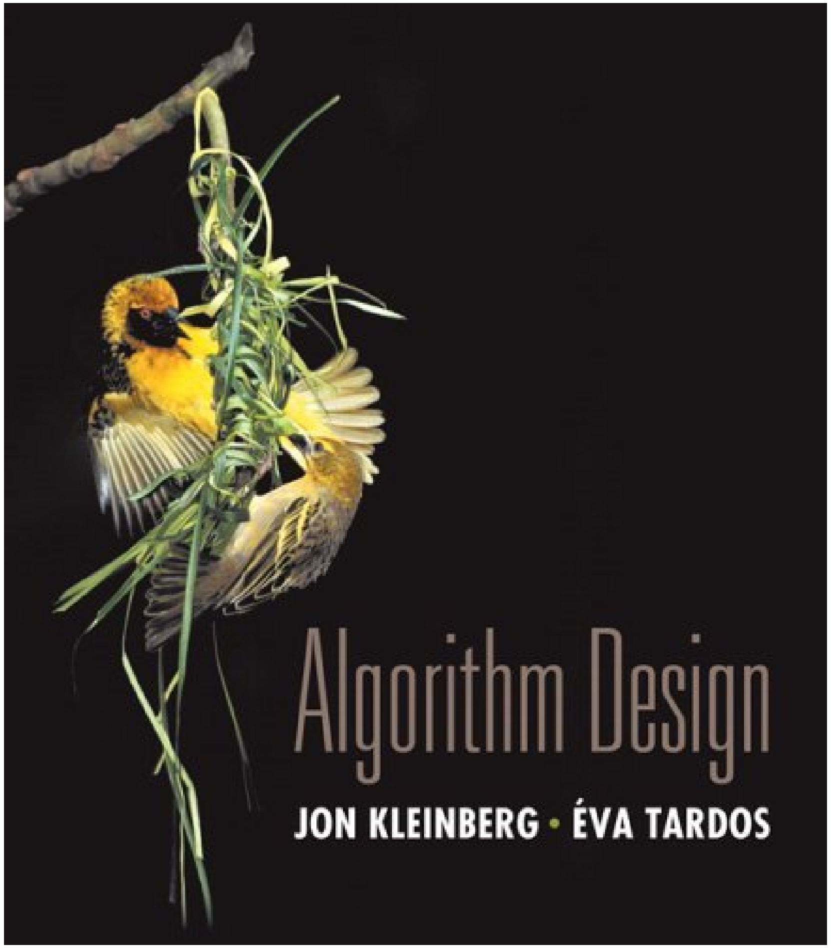
CS 381 – FALL 2019

Week 12.1, Monday, Nov 4

Homework 6: Planning to Release Soon



Network Flow

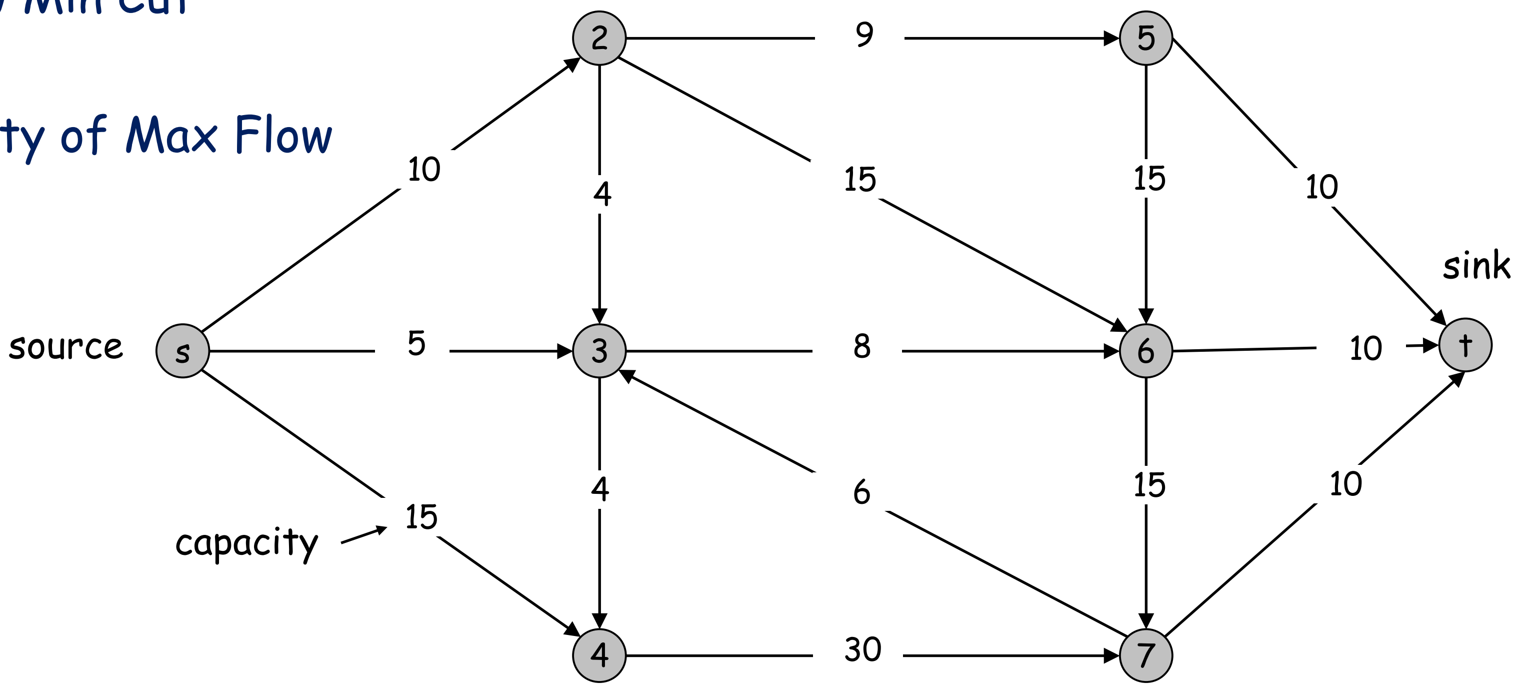
Recap

Residual Graph G_f

- Augmenting Path
- Ford-Fulkerson Algorithm
  - While the residual graph contains an augmenting path
    - Increase Flow (Augment)
    - Update Residual Graph

Max Flow Min Cut

Integrality of Max Flow



Running time

Assumption. Capacities are integers between 1 and C.

Integrality invariant. Throughout the algorithm, the flow values $f(e)$ and the residual capacities $c_f(e)$ are integers.

