



CSE 421

Introduction to Algorithms

Lecture 18

Winter 2024

Network Flow, Part 2

Outline

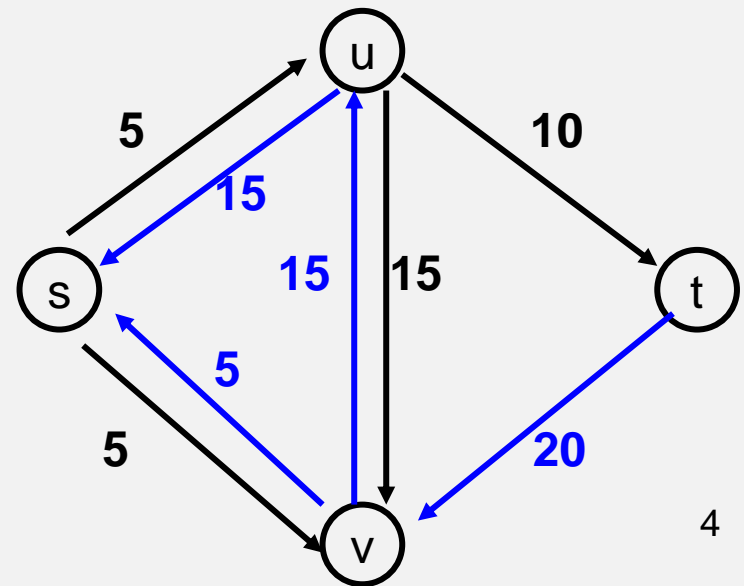
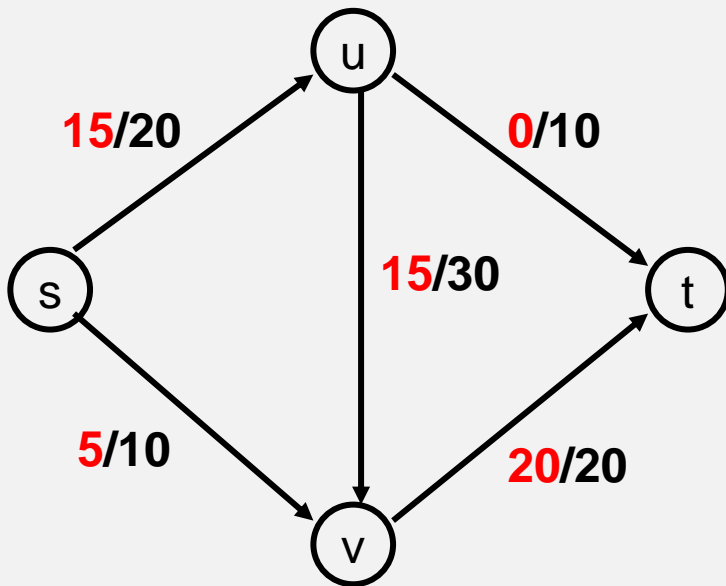
- ~~Network flow definitions~~
- ~~Flow examples~~
- ~~Augmenting Paths~~
- ~~Residual Graph~~
- ~~Ford Fulkerson Algorithm~~
- Cuts
- Maxflow-MinCut Theorem
- Worst Case Runtime for FF
- Improving Runtime bounds
 - Capacity Scaling
 - Fully Polynomial Time Algorithms

Network Flow Definitions

- Flowgraph: Directed graph with distinguished vertices s (source) and t (sink)
- Capacities on the edges, $c(e) \geq 0$
- Problem, assign flows $f(e)$ to the edges such that:
 - $0 \leq f(e) \leq c(e)$
 - Flow is conserved at vertices other than s and t
 - Flow conservation: flow going into a vertex equals the flow going out
 - The flow leaving the source is as large as possible

