CS 580
ALGORITHM DESIGN AND ANALYSIS

Max-Flow Min-Cut:
Part 2

Vassilis Zikas

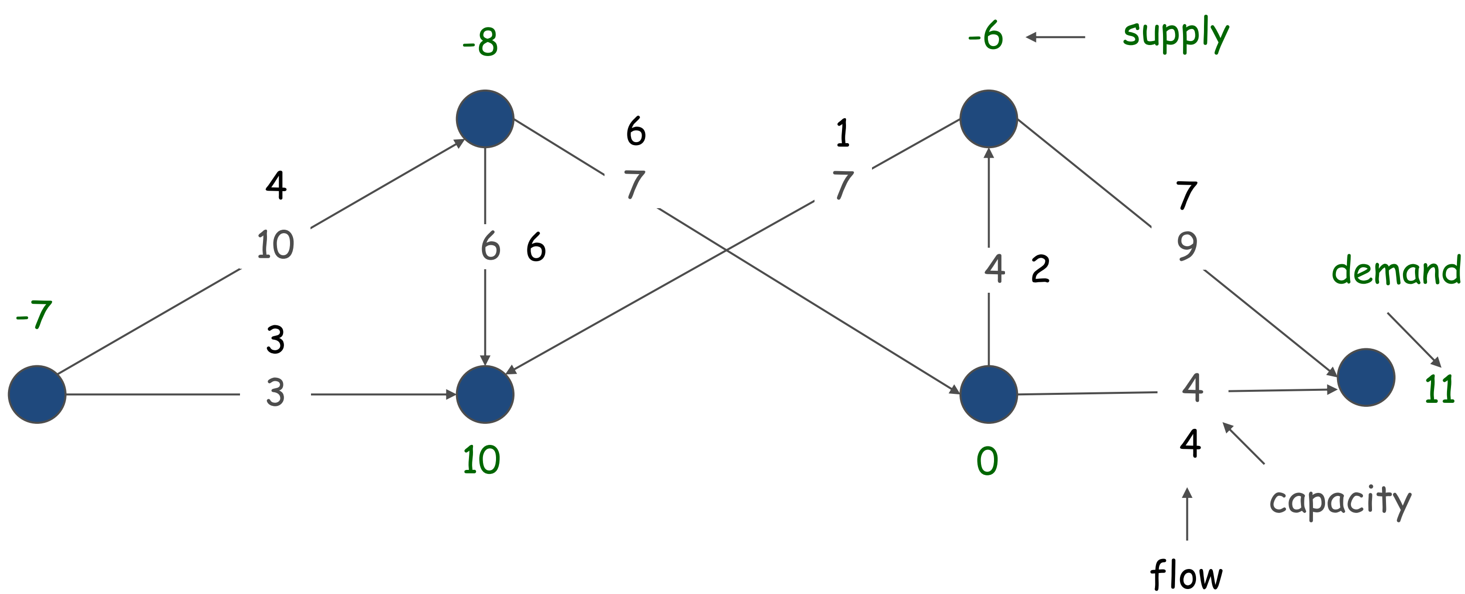
SO FAR

- Previously:
  - Max-Flow Min-cut
- This time:
  - Applications

CIRCULATION WITH DEMANDS

- Input:
  - Directed graph G
  - Edge capacities $c(e)$, for each $e \in$ E
  - Node supply/demand $d(v)$, for each $v \in V$
    - $d(v) > 0$: demand
    - $d(v) < 0$: supply
- Dfn: A circulation satisfies:
  - For each $e \in E$: $0 \leq f(e) \leq c(e)$
  - For each $v \in V$: $\sum_{e\ in\ to\ v} f(e) - \sum_{e\ out\ of\ v} f(e) = d(v)$
- Question: Does there exist a circulation?

