Maximum flow problem

CE 377K

February 26, 2015

REVIEW

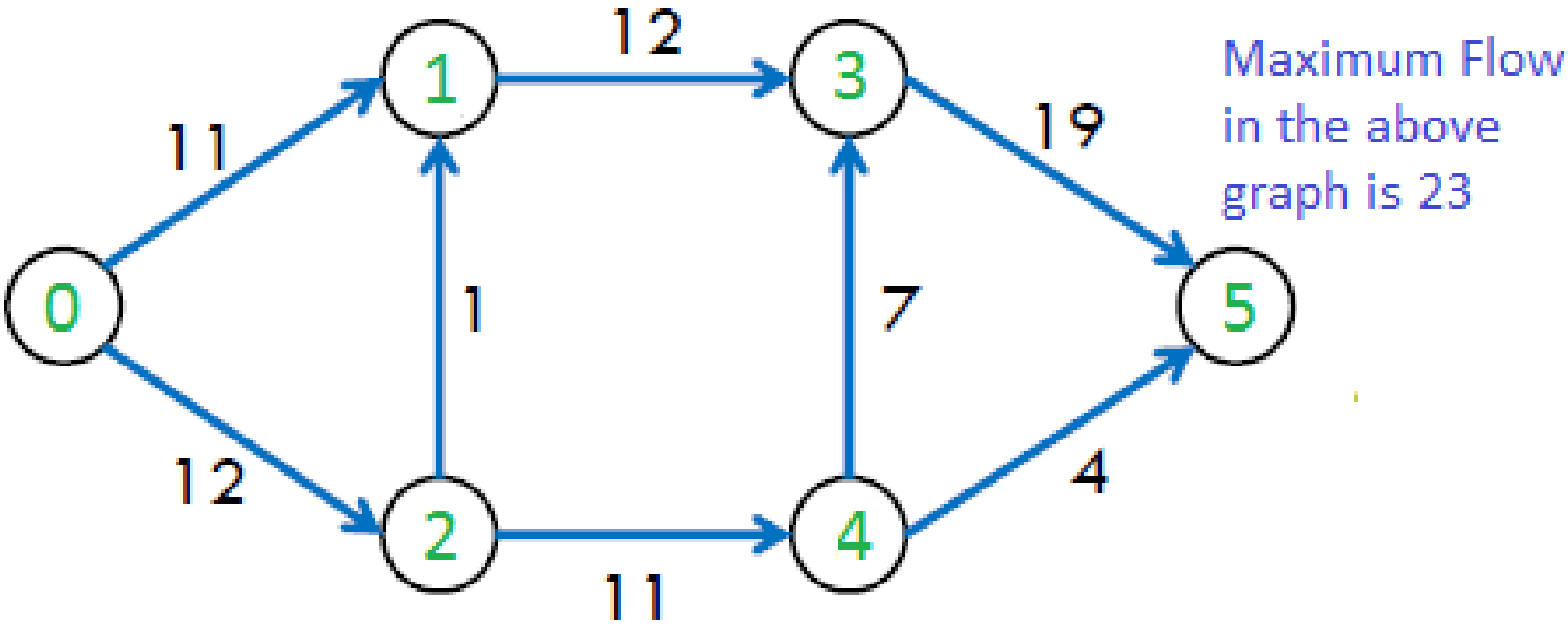
HW 2 due in 1 week

Label setting vs. label correcting

Bellman-Ford algorithm

MAXIMUM FLOW PROBLEM

Maximum Flow Problem



What is the greatest amount of flow that can be shipped between a given source and sink without exceeding link capacities?

Applications

- Determining the capacity of a network
- Identifying critical links in a network
- Rounding a matrix

In the maximum flow problem, we are given...

A network $G = (N, A)$ with a specified "source" node r and "sink" node s.

Each link (i, j) has a *capacity* u_{ij}. (Costs are irrelevant in the maximum flow problem.)

The objective is to ship as much flow as possible from r to s while respecting the link capacities.