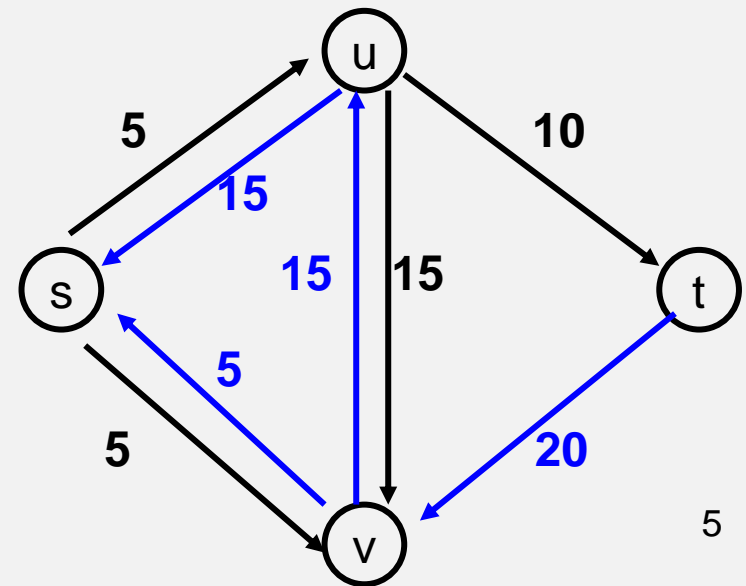
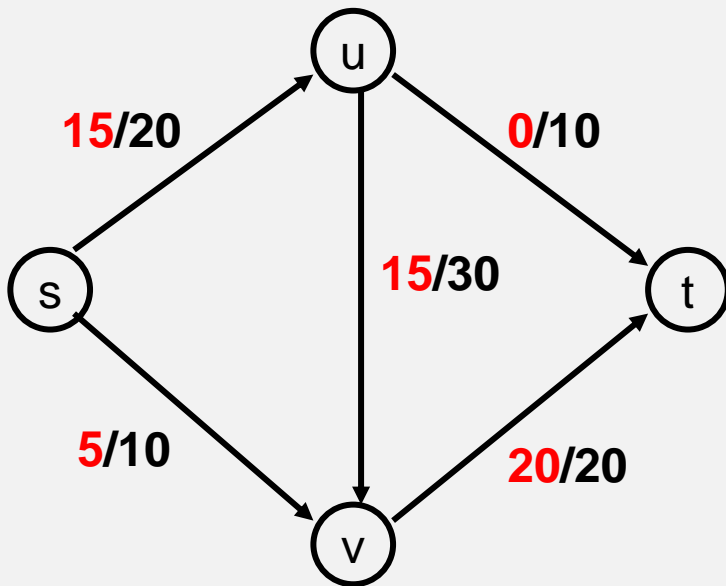
Residual Graph

- Flow graph showing the remaining capacity
- Flow graph G , Residual Graph G_R
 - G : edge e from u to v with capacity c and flow f
 - G_R : edge e' from u to v with capacity $c - f$
 - G_R : edge e'' from v to u with capacity f



Augmenting Path Algorithm

- Augmenting path in residual graph
 - Vertices v_1, v_2, \dots, v_k
 - $v_1 = s, v_k = t$
 - Possible to add b units of flow between v_j and v_{j+1} for $j = 1 \dots k-1$



Ford-Fulkerson Algorithm (1956)

while not done

 Construct residual graph G_R

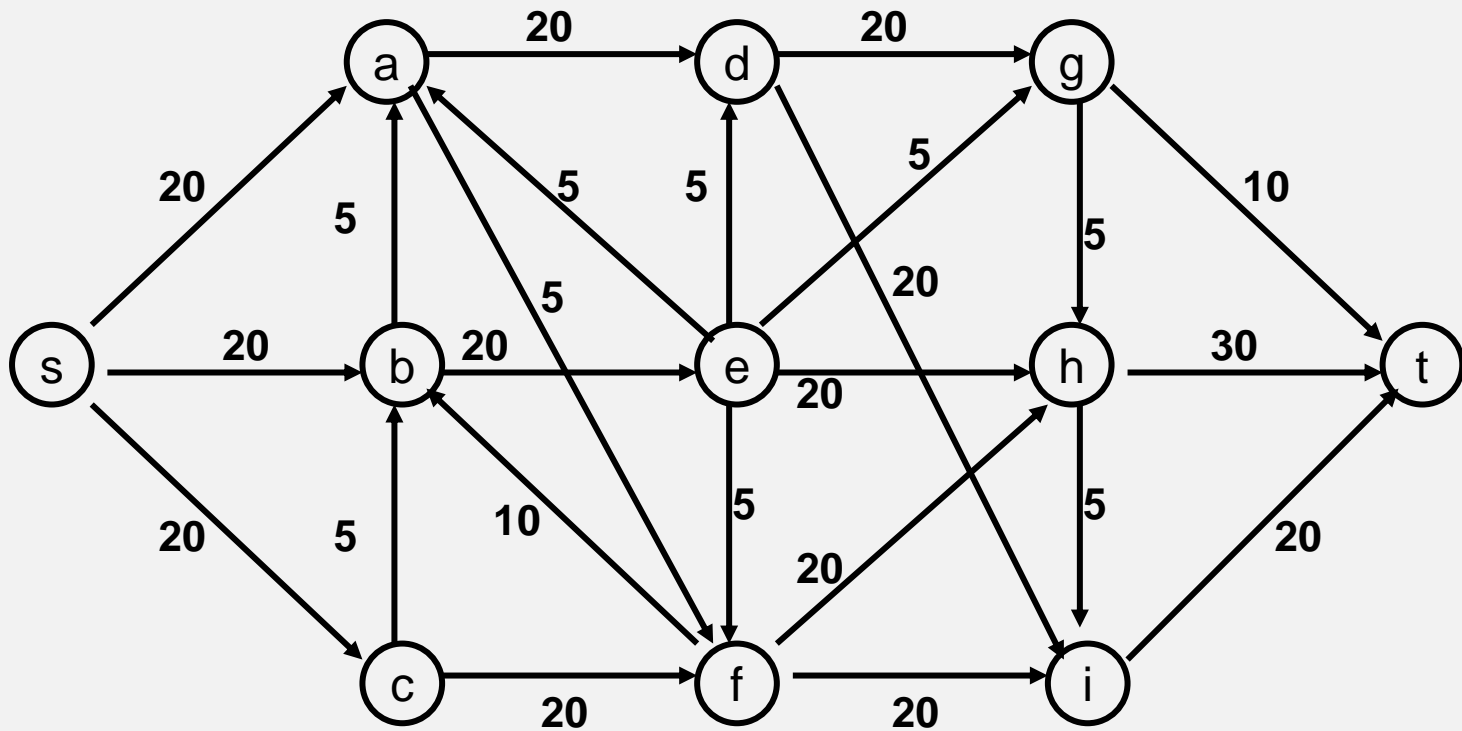
 Find an s-t path P in G_R with capacity $b > 0$