CIRCULATION WITH DEMANDS

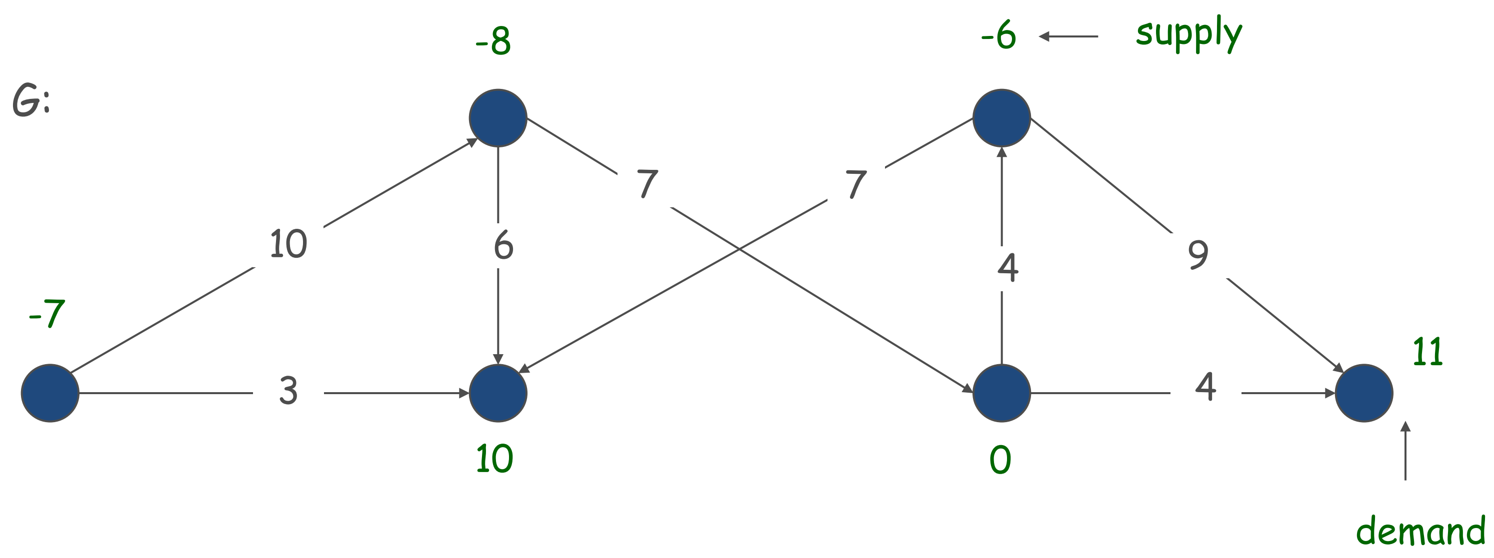


CIRCULATION WITH DEMANDS

- Observation: A necessary condition for a circulation to exist is that sum of supplies equals sum of demands
  - $\sum_{v:d(v)>0} d(v) = -\sum_{v:d(v)<0} d(v) = D$
- Proof: Sum up the circulation constraints over all nodes

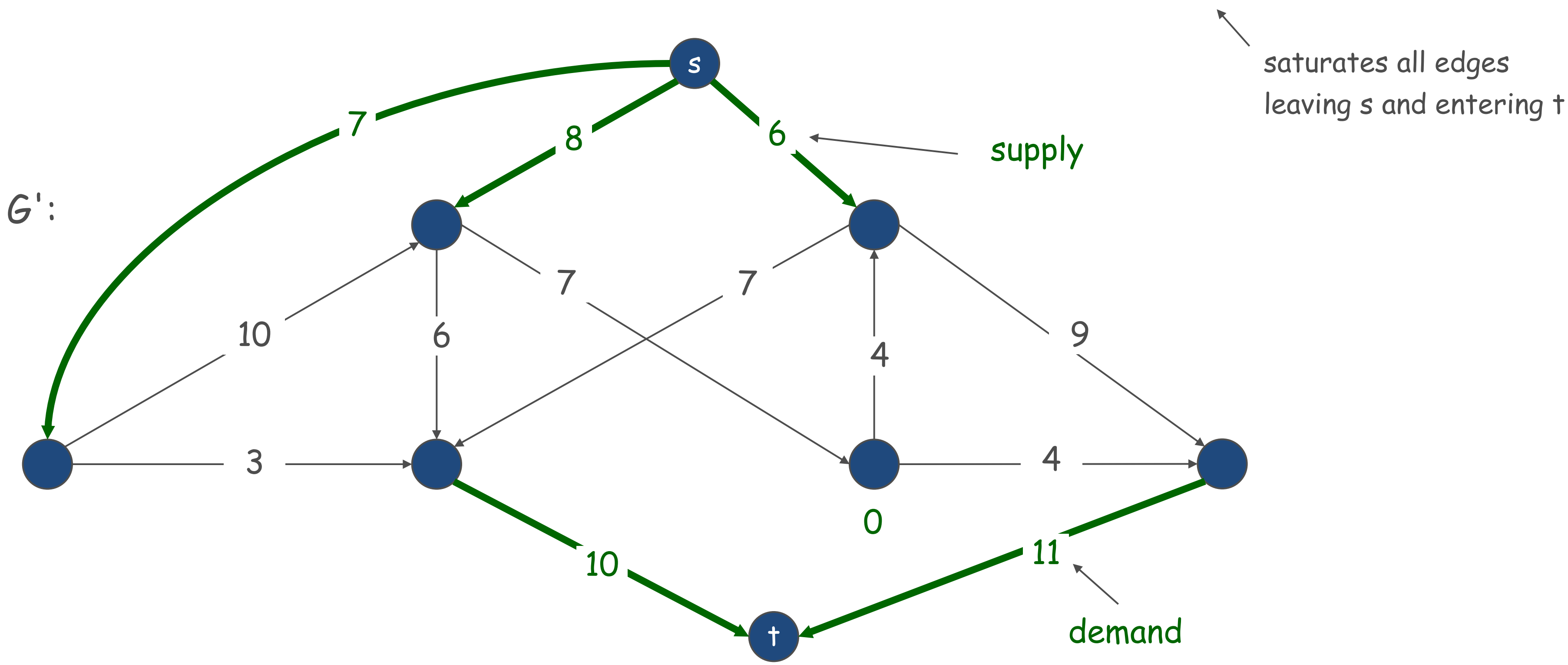
CIRCULATION WITH DEMANDS

- Max flow formulation.



CIRCULATION WITH DEMANDS

- Max flow formulation.
  - Add new source s and sink t.
  - For each v with $d(v) < 0$, add edge (s, v) with capacity $-d(v)$.
  - For each v with $d(v) > 0$, add edge (v, t) with capacity $d(v)$.
  - Claim: G has circulation iff G' has max flow of value D.



