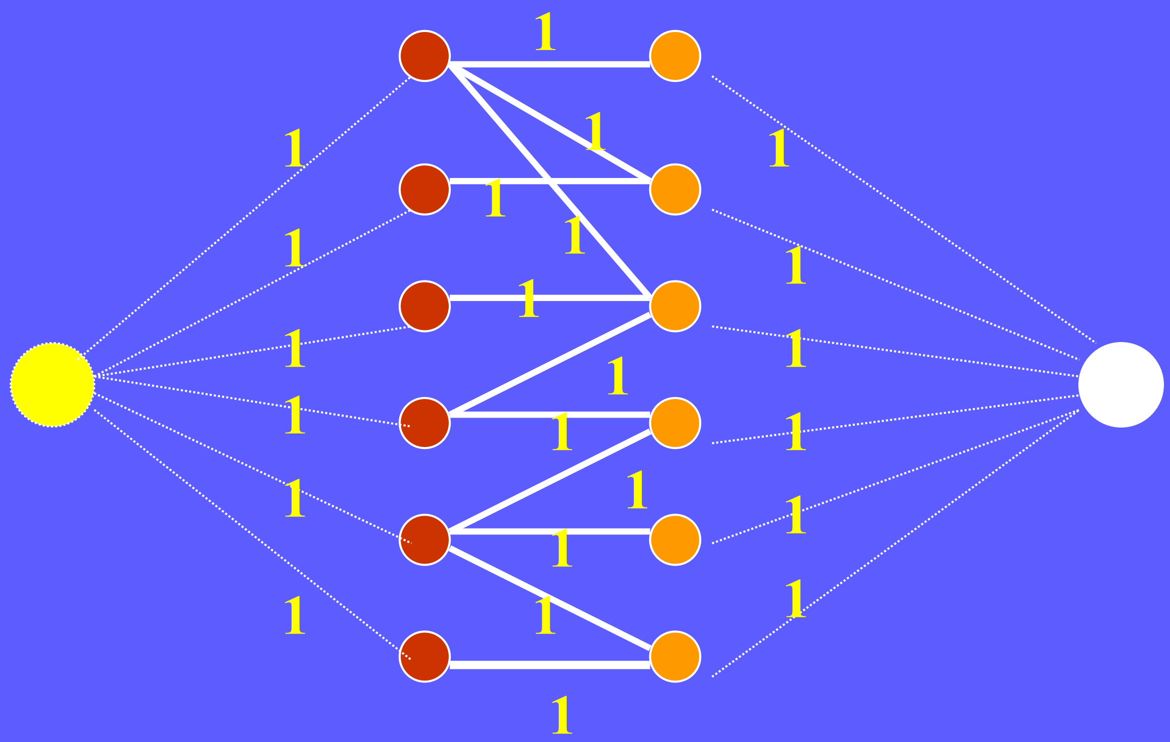
- **Maximum ?**





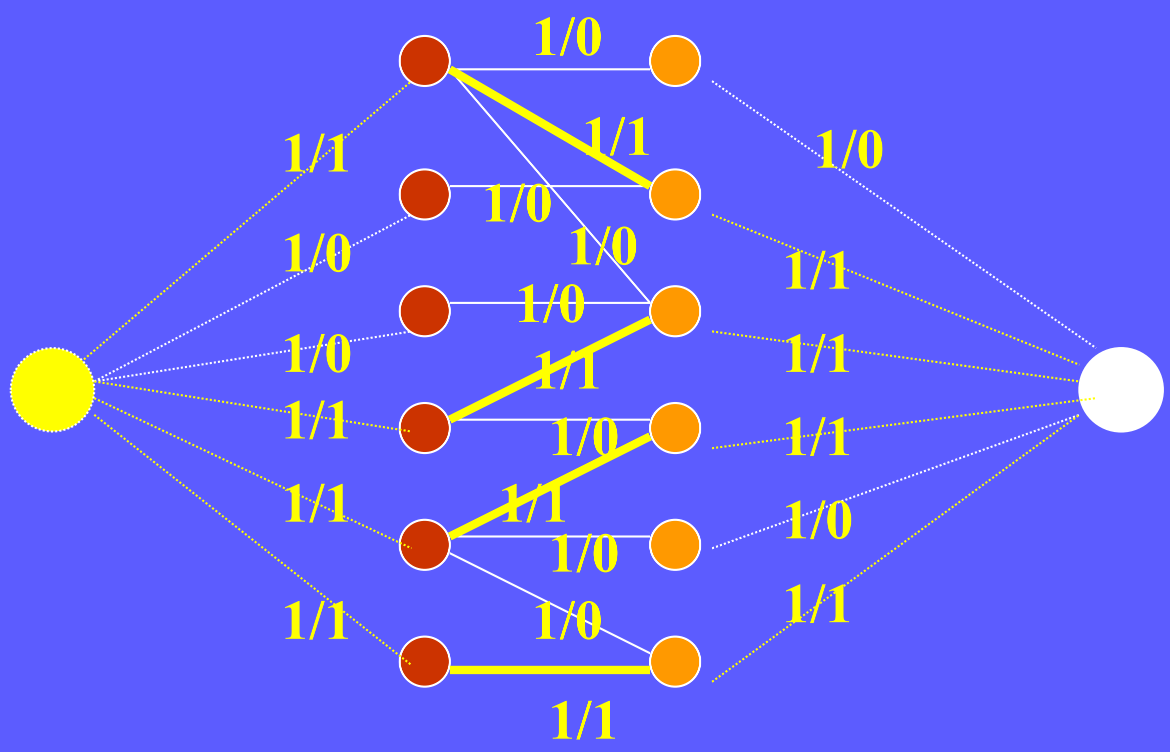
Observation: Reducing Partite Graph to Network Flow

- **M is a maximum matching in G iff the corresponding flow is a maximum flow in G'**



Observation: Reducing Partite Graph to Network Flow

- **M is a maximum matching iff the corresponding flow is a maximum flow in G'**



Proof of Alternating-Path Theorem

- **a matching is maximum**
  iff it has no alternating paths
- **Proof (reduce to maximum flow problem)**

(1) if flow is maximum, it is a maximum matching
Otherwise, there would be a larger flow

(2) if it is a maximum matching,
its corresponding flow is maximum
if M is maximum matching, no alternating path for it
=> no augmenting path in G'
=> flow is maximum

Augmenting Paths

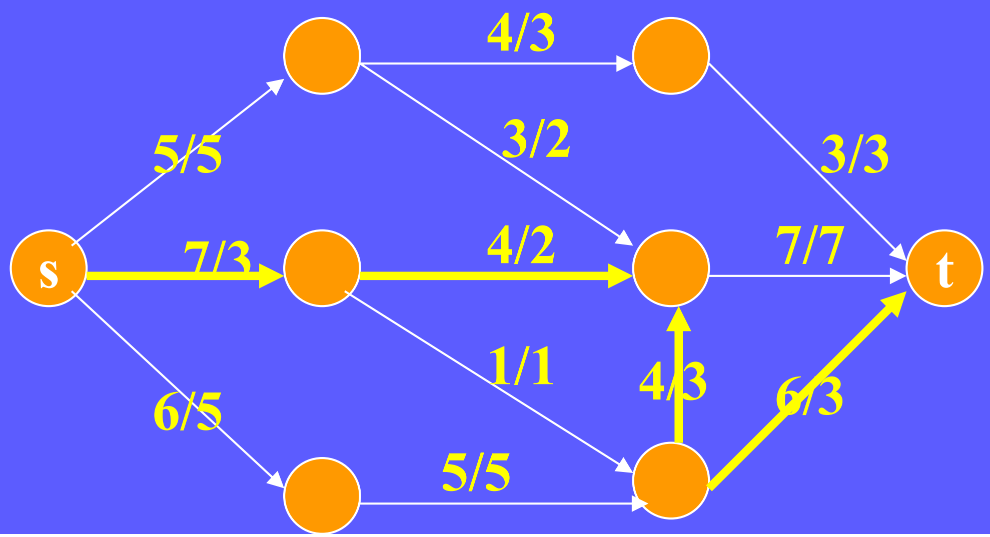
- **An augmenting path with respect to a given flow *f* is a directed path from *s* to *t*, each edge (v, u) satisfies**
  - **forward edge**

    (v, u) is in the same direction as it is in G, *f*(v, u) < c(v, u)

    *** slake of edge = c(v, u) - *f*(v, u), room for flow**
  - **backward edge**

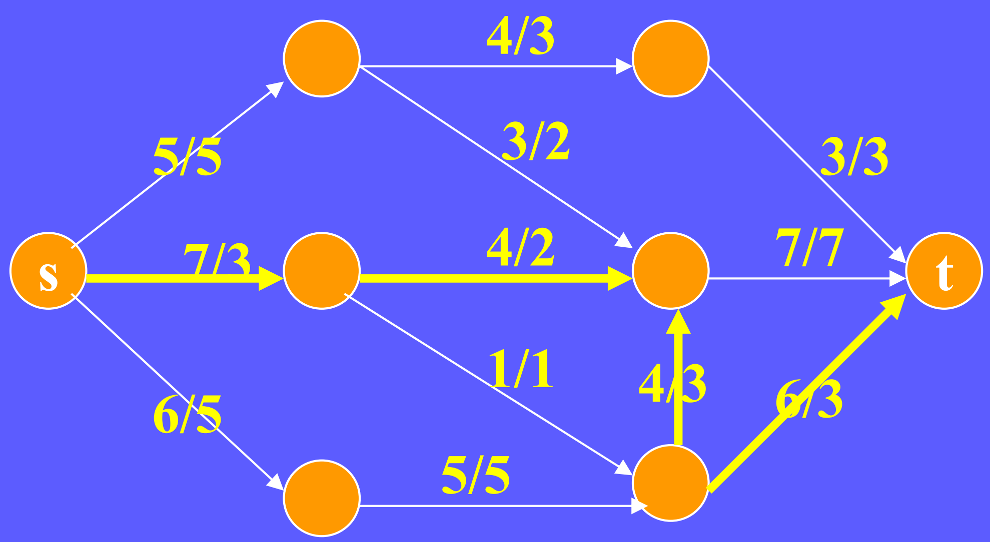
    (v, u) is in the opposite direction in G, *f*(u, v) > 0

    *** it is possible to borrow some flow from backward edge**



Increment of Flow in Augmenting Path

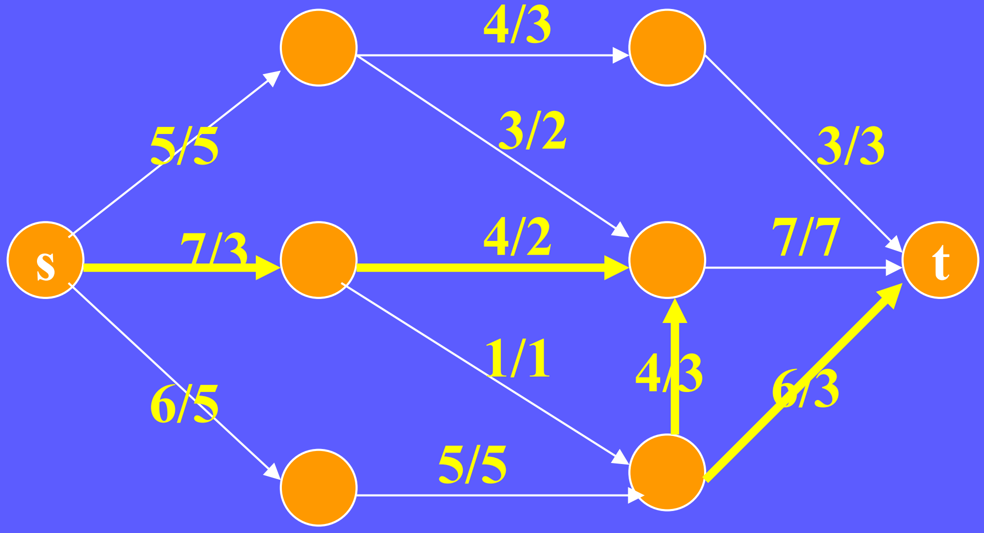
- **Increase is equal to minimum of either**
  - **the minimal slake of forward edges or**
  - **minimal current flow through backward edges**



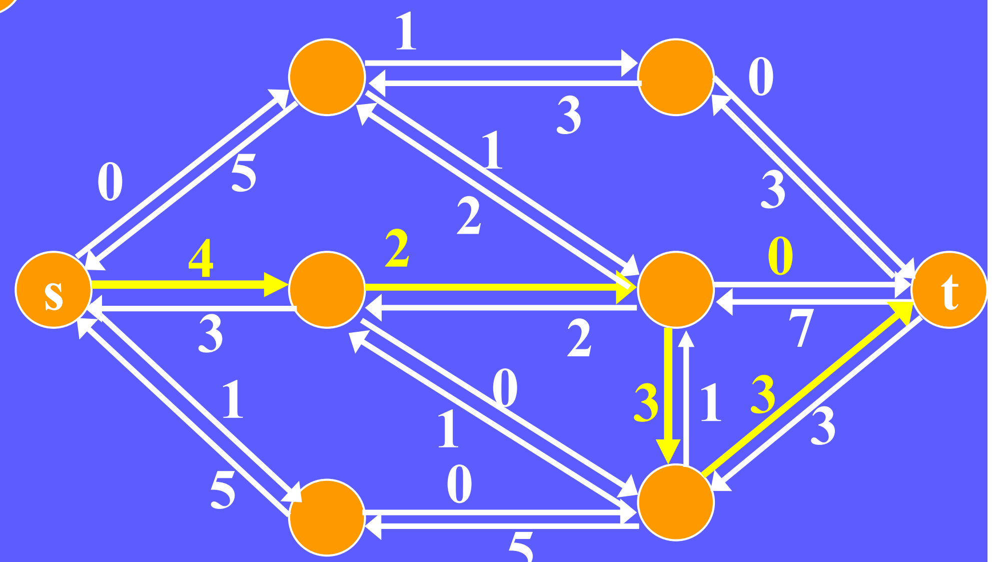
increase
= min(min{4, 2, 3}, min{3})
= 2

Algorithm for Searching for Augmenting Paths

- residual graph, R=(V, F)



residual graph



Algorithm for Searching for Augmenting Paths

- **residual graph, R=(V, F)**
  with respect to a network G=(V, E) and a flow f,
  but with different directions & capacities
- **An edge (v,w) belongs to F if it is either**
  a forward edge (capacity = c(v,w)-f(v,w)) or
  a backward edge (capacity F(v,w))
- **An augmenting path =**
  directed path from s to t in the residual graph
- **Constructing the residual graph requires |E| steps**