 Add b units of flow along path P in G

Runtime Analysis

- Assume the capacities are integers*
- Let C be the sum of edge capacities leaving s
- The total flow F is at most C
- Every iteration increases flow by at least 1, so there are at most C iterations
- Cost per iteration is $O(m+n)$
- Runtime is $O(C(m+n))$

Flow Example



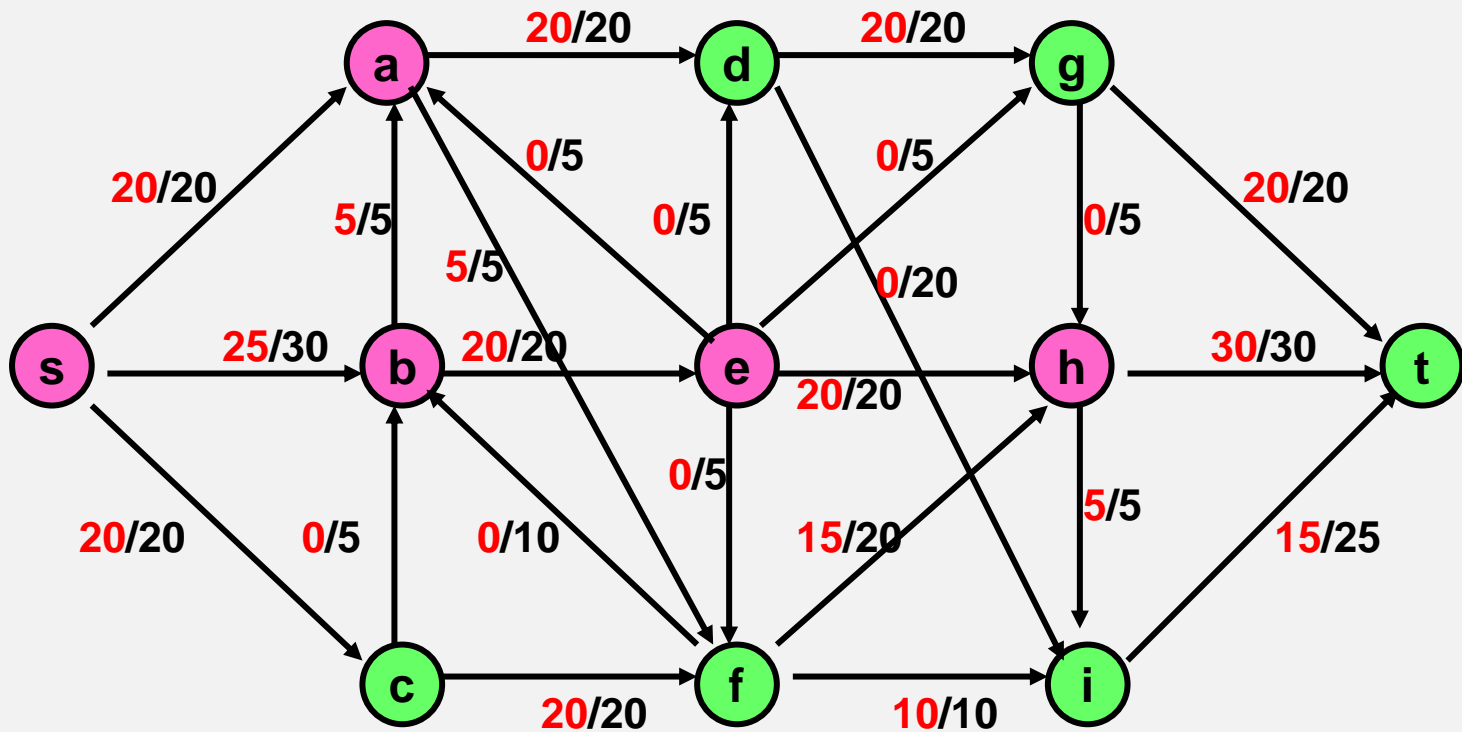
Cuts in a graph

- Cut: Partition of V into disjoint sets S , T with s in S and t in T .
- $\text{Cap}(S,T)$: sum of the capacities of edges from S to T
- $\text{Flow}(S,T)$: net flow out of S
 - Sum of flows out of S minus sum of flows into S