PROOF (IF)

- Say that G' has a flow f of value D
- Then, it must be that all edges adjacent to s satisfy $f(e) = c(e)$.
  - Furthermore, for all e $= (s, v)$ edges we have that $c(e) = -d(v)$
- This flow satisfies $\sum_{e\ in\ to\ v} f(e) - \sum_{e\ out\ of\ v} f(e) = 0$, for all v
- All supply nodes have an extra $-d(v)$ as incoming flow
- Therefore, for all supply nodes we have $\sum_{e\ in\ to\ v \backslash (s,v)} f(e) - \sum_{e\ out\ of\ v} f(e) = d(v)$
- Similarly for demand nodes.
- Therefore, f is a valid circulation for G

PROOF (ONLY IF)

- Say G has a valid circulation f
- Send flow $-d(v)$ from s to every supply node v
- Send flow $d(v)$ from every demand node v to t
- This gives a valid $s-t$ flow with value D
- There is no flow with higher value, since all the cut $\{ s \}$ has value D

CIRCULATION WITH DEMANDS AND LOWER BOUNDS

- Input:
  - Directed graph G
  - Edge capacities $c(e)$ and lower bounds $\ell(e)$, for each $e \in E$
  - Node supply/demand $d(v)$, for each $v \in V$
- Dfn: A circulation satisfies:
  - For each $e \in E$: $\ell(e) \leq f(e) \leq c(e)$
  - For each $v \in V$: $\sum_{e \ in \ to \ v} f(e) - \sum_{e \ out \ of \ v} f(e) = d(v)$
- Question: Does there exist a circulation?

CIRCULATION WITH DEMANDS AND LOWER BOUNDS

- Idea: Model lower bounds with demands.
  - Send ℓ(e) units of flow along edge e.
  - Update demands of both endpoints.

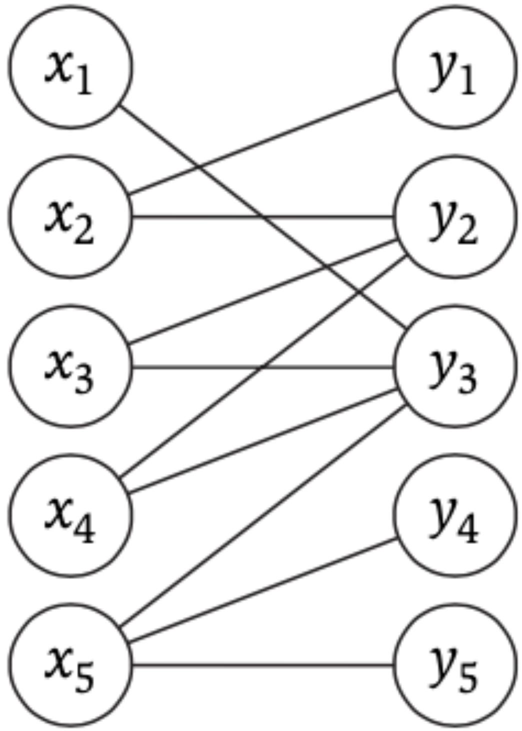
- Theorem. There exists a circulation in G iff there exists a circulation in G'. If all demands, capacities, and lower bounds in G are integers, then there is a circulation in G that is integer-valued.

MATCHINGS

BIPARTITE MATCHING

- Input:
  - Bipartite graph $G = (V, E)$
    - $V = X \cup Y$
- Output:
  - Maximum cardinality matching
- Dfn: A matching M in G is a subset of the edges $M \subseteq E$, such that each vertex appears in at most one edge

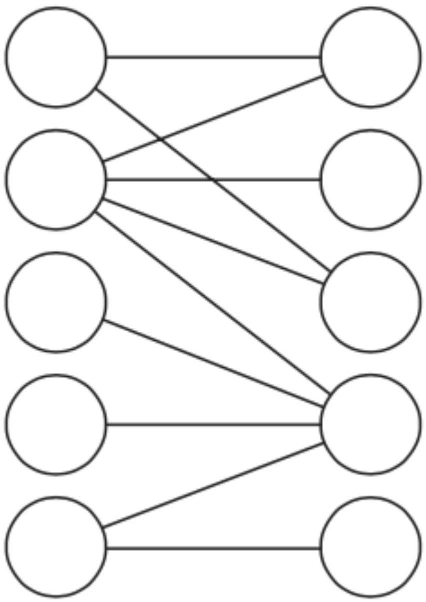
BIPARTITE MATCHING



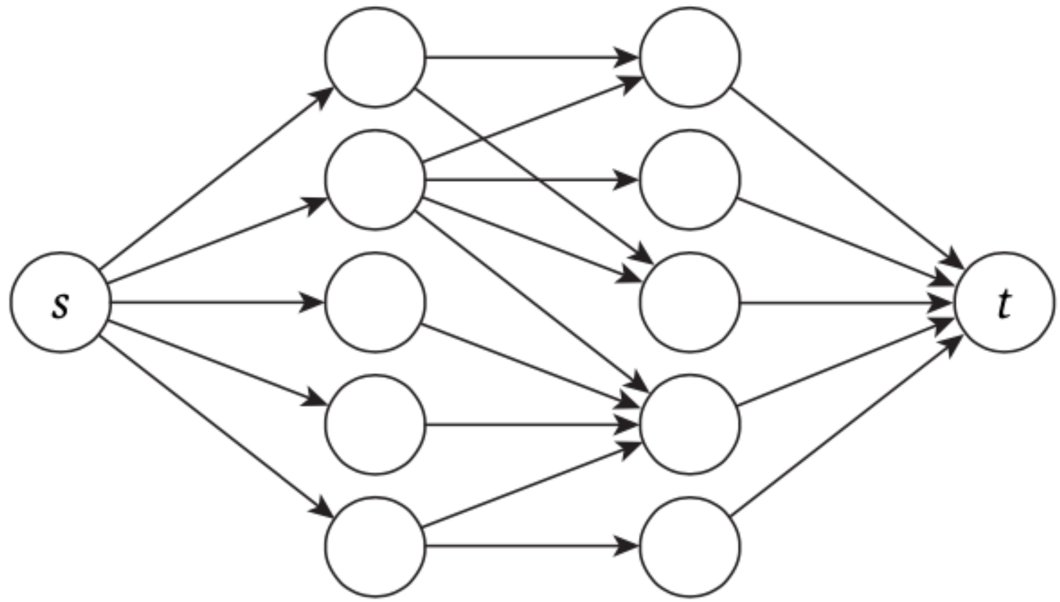
BIPARTITE MATCHING

- Same trick!
- Add a source node s and sink node t
- Add edges (s, v) for all $v \in X$, with capacity 1
- Add edges (v, t) for all $v \in Y$, with capacity 1
- Direct all edges $e = (u, v)$, for $u \in X$ and $v \in Y$, from u to v, and set $c(e) = 1$