Formulation

$$\max_{b,\mathbf{x}} \quad b$$

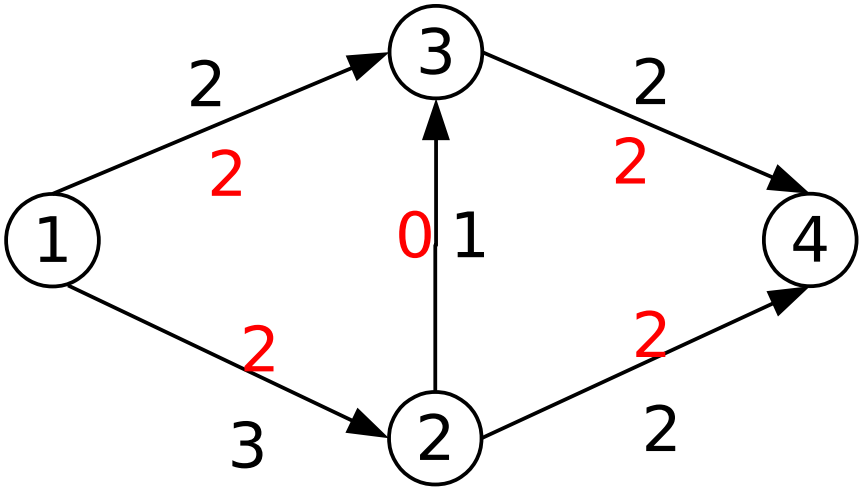
$$\text{s.t.} \quad \sum_{(h,i)\in A} x_{hi} - \sum_{(i,j)\in A} x_{ij} = \begin{cases} -b & i = r \\ +b & i = s \\ 0 & \text{otherwise} \end{cases}$$

$$0 \leq x_{ij} \leq u_{ij} \qquad \forall (i,j) \in A$$

A feasible solution $\mathbf{x}$ to this problem is called a *flow*. The amount of flow shipped is given by b.

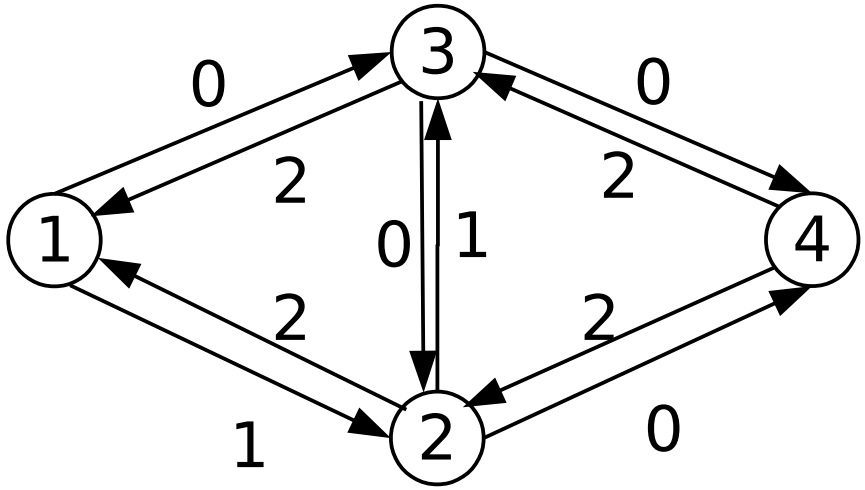
RESIDUAL GRAPHS

A very useful concept in solving maximum flow problems is the *residual graph* $\mathcal{R}(\mathbf{x})$