Algorithm of Maximum Flow

- Start with a flow of 0
- Repeat

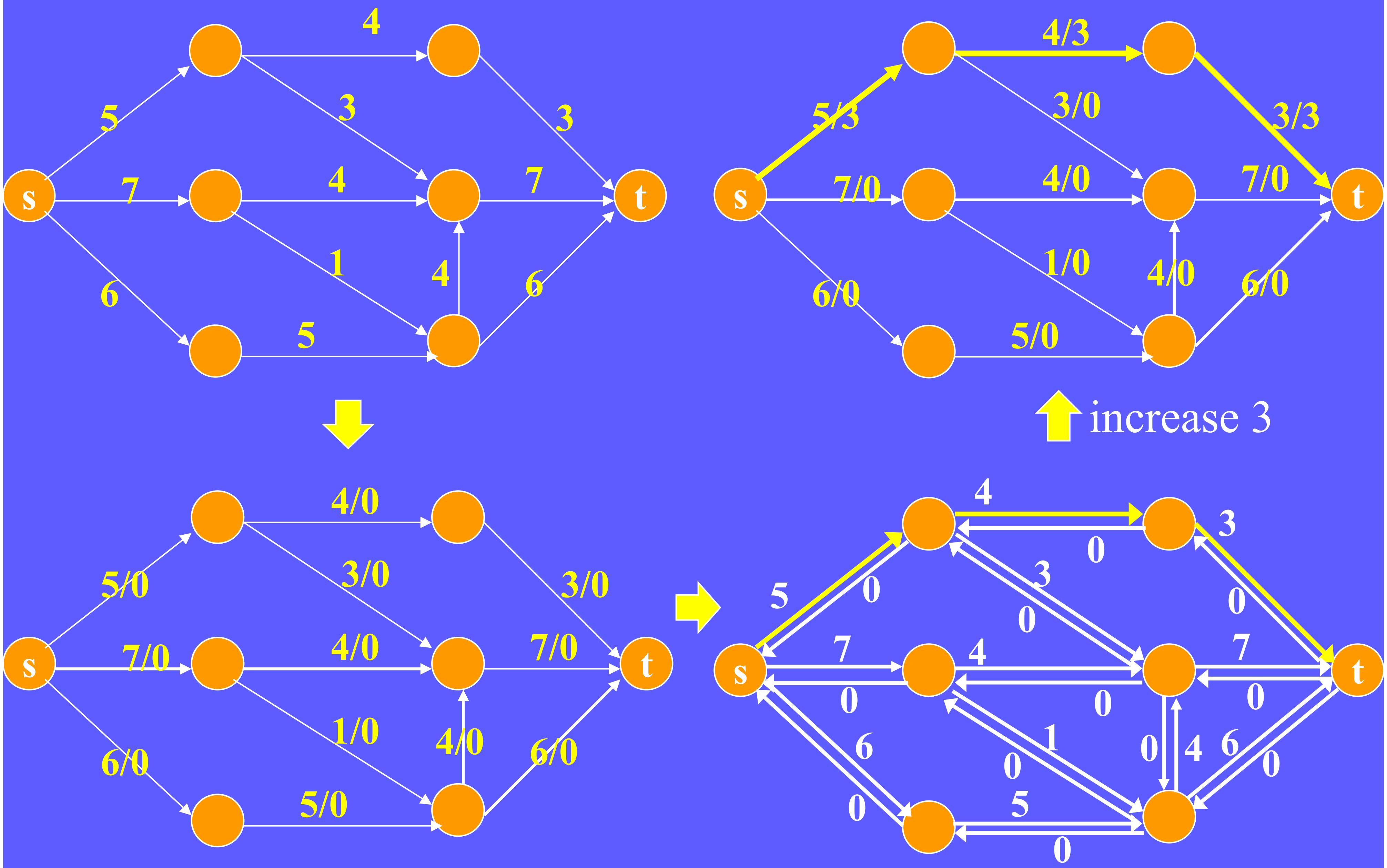
    construct the residual graph for the current flow

    search for augmenting paths

    augment the flow

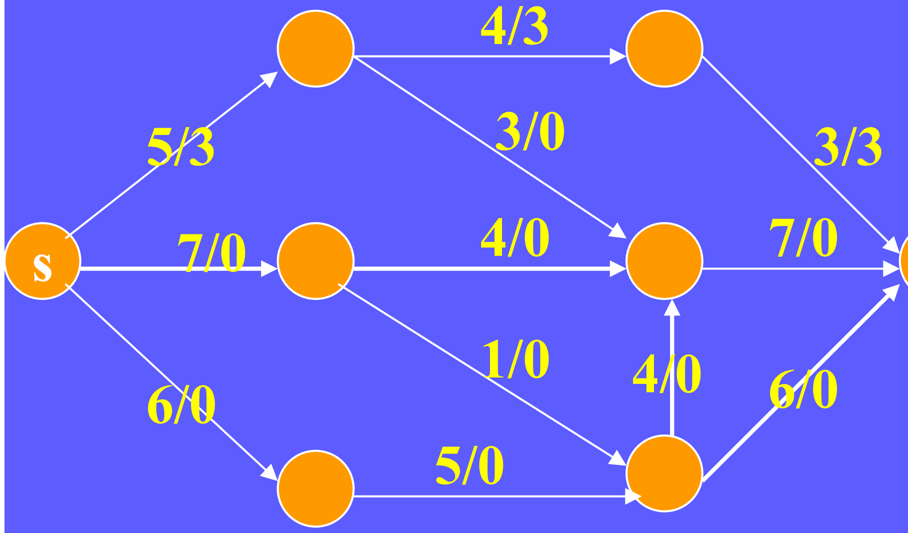
  Until no more augmenting paths

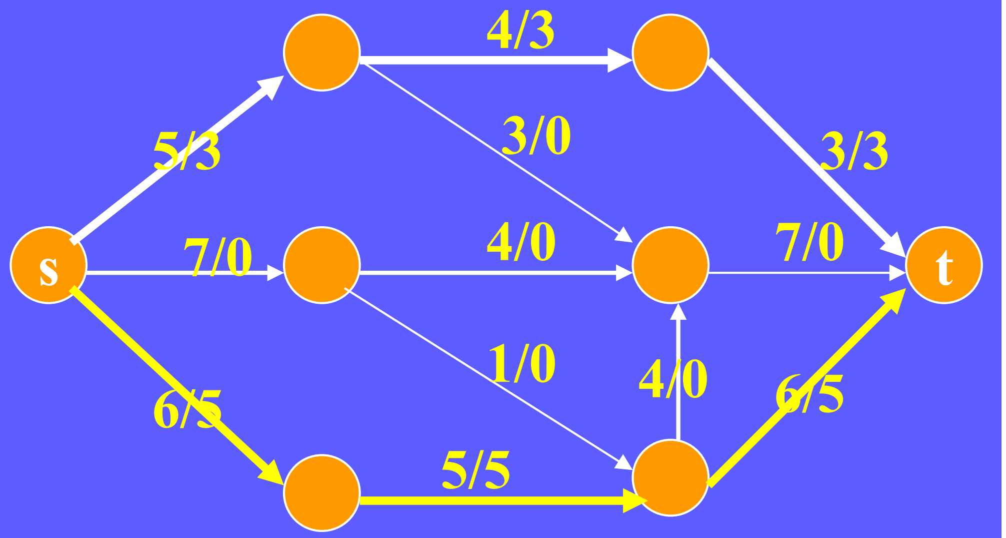
An Example



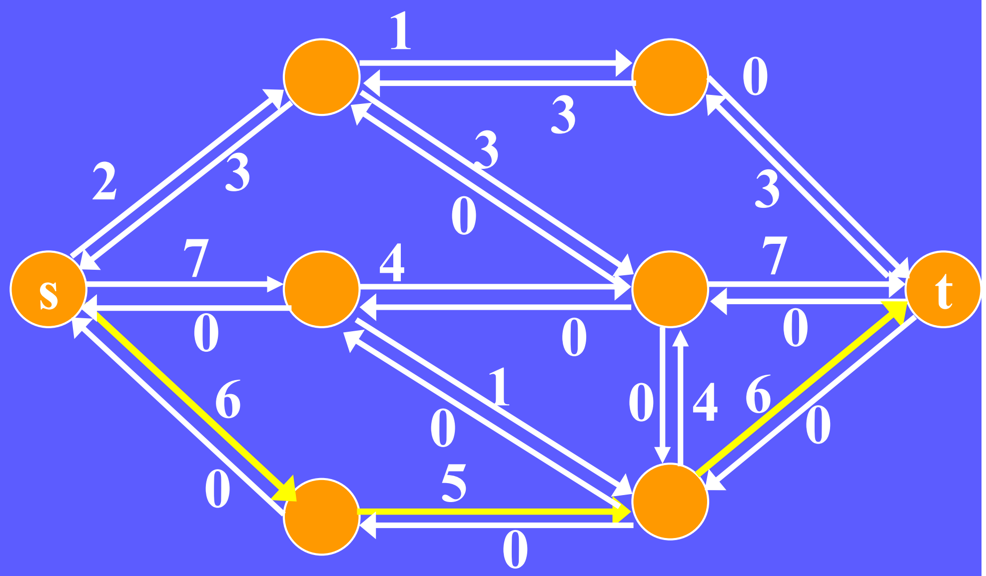
s
t
4
3
5
7
4
1
4
6
5
3
7
6
4/3
3/0
5/3
3/3
7/0
4/0
7/0
1/0
4/0
6/0
6/0
5/0
increase 3
4/0
3/0
5/0
3/0
7/0
4/0
7/0
1/0
4/0
6/0
6/0
5/0
4
0
3
5
0
3
0
0
7
4
7
0
0
0
1
0
4
6
0
6
0
5
0
0

An Example (cont.)