Theorem. The algorithm terminates in at most $val(f^*) \leq nC$ iterations.
Pf. Each augmentation increases the value by at least 1. ▪

Pseudo-polynomial

Corollary. The running time of Ford-Fulkerson is $O(mnC)$.
Corollary. If $C = 1$, the running time of Ford-Fulkerson is $O(mn)$.

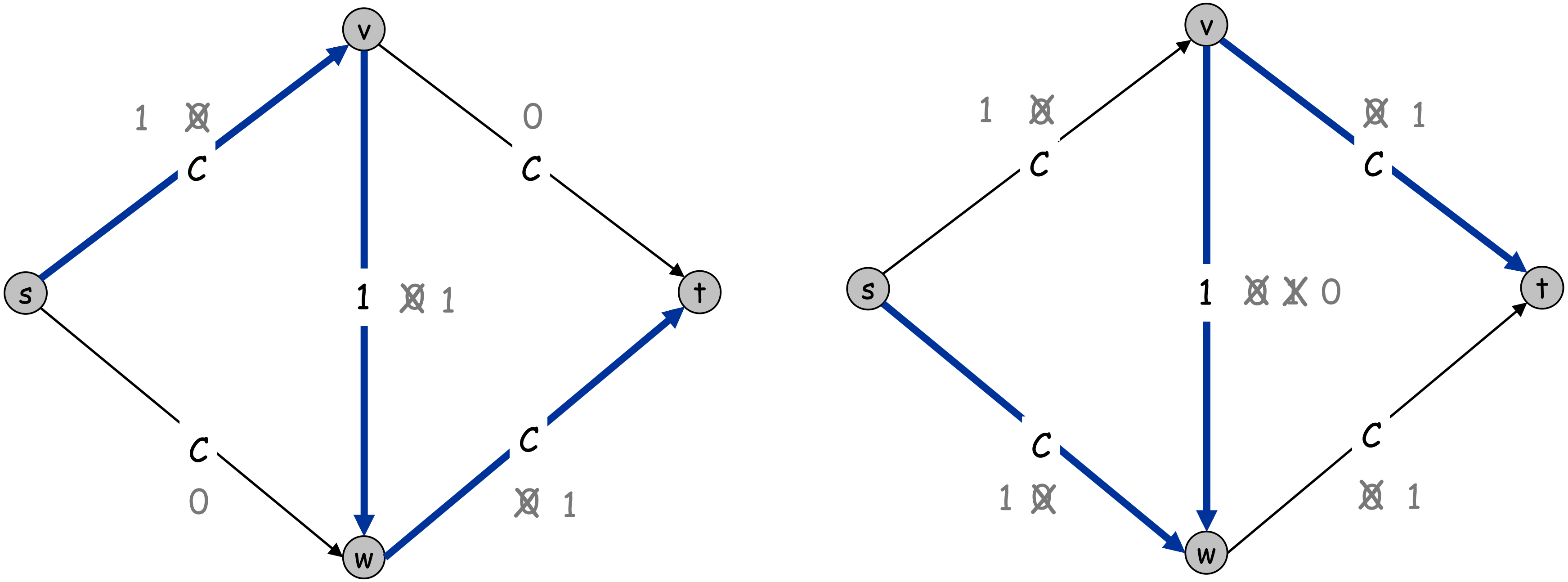
Integrality theorem. Then exists a max-flow f^* for which every flow value $f^*(e)$ is an integer.
Pf. Since algorithm terminates, theorem follows from invariant. ▪