- $\text{Flow}(S,T) \leq \text{Cap}(S,T)$

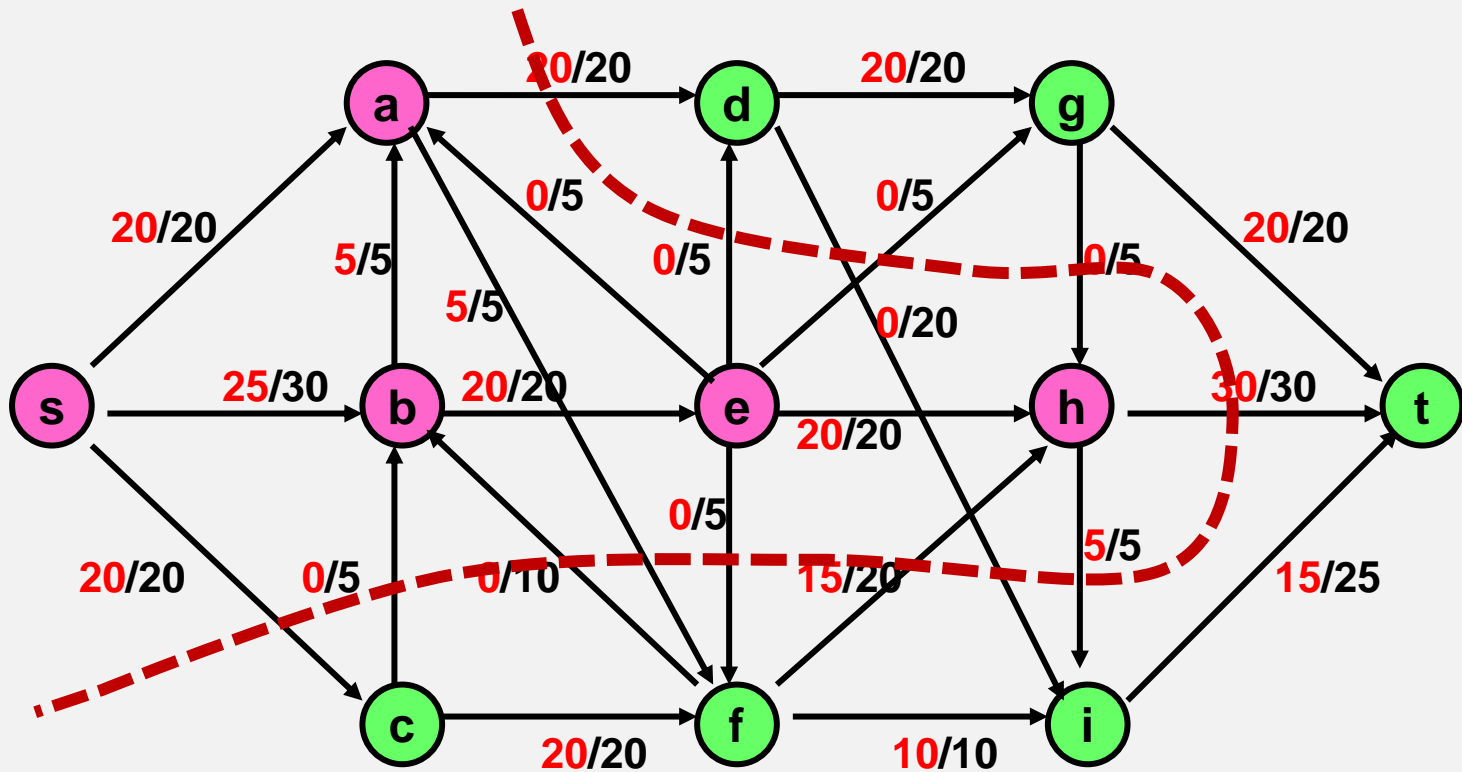
What is $\text{Cap}(S,T)$ and $\text{Flow}(S,T)$

$S = \{s, a, b, e, h\}$, $T = \{c, f, i, d, g, t\}$



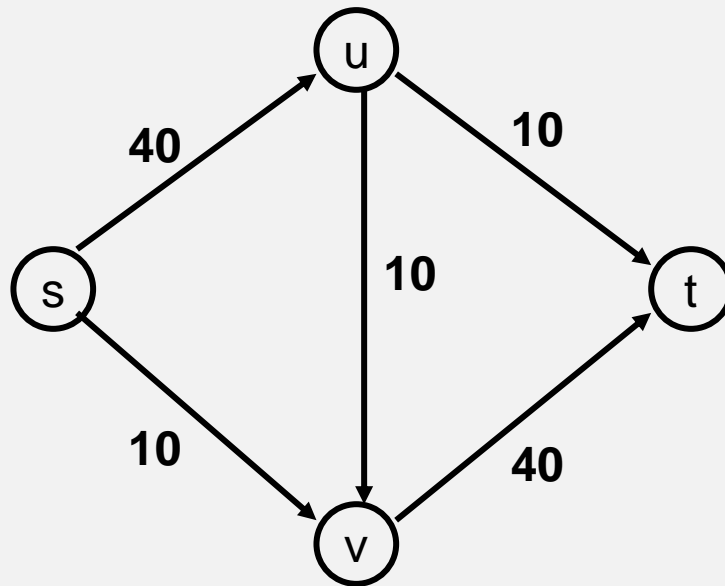
What is $\text{Cap}(S,T)$ and $\text{Flow}(S,T)$