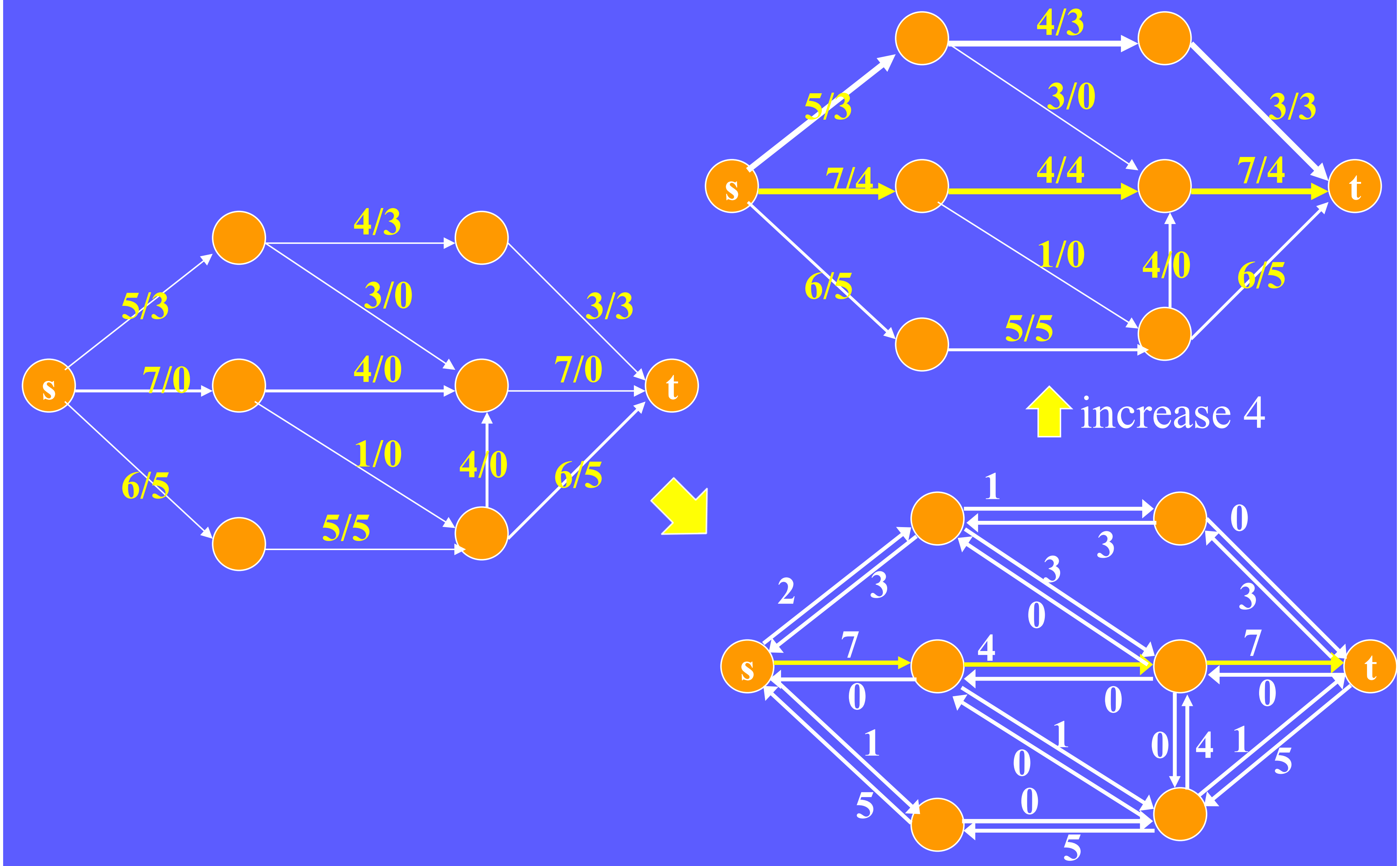




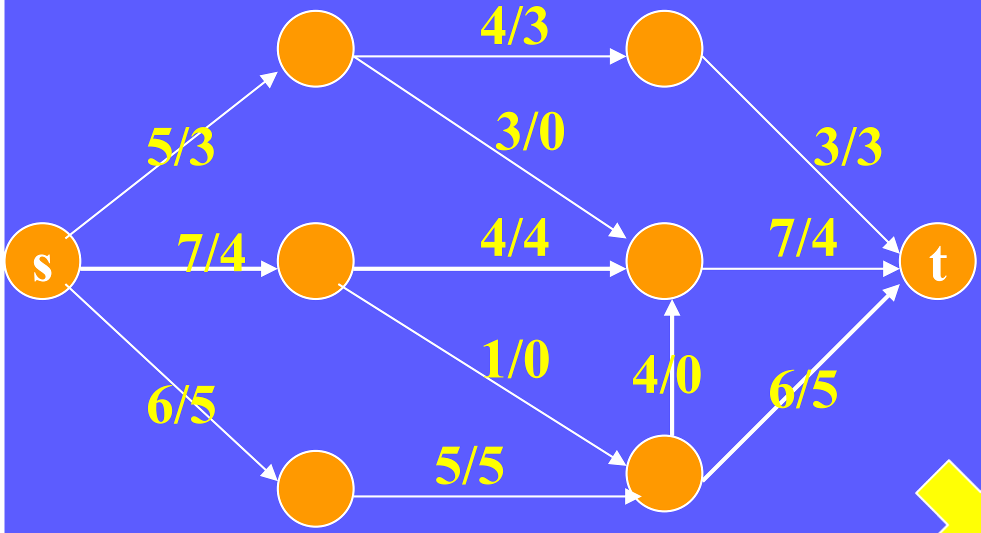
increase 5

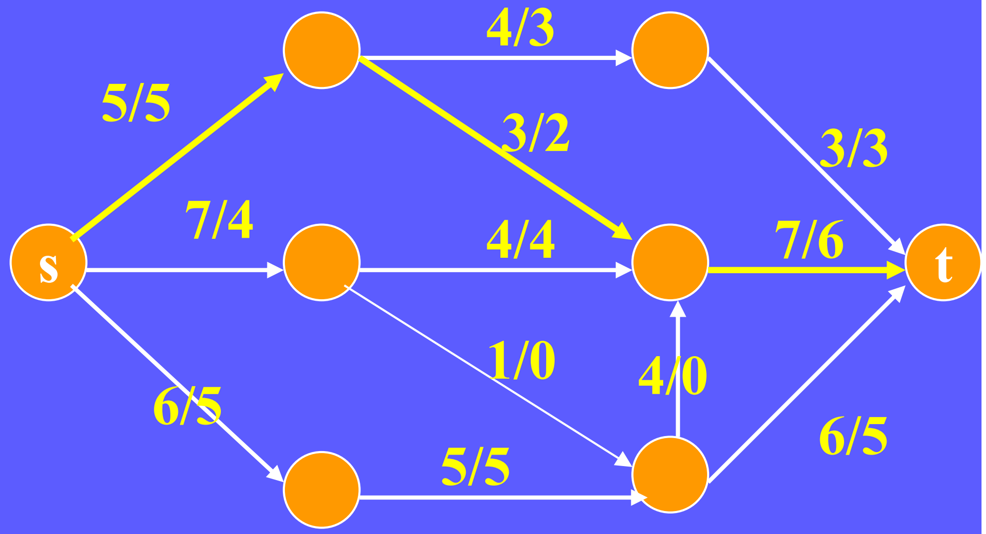


An Example (cont.)

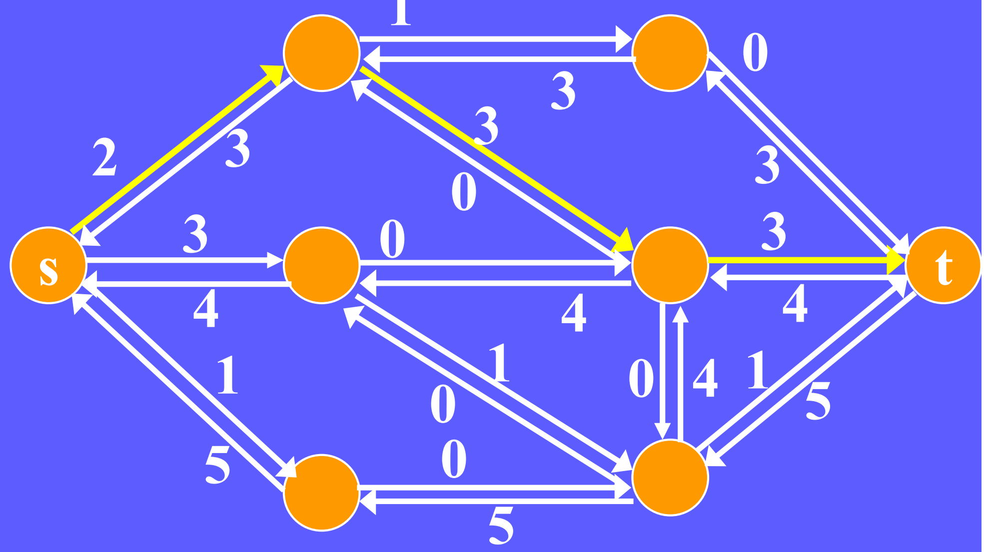


An Example (cont.)

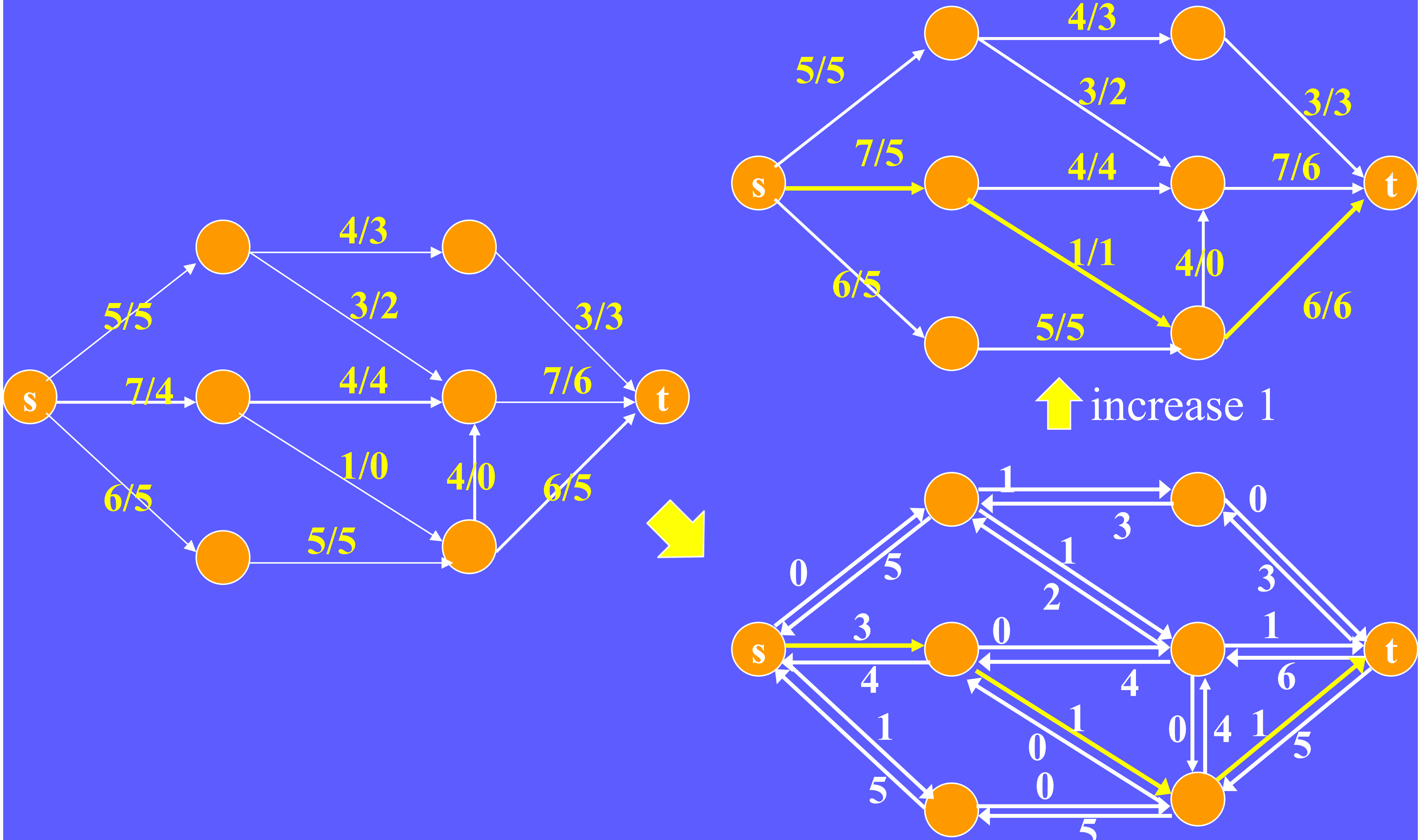




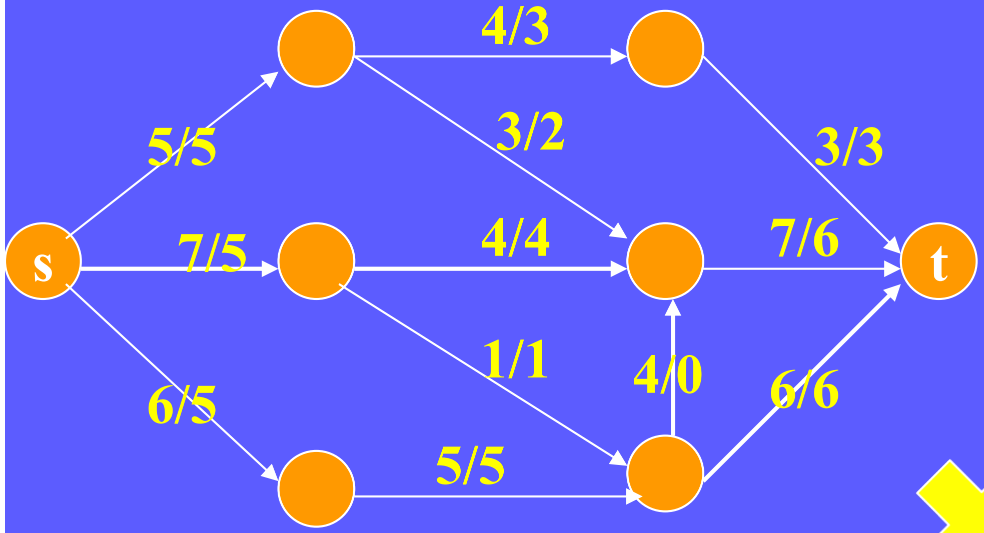
increase 2

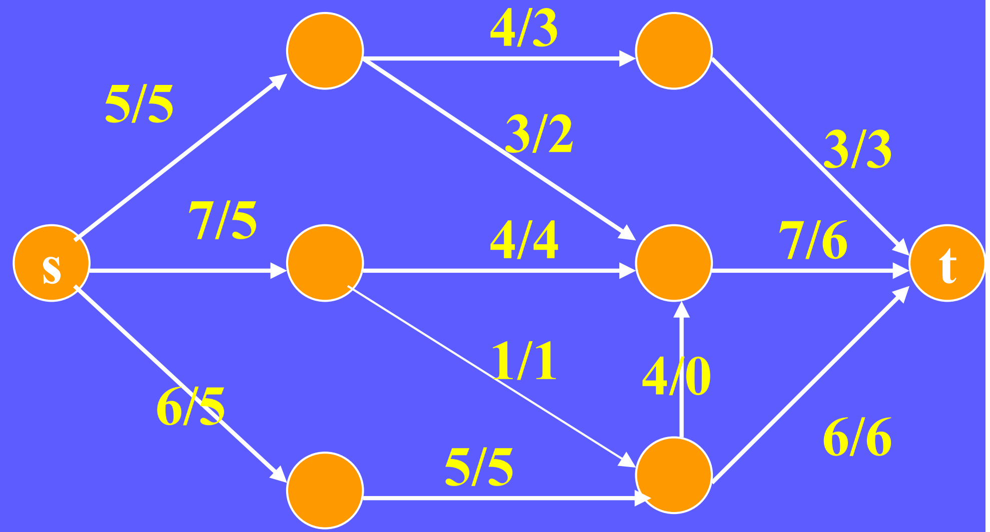


An Example (cont.)

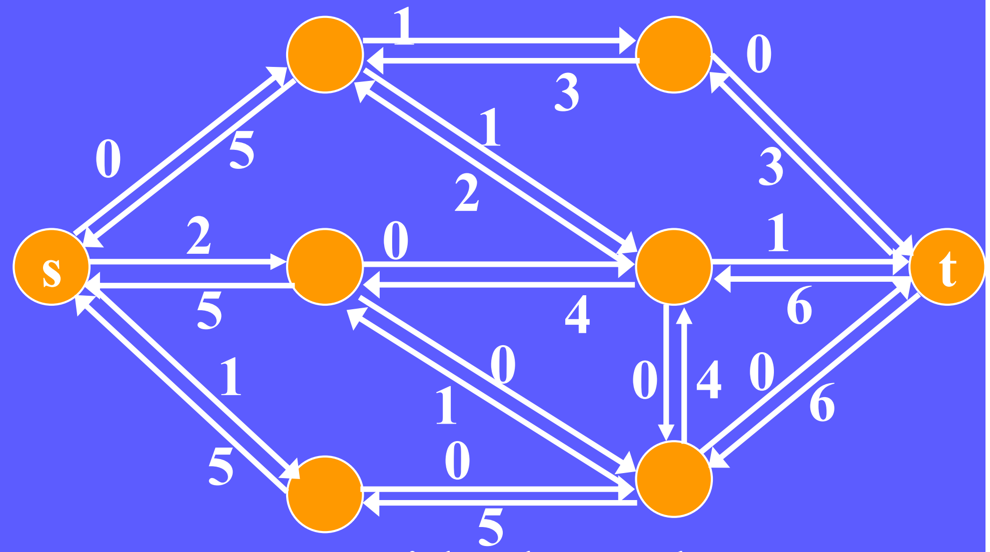


An Example (cont.)