BIPARTITE MATCHING

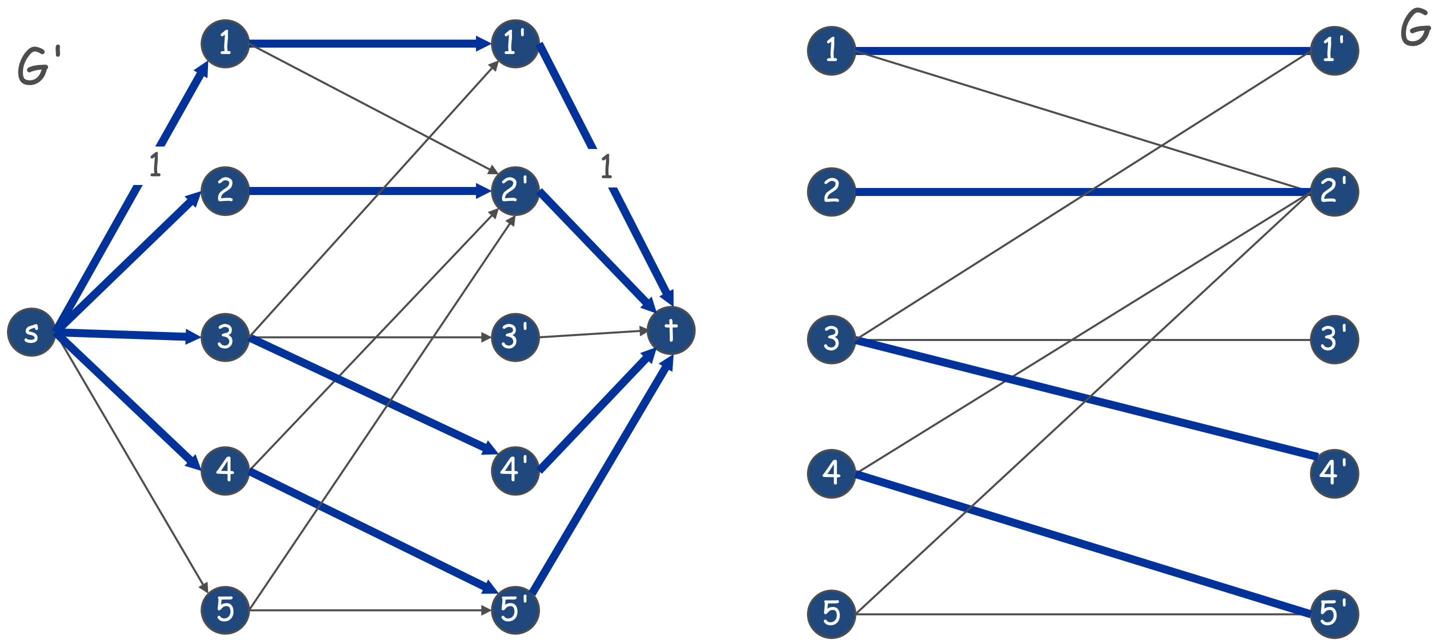


(a)



(b)

BIPARTITE MATCHING



BIPARTITE MATCHING

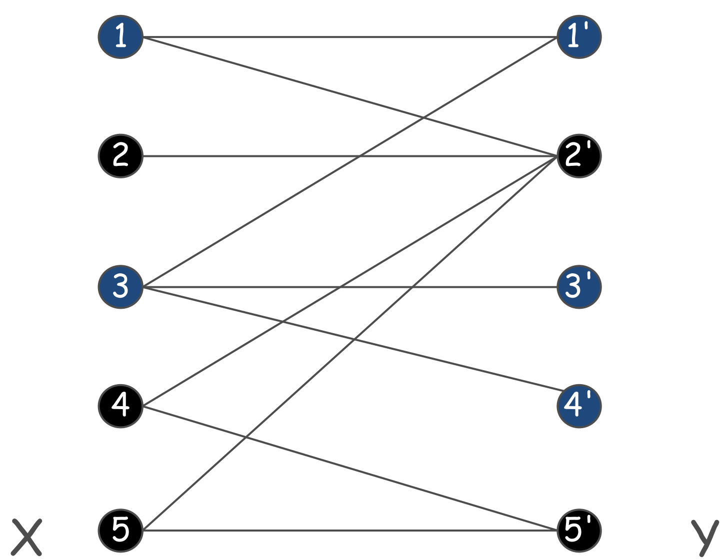
- Claim: There exists a matching of size D in G if and only if there an $s - t$ flow with value D in G'
  - Going from the matching to the flow is easy
  - Given a flow f' with value D we have:
  - There exists an integer flow of value D
    - Therefore, all edges are either full capacity or untouched
  - For each full edge (u, v) where $u \in X$ and $v \in Y$, there are no other full edges out of u or into v
    - The maximum possible incoming/outgoing flow is 1 for u and v, respectively.
  - Exactly D full edges out of s, and therefore exactly D full edges between X and Y

PERFECT MATCHING

- Dfn: A matching M is called perfect if each node appears in exactly one edge in M
- Question: When does a bipartite graph have a perfect matching?
- Obviously, we need $|X| = |Y|$
- What other conditions are necessary?
- What conditions are sufficient?

