Min Cost Flow

Contents.

- Min cost flow.
- Transportation problem.
- Assignment problem.
- Mail carrier problem.
- Klein's cycle-canceling algorithm.
- Network simplex.

Minimum Cost Flow

Minimum cost flow problem.

- Directed network.
- Each edge has a cost and a capacity.
- Each vertex has a supply or demand.
- Find best way to send flow from supply vertices to demand vertices.

Min cost flow generalizes:

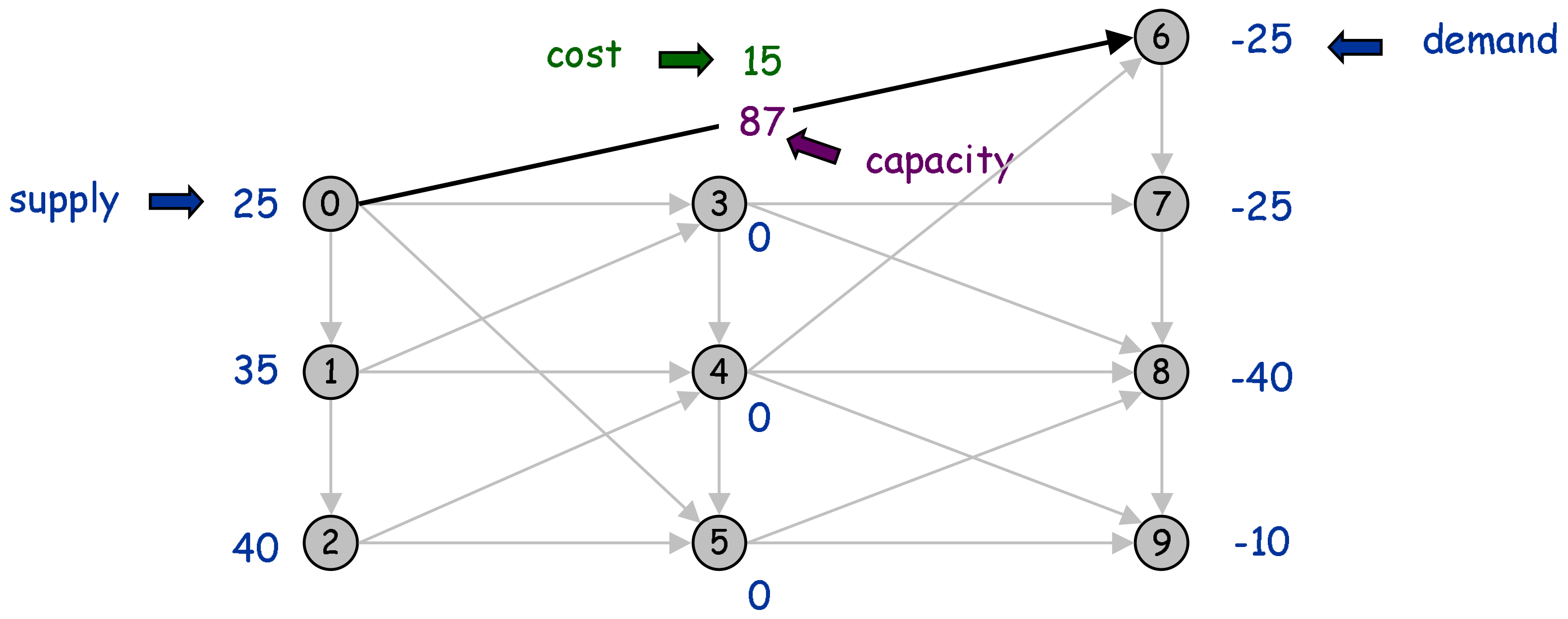
- Transportation problem.
- Assignment problem.
- Mail carrier problem.
- Max flow.
- Shortest path.

One step closer to single ADT for graph problems.

Minimum Cost Flow Problem

Min cost flow problem.

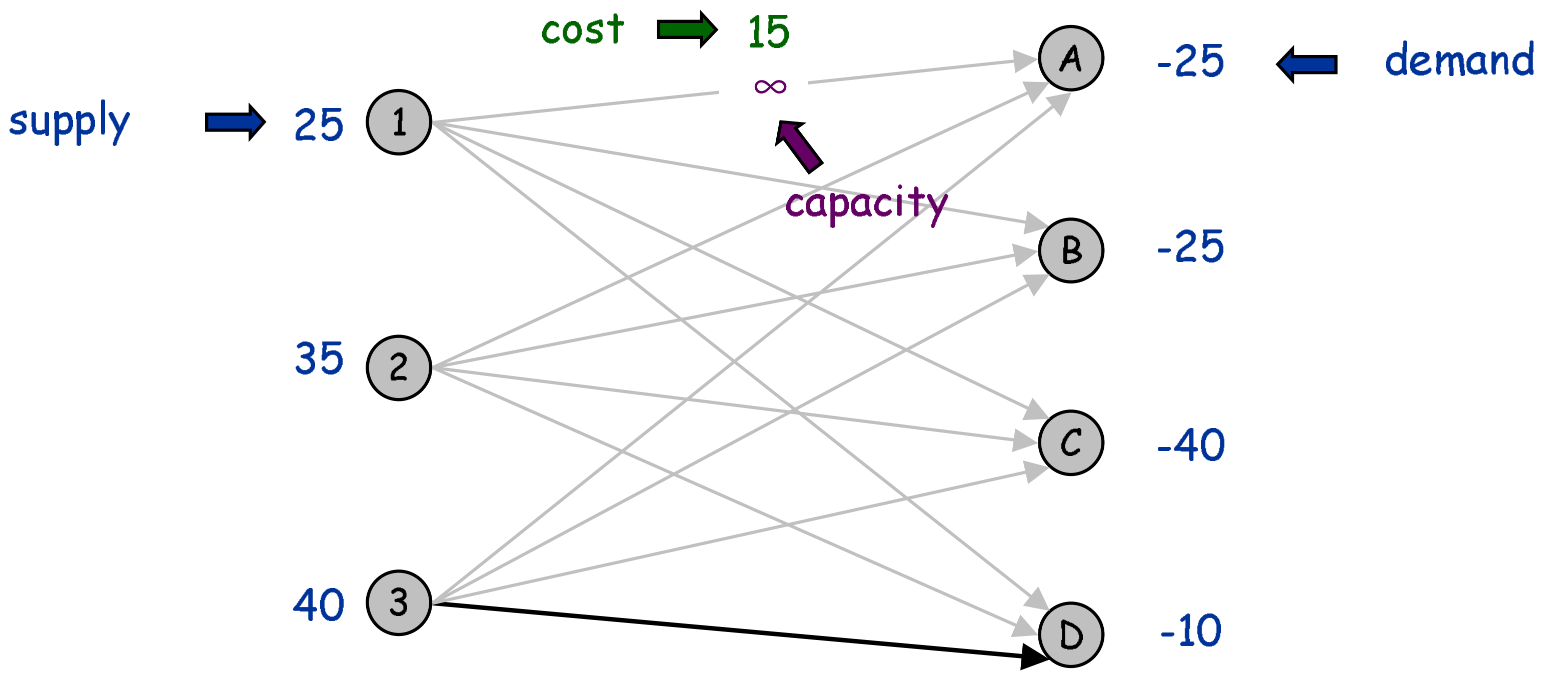
- Send finished good from plants to customers.
- s_v = net supply / demand at vertex v. (sum of supply = sum of demand)
- c_{vw} = unit shipping cost from v to w. (positive or negative)
- u_{vw} = capacity of edge v-w. (infinity ok)
- Goal: satisfy demand at minimize cost.



Transportation Problem

Transportation problem.

- Send finished good from plants to customers.
- s_v = amount produced at plant v.
- d_w = amount demanded by customer w.
- c_{vw} = unit shipping cost from plant v to customer w.
- Goal: minimize total cost.



Transportation Problem: Application

Assign 600 Princeton undergrads to 40 writing seminars.

- Each students ranks top 8 choices.
- Registrar assigns students to seminars.
- Goal: maximize happiness of students.

Model as a transportation problem.

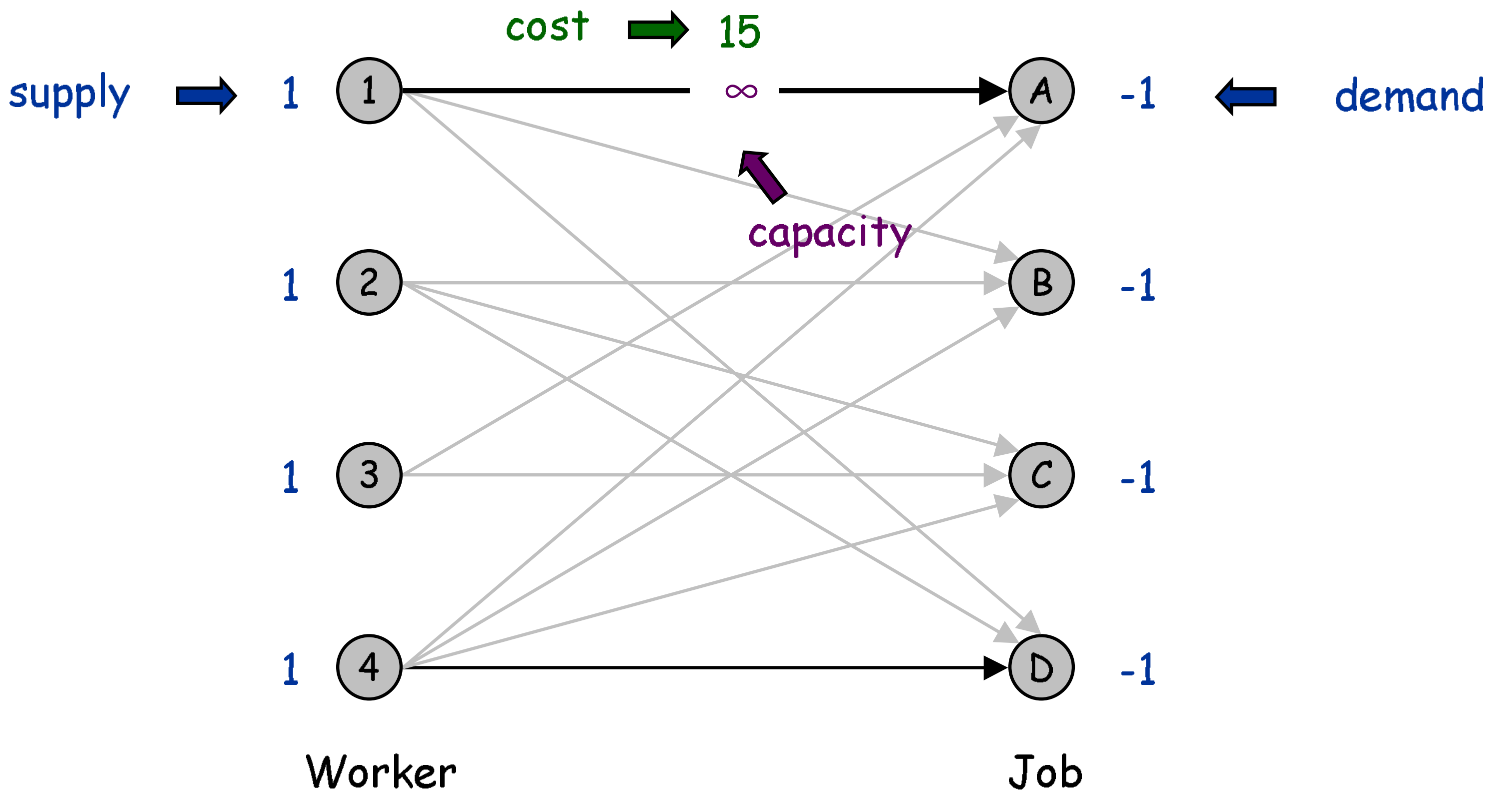
- Student vertices: supply = 1.
- Seminar vertices: supply = 15.
- Cost of assigning student i to seminar j:
  - ∞ if not among top 8 choices
  - r^8 where r = student i's rank of seminar j

choice of function determines tradeoff, e.g., between assigning one student their 2nd choice and another their 5th

Assignment Problem

Assignment problem.

- Assign workers to jobs.
- c_{vw} = cost of assigning worker v to job w.
- Goal: minimize total cost.