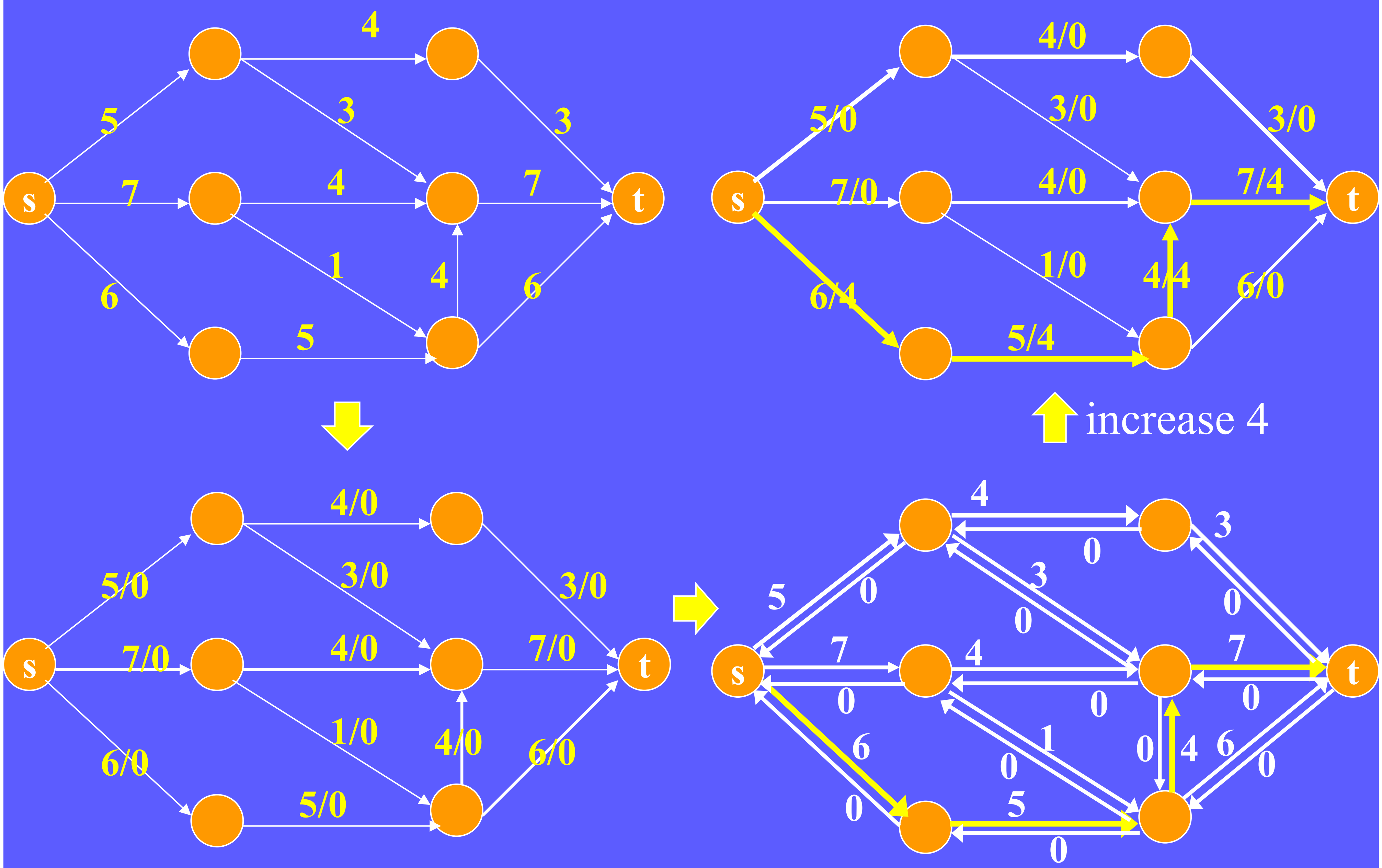


maximun



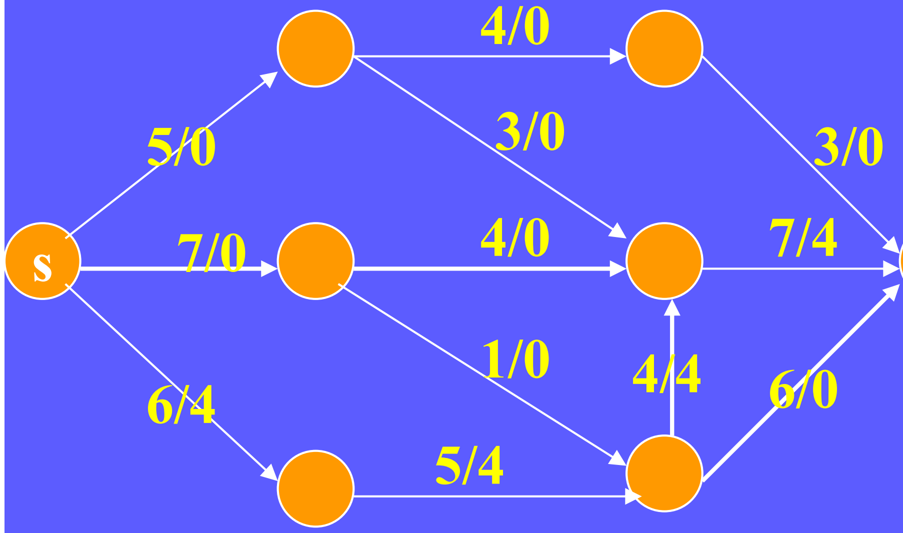
residual graph

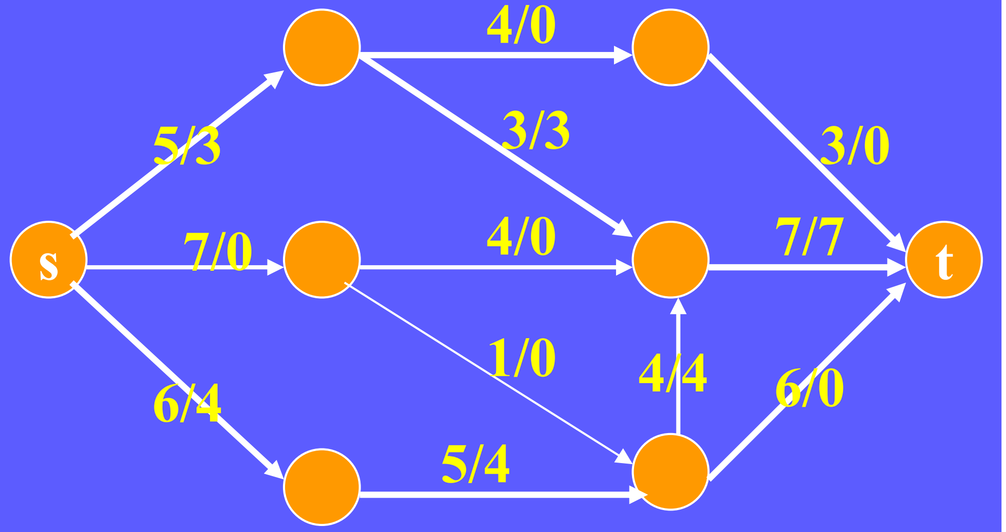
An Example



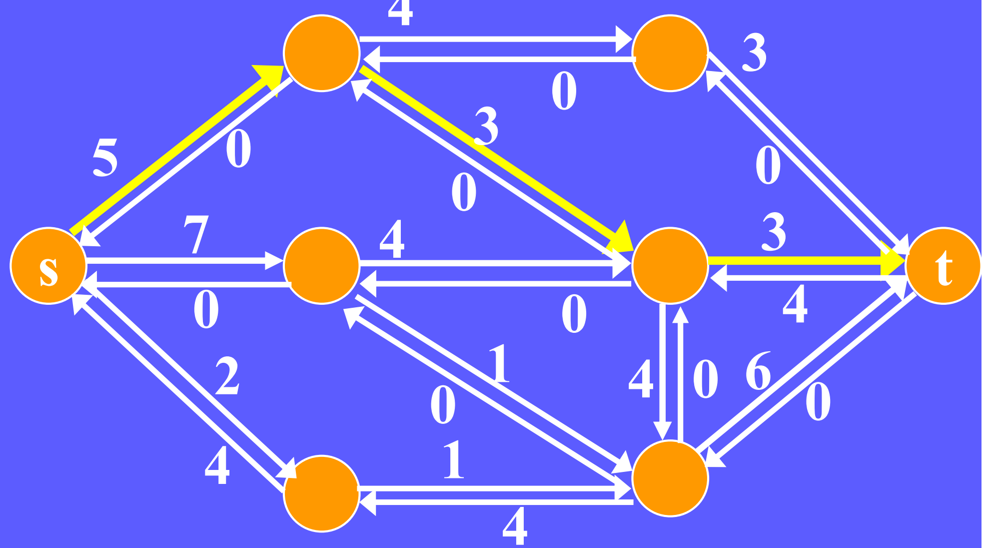
s
t
4
3
5
7
4
3
7
1
4
6
6
5
4/0
3/0
5/0
7/0
4/0
7/4
1/0
4/4
6/0
6/4
5/4
increase 4
4/0
3/0
3/0
5/0
7/0
4/0
7/0
1/0
4/0
6/0
6/0
5/0
4
0
3
5
0
3
0
0
7
4
7
0
0
0
1
0
4
6
0
6
0
5
0

An Example (cont.)