PERFECT MATCHING

- Notation: Given a subset of nodes S, let $N(S)$ be the set of nodes adjacent to nodes in S
- Observation: If a bipartite graph has a perfect matching, then $|N(S)| \geq |S|$ for all $S \subseteq X$.
  - Proof: Each node in S has to be matched to a different node in $N(S)$



<u>No perfect matching:</u>

S = { 2, 4, 5 }

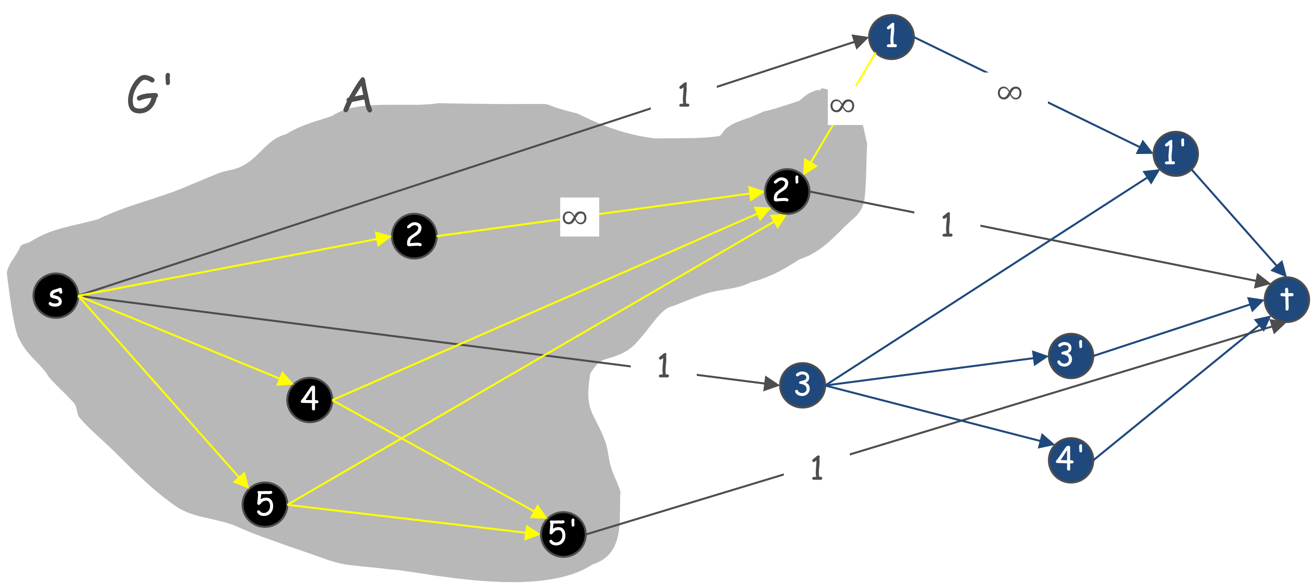
N(S) = { 2', 5' }.

MARRIAGE THEOREM

- Theorem(Frobenius 1917, Hall 1935): Let $G = (X \cup Y, E)$ be a bipartite graph. Then G has a perfect matching iff $|N(S)| \geq |S|$ for all $S \subseteq X$.
- Proof
  - "only if" we've already shown

PROOF OF MARRIAGE THEOREM

- Suppose that G does not have a perfect matching
- We need to show that if the value of maximum flow is less than $|X|$, there exists a subset S, with $|N(S)| < |S|$
- Formulate as a max-flow problem (same as the bipartite graph reduction, but with ∞ edges) and let (A, B) be a min-cut of G'
- $cap(A, B) \leq v(f) < |X|$
- Define $L_A = X \cap A, L_B = X \cap B, R_A = Y \cap A$