Assignment Problem: Applications

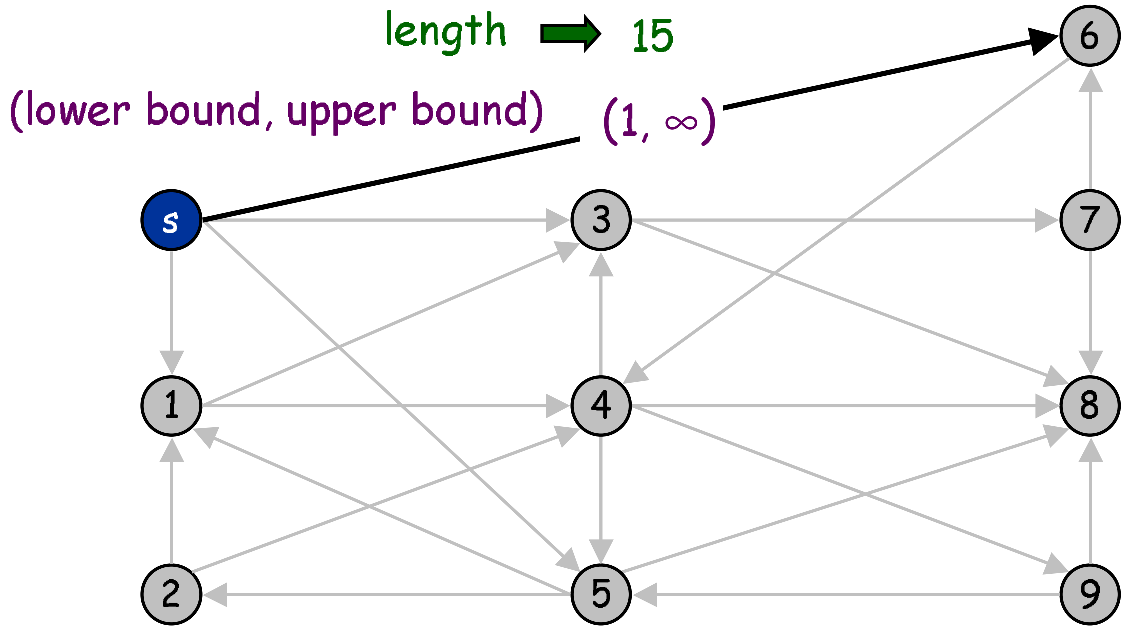
Many important real-world applications.

Left	Right	Optimize
jobs	machines	cost
people	projects	cost
students	dorm rooms	happiness
swimmers	events	chance of winning
service personnel	military postings	relocation cost
bachelors	bachelorettes	compatibility
translators	diplomatic meetings	cost
radar blip at time t	radar blip at time t+1	accuracy

Mail Carrier Problem

Mail carrier problem.

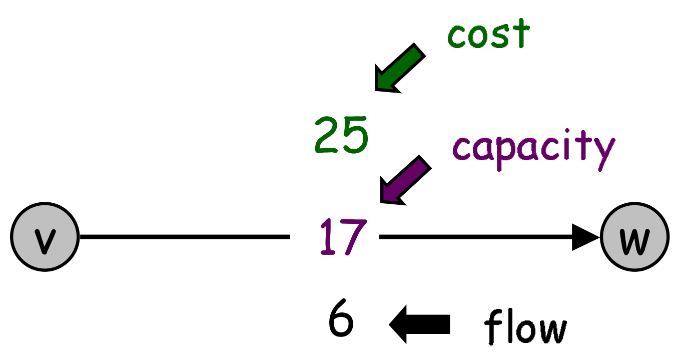
- Post office located at node s.
- Find minimum length route that starts and ends at s and visits each road at least once.
- Need to traverse roads more than once unless graph is Eulerian.



Residual Graph

Original graph.

- Flow f(e).
- Arc e = v-w.

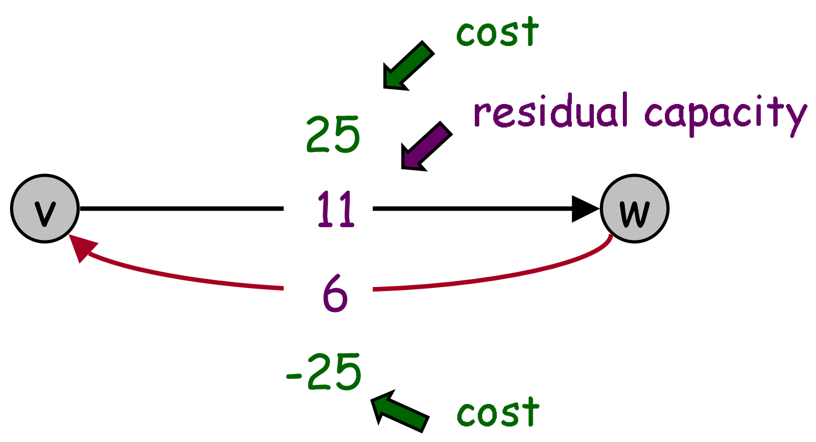


Residual arcs.

- v-w and w-v.
- "Undo" flow sent.

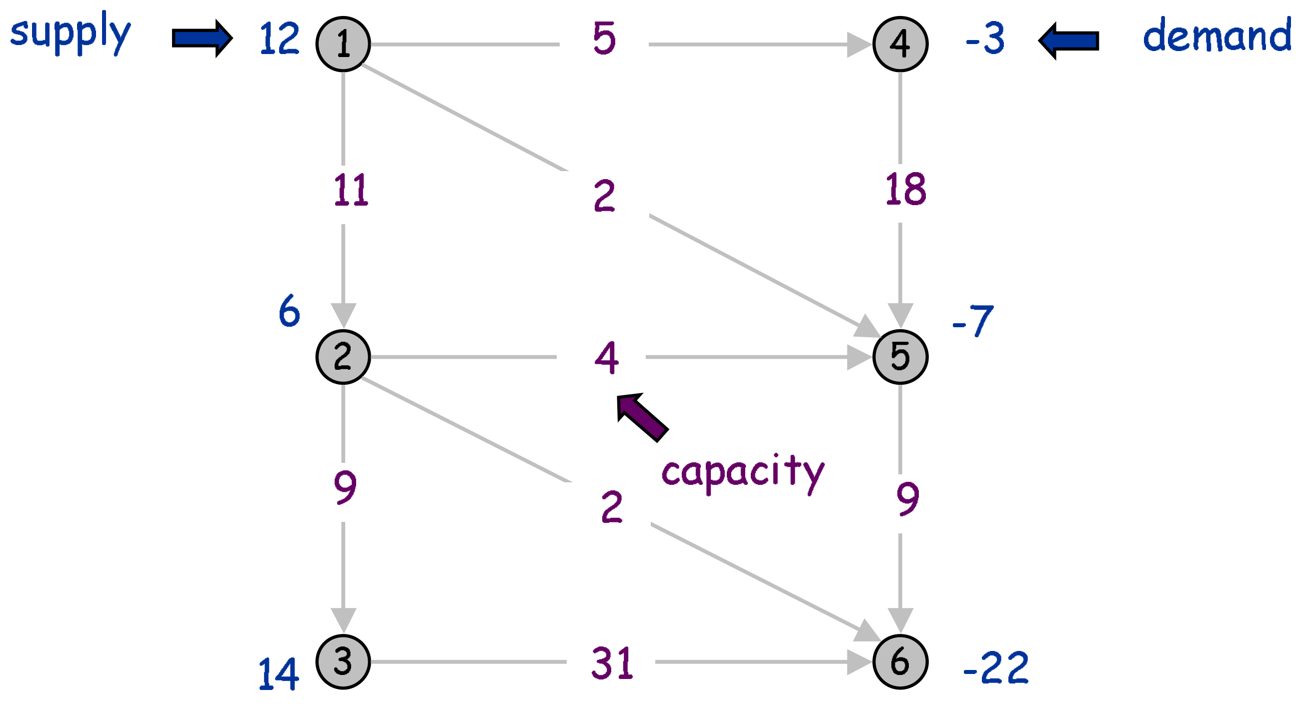
Residual graph.

- All residual arcs with positive capacity.



Finding a Feasible Flow

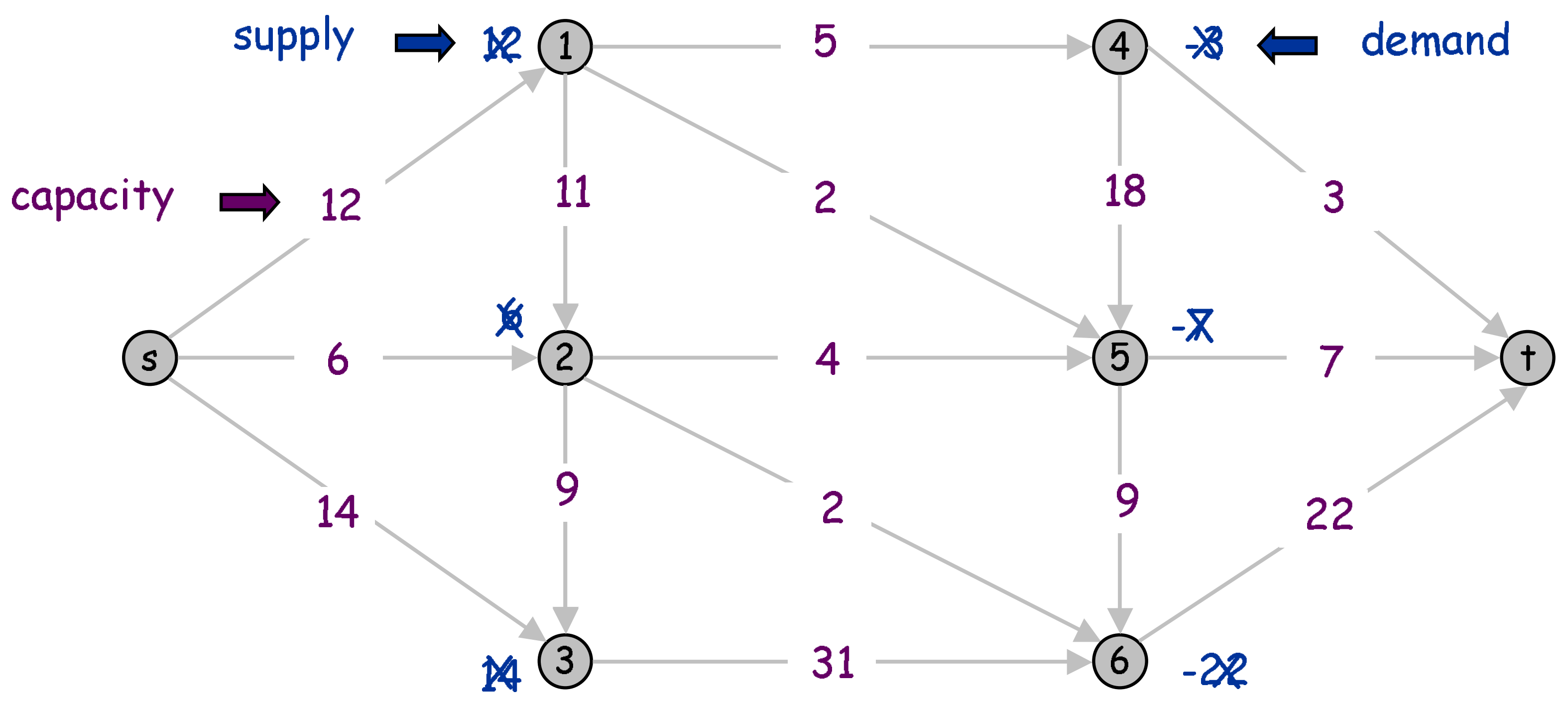
Feasible flow problem: Given a capacitated network with supplies and demands, find a feasible flow if one exists.



Finding a Feasible Flow

Feasible flow problem: Given a capacitated network with supplies and demands, find a feasible flow if one exists.