Ford-Fulkerson: Exponential Number of Augmentations

Q. Is generic Ford-Fulkerson algorithm polynomial in input size?

m, n, and log C

A. No. If max capacity is C, then algorithm can take C iterations.



Bad case for Ford-Fulkerson

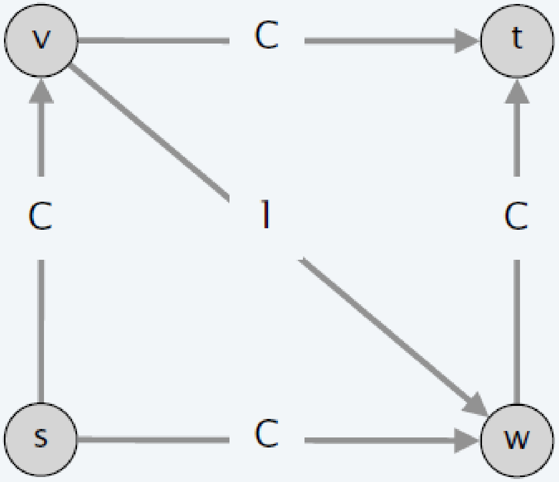
Q. Is generic Ford-Fulkerson algorithm poly-time in input size?

m, n, and log C

A. No. If max capacity is C, then algorithm can take $\geq C$ iterations.

- $s \to v \to w \to t$
- $s \to w \to v \to t$
- $s \to v \to w \to t$
- $s \to w \to v \to t$
- …
- $s \to v \to w \to t$
- $s \to w \to v \to t$

each augmenting path sends only 1 unit of flow (# augmenting paths = 2C)



7.3 Choosing Good Augmenting Paths

Choosing Good Augmenting Paths

Use care when selecting augmenting paths.

- Some choices lead to exponential algorithms.
- Clever choices lead to polynomial algorithms.
- If capacities are irrational, algorithm not guaranteed to terminate!

Goal: choose augmenting paths so that:

- Can find augmenting paths efficiently.
- Few iterations.

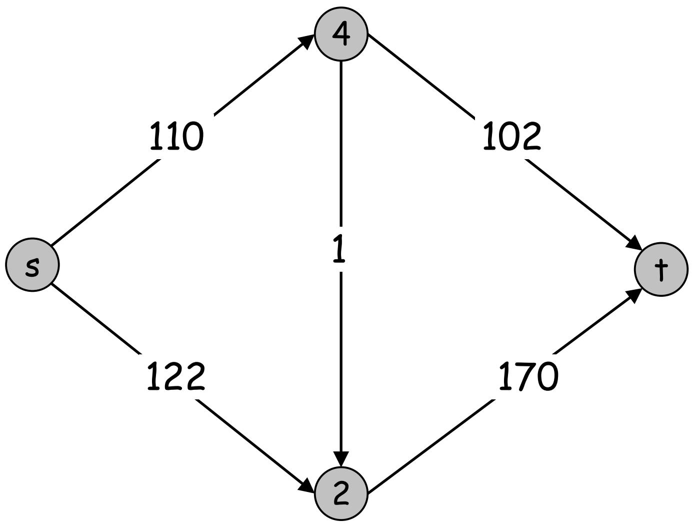
Choose augmenting paths with: [Edmonds-Karp 1972, Dinitz 1970]

- Max bottleneck capacity.
- Sufficiently large bottleneck capacity.
- Fewest number of edges.

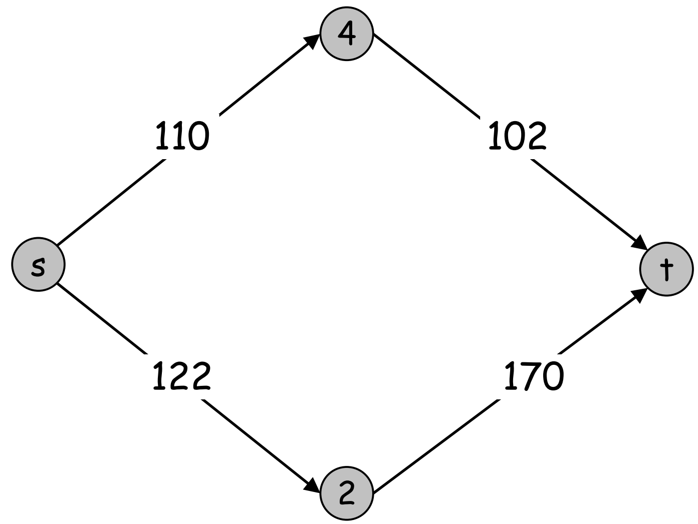
Capacity Scaling

Intuition. Choosing path with highest bottleneck capacity increases flow by max possible amount.

- Don't worry about finding exact highest bottleneck path.
- Maintain scaling parameter Δ.
- Let $G_f(\Delta)$ be the subgraph of the residual graph consisting of only arcs with capacity at least Δ.



G_f



$G_f(100)$

Capacity Scaling

```
Scaling-Max-Flow(G, s, t, c) {
   foreach e ∈ E  f(e) ← 0
   Δ ← smallest power of 2 greater than or equal to C //max capacity
   Gf ← residual graph

   while (Δ ≥ 1) {
      Gf(Δ) ← Δ-residual graph
      while (there exists augmenting path P in Gf(Δ)) {
         f ← augment(f, c, P)
         update Gf(Δ)
      }
      Δ ← Δ / 2
   }
   return f
}
```

Capacity Scaling: Correctness

Assumption. All edge capacities are integers between 1 and C.