$L_A = \{2, 4, 5\}$

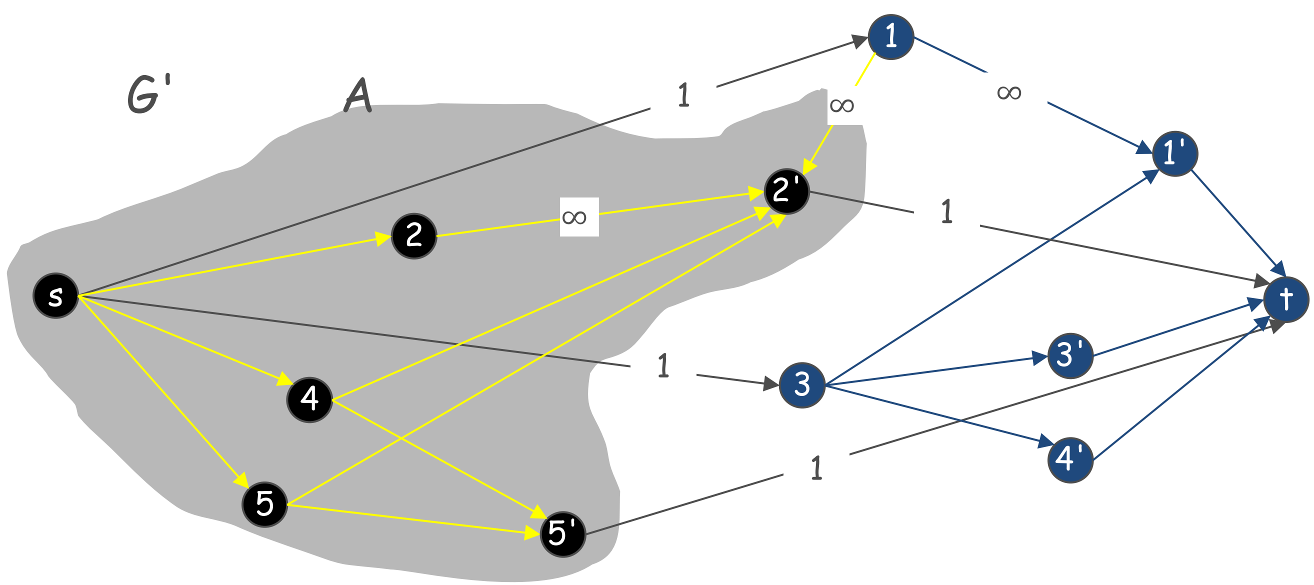
$L_B = \{1, 3\}$

$R_A = \{2', 5'\}$

$N(L_A) = \{2', 5'\}$

PROOF OF MARRIAGE THEOREM

- The cut shouldn't use any ∞ edges, i.e. $N(L_A) \subseteq R_A$
- $cap(A, B) = |L_B| + |R_A|$
- $|N(L_A)| \leq |R_A| = cap(A, B) - |L_B| < |L| - |L_B|$
- $|L| - |L_B| = |L_A|$
- $S = L_A$ satisfies $|N(S)| < |S|$



$L_A = \{2, 4, 5\}$

$L_B = \{1, 3\}$

$R_A = \{2', 5'\}$

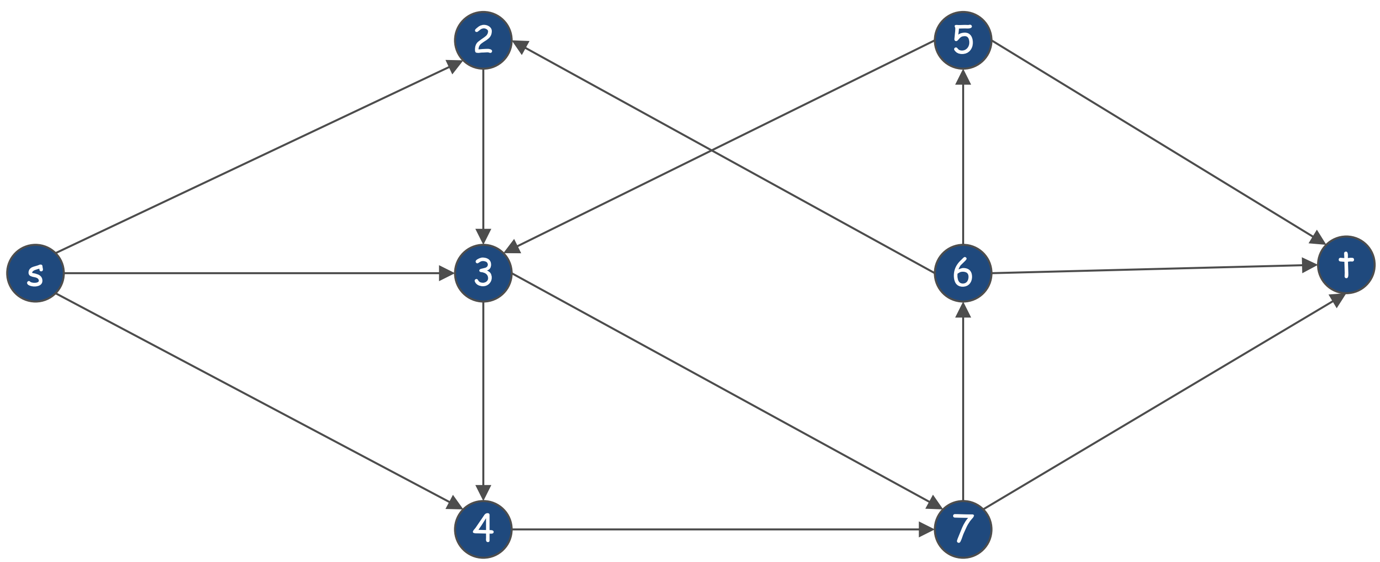
$N(L_A) = \{2', 5'\}$

MARRIAGE THEOREM

- We can use max-flow to find a perfect matching in a bipartite graph
- But, we also used max-flow in the proof of a structural characterization of perfect matchings!
- What about non-bipartite matchings?
  - More complicated, but well-understood
  - Blossom algorithm $O(n^4)$ [Edmonds 1965]
  - Current champion O($mn^{\frac{1}{2}}$) [Micali-Vazirani 1980]