Capacity

Flow

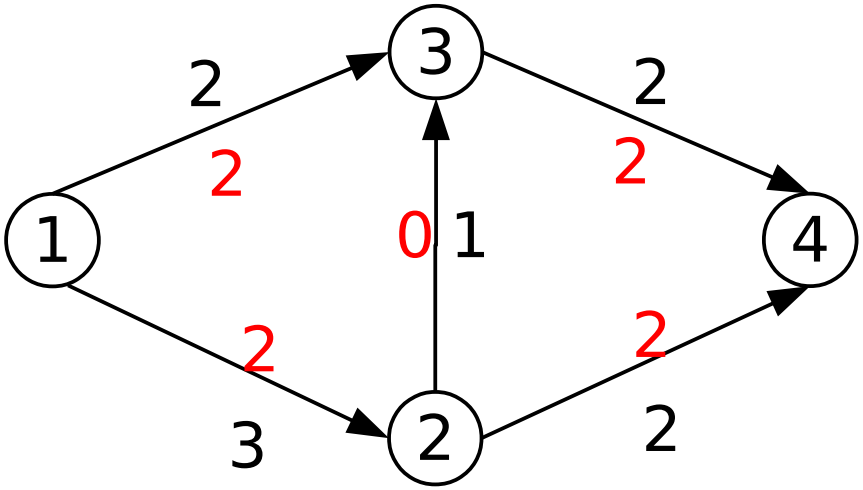


Residual Graph

$b = 4$

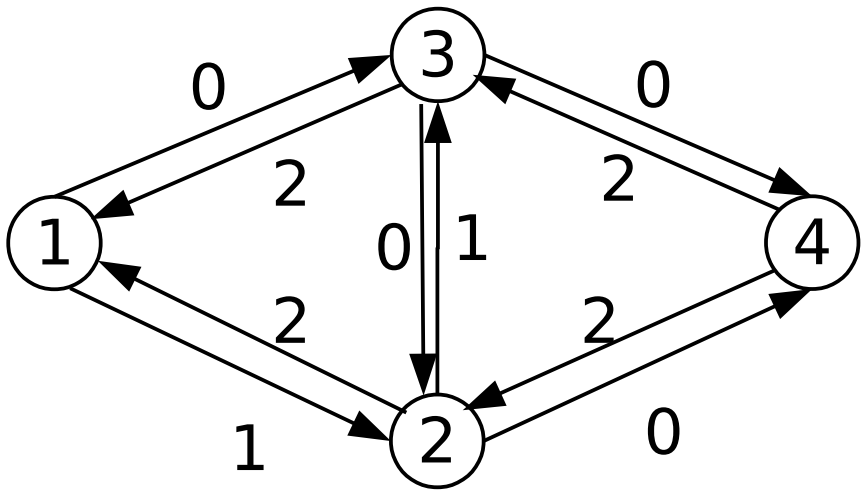
In a residual graph, each link is replaced by two links: a *forward link* with capacity $u_{ij} - x_{ij}$, and a *reverse link* with capacity x_{ij}.

Residual graphs tell us how we can change the flow **x** (usually with the aim of trying to add more.) Forward links tell us how much more flow can be added to a link; reverse links tell us how much can be removed.



Capacity

Flow



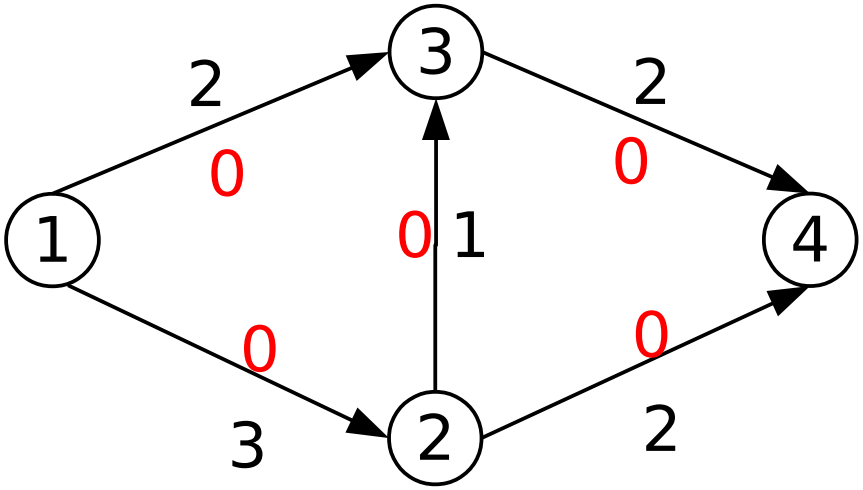
Residual Graph

$b = 4$

Note that the same network will have a different residual graph for each possible flow **x**.

