MINIMUM COST FLOW

Sarina Heshmati

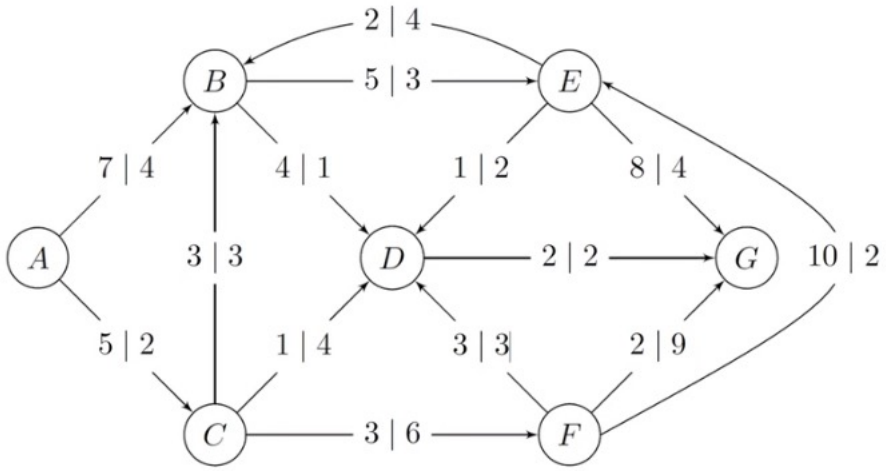
sarinaheshmatii@gmail.com

PROBLEM DEFINITION

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

PROBLEM DEFINITION

- It generalizes maximum flow problem
- It also includes the concept of shortest path
- In a nutshell each edge has both cost and capacity



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

APPLICATIONS

- Transportation and logistics
- Supply chain management
- Network design

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

PROBLEM DEFINITION

- b-flow instead of s-t-flow
- No specific source or sink!
- Hence the concept of node balances

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

NODE BALANCES

- The sum is equal to zero
- Basically the balance indicates how much flow is created or destroyed by each node
- On the contrary, in the max-flow problem, except for s and t, on each node, no flow is created nor destroyed, it is only distributed
- By conclusion, there could be many sources and many sinks in the graph

PROBLEM DEFINITION

- Edge capacities. An edge cannot carry more flow than its capacity.

$$f(e) \leq c(e) \quad \forall e \in E$$

- Flow balance. The difference in flow entering and leaving a node v must equal $b(v)$.

$$\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) = b(v) \quad \forall v \in V$$

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

PROBLEM DEFINITION
FLOW BALANCE

- If b(v) for all the v is equal to zero => max-flow
- If b(v) > 0 => v is a source vertex (it creates flow)
- If b(v) < 0 => v is a sink vertex (it destroys flow)
- Generalization is possible!
- Also known as mass balance constraints

- Flow balance. The difference in flow entering and leaving a node v must equal $b(v)$.

$$\sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) = b(v) \quad \forall v \in V$$

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

PROBLEM DEFINITION
EDGE CAPACITIES

- The flow must be feasible according to the given capacities
  - Also known as flow bound constraints

- Edge capacities. An edge cannot carry more flow than its capacity.

$$0 <= f(e) \leq c(e) \quad \forall e \in E$$

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

PROBLEM DEFINITION
OBJECTIVE FUNCTION

find a b-flow f that minimizes $\sum_{e \in E} w(e) f(e)$

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

PROBLEM DEFINITION

Let $G = (N, A)$ be a directed network with a *cost* c_{ij} and a *capacity* u_{ij} associated with every arc $(i, j) \in A$. We associate with each node $i \in N$ a number $b(i)$ which indicates its supply or demand depending on whether $b(i) > 0$ or $b(i) < 0$. The minimum cost flow problem can be stated as follows:

$$\text{Minimize} \quad z(x) = \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{9.1a}$$