Integrality invariant. All flow and residual capacity values are integral.

Correctness. If the algorithm terminates, then f is a max flow.
Pf.

- By integrality invariant, when $\Delta = 1 \Rightarrow G_f(\Delta) = G_f$.
- Upon termination of $\Delta = 1$ phase, there are no augmenting paths. ▪

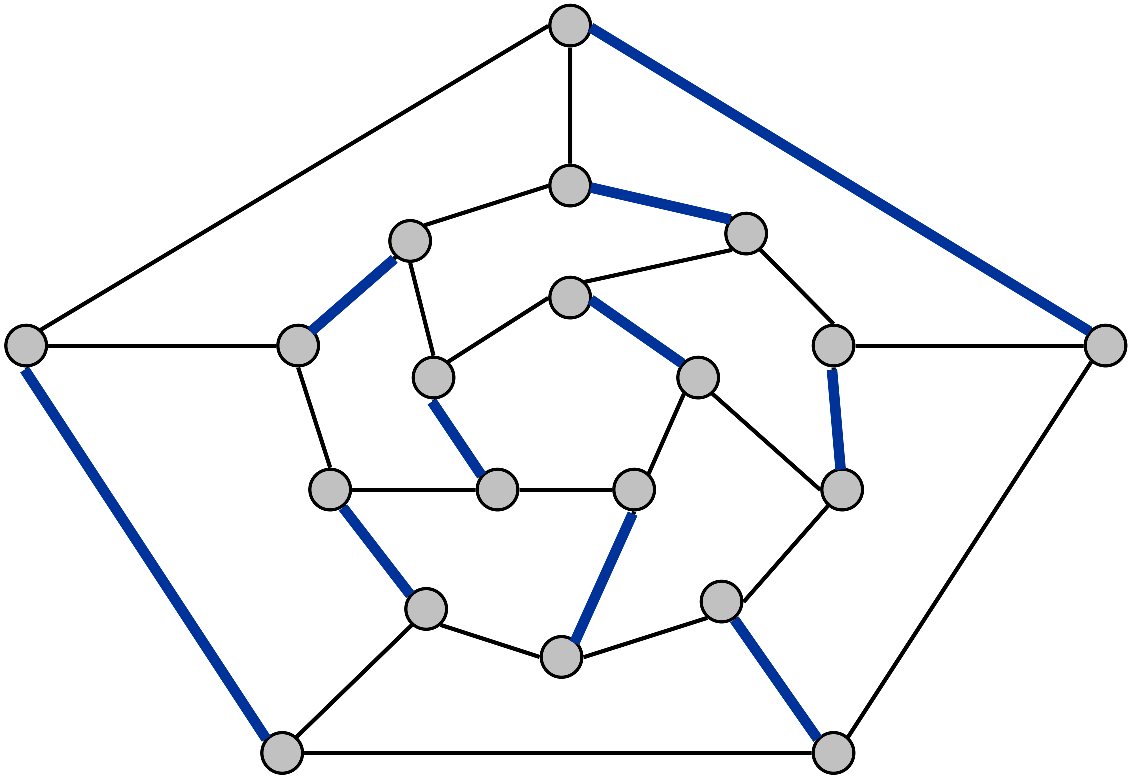
Fact: The algorithm terminates in polynomial time in n, m and log(C)

Proof: Homework 6! (We provide the hints you provide the proof)

7.5 Bipartite Matching

Matching.