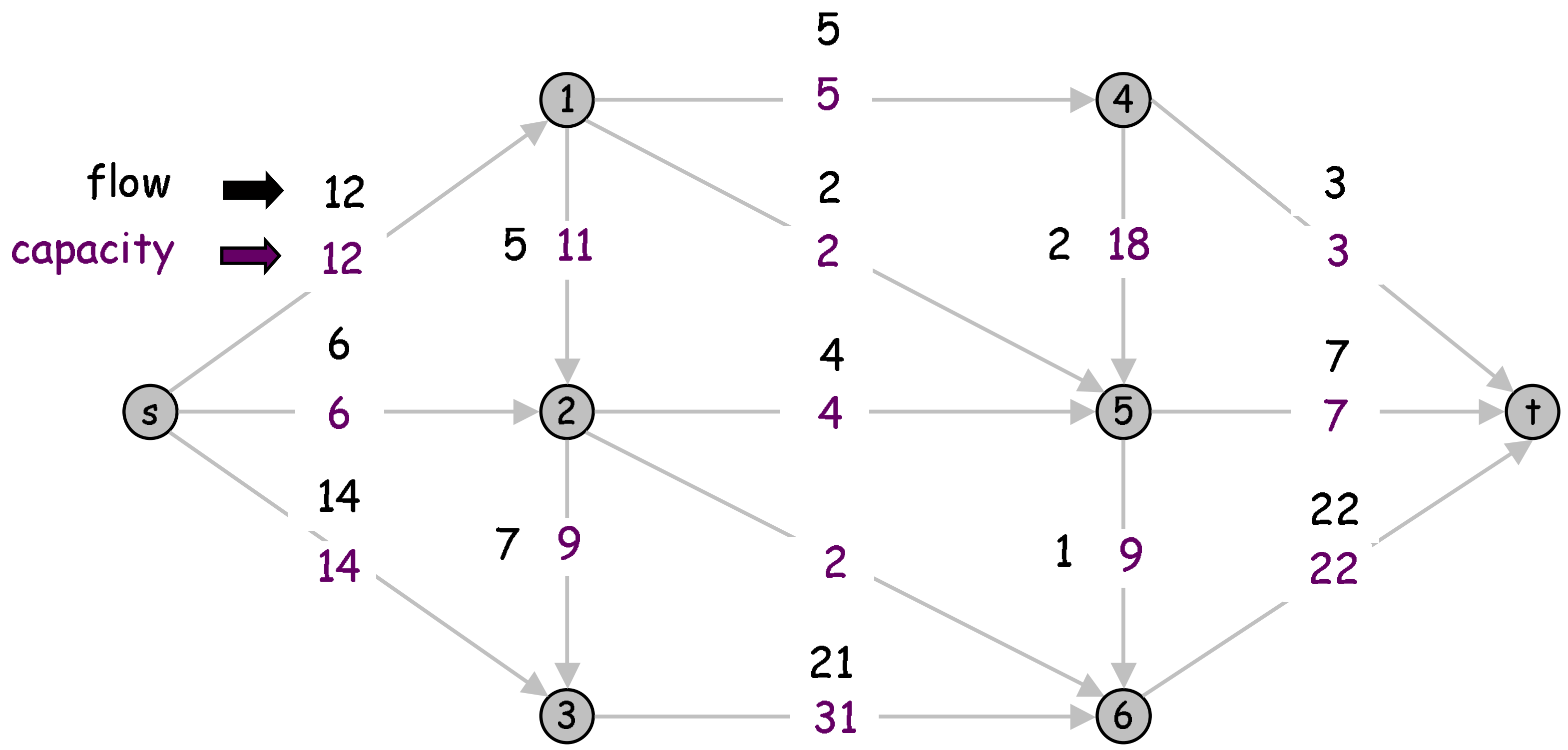
One solution: Solve a maximum flow problem in a related network!



Finding a Feasible Flow

Feasible flow problem: Given a capacitated network with supplies and demands, find a feasible flow if one exists.

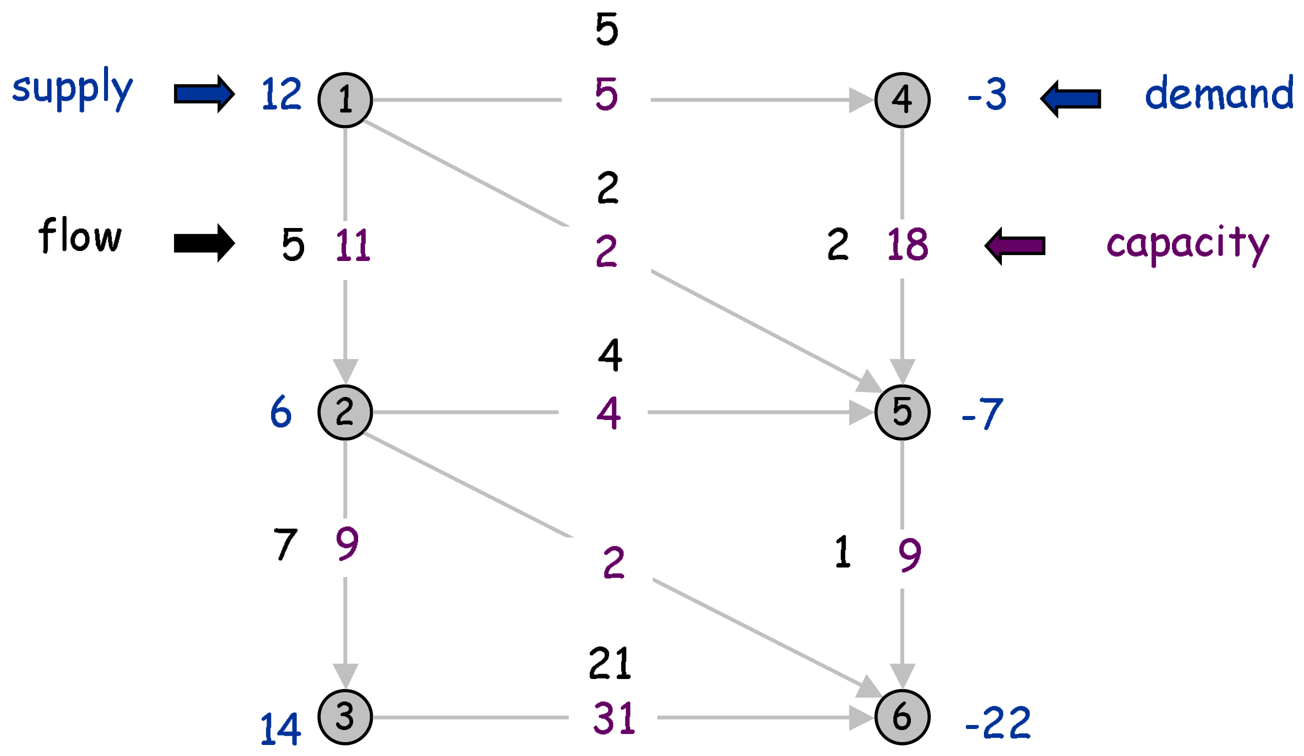
One solution: Solve a maximum flow problem in a related network!



Finding a Feasible Flow

Feasible flow problem: Given a capacitated network with supplies and demands, find a feasible flow if one exists.

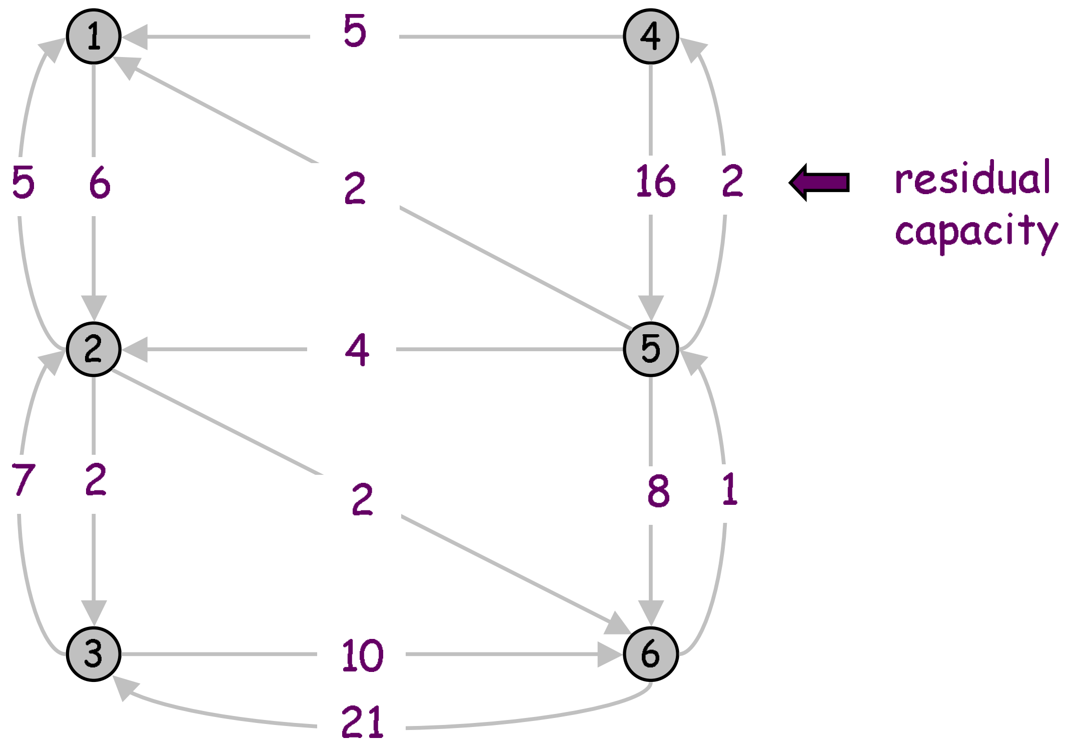
One solution: Solve a maximum flow problem in a related network!



Min Cost Flow Assumptions

Useful assumption for min cost flow problems.

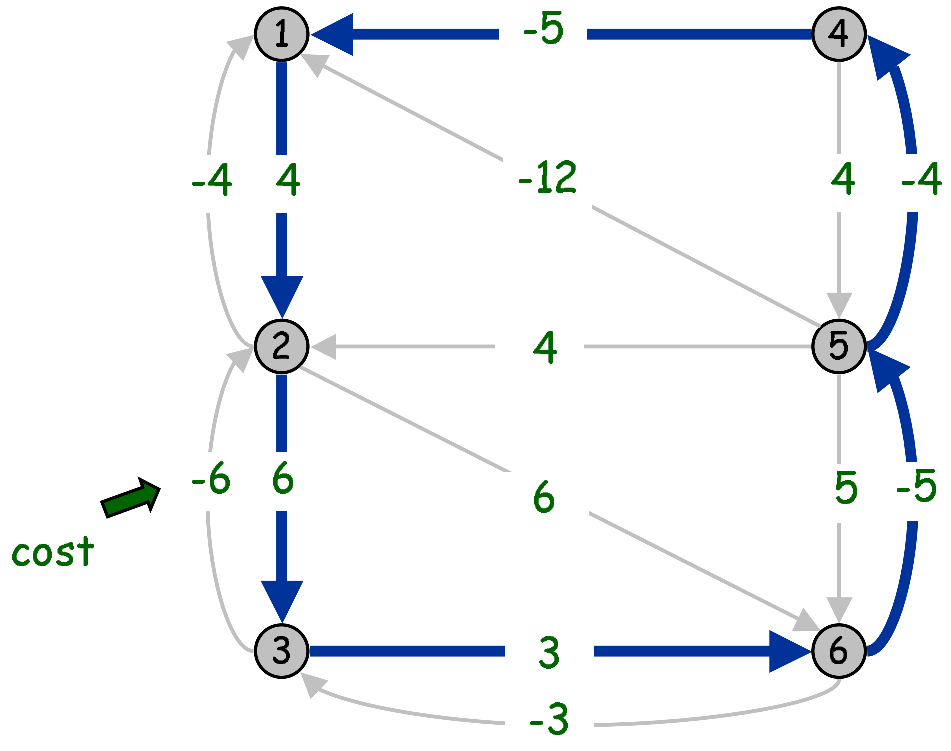
- Underlying graph is connected.
- No supply or demand vertices.
  - find a feasible solution
  - solve problem in residual graph and then translate back



Cycle Canceling Algorithm

How to improve the current feasible flow while maintaining feasibility?