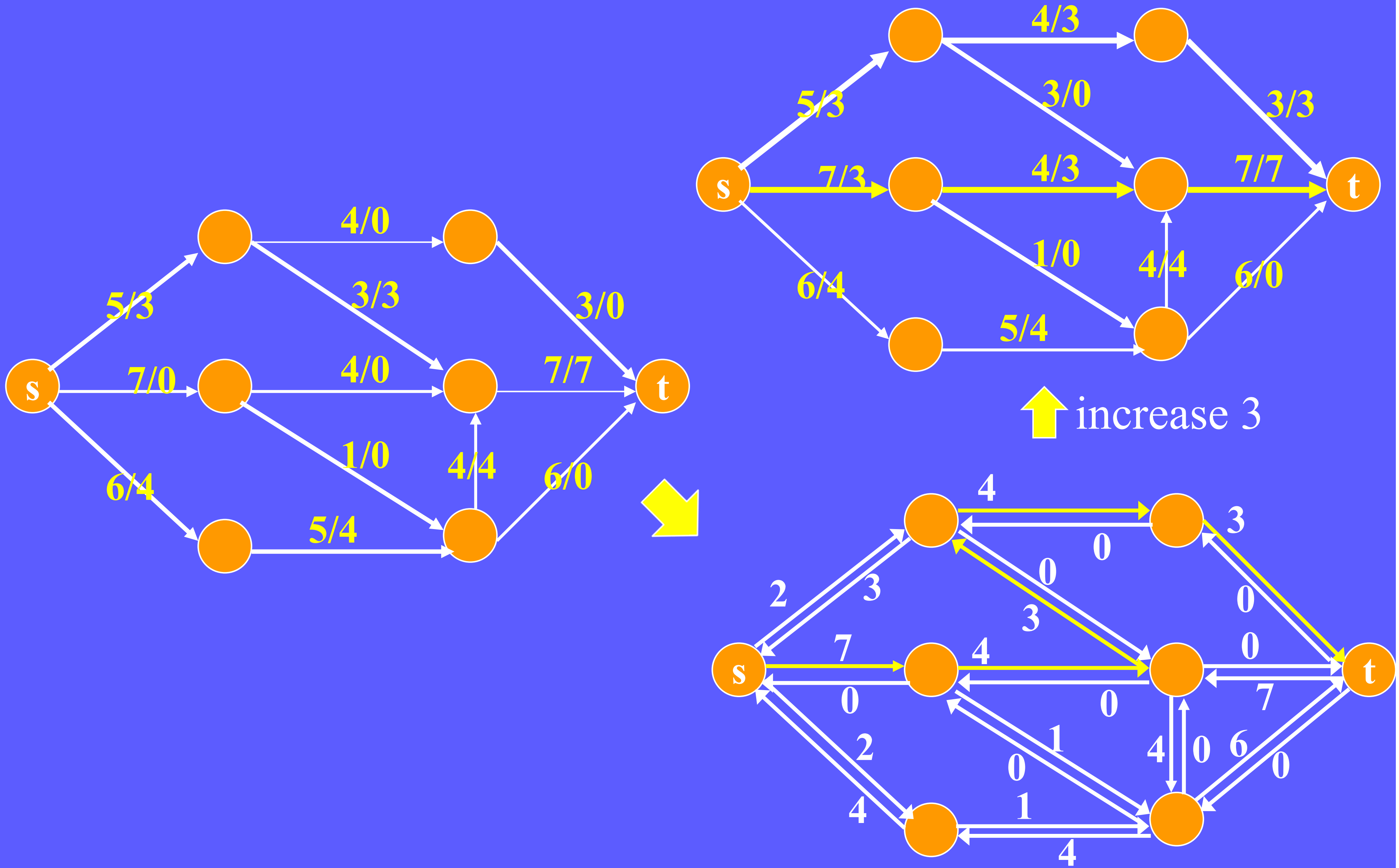




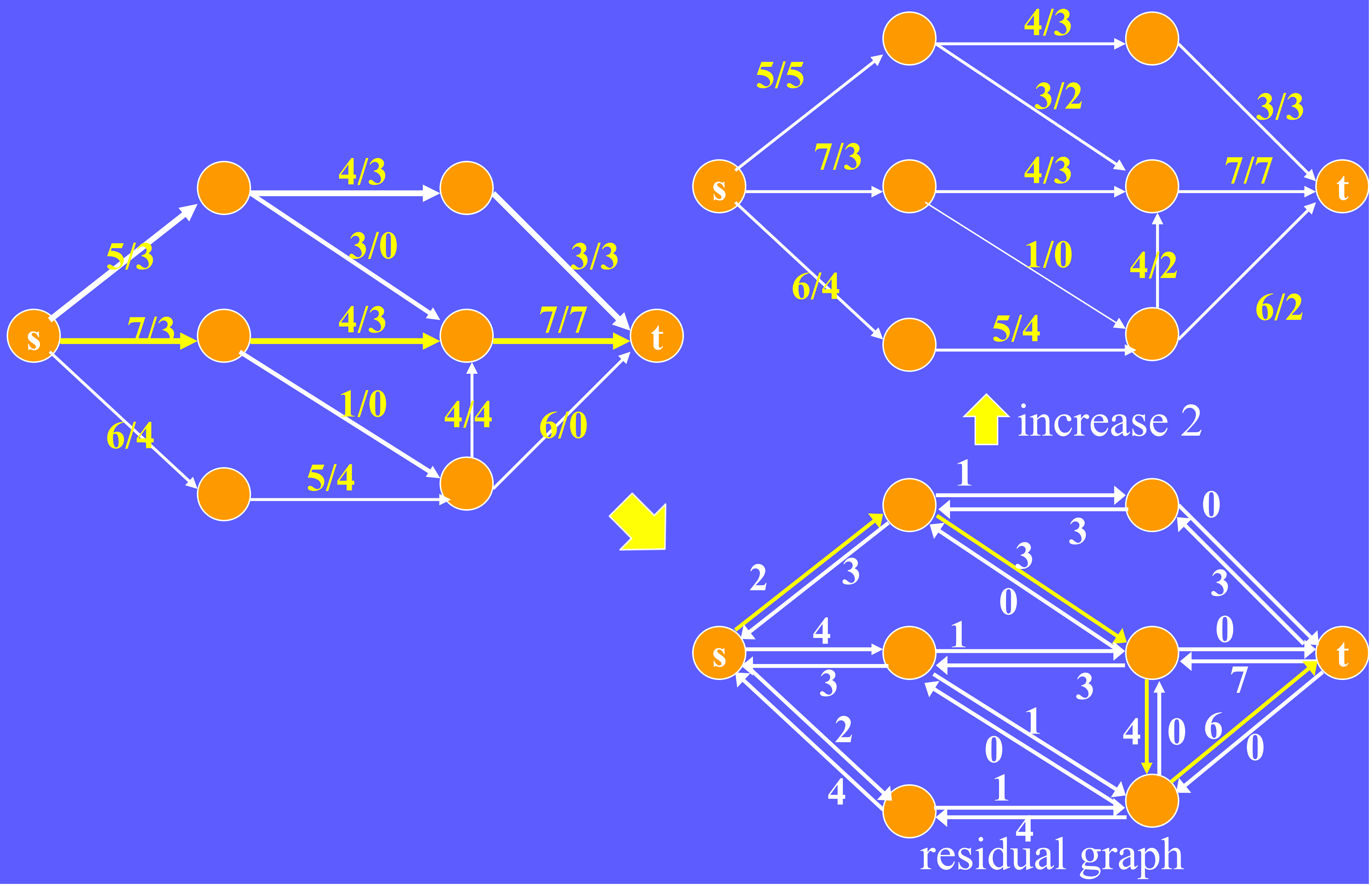
increase 3



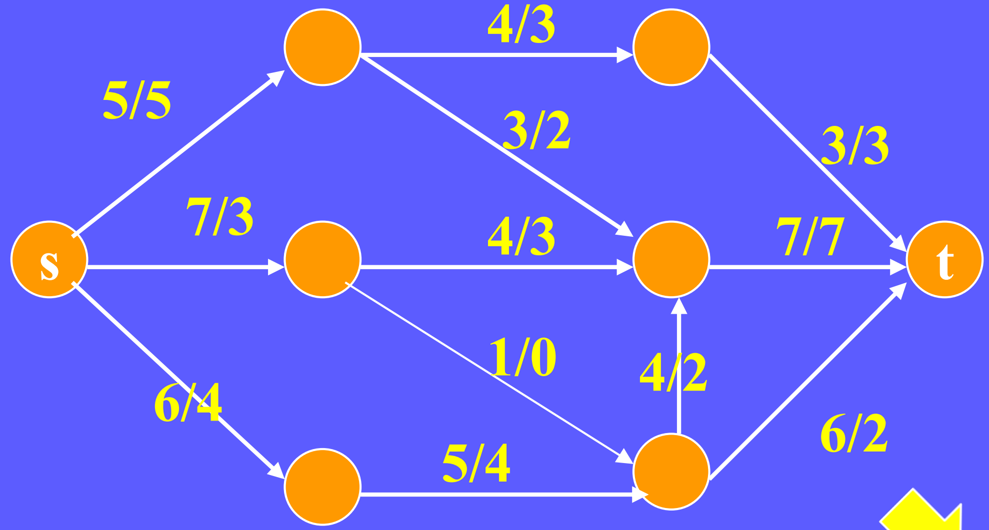
An Example (cont.)

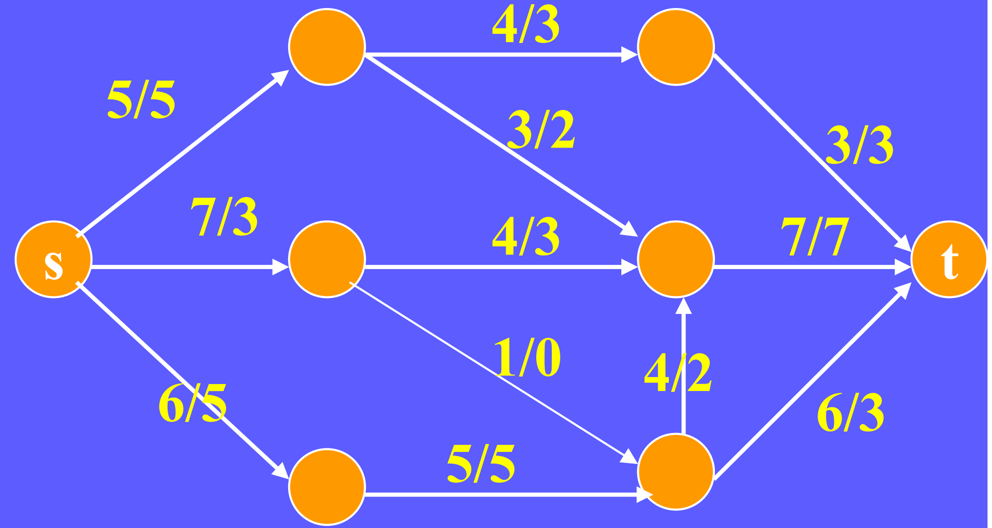


An Example (cont.)

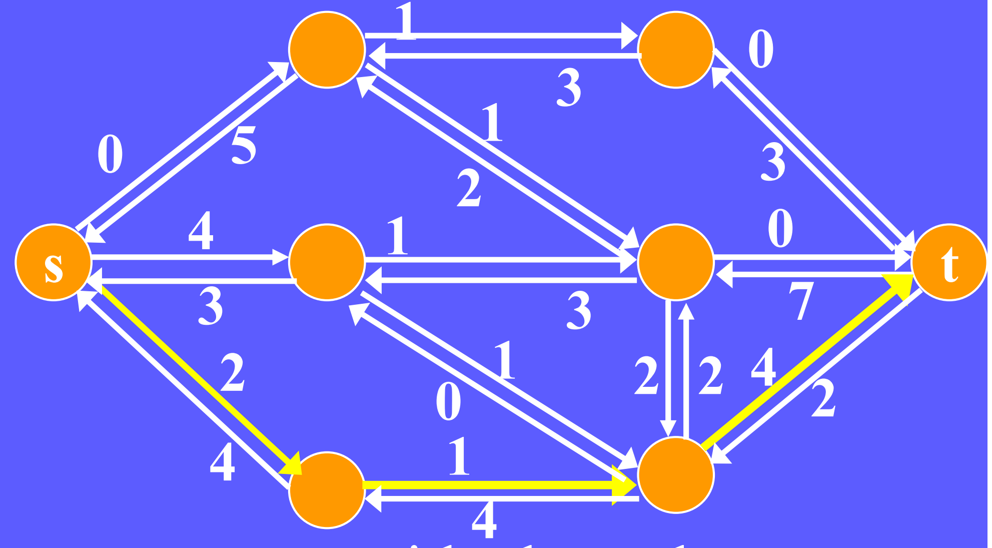


An Example (cont.)



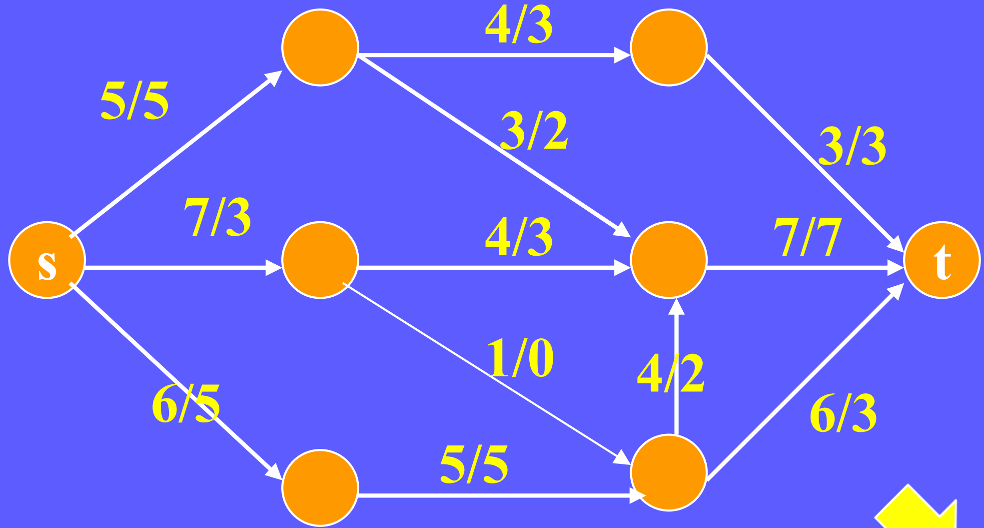


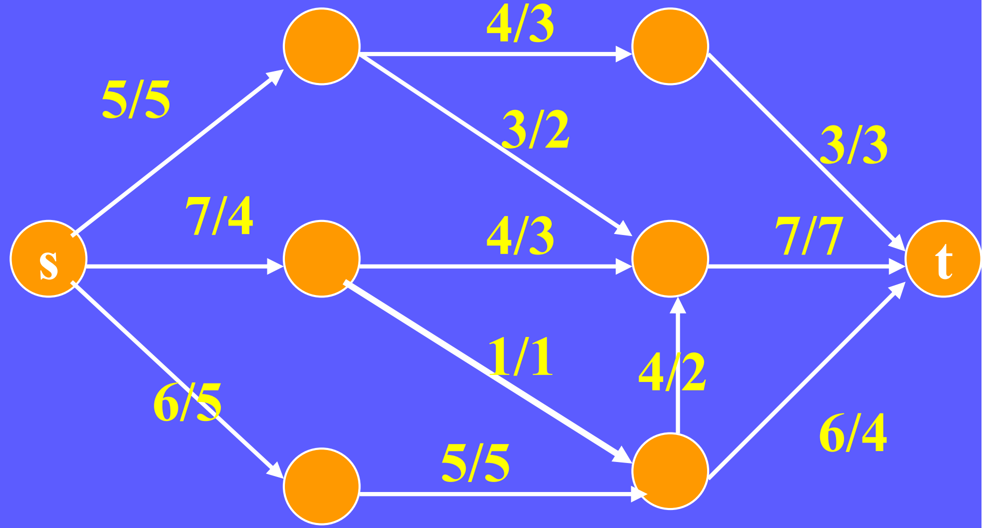
increase 1



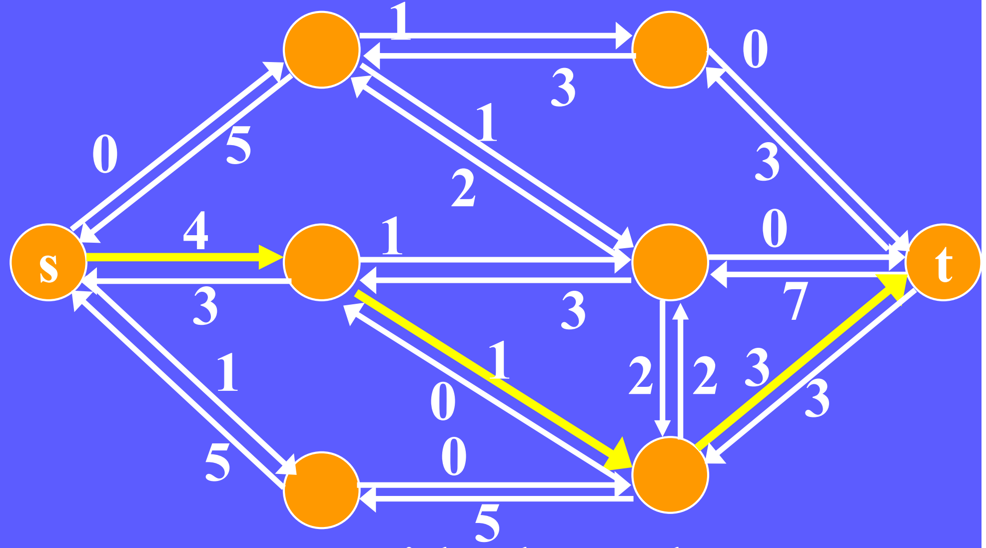
residual graph

An Example (cont.)



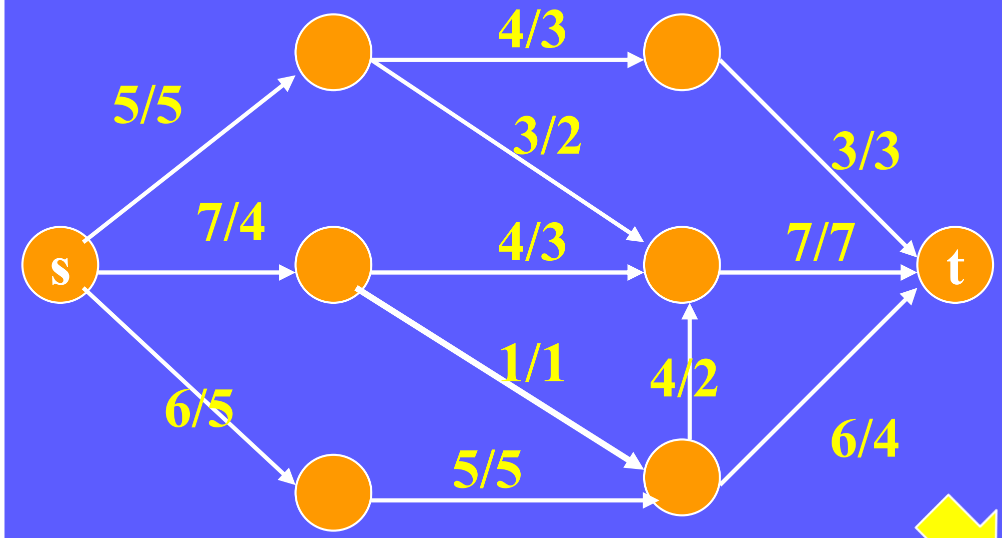


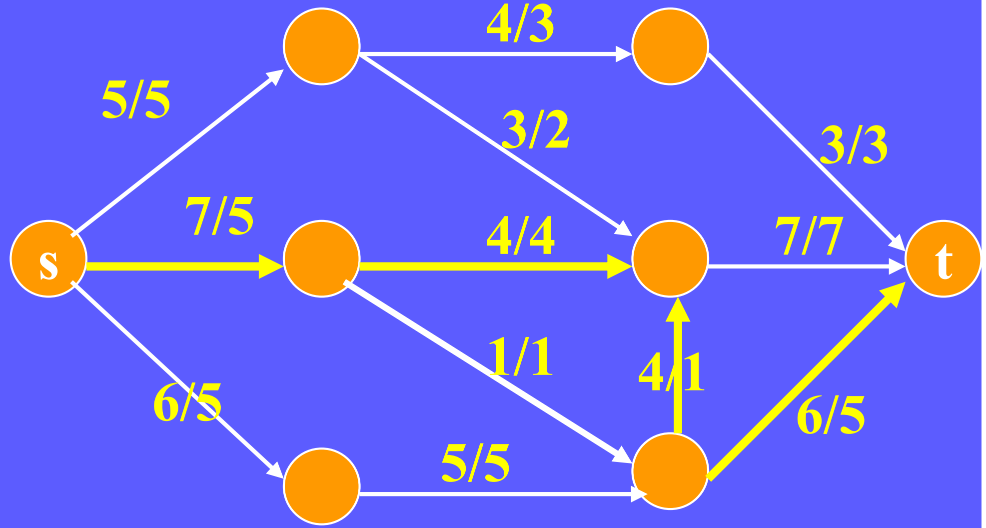
increase 1



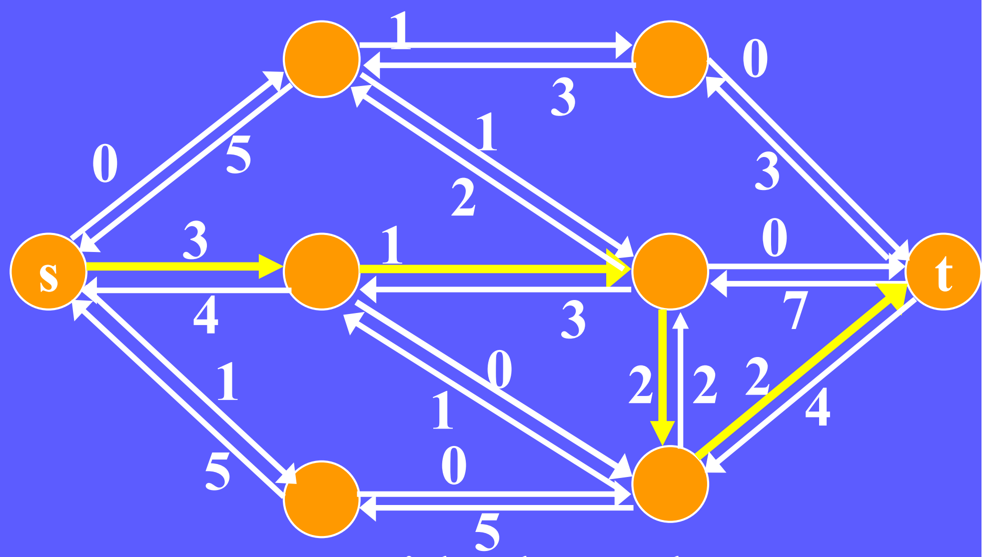
residual graph

An Example (cont.)