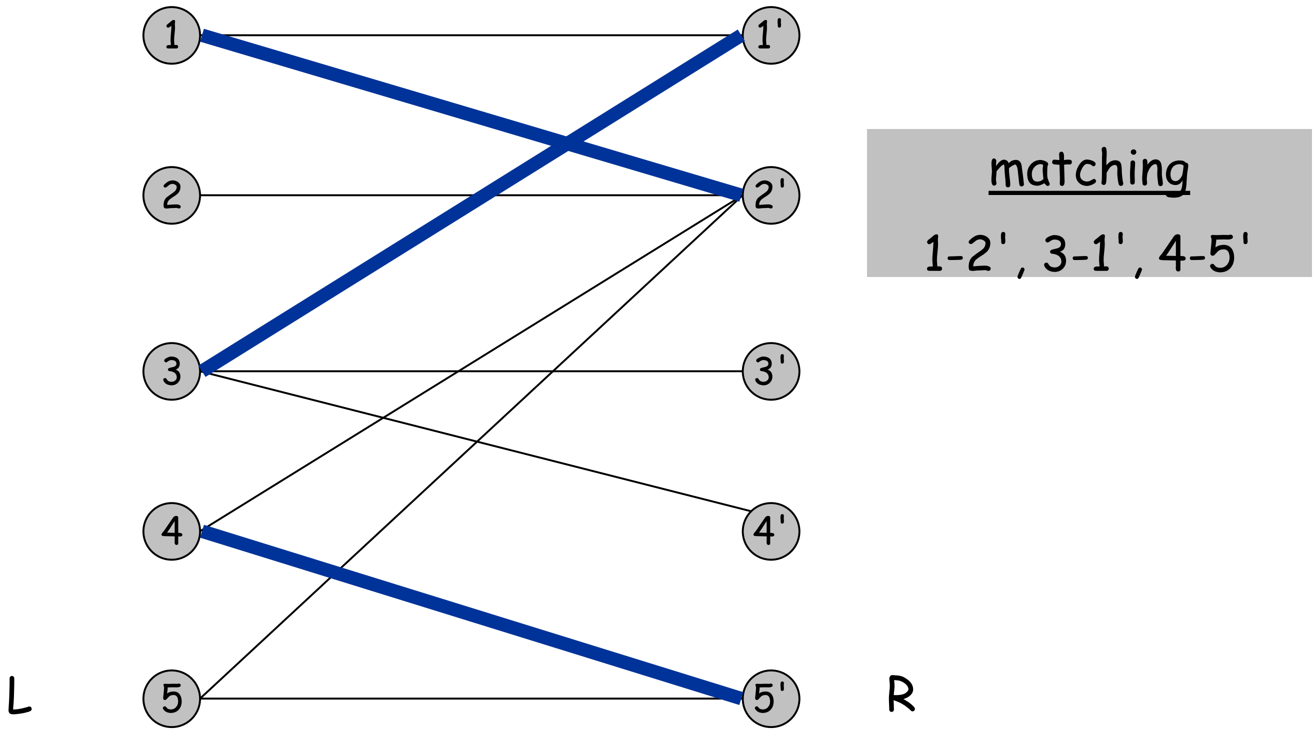
- Input: undirected graph $G = (V, E)$.
- $M \subseteq E$ is a matching if each node appears in at most edge in M.
- Max matching: find a max cardinality matching.



Bipartite Matching

Bipartite matching.

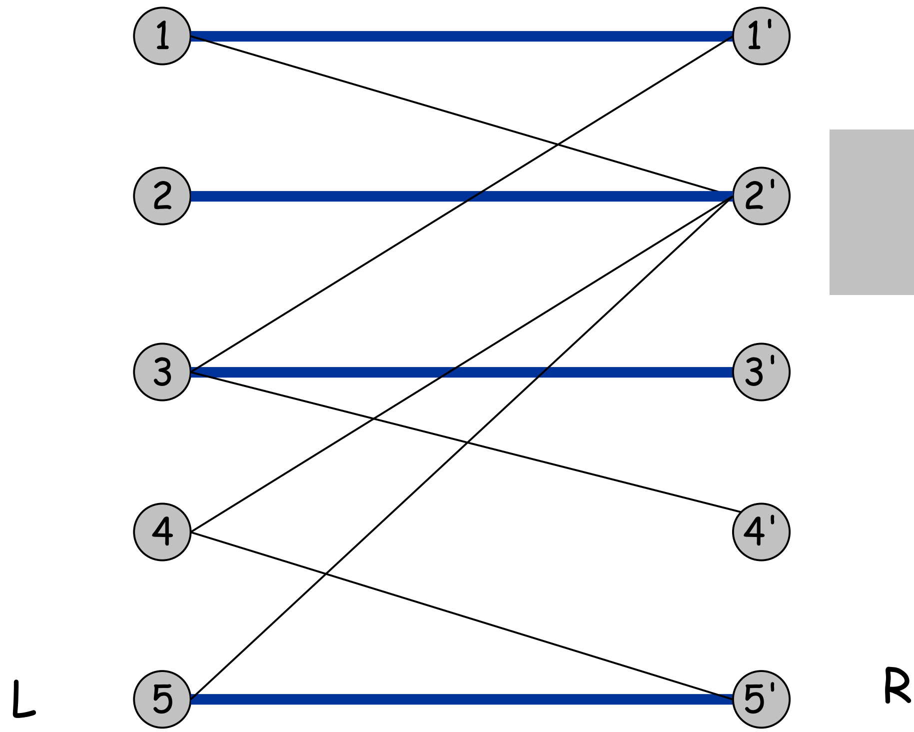
- Input: undirected, bipartite graph $G = (L \cup R, E)$.
- $M \subseteq E$ is a matching if each node appears in at most edge in M.
- Max matching: find a max cardinality matching.



Bipartite Matching

Bipartite matching.

- Input: undirected, bipartite graph $G = (L \cup R, E)$.
- $M \subseteq E$ is a matching if each node appears in at most edge in M.
- Max matching: find a max cardinality matching.