- AUGMENTING CYCLE: negative cost cycle in residual graph.
- Can send flow around cycle.
  - strictly decreases cost
  - preserves feasibility



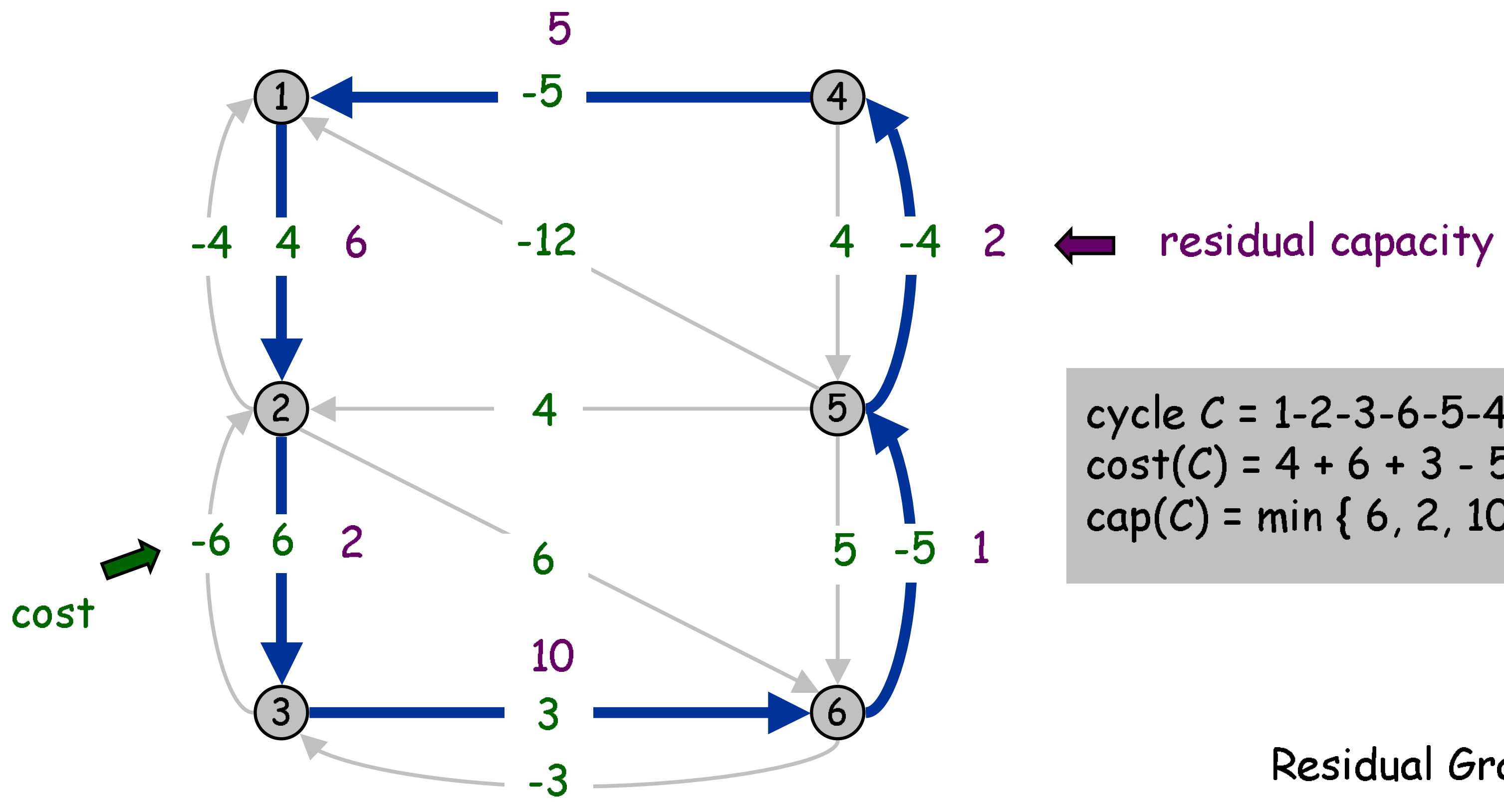
cycle C = 1-2-3-6-5-4-1
cost(C) = 4 + 6 + 3 - 5 - 4 - 5 = -1

Residual Graph

Cycle Canceling Algorithm

How to improve the current feasible flow while maintaining feasibility?

- AUGMENTING CYCLE: negative cost cycle in residual graph.
- Can send flow around cycle.
  - strictly decreases cost
  - preserves feasibility



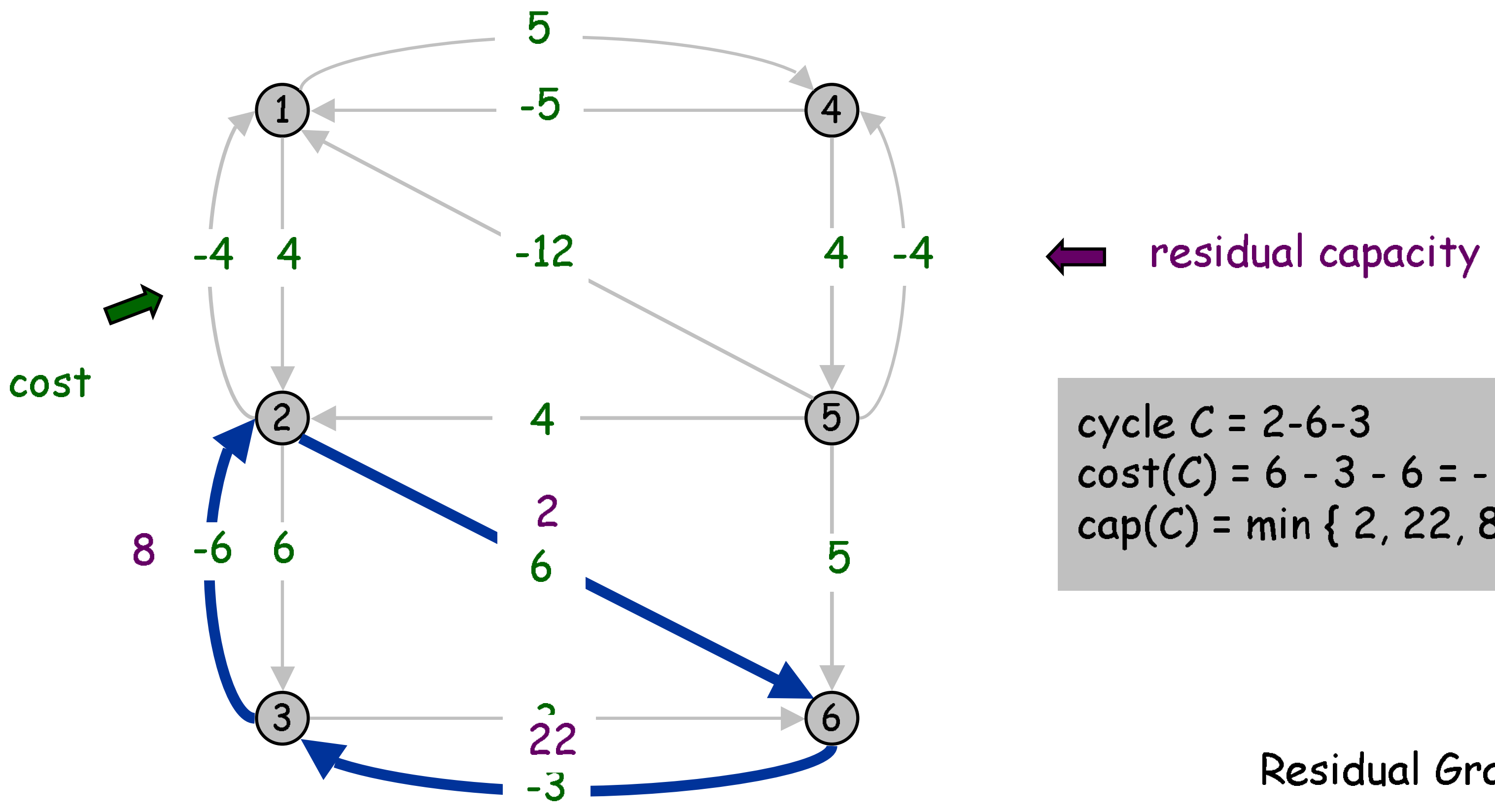
cycle C = 1-2-3-6-5-4-1
cost(C) = 4 + 6 + 3 - 5 - 4 - 5 = -1
cap(C) = min { 6, 2, 10, 1, 2, 5 } = 1

Residual Graph

Cycle Canceling Algorithm

How to improve the current feasible flow while maintaining feasibility?

- AUGMENTING CYCLE: negative cost cycle in residual graph.



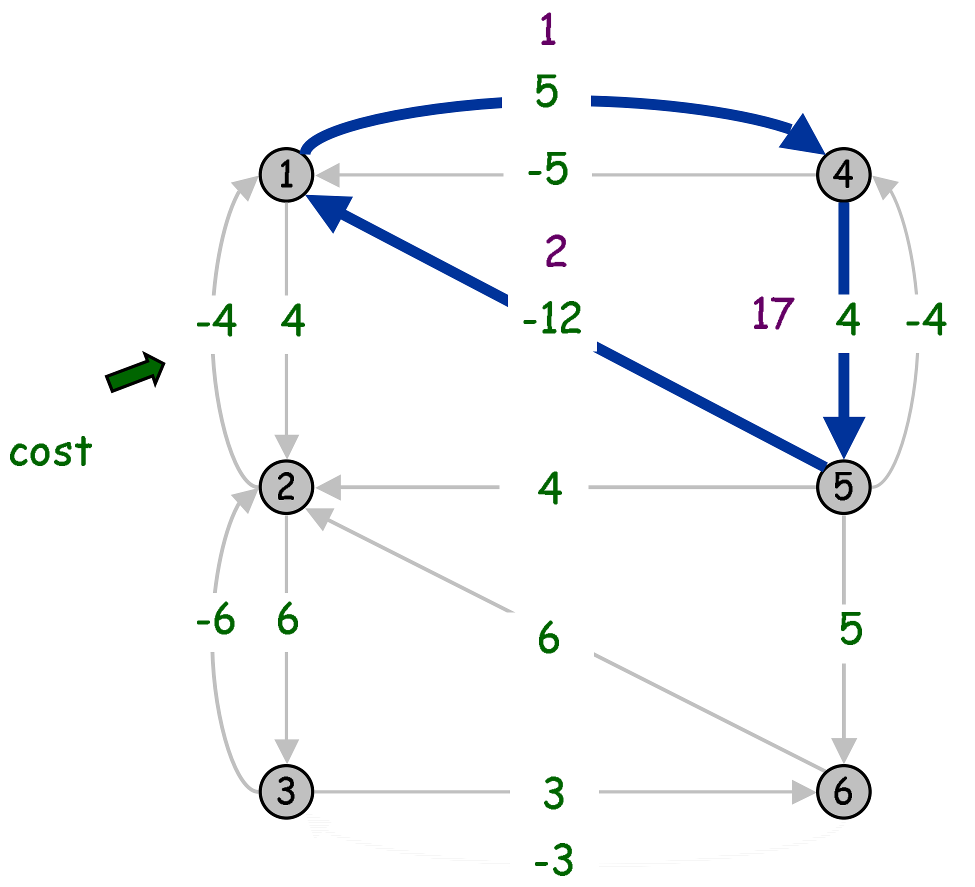
cycle C = 2-6-3
cost(C) = 6 - 3 - 6 = - 3
cap(C) = min { 2, 22, 8 } = 2

Residual Graph

Cycle Canceling Algorithm

How to improve the current feasible flow while maintaining feasibility?