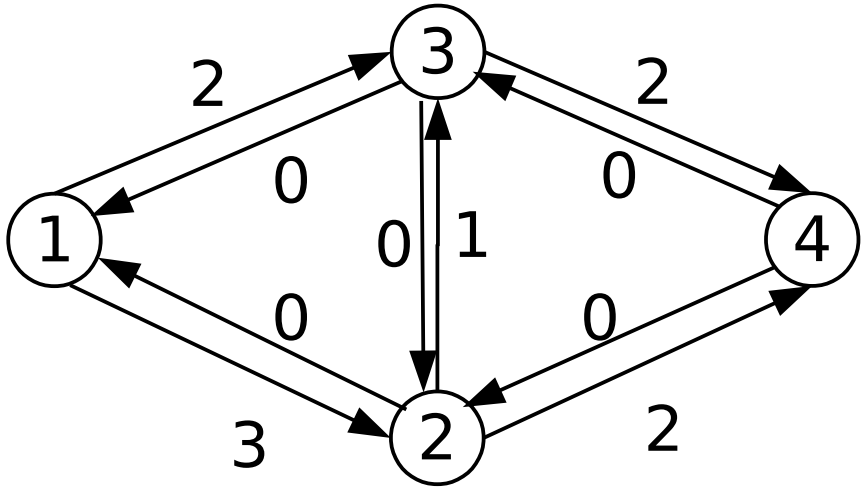
AUGMENTING PATH ALGORITHM

Example



Capacity

Flow



Residual Graph

Algorithm

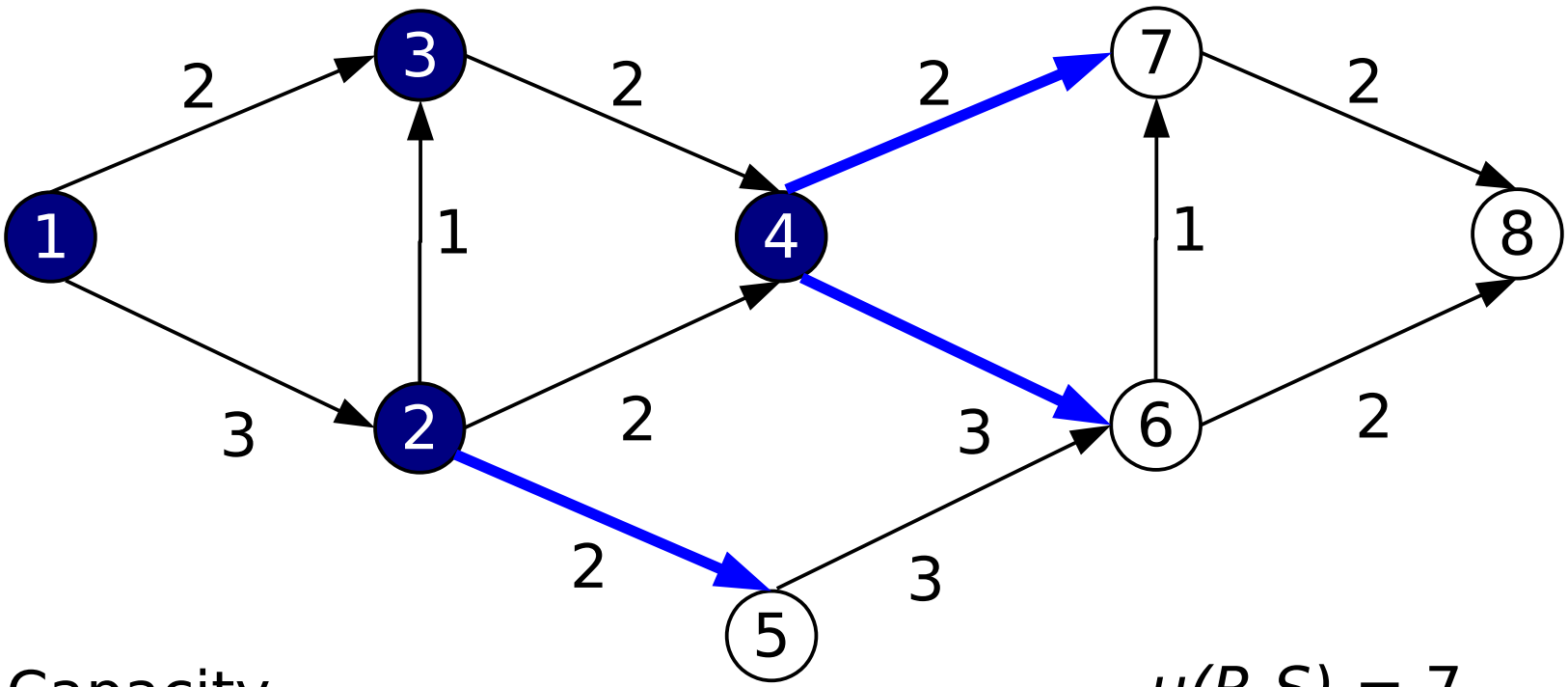
1. Initialize the flow with $\mathbf{x} \leftarrow \mathbf{0}$, $b^k \leftarrow 0$.
2. Construct the residual network $\mathcal{R}(\mathbf{x})$ corresponding to flows $\mathbf{x}$.
3. Identify a path π in $\mathcal{R}$ connecting s and t which has positive capacity on each of its arcs. (Terminate if no such path exists.)
4. Find the minimum capacity u^* of all of the arcs in π (whether forward or reverse).
5. For each forward arc (i, j) in π^k, $x_{ij} \leftarrow x_{ij} + u^*$. For each reverse arc (j, i) in π^k, $x_{ij} \leftarrow x_{ij} - u^*$.
6. Increase b by u^*, and return to step 2.

If you were writing code to do this, how would you implement step 3?