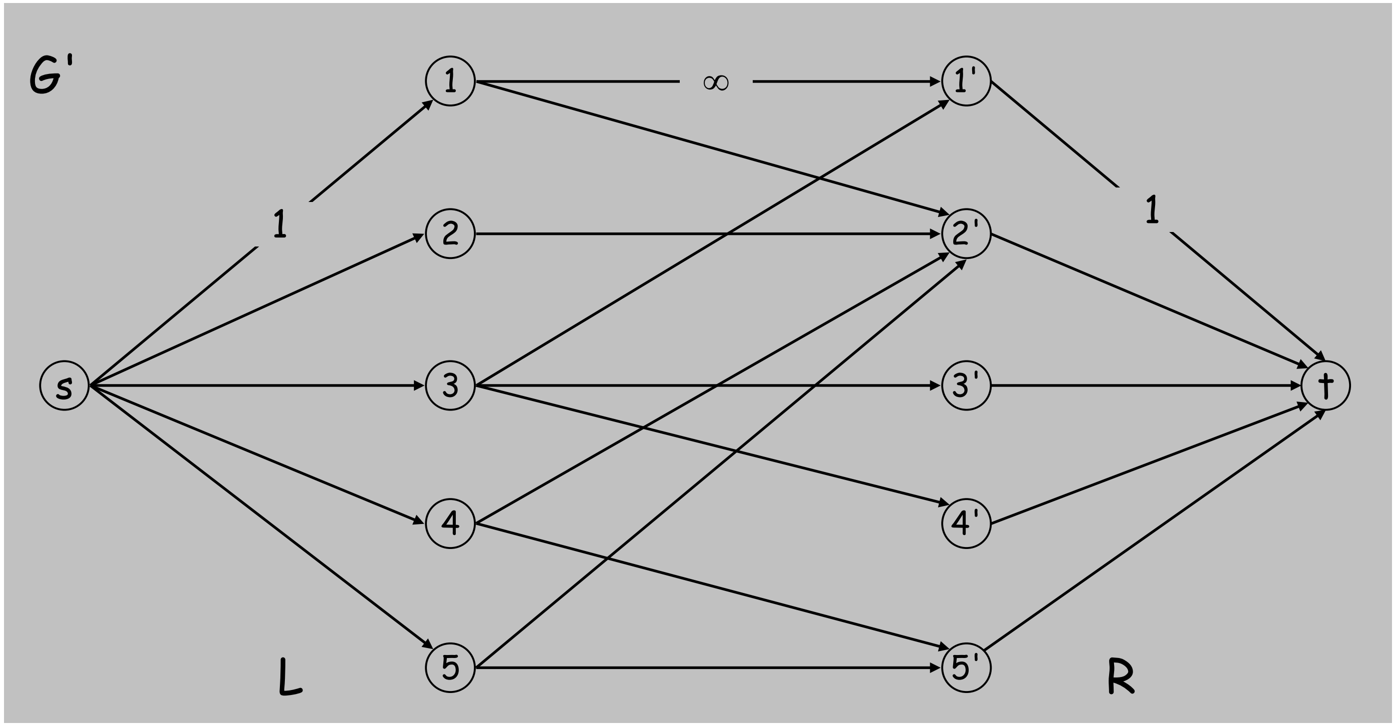


max matching

1-1', 2-2', 3-3' 4-4'

Bipartite Matching

Max flow formulation.
- Create digraph $G' = (L \cup R \cup \{s, t\},\ E')$.
- Direct all edges from L to R, and assign infinite (or unit) capacity.
- Add source s, and unit capacity edges from s to each node in L.
- Add sink t, and unit capacity edges from each node in R to t.

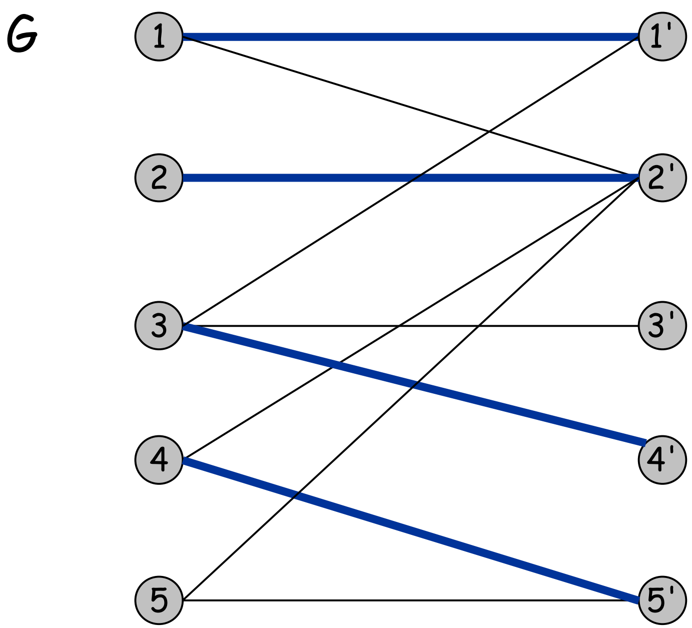


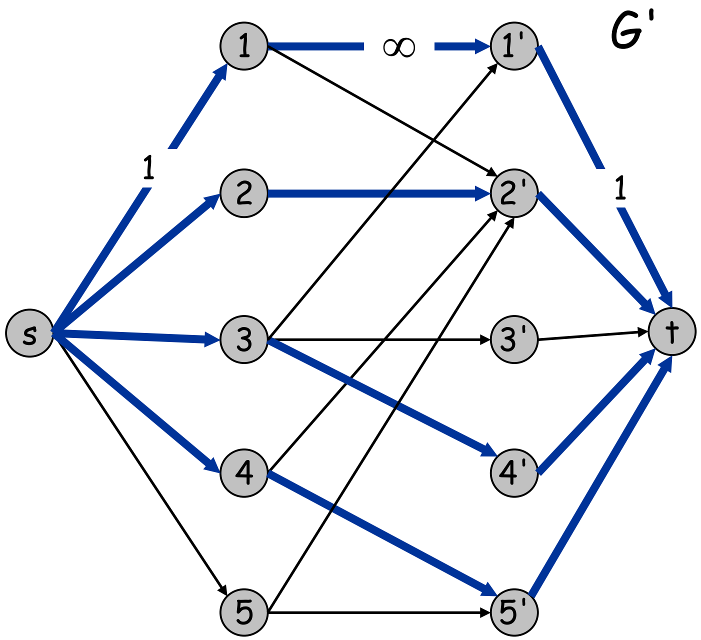
Bipartite Matching: Proof of Correctness