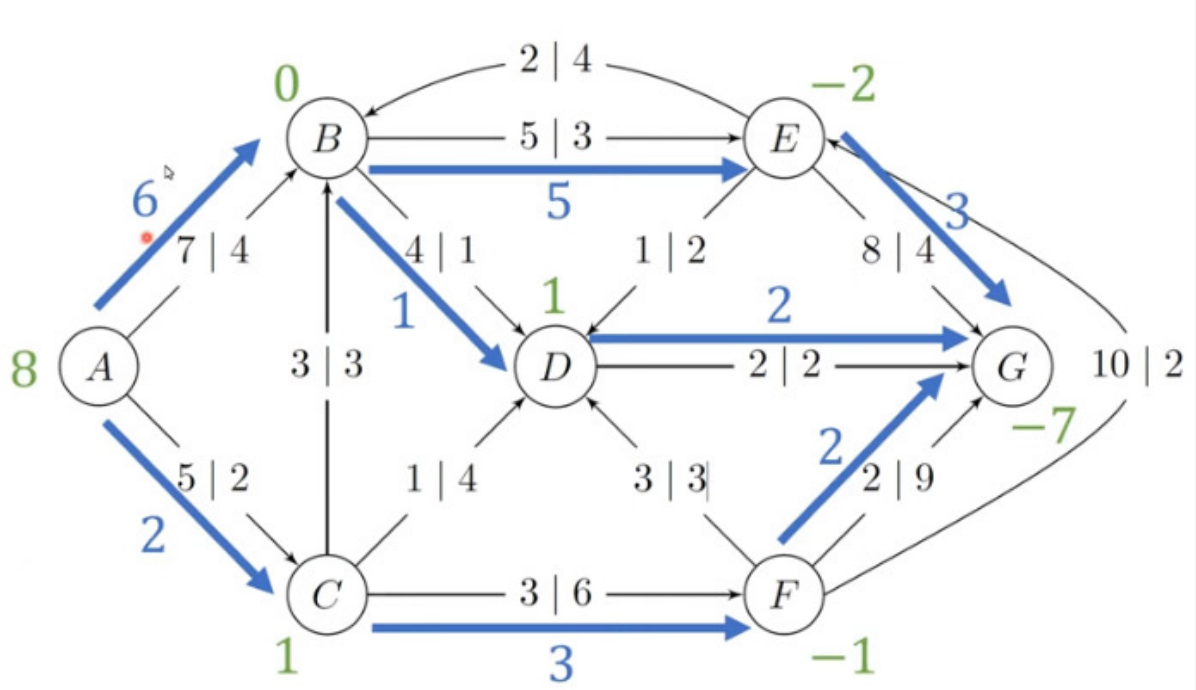
subject to

$$\sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i) \qquad \text{for all } i \in N, \tag{9.1b}$$