MAX-FLOW/MIN-CUT DUALITY

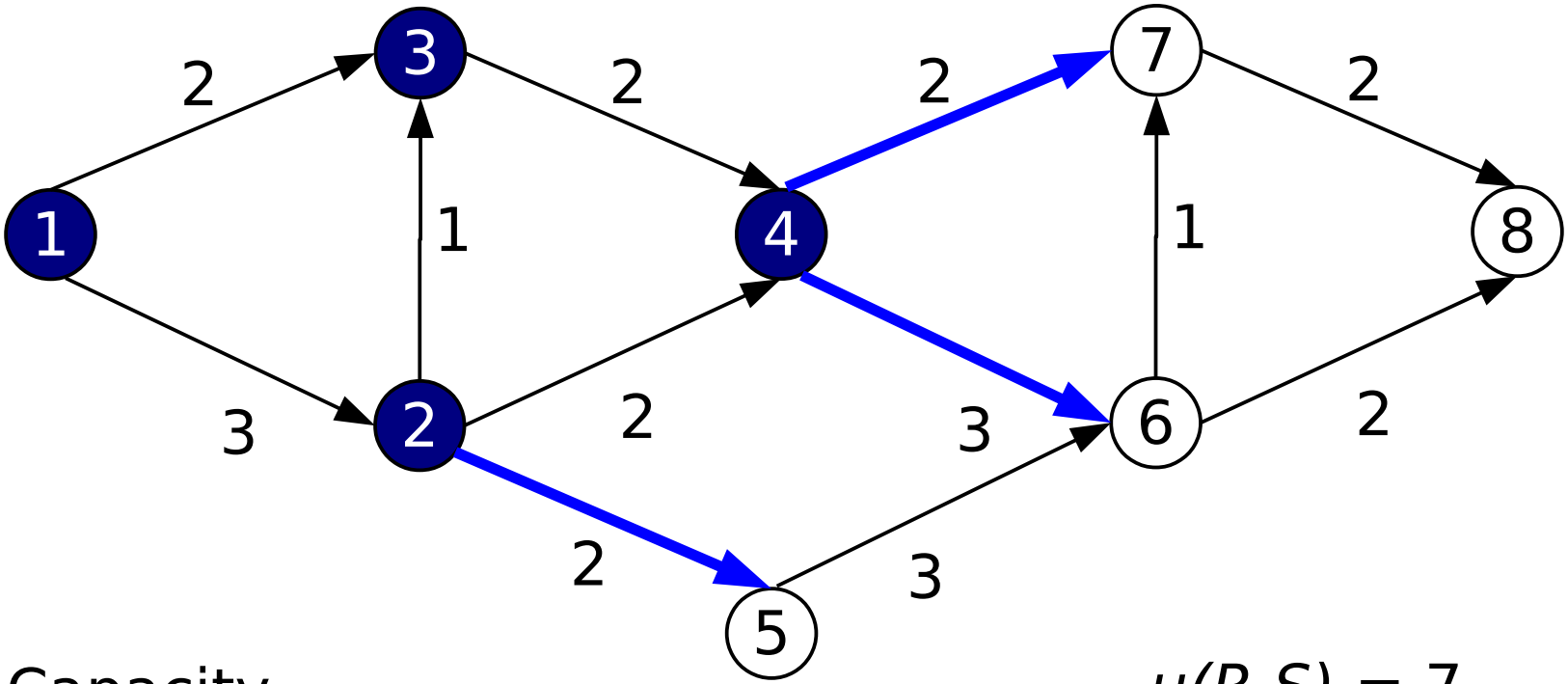
To prove correctness of the augmenting path algorithm, we'll take a short detour.