Theorem. Max cardinality matching in G = value of max flow in G'.

Pf. $\leq$

- Given max matching M of cardinality k.
- Consider flow f that sends 1 unit along each of k paths.
- f is a flow, and has cardinality k. ▪



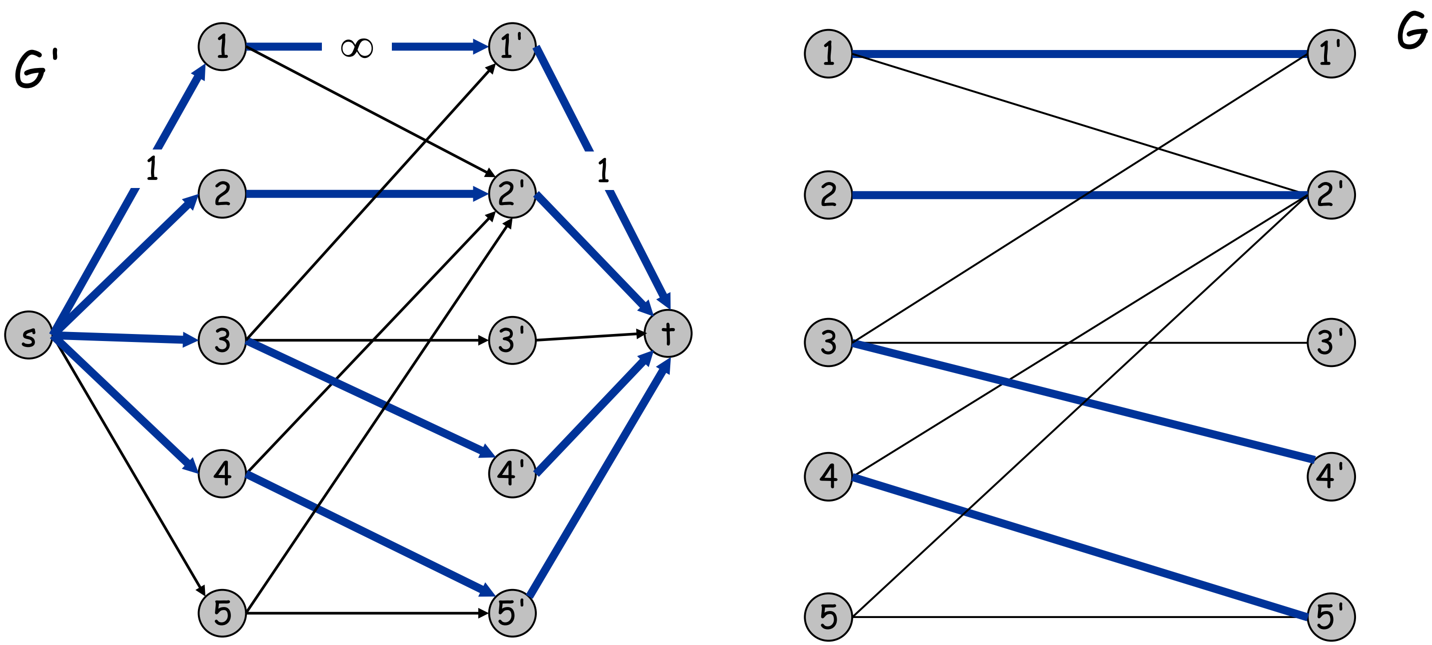


Bipartite Matching: Proof of Correctness

Theorem. Max cardinality matching in G = value of max flow in G'.

Pf. $\geq$

- Let f be a max flow in G' of value k.
- Integrality theorem $\Rightarrow$ k is integral and can assume f is 0-1.
- Consider M = set of edges from L to R with f(e) = 1.
  - each node in L and R participates in at most one edge in M
  - |M| = k: consider *flow across* the cut $(L \cup s, R \cup t)$ ▪



Perfect Matching

Def. A matching $M \subseteq E$ is perfect if each node appears in exactly one edge in M.

Q. When does a bipartite graph have a perfect matching?

Structure of bipartite graphs with perfect matchings.

- Clearly we must have $|L| = |R|$.
- What other conditions are necessary?
- What conditions are sufficient?

Bipartite Matching: Running Time

Which max flow algorithm to use for bipartite matching?

- Generic augmenting path: $O(m\ val(f^*)) = O(mn)$.
- Capacity scaling: $O(m^2 \log C) = O(m^2)$.
- Shortest augmenting path: $O(m\ n^{1/2})$.