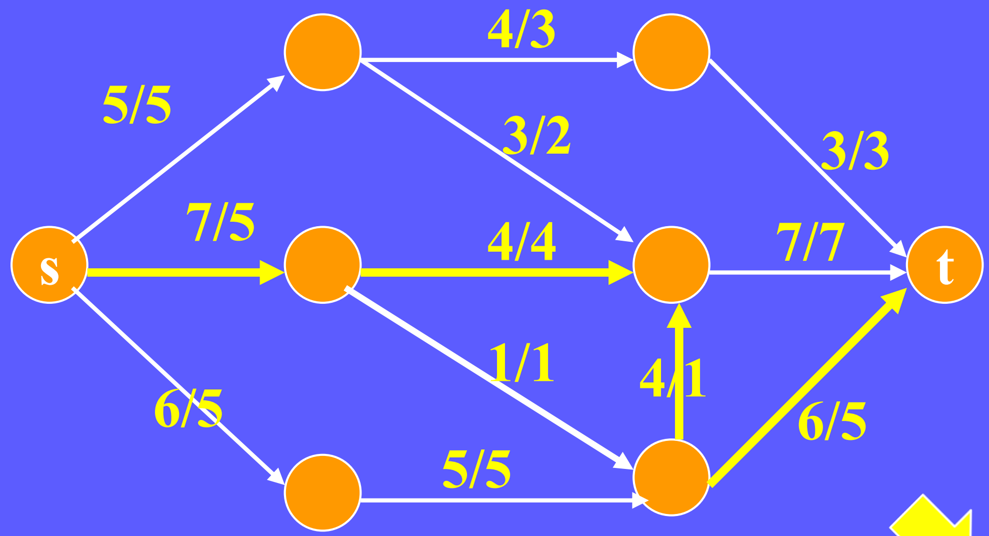


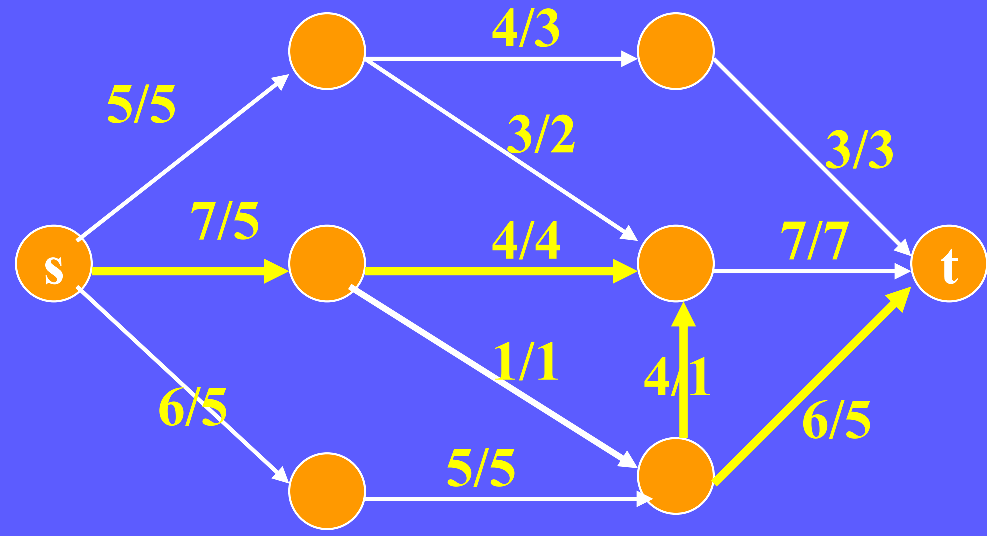
increase 1



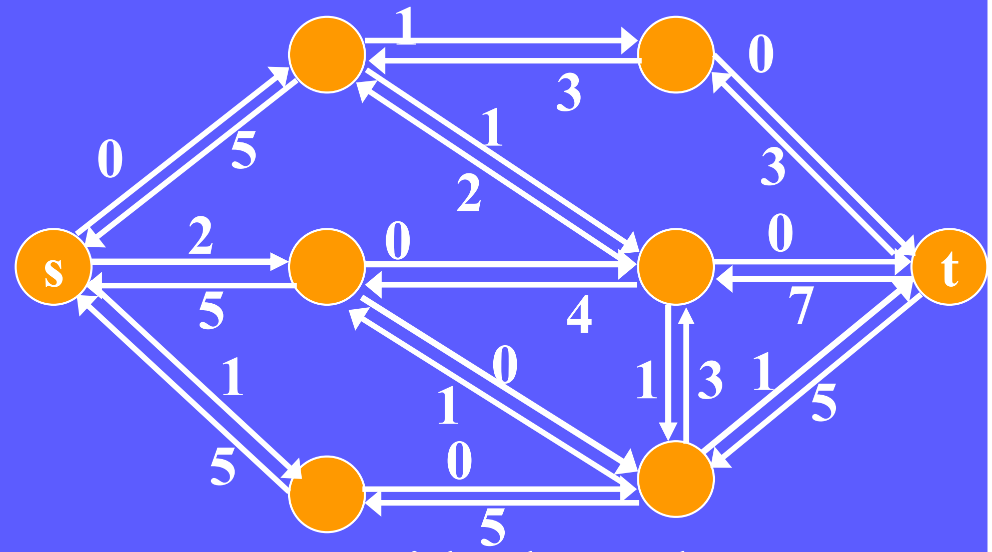
residual graph

An Example (cont.)





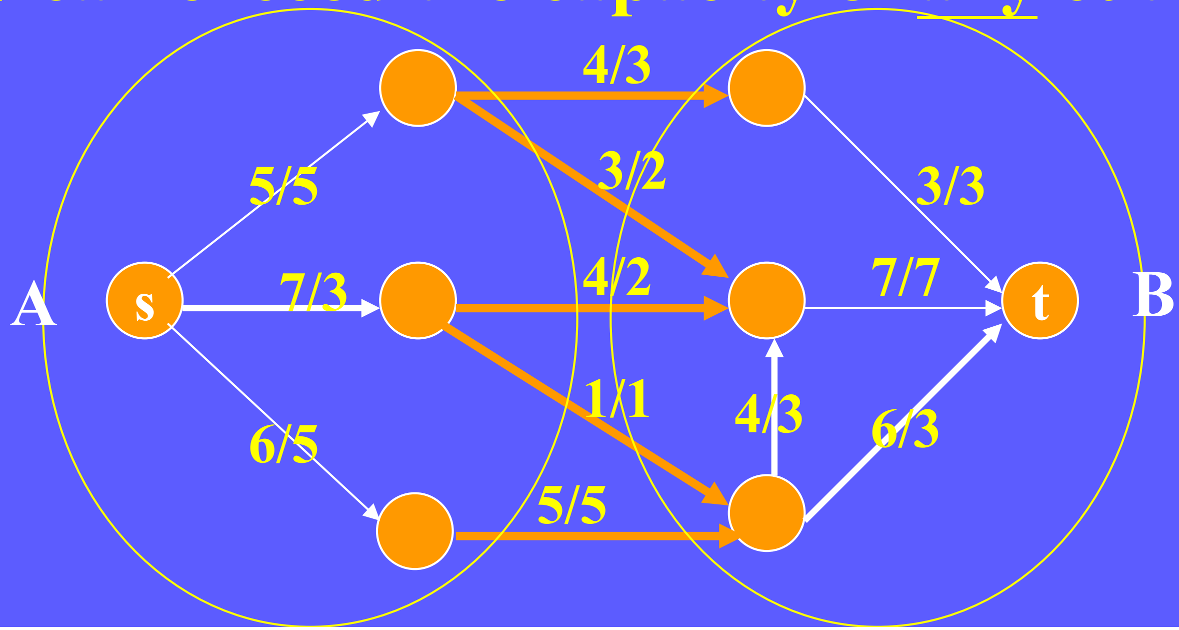
maximum



residual graph

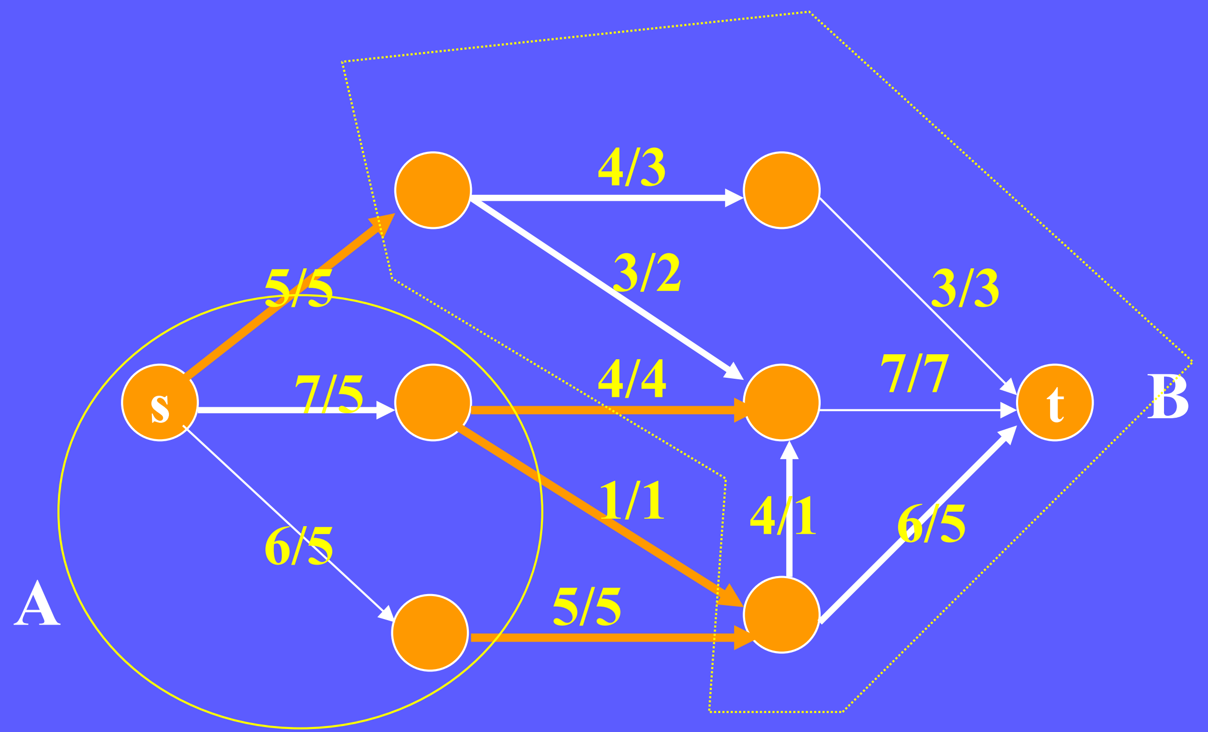
Cut in Graph

- cut: a set of edges that separate s from t
- precise definition of cut
  - let A be a set of vertices of V such that $s \in A$ and $t \notin A$
  - B=V-A, the rest of vertices
  - A cut is the set of edges $\{(v, w) \in E\}$ such that $v \in A$ and $w \in B$
- capacity of the cut: sum of capacities of its edges
- no flow can exceed the capacity of any cut



M. K. Shan, CS, NCCU

Augmenting-Path Theorem (cont.)