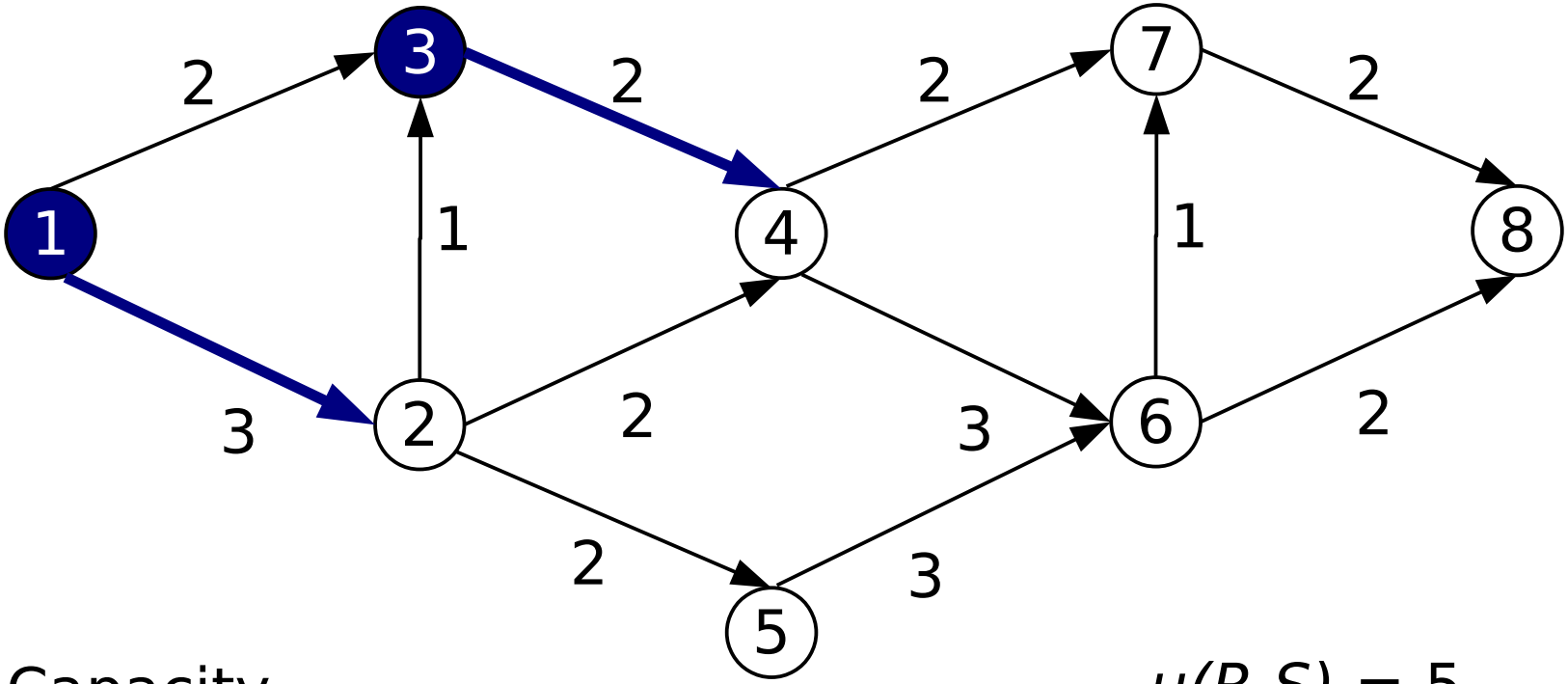
To prove correctness of the augmenting path algorithm, we'll take a short detour.