$S = \{s, a, b, e, h\}$, $T = \{c, f, i, d, g, t\}$

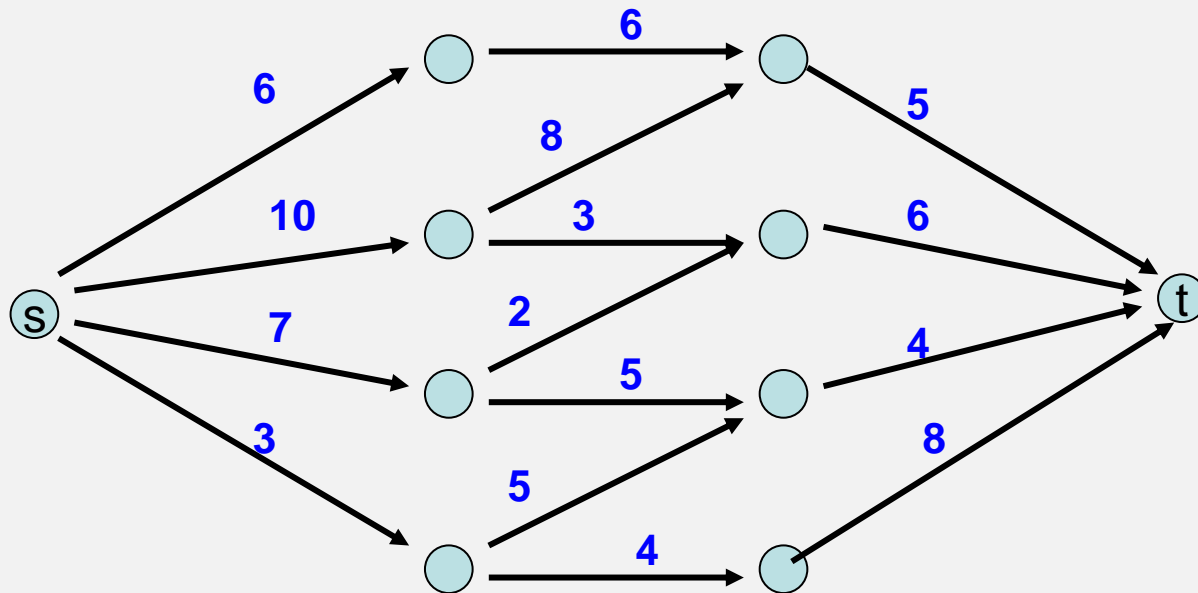


$\text{Cap}(S,T) = 95$, $\text{Flow}(S,T) = 80 - 15 = 65$

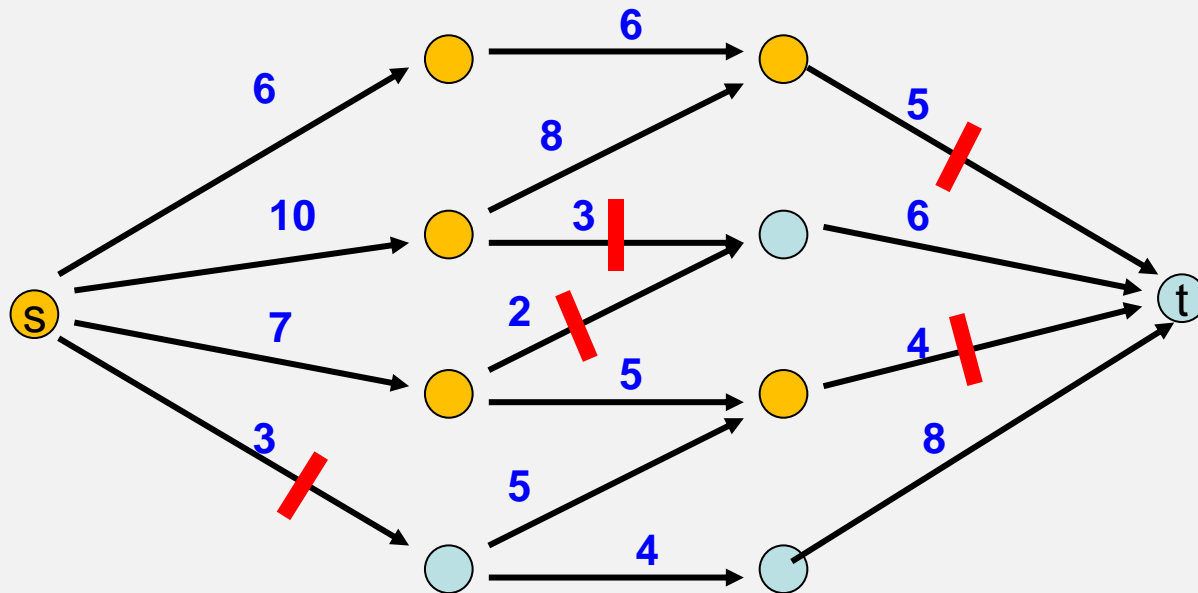