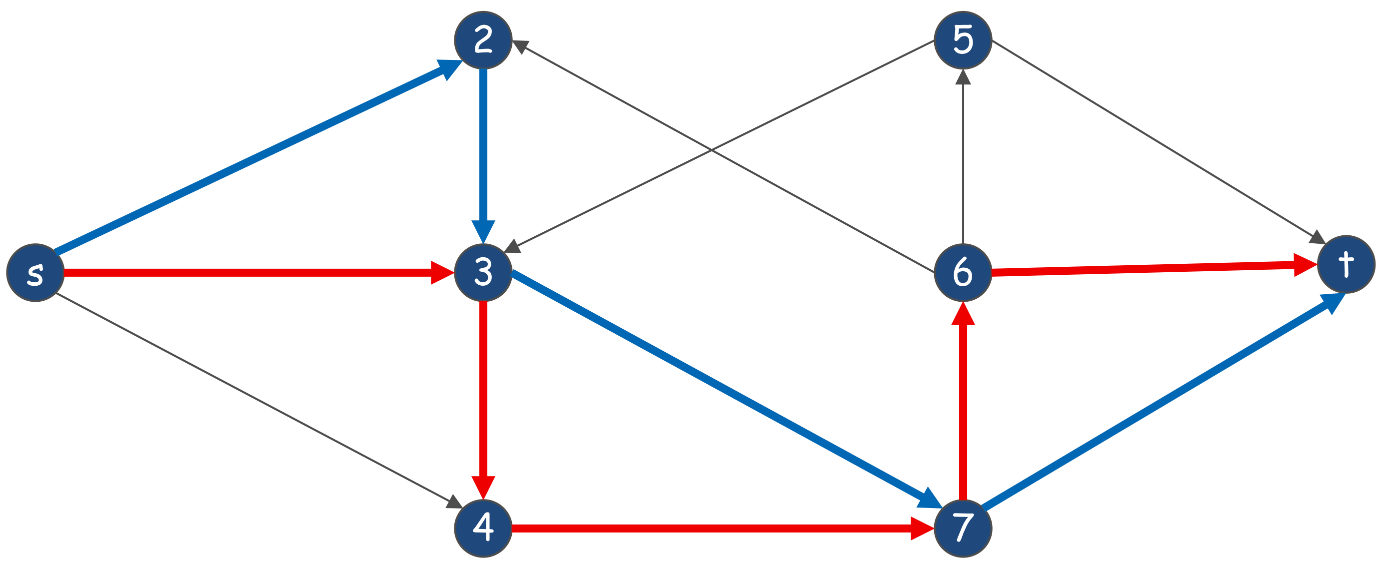
EDGE DISJOINT PATHS

- Input: directed graph G and two nodes s, t
- Output: the maximum number of edge disjoint paths from s to t
- Dfn: Two paths are edge-disjoint if they have no edge in common



EDGE DISJOINT PATHS

- Input: directed graph G and two nodes s, t
- Output: the maximum number of edge disjoint paths from s to t
- Dfn: Two paths are edge-disjoint if they have no edge in common



EDGE DISJOINT PATHS

- Max flow formulation: assign unit capacity to each edge
- **Theorem**: Max-flow value equals to the number of edge-disjoint paths
- Proof (flow $\geq$ number of paths)
  - Assume that there are k edge-disjoint paths $P_1, \dots, P_k$
  - Set $f(e) = 1$ if e is in some path P_i, else $f(e) = 0$
  - Since paths are edge-disjoint, this flow is valid, and its value is exactly k

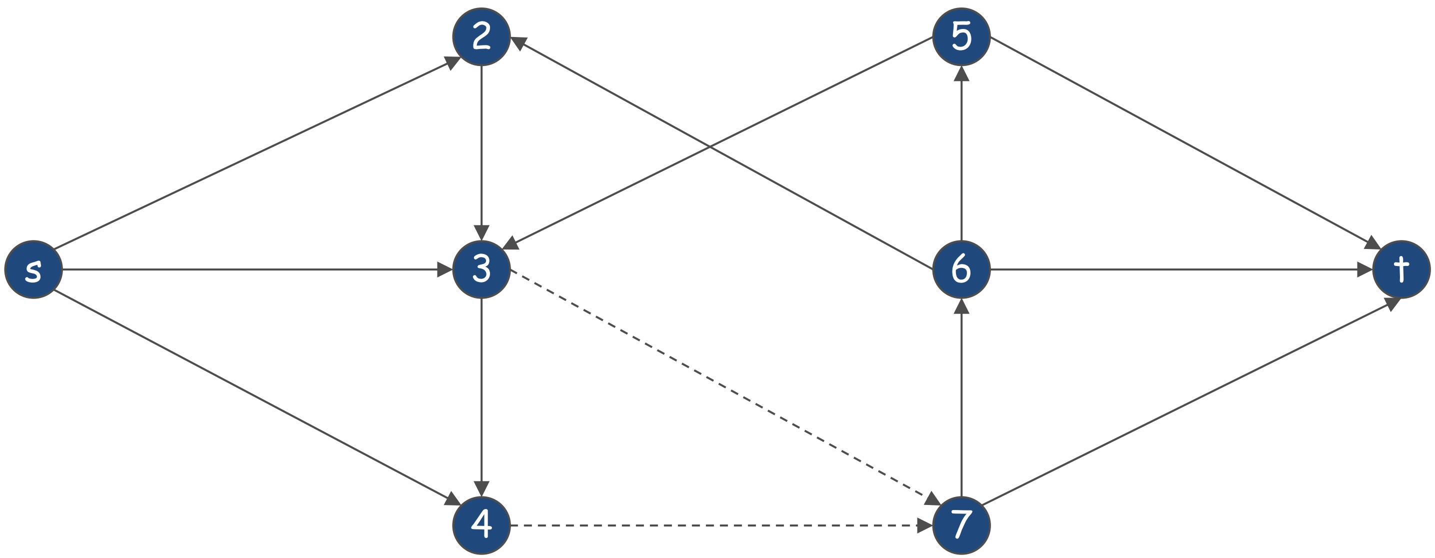
EDGE DISJOINT PATHS

- Max flow formulation: assign unit capacity to each edge
- Theorem: Max-flow value equals to the number of edge-disjoint paths
- Proof (flow $\leq$ number of paths)
  - Suppose max-flow value is k
  - Since capacities are integral, there exists a 0-1 flow with value k
  - Consider edge (s, u) with $f(s, u) = 1$
    - By conservation of flow, there exists an edge $f(u, v) = 1$
    - Keep going until you reach t, picking a new edge every time
  - Pick a different (s, x) edge, and repeat
  - We will get k (not necessarily simple) edge-disjoint paths
    - Eliminate any cycles to get simple paths

NETWORK CONNECTIVITY

- Input: directed graph G and two nodes s, t
- Output: the minimum number of edges whose removal disconnects t from s