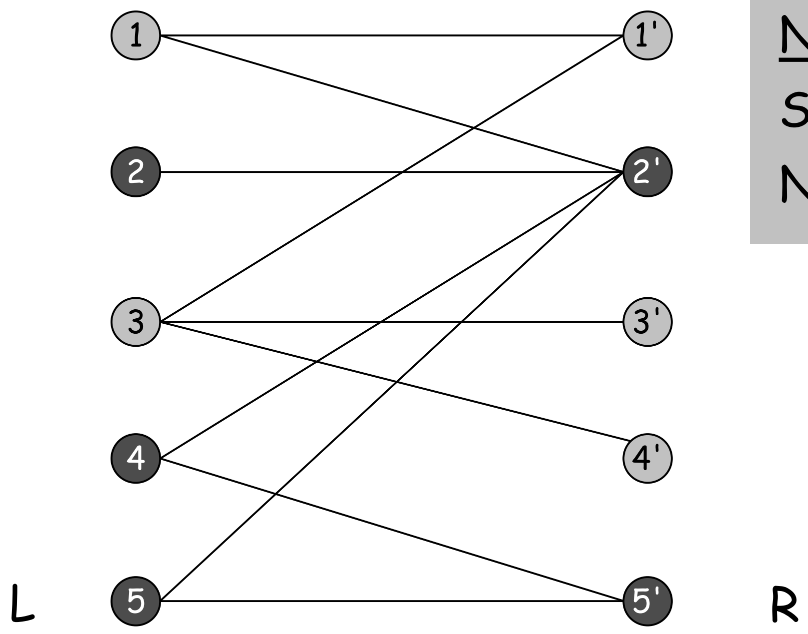
Non-bipartite matching.

- Structure of non-bipartite graphs is more complicated, but well-understood. [Tutte-Berge, Edmonds-Galai]
- Blossom algorithm: $O(n^4)$. [Edmonds 1965]
- Best known: $O(m\ n^{1/2})$. [Micali-Vazirani 1980]

Perfect Matching

Notation. Let S be a subset of nodes, and let N(S) be the set of nodes adjacent to nodes in S.

Observation. If a bipartite graph $G = (L \cup R, E)$, has a perfect matching, then $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.
Pf. Each node in S has to be matched to a different node in N(S).



No perfect matching:

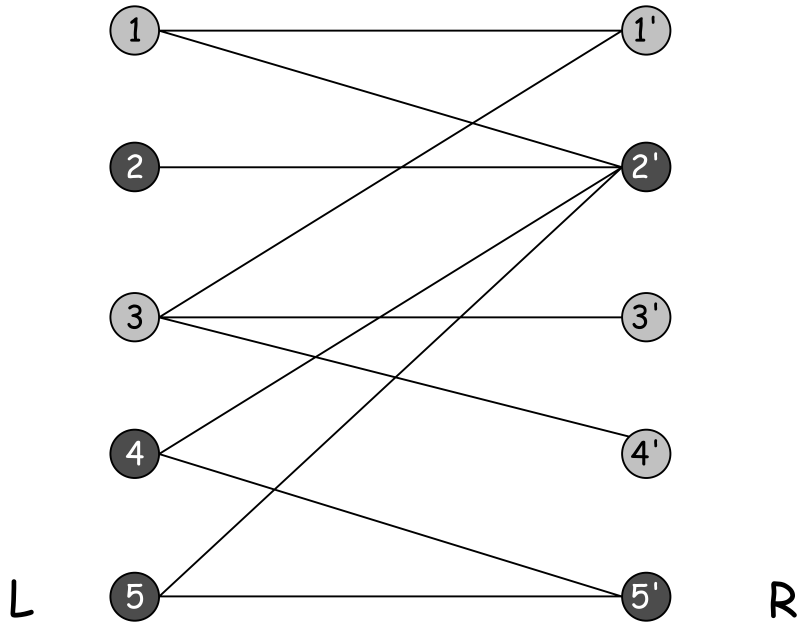
S = { 2, 4, 5 }

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

Marriage Theorem

Marriage Theorem. [Frobenius 1917, Hall 1935] Let $G = (L \cup R, E)$ be a bipartite graph with $|L| = |R|$. Then, G has a perfect matching iff $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.

Pf. $\Rightarrow$ This was the previous observation.



No perfect matching:

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

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

Proof of Marriage Theorem

Pf. $\Leftarrow$ Suppose G does not have a perfect matching.

- Formulate as a max flow problem and let (A, B) be min cut in G'.
- By max-flow min-cut, cap(A, B) < | L |.
- Define $L_A = L \cap A$, $L_B = L \cap B$, $R_A = R \cap A$.
- Since min cut can't use ∞ edges: $N(L_A) \subseteq R_A$.
- cap(A, B) = $| L_B | + | R_A |$ (again, since min cut can't use ∞ edges).
- $|N(L_A)| \leq | R_A | = \text{cap}(A, B) - | L_B | < | L | - | L_B | = | L_A |$.
- Choose $S = L_A$. ▪

L_A = {2, 4, 5}

L_B = {1, 3}

R_A = {2', 5'}

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