Minimum value cut



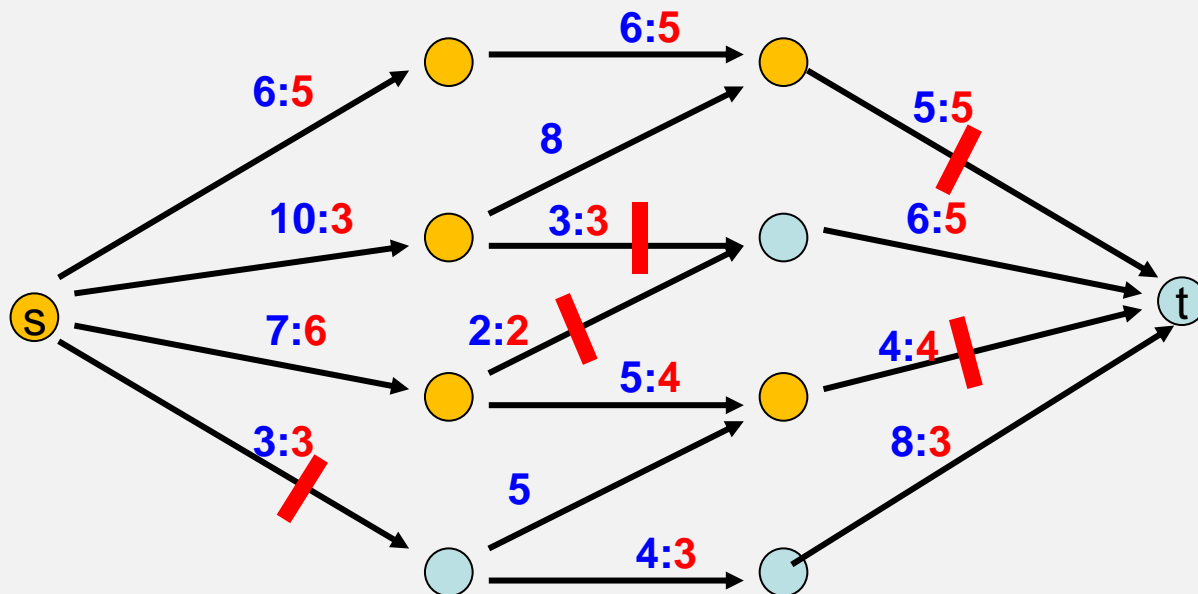
Find a minimum value cut



Find a minimum value cut

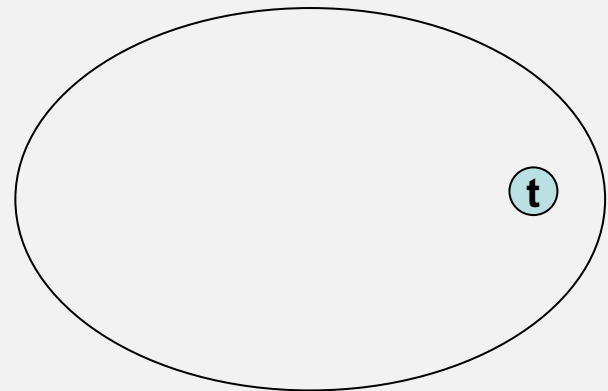
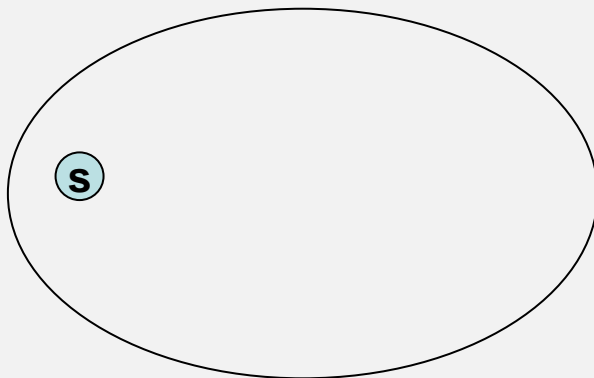