- AUGMENTING CYCLE: negative cost cycle in residual graph.



cycle C = 1-4-5
cost(C) =5 + 4 - 12 = - 3
cap(C) = min { 1, 17, 2 } = 1

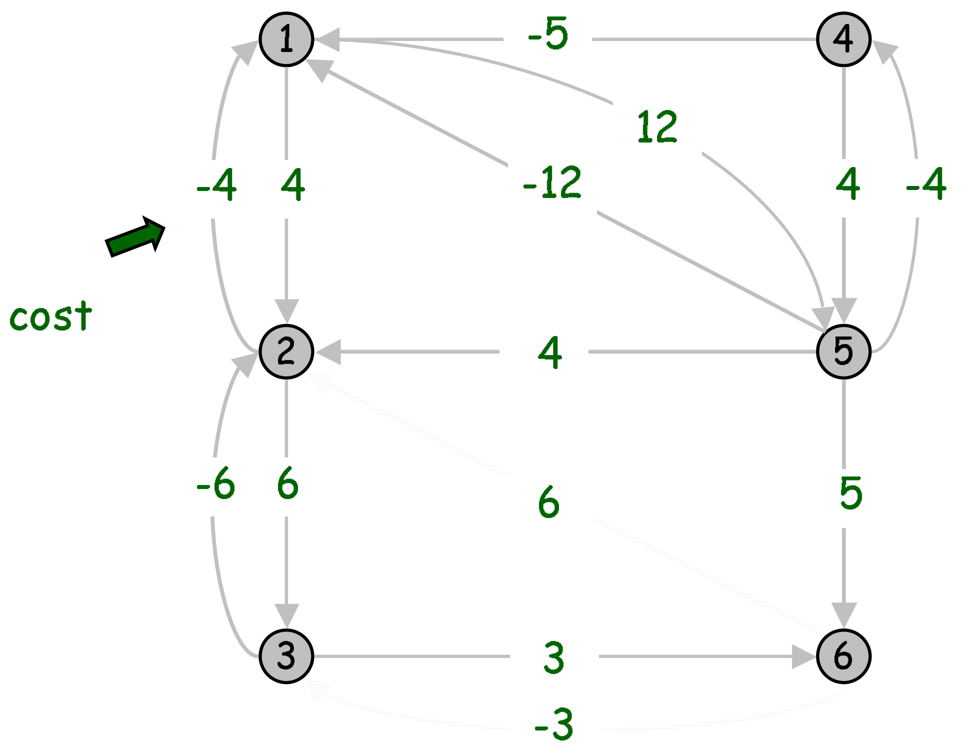
Residual Graph

Cycle Canceling Algorithm

How to improve the current feasible flow while maintaining feasibility?

- AUGMENTING CYCLE: negative cost cycle in residual graph.

Is flow optimal when no more augmenting cycles?



Residual Graph

Cycle Canceling Algorithm

Klein's cycle canceling algorithm.

- Generic method for solving min cost flow problem.
- Analog of Ford-Fulkerson augmenting path algorithm for max flow.

Klein's Cycle Canceling Algorithm

```
Start with a feasible flow f.

REPEAT (until no augmenting cycles)
   Find an augmenting cycle C.
   Augment flow along C.
```

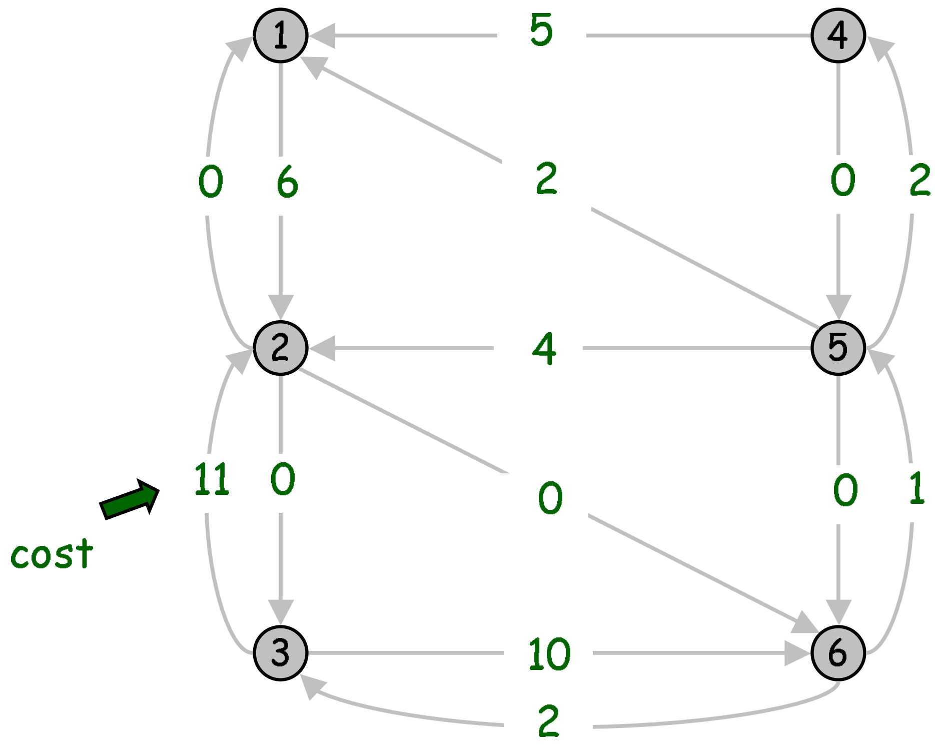
Questions.

- Does this lead to a min cost flow?
- How do we find an augmenting cycle?
- How many augmenting cycles does it take?

Min Cost Flow: Optimality Conditions

Observation. If all residual arcs have ≥ 0 cost, then flow is optimal.

- Current flow is always feasible.
- Any change in flow can only increase cost.

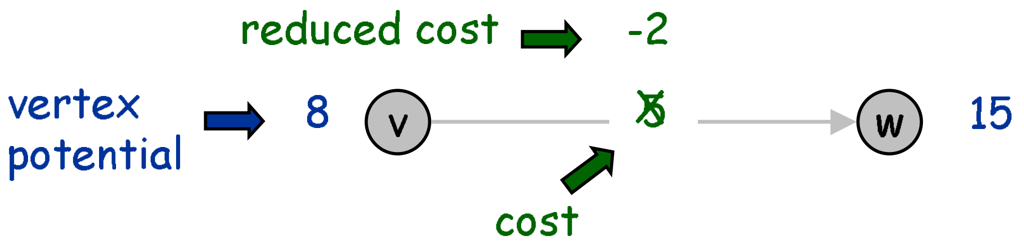


Residual Graph

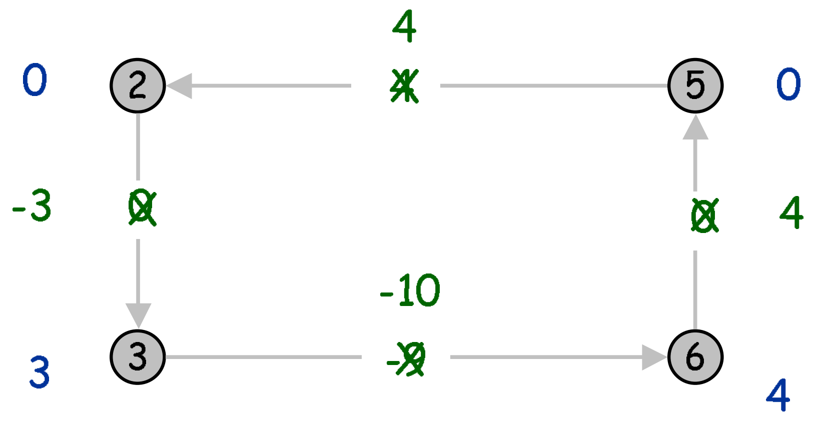
Min Cost Flow: Reduced Cost

Reduced cost: given vertex potentials φ(v), the reduced cost of edge v-w is c(v, w) + φ(v) - φ(w).

Intuition. φ(v) = market price for one unit of flow a v.



Observation. Cost of cycle = reduced cost of cycle.



cost = -5
reduced cost = -5

Min Cost Flow: Optimality Conditions

Theorem. A feasible flow f is optimal if and only if there are no augmenting cycles.

Corollary. If Klein's algorithm terminates, it terminates with an optimal flow.

Proof.

- If augmenting cycle, decrease cost by sending flow around cycle.
- If no augmenting cycle, compute shortest path $\phi(v)$ from s to every node v in residual graph.
  - $\phi(w) \le \phi(v) + c(v, w)$
  - using ϕ as vertex potentials, all arcs have reduced cost ≥ 0
  - thus, current flow is optimal

Running Time

Assumption: all capacities are integers between 1 and U; all costs are integers between –C and C.

Invariant: every flow value and every residual capacity remain integral throughout Klein's algorithm.

Theorem: Klein's algorithm terminates after at most E U C iterations.

- Each augmenting cycle decrease cost by at least 1.

not polynomial in input size!

Integrality theorem: if all arc capacities, supplies, and demands are integers, then there exists an integral min cost flow.

- Assignment problem formulation relies on this fact.
- Can't route 1/2 airplane from Princeton to Palo Alto.

Finding A Negative Cost Cycle

How to find an augmenting cycle?

- Run Bellman-Ford in residual graph.
- O(E V) time per cycle.

How many cycles will we need to cancel?

- Some rules lead to exponential algorithms.
- Clever rules lead to polynomial algorithms.
  - generalize shortest augmenting path
  - generalize fattest augmenting path

Can we reduce the time needed to find a negative cycle?

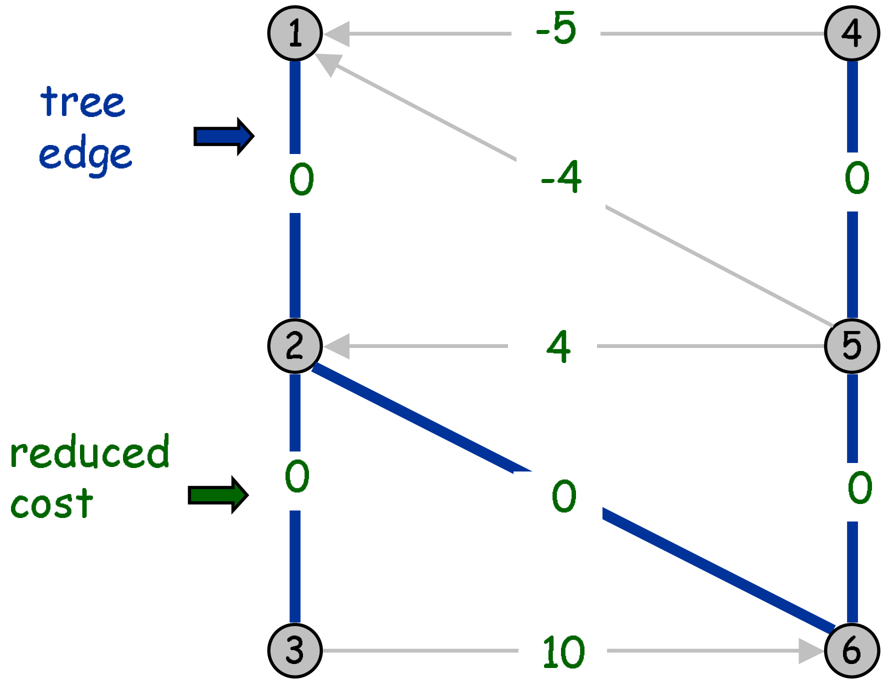
- No, unless we solve a major open research problem.
- Yes, since we can reuse information from iteration to iteration.
- Result: network simplex method.

Network Simplex

Maintain a spanning tree and vertex potentials π such that:

- All non-tree arcs e either have flow(e) = 0 or flow(e) = cap(e).
- All tree arcs have 0 reduced cost.
- Always possible since it's a tree.

ANY residual arc with neg reduced cost completes a neg cost cycle.