Augmenting-Path Theorem

- A flow is maximum iff it admits no augmenting path
- Proof

(1) if flow admits an augmenting path, then it is not maximum

(if a flow is maximum, it admits no augmenting path)

(2) if it admits no augmenting path, a flow is maximum

- if a flow admits no augmenting paths, it is equal to capacity of a cut
  - Let $A \subset V$ be a set of vertices such that $\forall\ v \in A$, $\exists$ an augmenting path, with respect to the flow f, from s to v
  - A defines a cut ($s \in A$, $t \notin A$)
  - for all edges (v, w) in that cut, f(v, w) = c(v,w)

    otherwise, (v, w) would be a forward edge, an augmenting path to w or a backward edge
- if a flow equals to capacity of a cut, it is maximum

More Theorems concerning Maximum Flow

- **Max-Flow Min-Cut Theorem**

  The value of a maximum flow in a network is equal to the minimum capacity of a cut

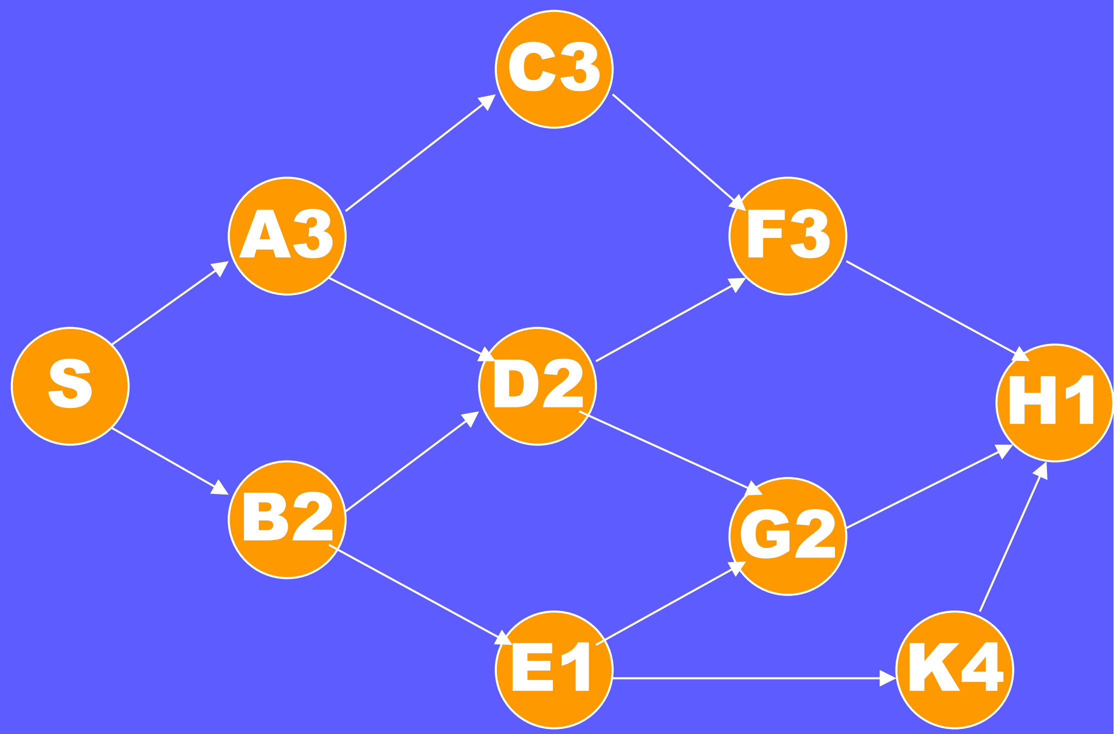
- **Integral-Flow Theorem**

  If the capacities of all edges in the network are integers then there is a maximum flow whose value is an integer

Critical Path

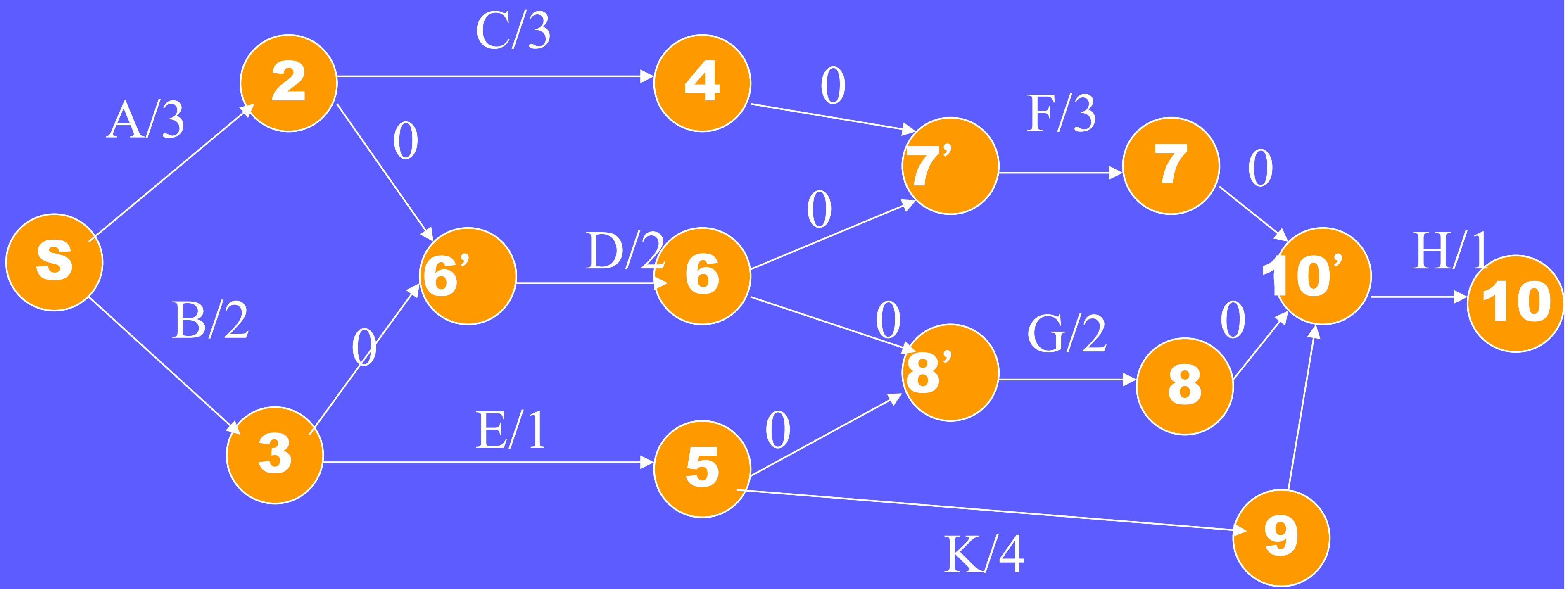
Critical Path Analysis

- **Activity-node graph**



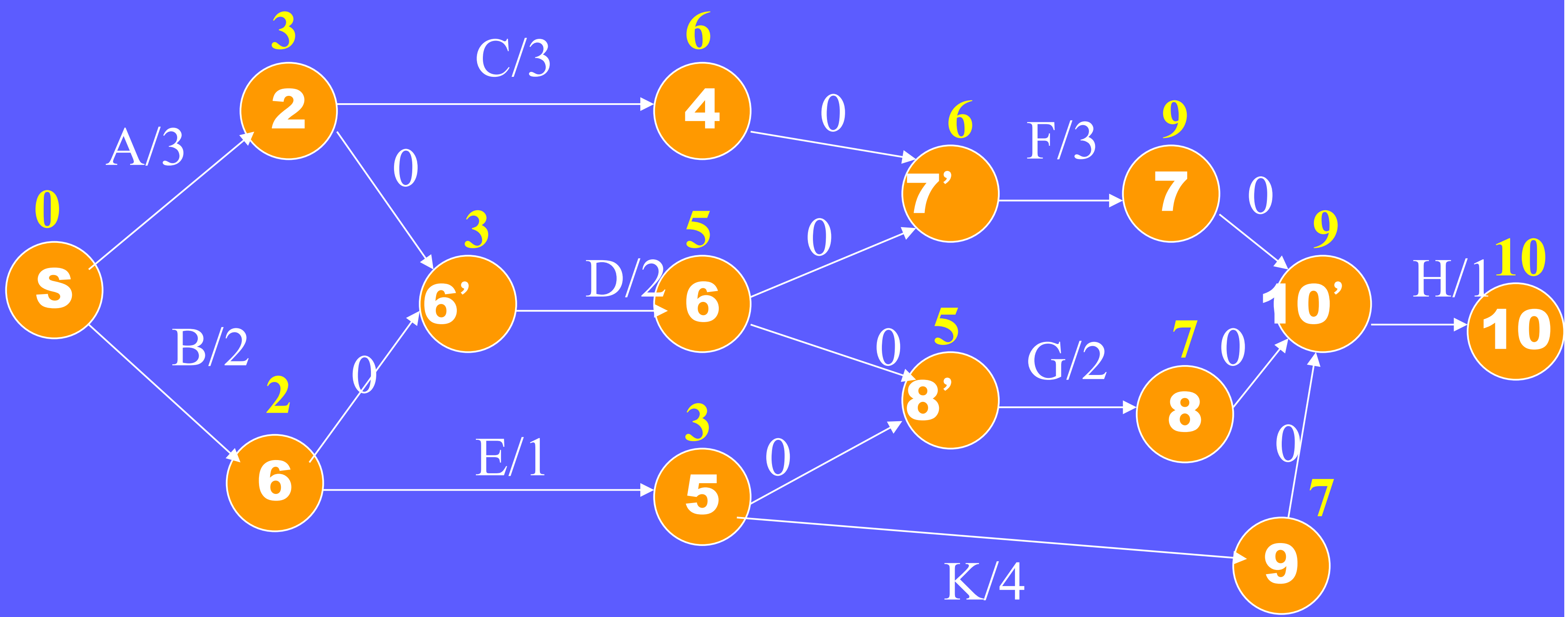
Critical Path Analysis (cont.)

- Event-node graph



Critical Path Analysis (cont.)

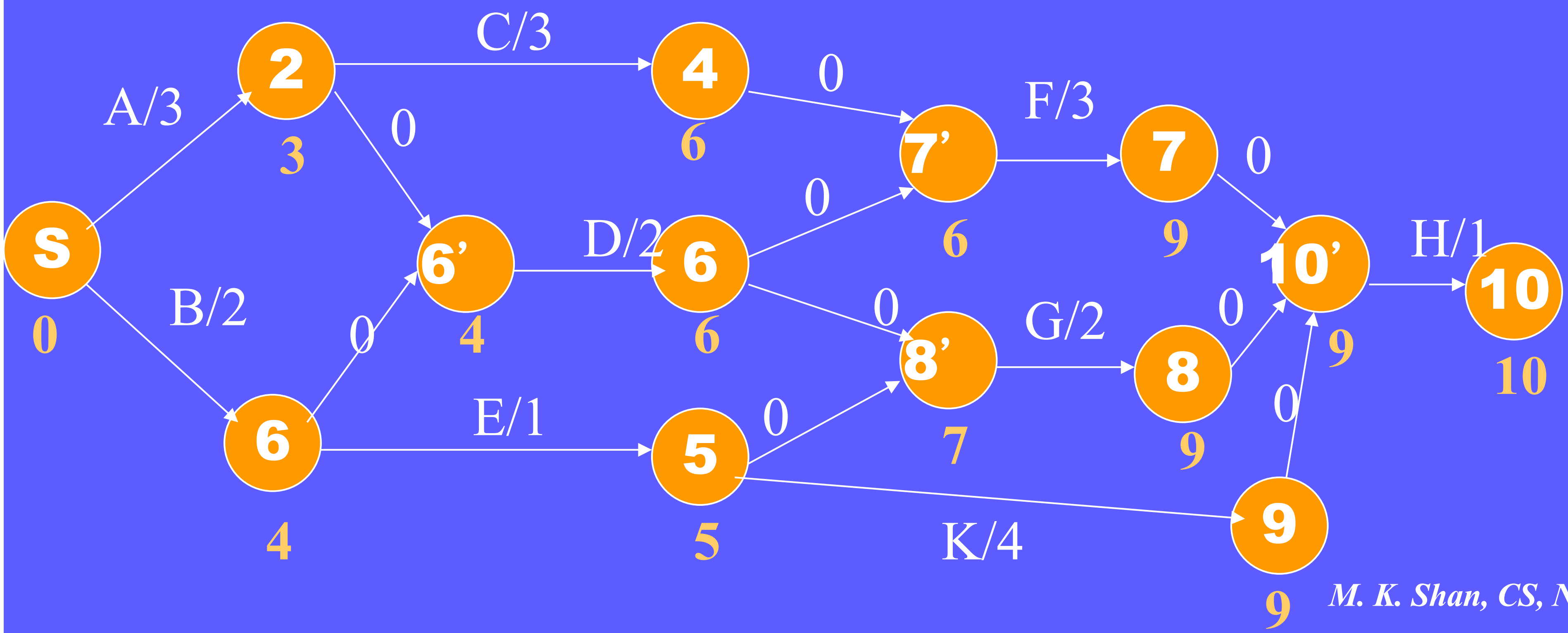
- **Earliest completion times: longest path**
  - **computed by topological order**
  - **EC_1=0**
  - **$EC_w=\max(EC_v+C_{v,w})$**



M. K. Shan, CS, NCCU

Critical Path Analysis (cont.)

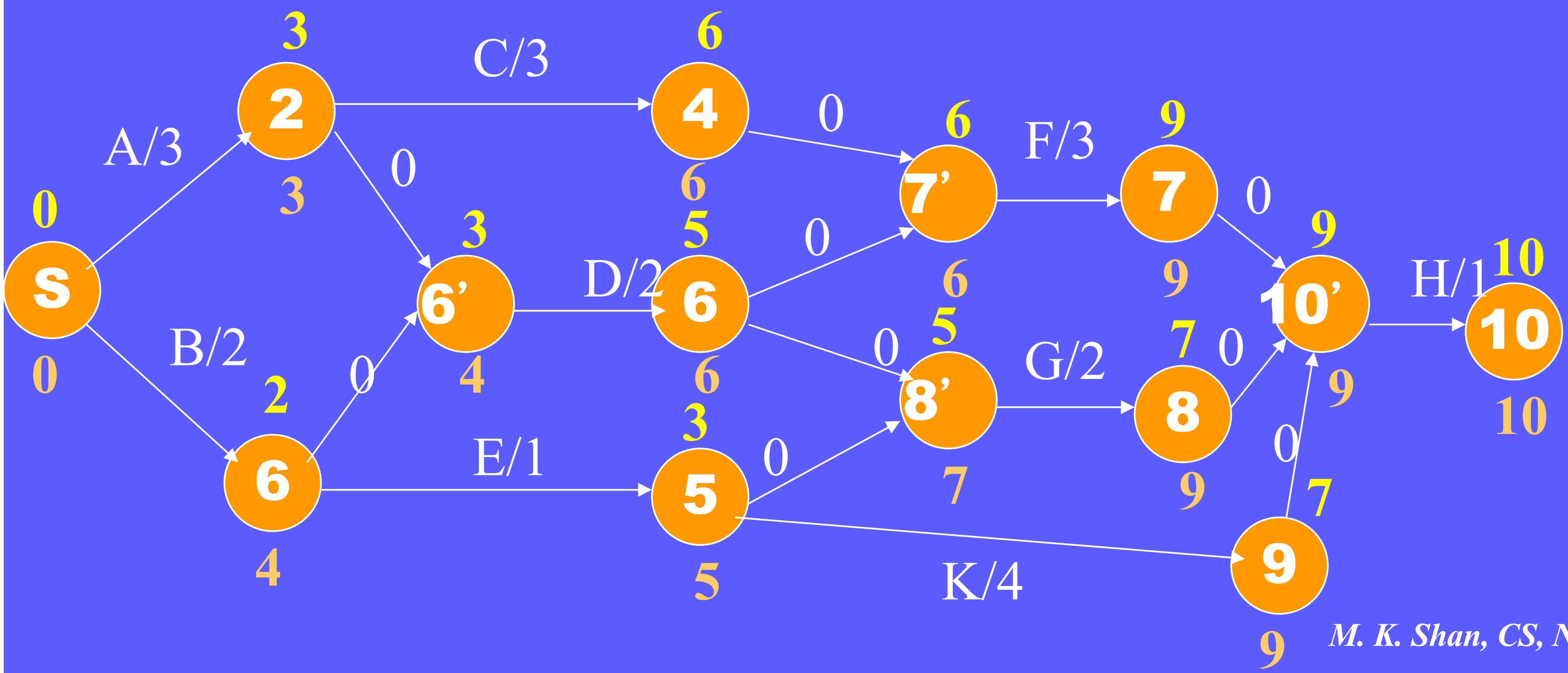
- **Latest completion times:**
  - **latest time without affecting final completion time**
  - **computed by reverse topological order**
  - $LC_n = EC_n$
  - $LC_v = \min(LC_w - C_{v,w})$