Find a minimum value cut

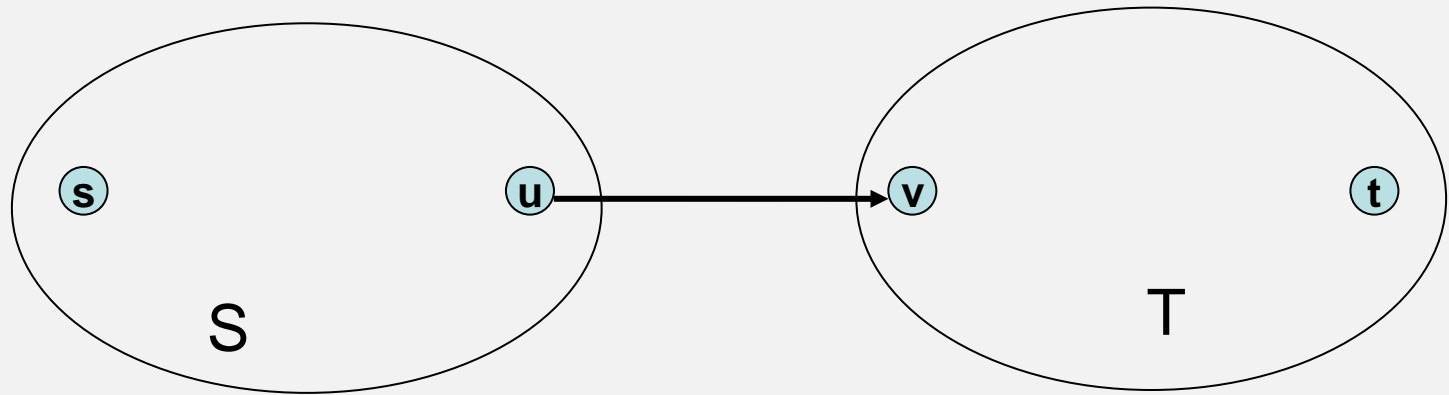


MaxFlow – MinCut Theorem

- There exists a flow which has the same value as the minimum cut
- Proof: Consider a flow where the residual graph has no s-t path with positive capacity
- Let S be the set of vertices in G_R reachable from s with paths of positive capacity



Let S be the set of vertices in G_R reachable from s with paths of positive capacity



What can we say about the flows and capacity between u and v ?

Max Flow - Min Cut Theorem

- Ford-Fulkerson algorithm finds a flow where the residual graph is disconnected, hence FF finds a maximum flow.
- If we want to find a minimum cut, we begin by looking for a maximum flow.

History

- Ford / Fulkerson studied network flow in the context of the Soviet Rail Network