NETWORK CONNECTIVITY



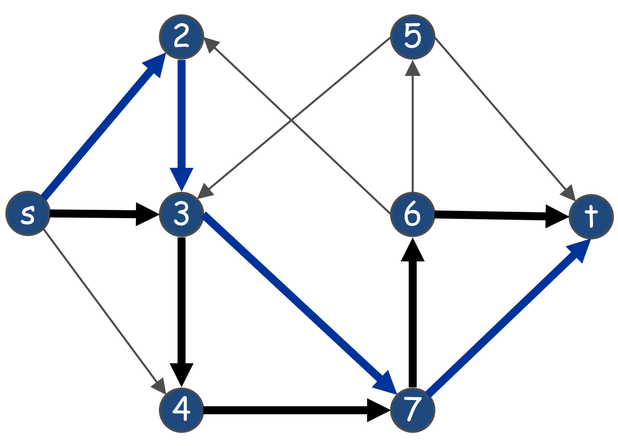
NETWORK CONNECTIVITY

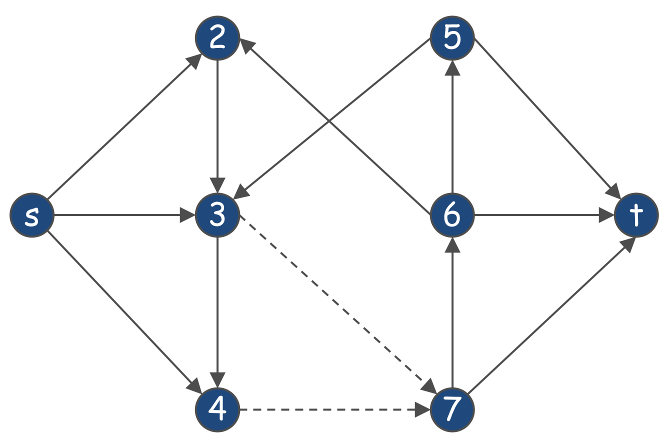
- Input: directed graph G and two nodes s, t
- Output: the minimum number of edges whose removal disconnects t from s

- Observation: A set of edges $F \subseteq \mathrm{E}$ disconnects t from s if every $s - t$ path uses at least one edge in F

NETWORK CONNECTIVITY

- Theorem [Merger 1927]: The max number of edge-disjoint $s - t$ paths is equal to the min number of edges whose remove disconnects t from s



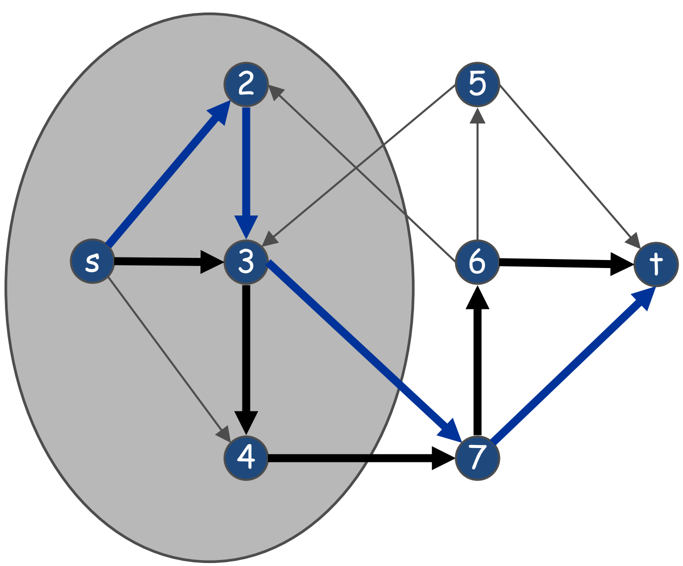


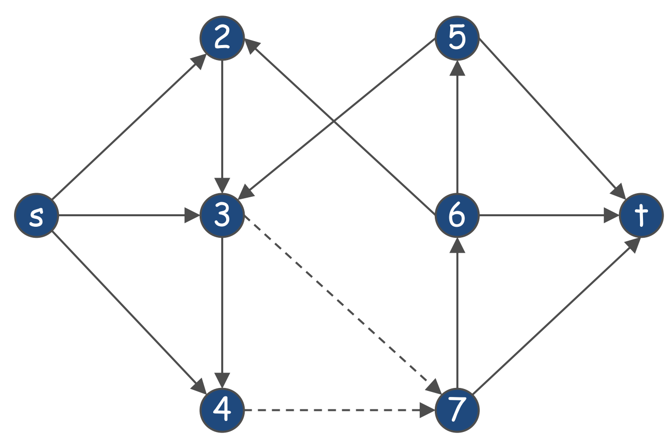
NETWORK CONNECTIVITY

- Theorem [Merger 1927]: The max number of edge-disjoint $s-t$ paths is equal to the min number of edges whose remove disconnects t from s
- Proof ($\leq$)
  - Suppose that the removal of $F \subseteq E$ disconnects t from s, and $|F| = k$
  - Every $s-t$ path uses at least one edge in F
  - Then there are at most k edge disjoint paths

NETWORK CONNECTIVITY

- Proof ($\geq$)
  - Suppose the max number of edge-disjoint paths is k
  - Then max-flow value is equal to k
  - Then, there exists an (A, B) cut with capacity k
  - Let F be the edges going from A to B
  - $|F| = cap(A, B) = k$, and F disconnects t from s





SUMMARY OF MAX-FLOW

- Max-flow Min-cut
    - Ford-Fulkerson algorithm
- A few applications:
    - Bipartite matching
    - Perfect matching
    - Edge disjoint paths
    - Network connectivity
    - Circulations with demands
    - Circulations with demands and lower bounds