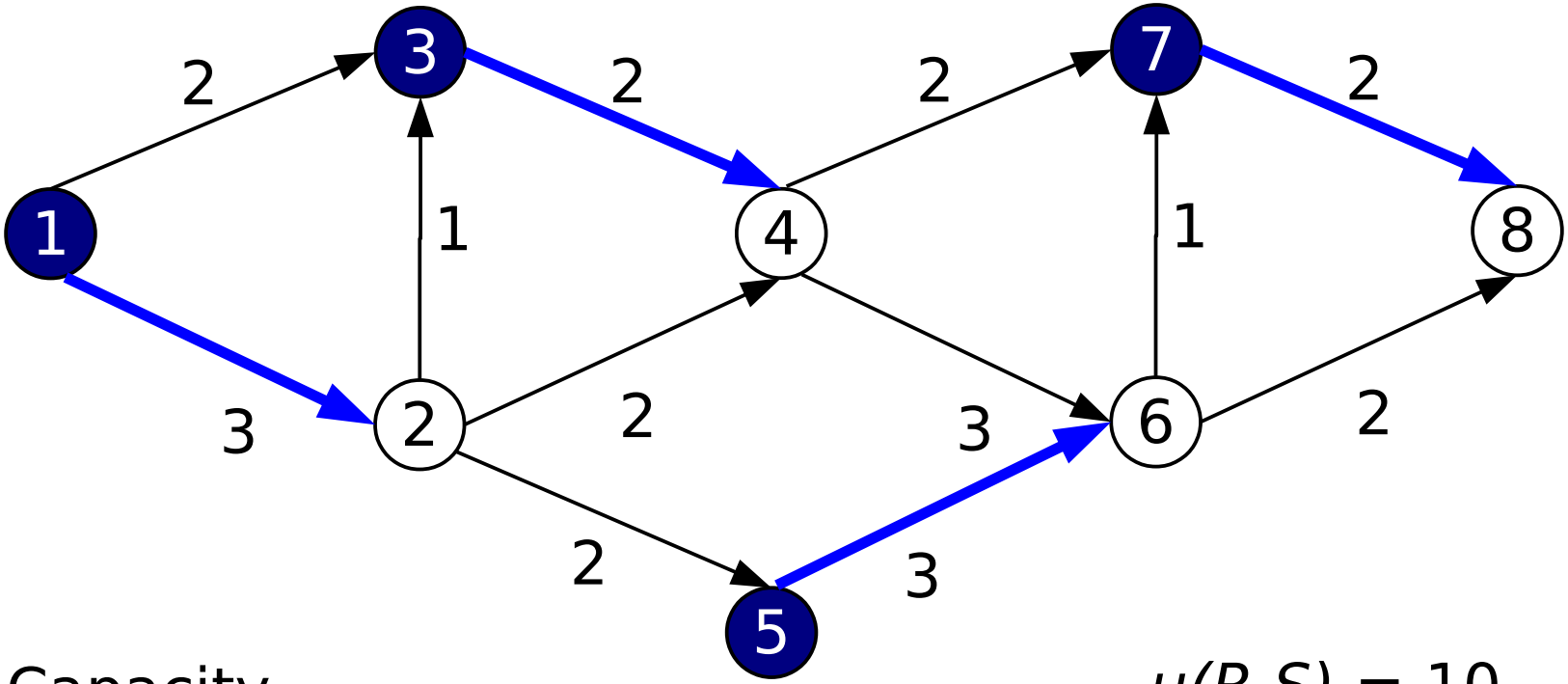


In a network, a *cut* is a partitioning of the nodes into two sets R and S, where the source is in R and the sink is in S.
A *cut link* is a link whose tail node is in R and whose head node is in S.

The capacity of a cut $u(R, S)$ is the sum of the capacities in the cut links.




1
2
3
4
5
6
7
8
2
3
2
1
2
2
3
2
3
2
1
2
Capacity
u(R,S) = 5



Capacity

$u(R,S) = 10$

Weak max-flow/min-cut theorem

.

If $(b, \mathbf{x})$ is a feasible solution to the max flow problem, and if (R, S) is a cut, then $b \leq u(R, S)$

Proof sketch: Any unit of flow shipped from r to s must cross one of the cut links. So, the total amount of flow moving from r to s cannot exceed the sum of the capacities in the cut links.

Strong max-flow/min-cut theorem

.

If $(b^*, \mathbf{x}^*)$ is an optimal solution to the max flow problem, and if $u^* = \min\{u(R,S)\}$ among all possible cuts, then $b^* = u^*$

This is a "duality" type of result: finding the maximum flow in a network is equivalent to finding the cut with minimum capacity.

We will prove this theorem in conjunction with the correctness of the augmenting path algorithm.

Assume for now that the augmenting path algorithm terminates with some flow $\mathbf{x}$. When it does so, there are no paths in $\mathcal{R}(\mathbf{x})$ from r to s with positive capacity.

Let R be the set of nodes for which paths with positive capacity *do* exist in the residual network (including r itself), and let S be all other nodes.