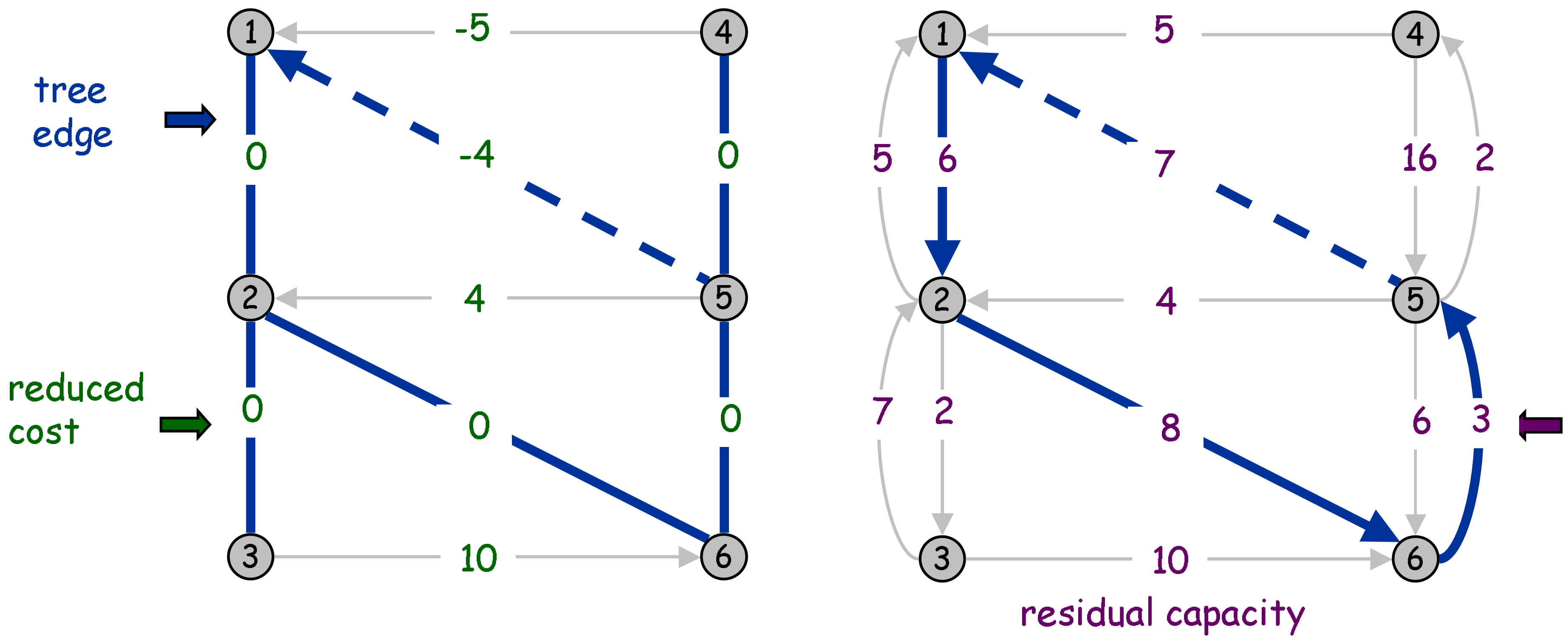


edge 5-1 has reduced cost -4
cycle *C* = 1-2-6-5-1
redcost(*C*) = cost(*C*) = -4

Network Simplex

How to update spanning tree?

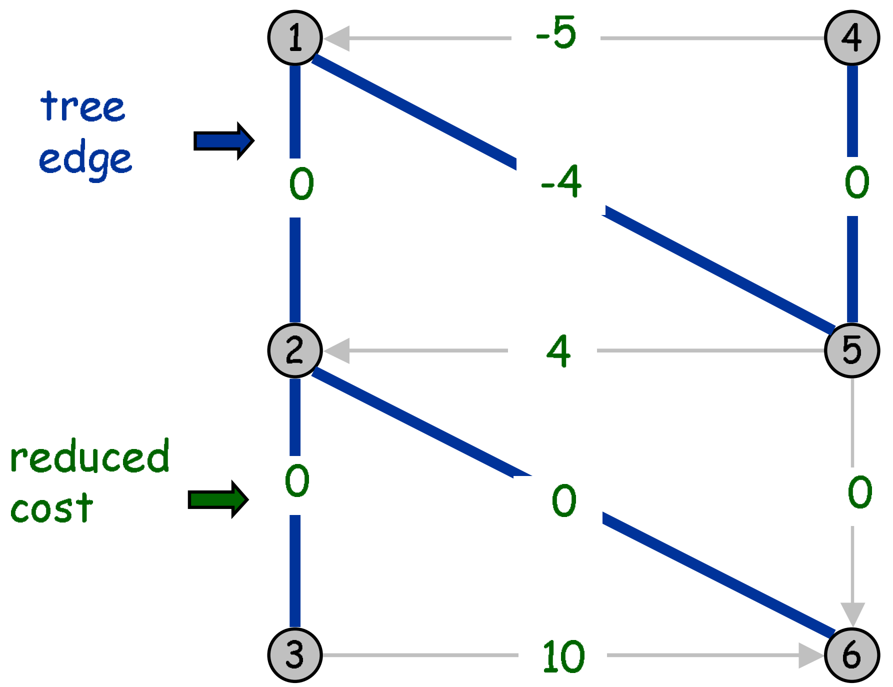
- Find bottleneck capacity θ.

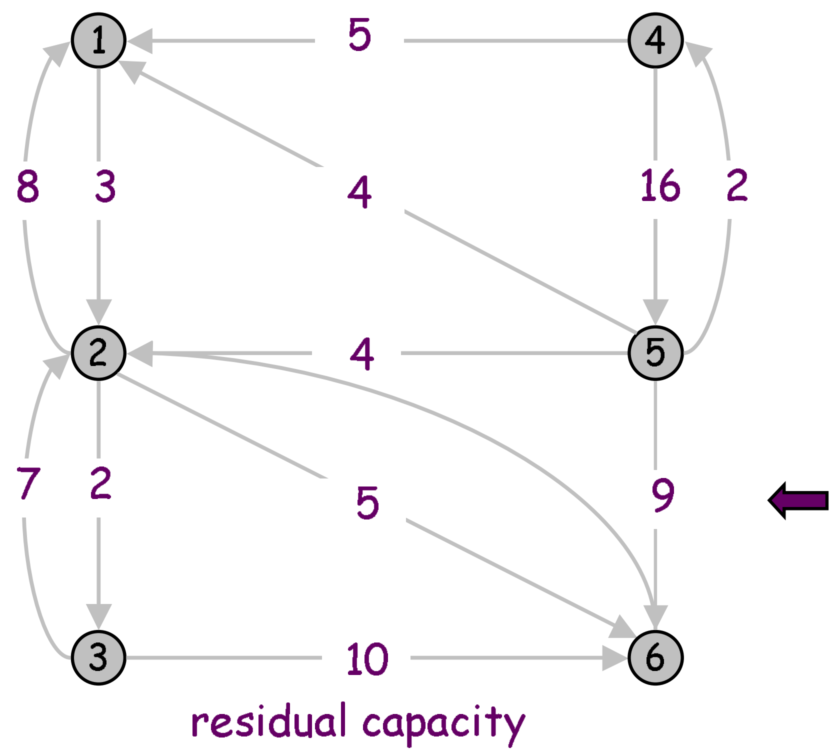


Network Simplex

How to update spanning tree?

- Find bottleneck capacity θ.
- Decrease flow on some edges by θ, increase it by θ on others.
- Delete a bottleneck edge from spanning tree; insert new edge.

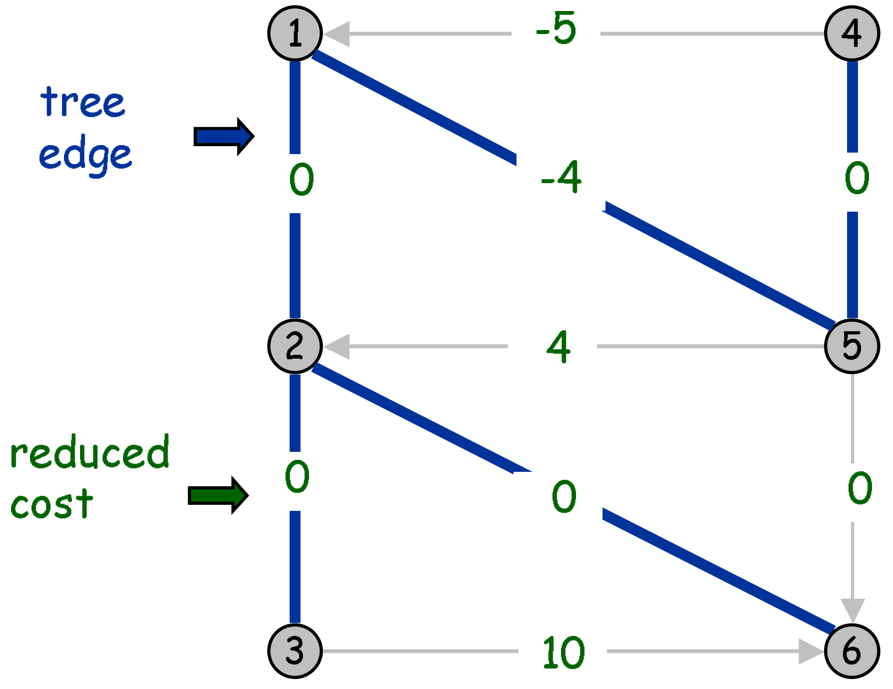


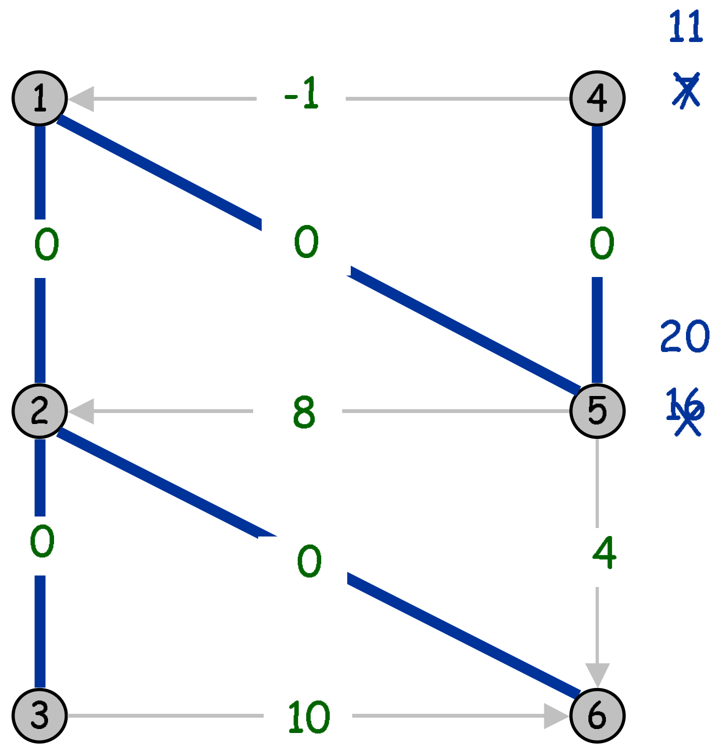


Network Simplex

How to update spanning tree?

- Find bottleneck capacity θ.
- Decrease flow on some edges by θ, increase it by θ on others.
- Delete a bottleneck edge from spanning tree; insert new edge.
- Recompute vertex potentials.





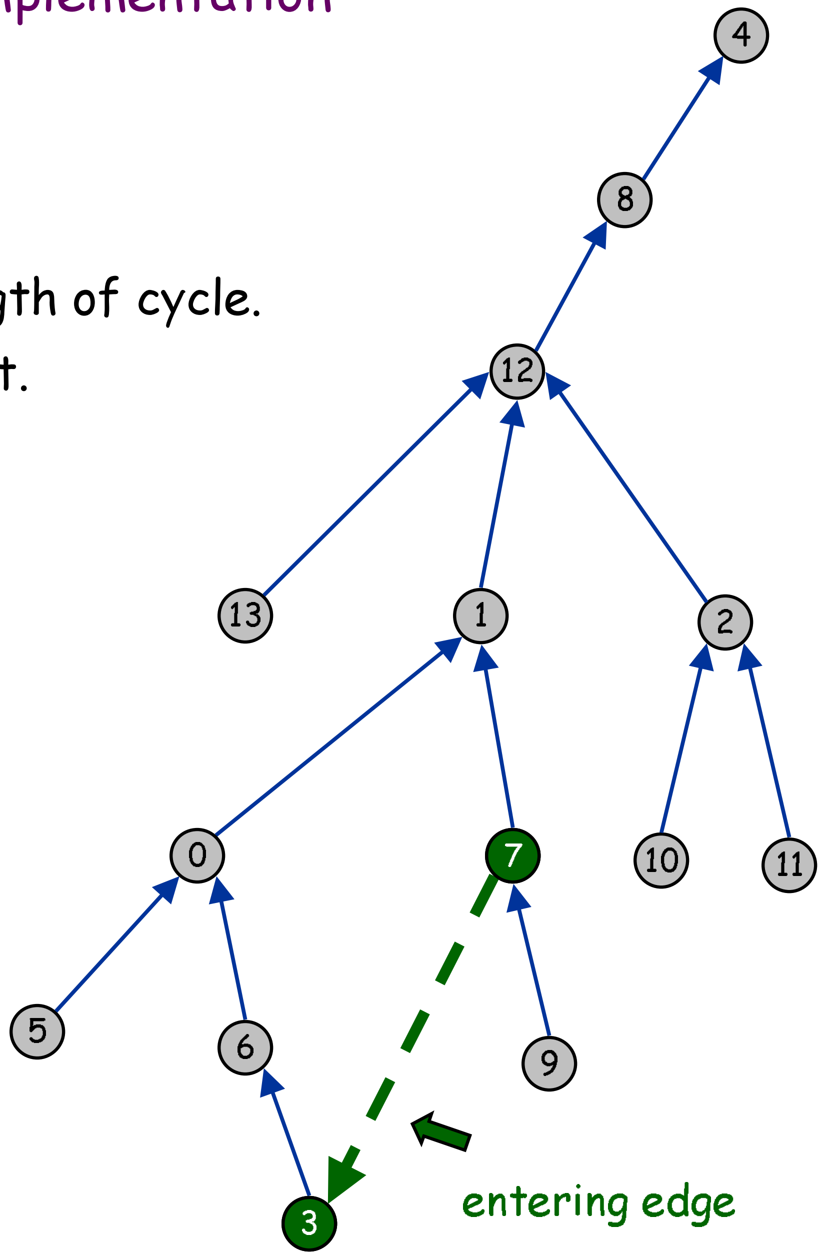
Network Simplex: Implementation

How to find cycle?