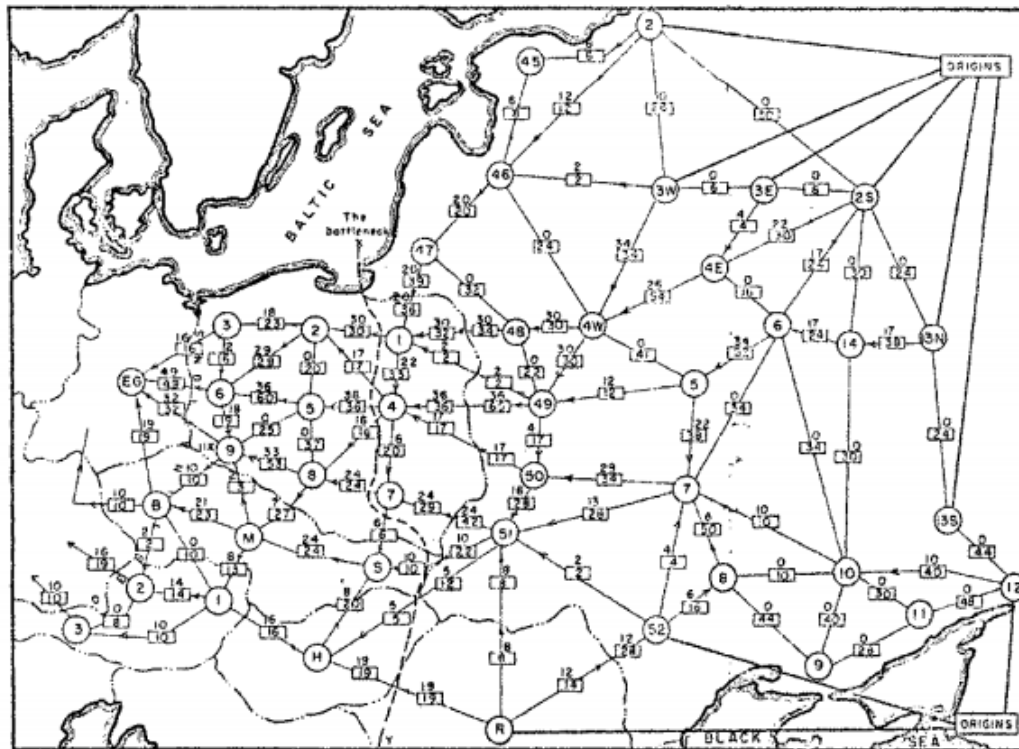


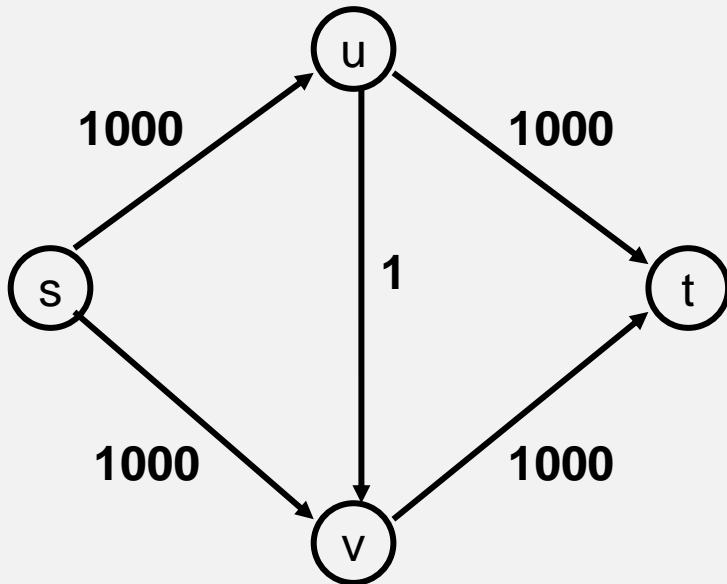
Figure 9

Ford Fulkerson Runtime

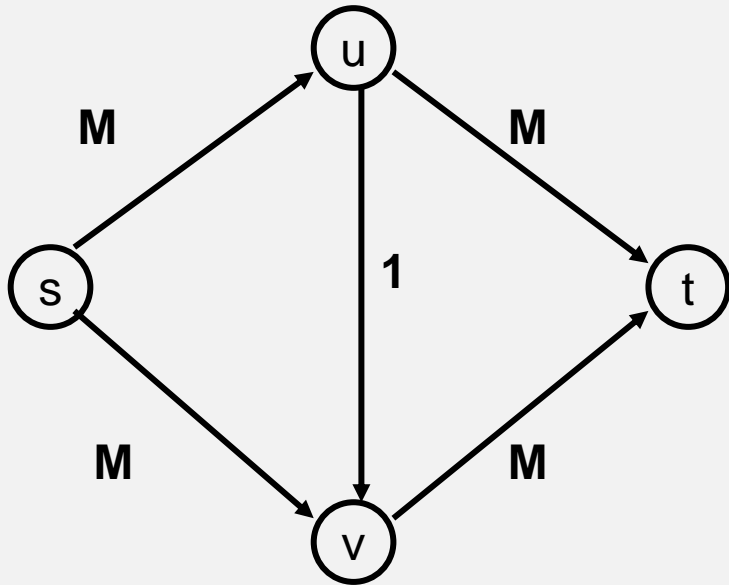
- Cost per phase \times number of phases
- Phases
 - Capacity leaving source: C
 - Add at least one unit per phase
- Cost per phase
 - Build residual graph: $O(m)$
 - Find s-t path in residual: $O(m)$

Performance

- The worst case performance of the Ford-Fulkerson algorithm is horrible



Improving path selection



Better methods of finding augmenting paths

- Find the maximum capacity augmenting path
 - $O(m^2 \log(C))$ time algorithm for network flow
- Find the shortest augmenting path
 - $O(m^2 n)$ time algorithm for network flow
- Find a blocking flow in the residual graph
 - $O(mn \log n)$ time algorithm for network flow

Polynomial Time Algorithms

- Input of size n , runtime $T(n) = O(n^k)$
- Input size measures
 - Bits of input
 - Number of data items
- Maximum item size C
 - $O(Cn^k)$: Exponential
 - $O(n^k \log C)$: Polynomial
 - $O(n^k)$: Fully polynomial

Capacity Scaling Algorithm

- Choose $\Delta = 2^k$ such that all edges in G_R have capacity less than 2Δ

while $\Delta \geq 1$

 while there is a path P in G_R with capacity Δ

 Add Δ units of flow along path P in G

 Update G_R

$\Delta = \Delta / 2$

Analysis

- If capacities are integers, then graph is disconnected when $\Delta = \frac{1}{2}$
- If largest edge capacity is C , then there are at most $\log C$ outer phases
- At the start of each outer phase, the flow is within $2m\Delta$ of the maximum
 - So there are at most $2m$ inner phases for each Δ

Shortest Augmenting Path

- Find augmenting paths by BFS

for $k = 1$ to n

while there is a path P in G_R of length k and capacity $b > 0$

 Add b units of flow along path P in G

 Update G_R

- Need to show:
 - The length of the shortest augmenting path is non-decreasing
 - Each while loop finds at most m paths

Analysis

- Augmenting along shortest path from s to t does not decrease distance from s to t

Analysis

- The distance from s to t must increase in G_R after m augmentations by shortest paths

Improving the shortest augmenting path algorithm

- Find a blocking flow in one phase to increase the length of augmenting paths
 - Dinitz (Ефим Абрамович Диниц) Algorithm
 - $O(n^2m)$
- Dynamic Trees to decrease cost per augmentation
 - $O(nm \log n)$