M. K. Shan, CS, NCCU

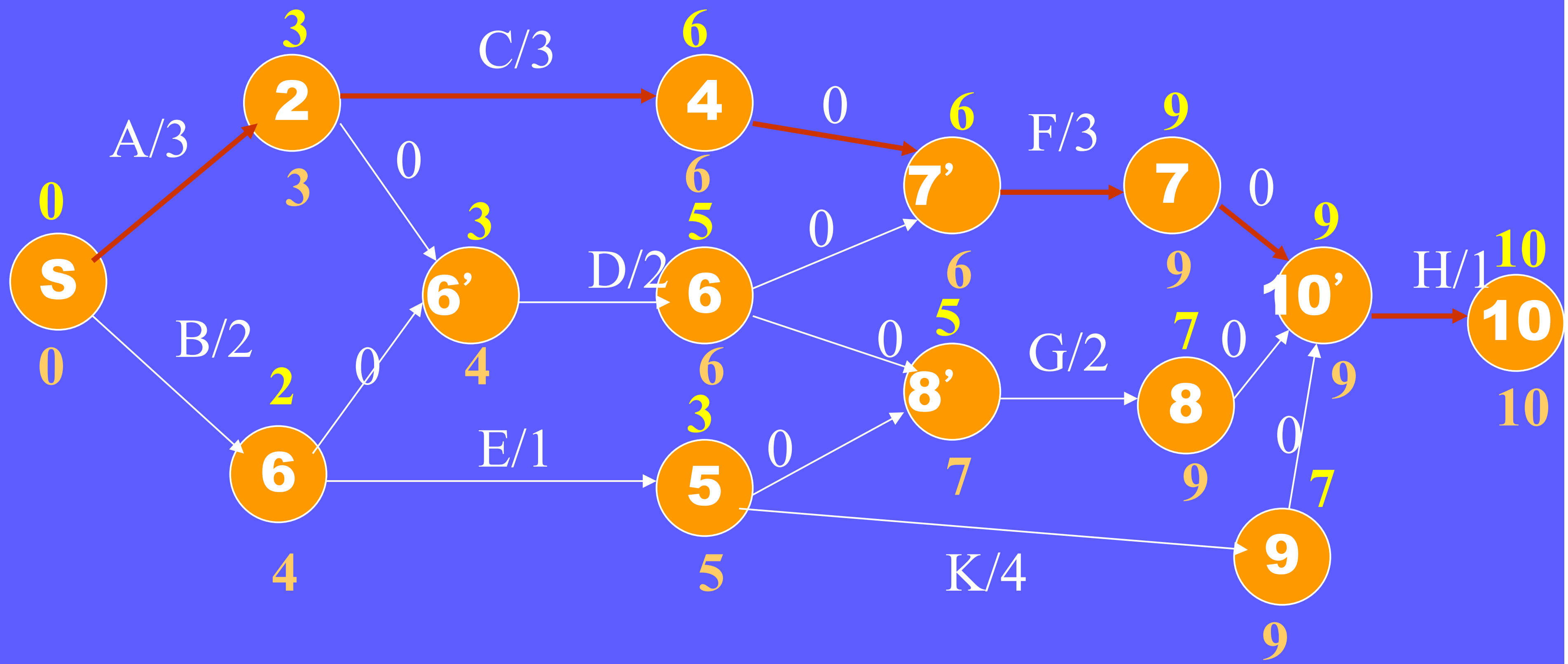
Critical Path Analysis (cont.)

- **Slack time(v,w)=$LC_w - EC_v - C_{v,w}$**



Critical Path Analysis (cont.)

- **Critical path = zero slack time**



Graph Decomposition

Decompositions of Graph

- Graph decomposition
  - partition graph into subgraphs

    such that

    each subgraph satisfies a certain desirable property.
  - examples
    - partition to connected components
    - partitioned undirected graph to biconnected components
    - partitioned directed graph to strongly connected components

18年前因為「這座電塔」 也造成全台大停電

18年前因為這座超高壓電塔倒塌，也造成全台大停電。（記者吳俊鋒攝）

1999/07/29
臺南縣左鎮
編號第326
輸電鐵塔傾斜
全台大停電

2017/08/15 22:33

〔記者吳俊鋒／台南報導〕桃園大潭電廠機組跳電，造成各縣市大規模停電，彷彿讓人回到18年前「729全台大停電」的夢魘！當時，台南市左鎮區澄山里過嶺附近一座345KV超高壓電塔倒塌，也造成全台大停電，郭姓在地居民回憶，當時發生時間是深夜，屋內外一片漆黑，街坊鄰居驚恐不已，甚至還謠傳「阿共ㄚ打過來了」！

1999年的7月29日，台電在左鎮區編號第326的輸電鐵塔，因連日下雨導致地基土壤流失，約於晚間11點半傾塌，中北部各發電廠因保護機制而跳脫，導致全台5分之4以上電廠因輸電系統低壓震盪跳機，引發全台大停電。

左鎮區郭姓民眾說，這座超高壓電塔所在位置，是名為「山豹」的部落，屬於砂質土與白堊土的混合地形，容易因豪雨沖刷而流失，基座不穩，造成傾斜。

729全台大停電事件，許多左鎮耆老仍記憶猶新，當時民眾發現停電事態嚴重，有人懷疑是附近龍崎區的超高壓變電所爆炸，或北部核電廠故障。

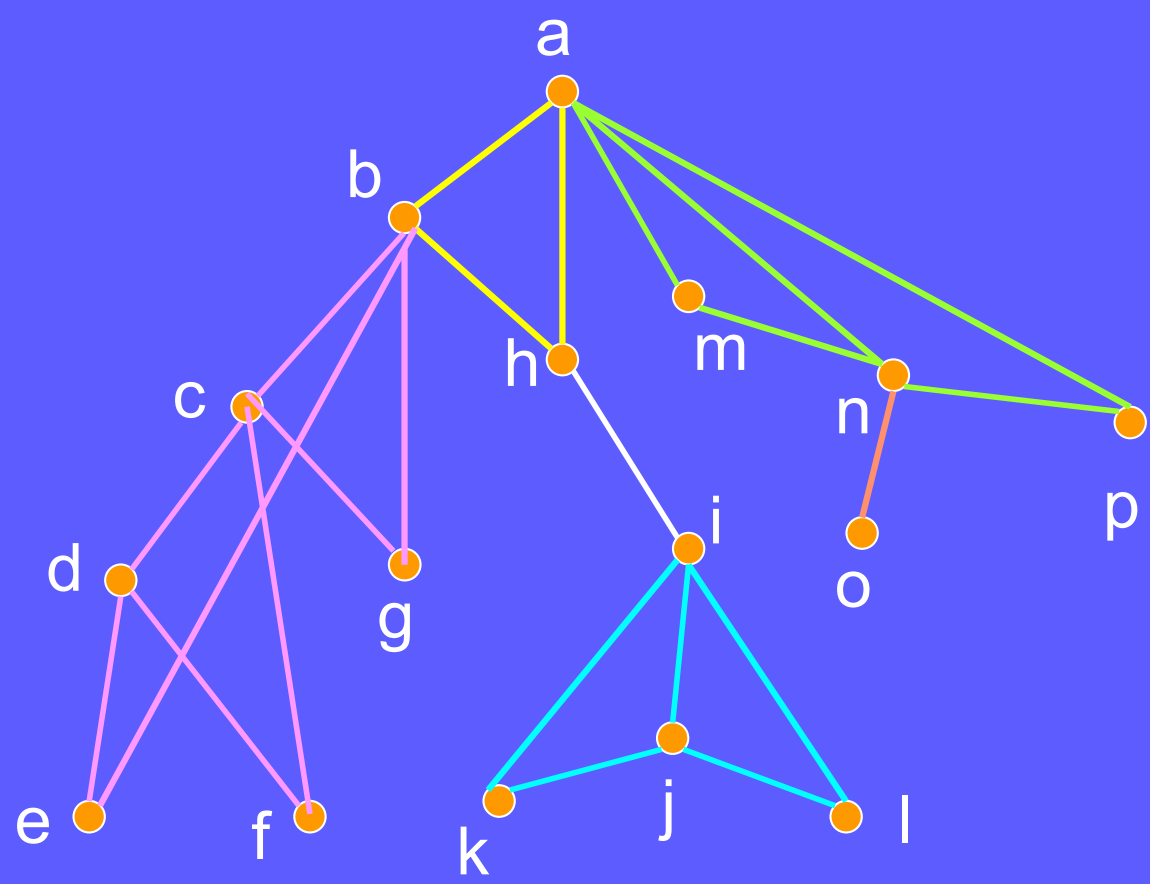
住在電塔附近的蔡姓婦人說，當時坊間還流傳是中國發射導彈攻擊，「兩岸開打了」，但也有人幫忙闢謠，斥為無稽之談，停電原因未明朗前，眾說紛紜。

後來台電努力修復倒塌電塔，並全面體檢山區超高壓電塔的基座，之前受制於用地取得不易，執行困難的第3迴路超高壓輸電幹線，也因這起事件而順利推動。

Biconnected Components

- Connected
  - An undirected graph is connected
    if there is a path from every vertex to every other vertex
- Biconnected
  - An undirected graph is biconnected
    if there are at least <u>two vertex disjoint paths</u>
    from every vertex to every other vertex
  - If a graph is not biconnected,
    then it can be partitioned into subgraphs,
    each of which is biconnected
- K-connected
  - An undirected graph is k-connected
    if there are at least k vertex disjoint paths between every two vertices.

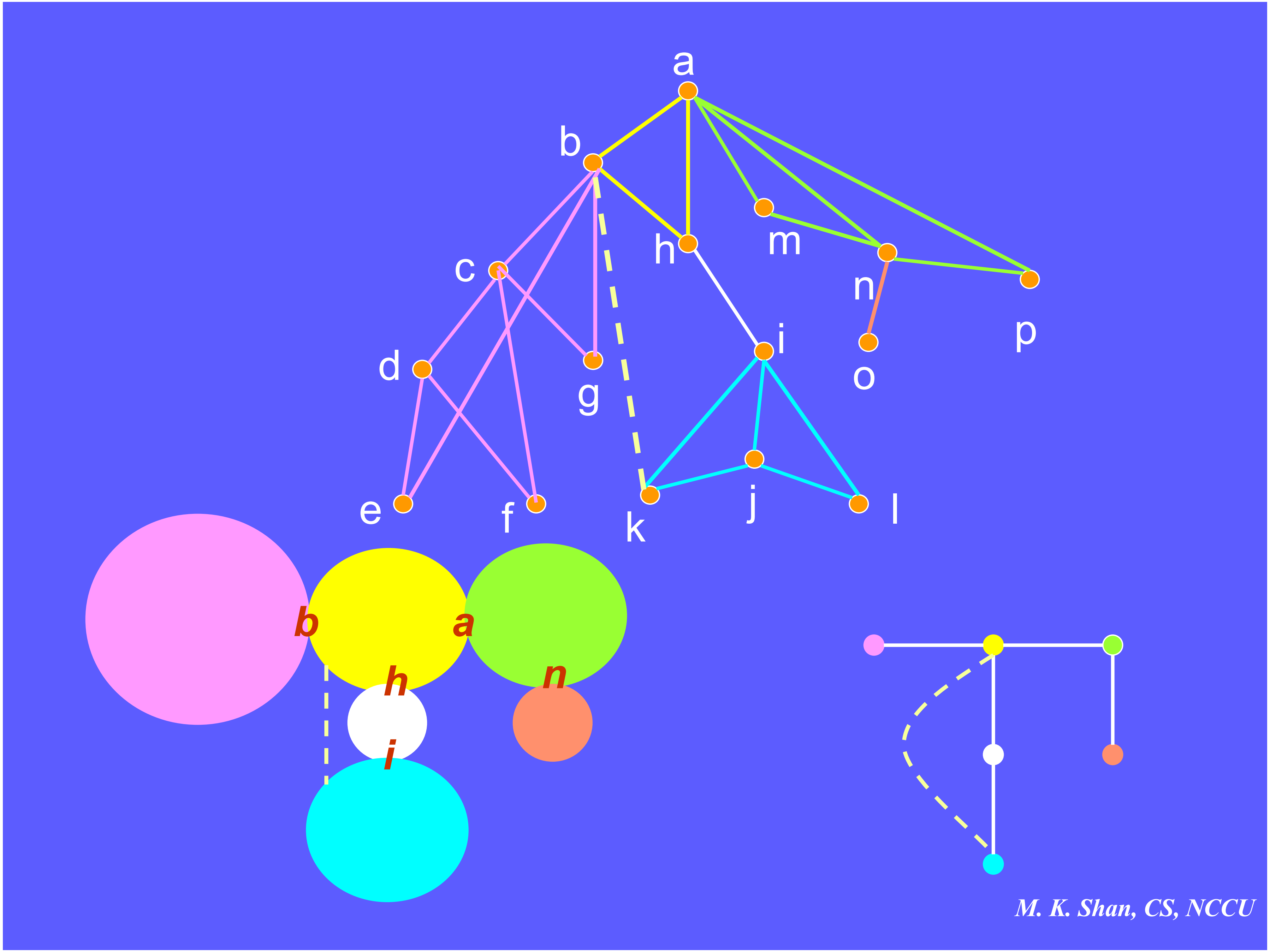
Biconnected Components (cont.)



Biconnected Components (cont.)

- **A graph is not biconnected**
  iff there is an articulation point
  - **<u>Articulation point</u>: vertex whose removal disconnects the graph**

- **Biconnected component**
  - **a maximal subset of the edges**
    such that its induced subgraph is biconnected


a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
b
a
h
n
i
M. K. Shan, CS, NCCU

Summary

- Graph Representation
- Graph Traversal (DFS, BFS)
- Finding Cycle in a Graph
- Topological Sorting
- Shortest Path
- Minimum Spanning Tree
- Graph Matching
- Graph Coloring
- Maximum Flow
- Graph Partitioning