- Can find in O(V) time using DFS.
- Goal: find in time proportional to length of cycle.
- In practice, length of cycle very short.

Use parent-link representation of tree.

- Climb tree from two endpoints until you hit least common ancestor.



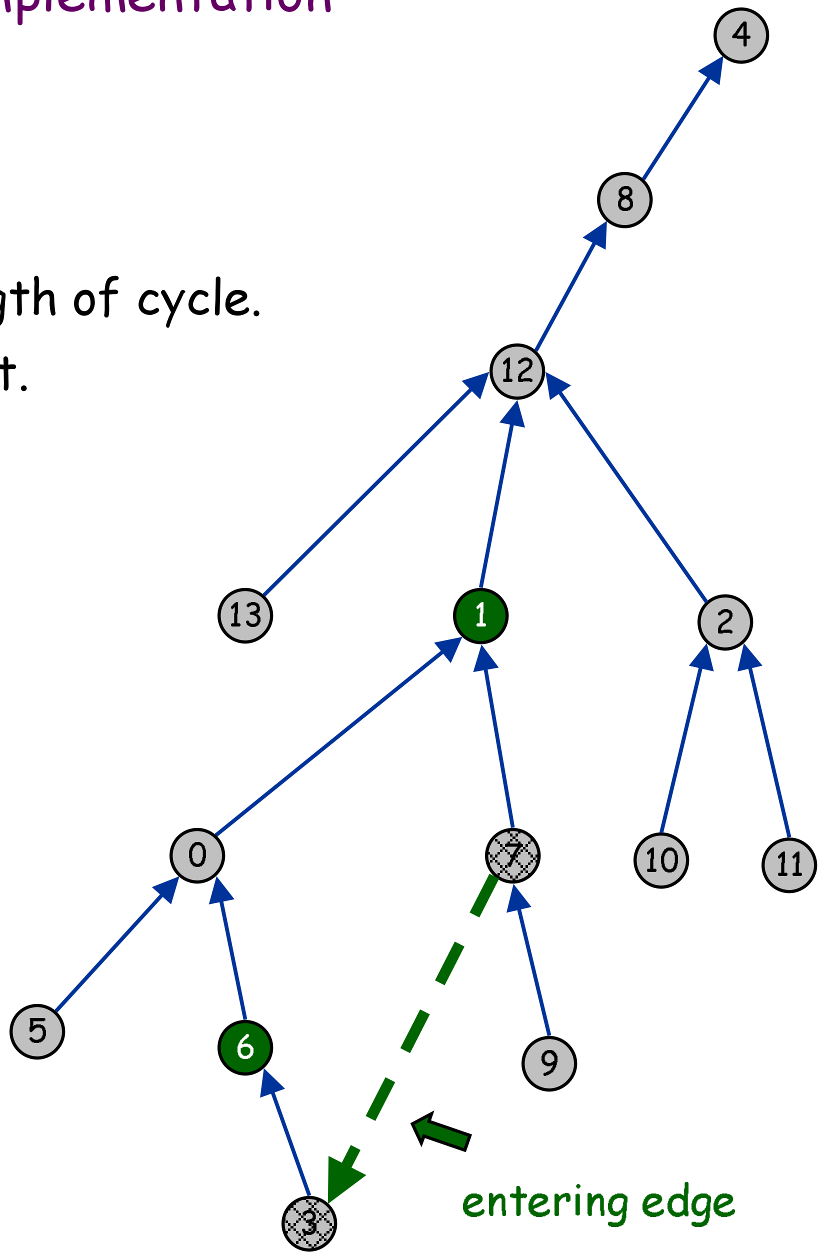
Network Simplex: Implementation

How to find cycle?

- Can find in O(V) time using DFS.
- Goal: find in time proportional to length of cycle.
- In practice, length of cycle very short.

Use parent-link representation of tree.

- Climb tree from two endpoints until you hit least common ancestor.



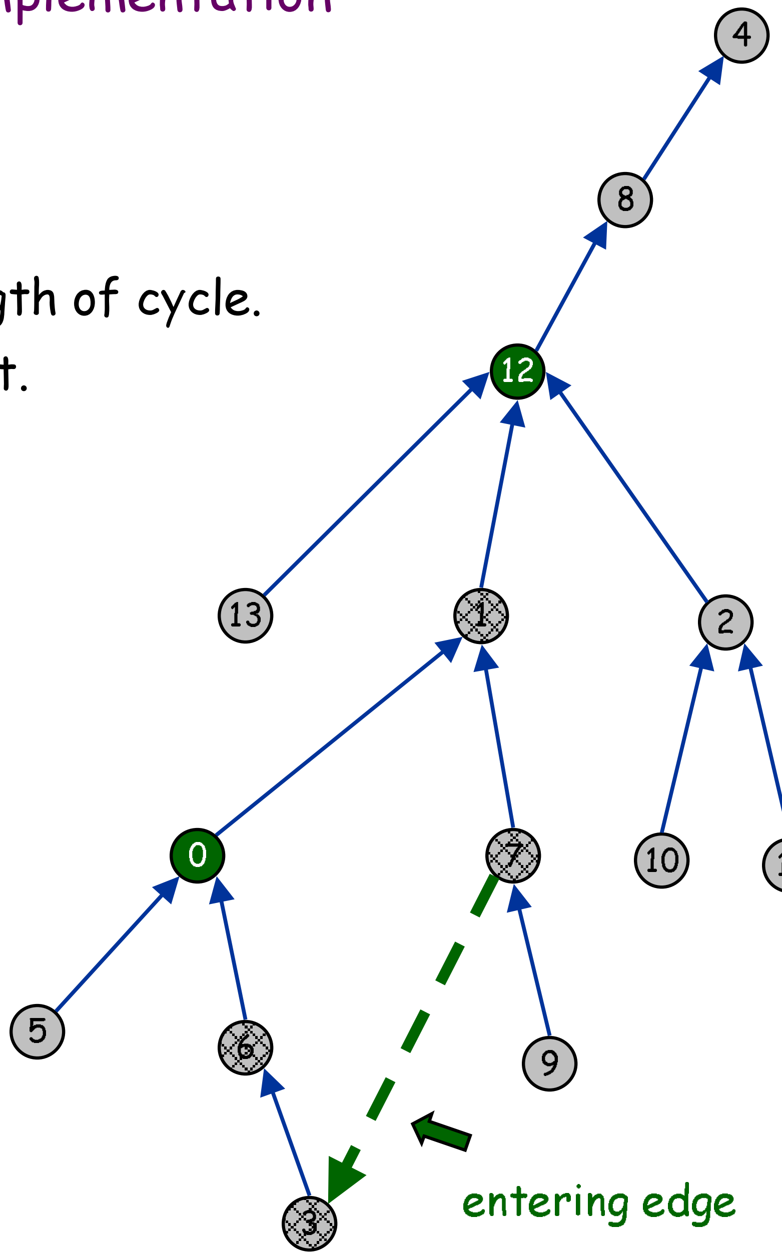
Network Simplex: Implementation

How to find cycle?

- Can find in O(V) time using DFS.
- Goal: find in time proportional to length of cycle.
- In practice, length of cycle very short.

Use parent-link representation of tree.

⇒ - Climb tree from two endpoints until you hit least common ancestor.



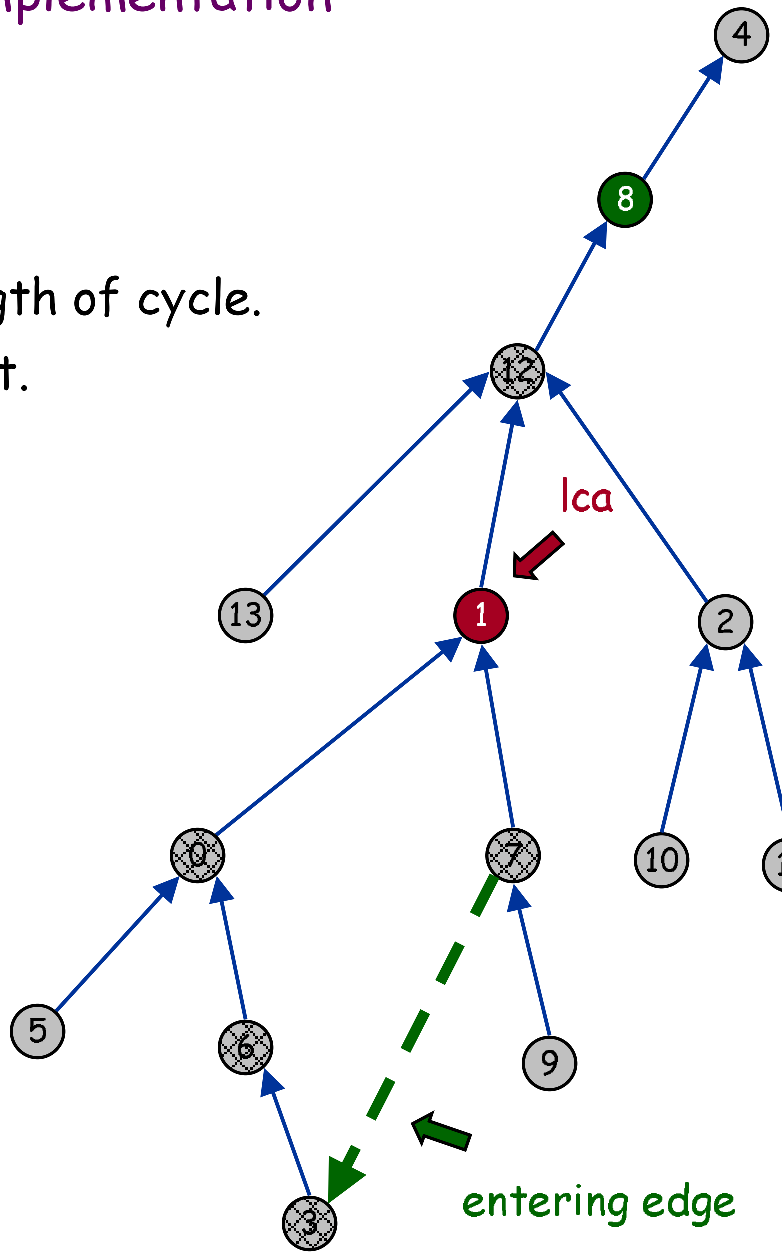
Network Simplex: Implementation

How to find cycle?

- Can find in O(V) time using DFS.
- Goal: find in time proportional to length of cycle.
- In practice, length of cycle very short.

Use parent-link representation of tree.

➡ - Climb tree from two endpoints until you hit least common ancestor.