Since $r \in R$ and $s \in S$, (R, S) is a valid cut. Furthermore, by definition every cut link has zero capacity.

Therefore, the capacity of the cut is exactly equal to the sum of the flow on the cut arcs, so $b = u(R, S)$.

By the weak max-flow/min-cut theorem, this means b must be maximal and $u(R, S)$ must be minimal. (Strong theorem has been proved.)

Therefore, the augmenting path algorithm terminates with the correct answer.

COMPLEXITY

The other issue is proving that the augmenting path algorithm must terminate.

This is easy to do if we assume that the capacities are integers.

(If they are fractional, multiply by a sufficiently large number.)

Each iteration adds at least 1 to b, so b will reach its maximum value b^* in a finite number of iterations.

If U is the largest capacity on any link, then the capacity of the min cut is bounded by nU. (Look at a cut where R is just the source.)

So, the augmenting path algorithm requires $O(nU)$ iterations.

Each iteration must construct a residual network ($O(m)$ steps), and identify a path with positive capacity.

The basic search algorithm for finding a path also has $O(m)$ steps.

So, the augmenting path algorithm in all requires $O(nmU)$ steps.

INFORMAL EVALUATIONS

- Pace of class? Fast/slow/OK
- Most unclear concept at this point
- What is working well?
- What can I improve on?
- Any other comments or suggestions