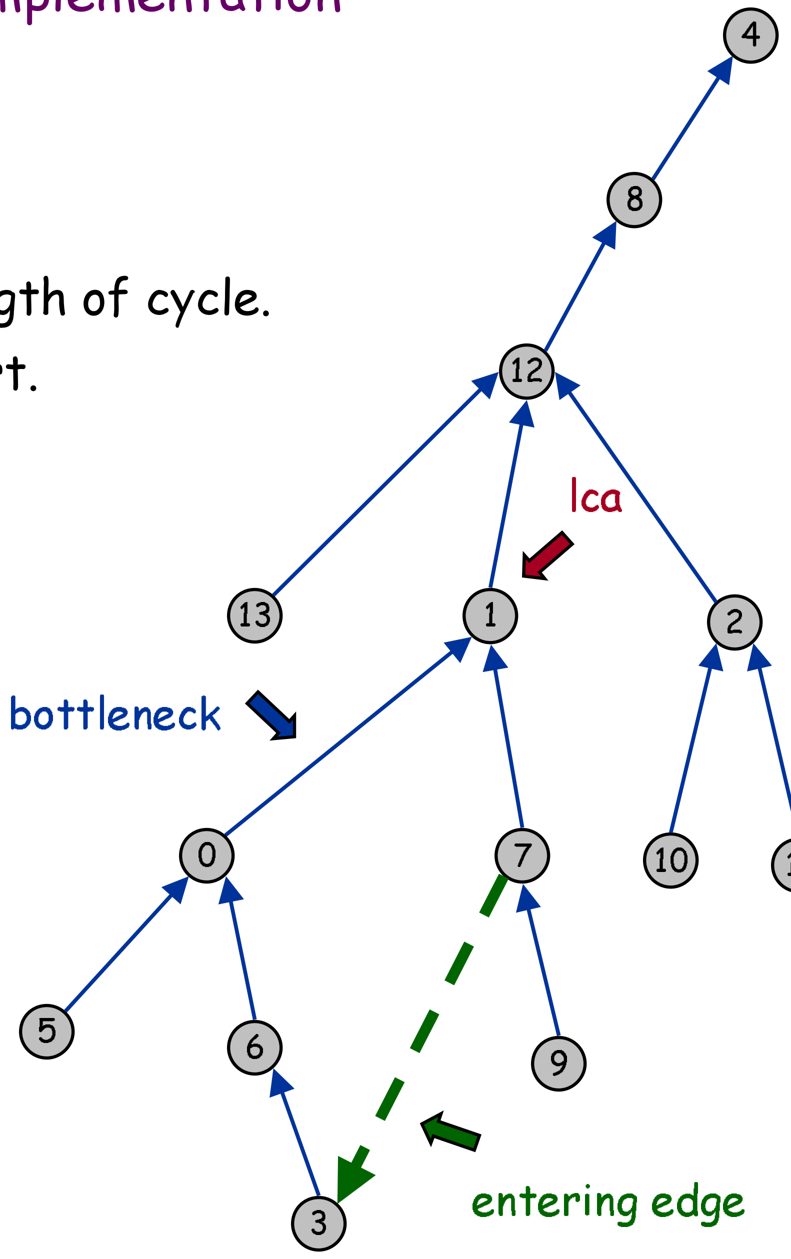
Network Simplex: Implementation

How to find cycle?

- Can find in O(V) time using DFS.
- Goal: find in time proportional to length of cycle.
- In practice, length of cycle very short.

Use parent-link representation of tree.

- Climb tree from two endpoints until you hit least common ancestor.
- ➡ Delete bottleneck edge.



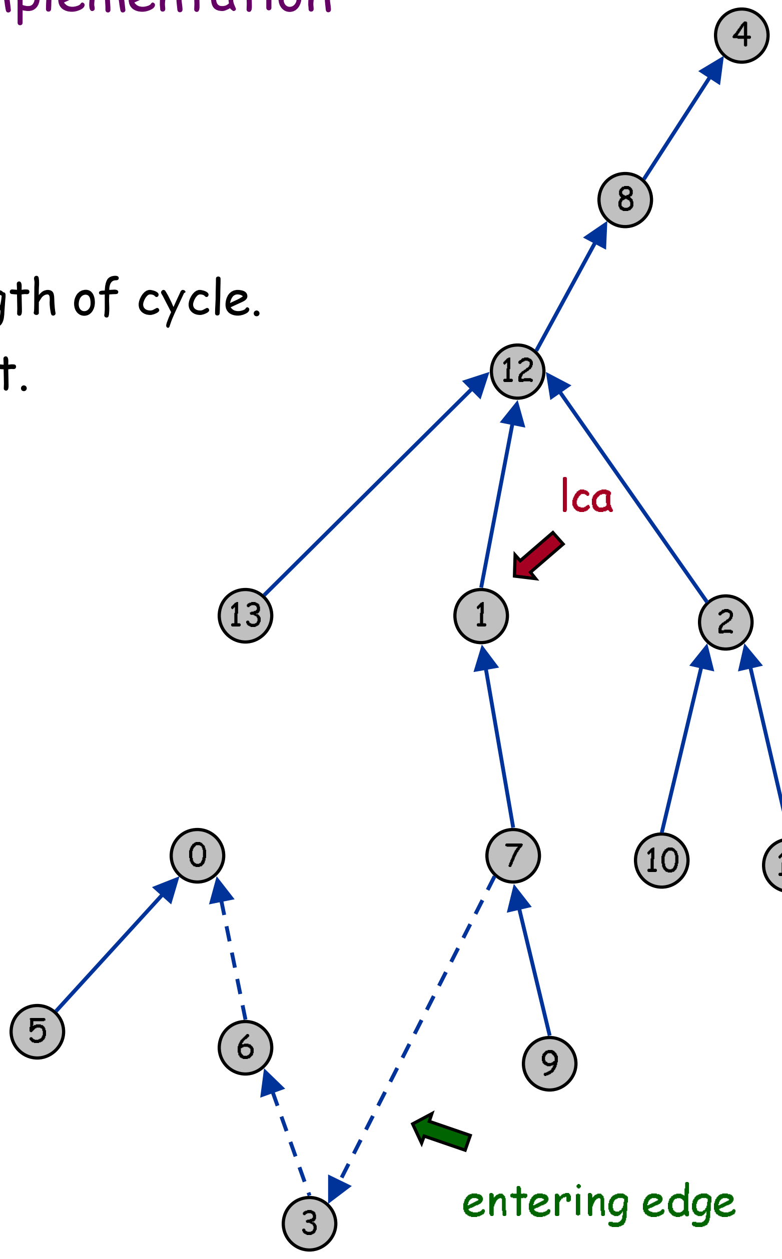
Network Simplex: Implementation

How to find cycle?

- Can find in O(V) time using DFS.
- Goal: find in time proportional to length of cycle.
- In practice, length of cycle very short.

Use parent-link representation of tree.

- Climb tree from two endpoints until you hit least common ancestor.
- Delete bottleneck edge.
- ➡ Update tree by reversing subpath.



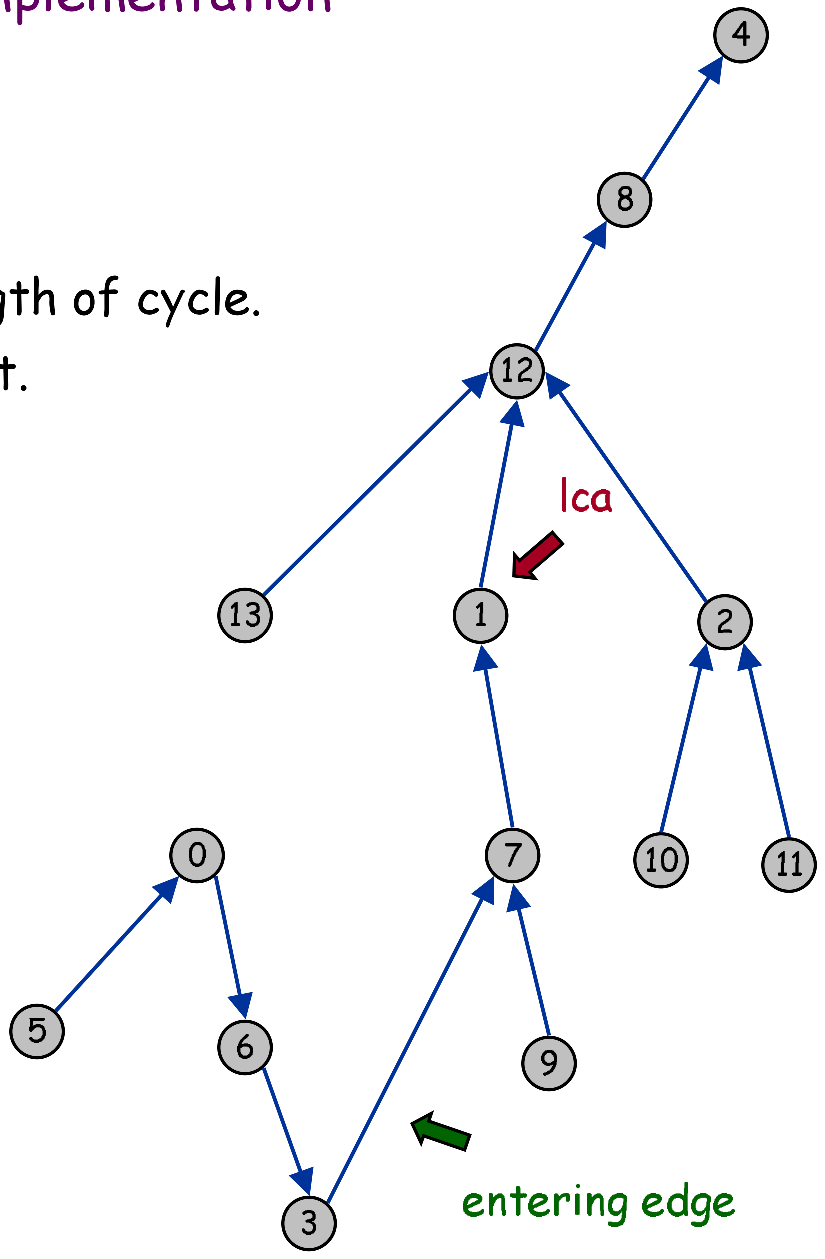
Network Simplex: Implementation

How to find cycle?

- Can find in O(V) time using DFS.
- Goal: find in time proportional to length of cycle.
- In practice, length of cycle very short.

Use parent-link representation of tree.

- Climb tree from two endpoints until you hit least common ancestor.
- Delete bottleneck edge.
- Update tree by reversing subpath.



Network Simplex Issues

Which edge should I add to tree?

- Any one with negative reduced cost works.
- Use first one $\Rightarrow$ less time searching for cycle.
- Use most negative one $\Rightarrow$ maximize rate at which cost decreases.
- Candidate list $\Rightarrow$ practical tradeoff.

Degeneracy: when bottleneck capacity = 0.

- Can happen if tree arc is at upper or lower bound.
- Can still make progress since spanning tree changes.
- Common in practice, slows down algorithm.
  - up to 90% degenerate pivots

Can degeneracy lead to infinite loop? Yes, but cycling rare in practice.
Can this be avoided? Yes, choose leaving edge using Cunningham's rule.

Development of Min Cost Flow Algorithms

Assumptions.

- Arc capacities between 1 and U, costs between $-C$ and C.
- Ignore log V factors.

Year	Discoverer	Method	Big-Oh ~
1951	Dantzig	Network simplex	$E^2 V^2 U$
1960	Minty, Fulkerson	Out of kilter	$E V U$
1958	Jewell	Successive shortest path	$E V U$
1962	Ford-Fulkerson	Primal dual	$EV^2 U$
1967	Klein	Cycle canceling	$E^2 C U$
1972	Edmonds-Karp, Dinitz	Capacity scaling	$E^2 \log U$
1973	Dinitz-Gabow	Improved capacity scaling	$E V \log U$
1980	Rock, Bland-Jensen	Cost scaling	$E V^2 \log C$
1985	Tardos	ε-optimality	poly(E, V)
1988	Orlin	Enhanced capacity scaling	E^2

Hard to beat optimized network simplex in practice . . .
But fastest algorithms use sophisticated "scaling" techniques.

Conclusions

Min cost flow is important because:

- It's a very general problem solving model.
- There are many fast and practical algorithms.

Min cost flows relies on algorithmic machinery we've been building up:

- Graph.
- Shortest path problem.
- Max flow problem.
- Parent-link representation.