$$0 \le x_{ij} \le u_{ij} \qquad \text{for all } (i, j) \in A. \tag{9.1c}$$

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

PROBLEM DEFINITION



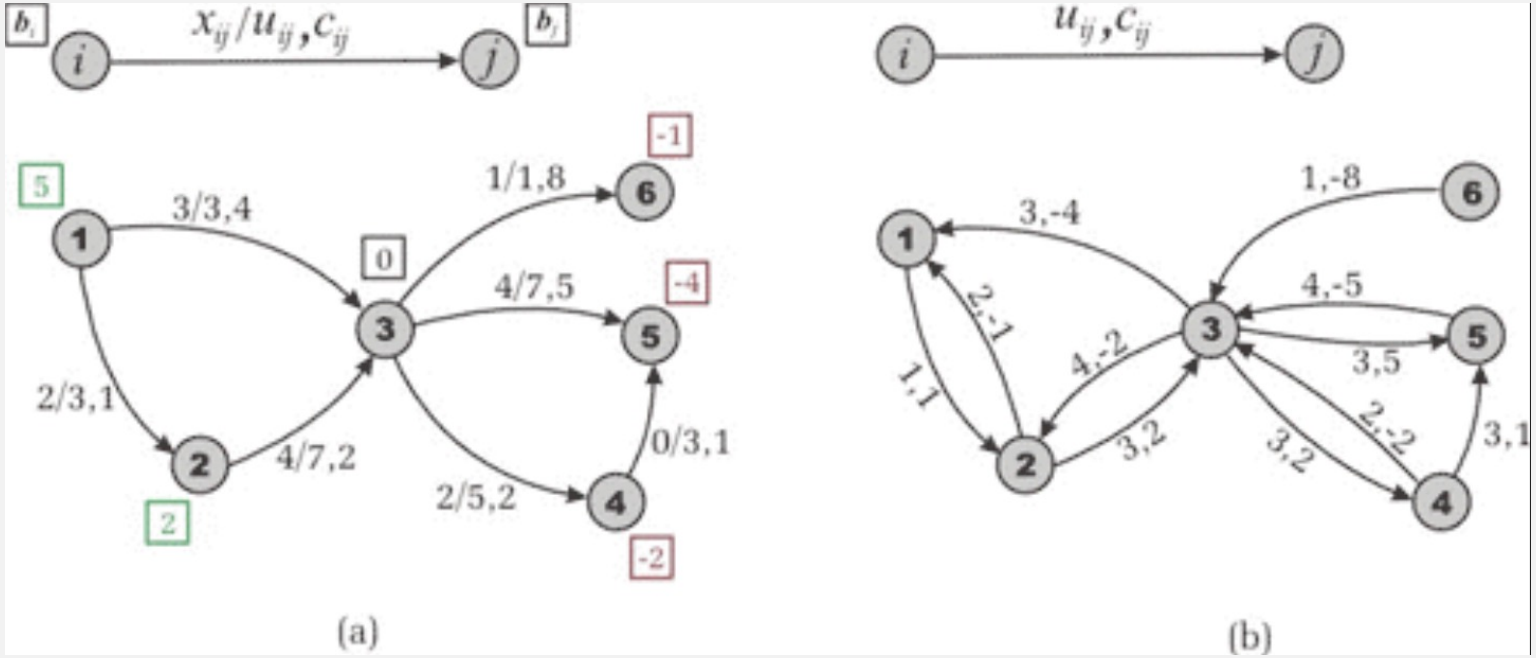
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

ASSUMPTIONS

- Assumption 1 : All data (cost, supply/demand. and capacity) are integral.
- Assumption 2 :The network is directed.
- Assumption 3 : The supplies/demands at the nodes satisfy the condition $\sum b_i = 0$ and the minimum cost flow problem has a feasible solution.
- Assumption 4 : The network G contains an uncapacitated directed path (i.e., each arc in the path has infinite capacity) between every pair of nodes.
- Assumption 5 : All arc costs are nonnegative.

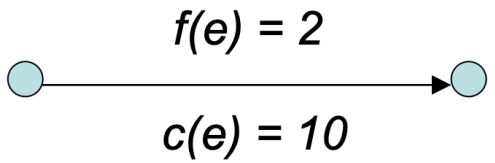
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

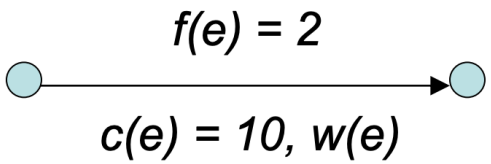
RESIDUAL NETWORK

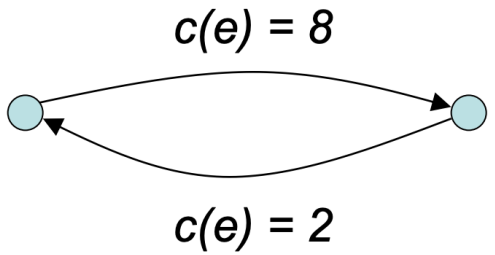


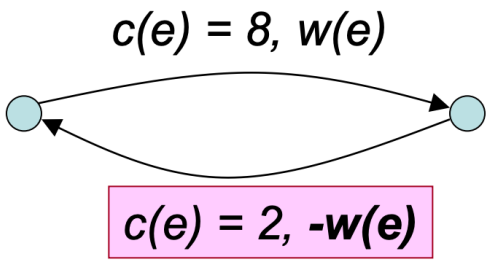
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

RESIDUAL NETWORK









Max Flow

Min-cost Flow

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

MCF MAY NOT BE SOLVABLE

- The given constraints may not be met
- Even if they are all met, there is no guarantee for having a feasible solution
- For example all the edges' capacities might be zero

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

OPTIMALITY CONDITIONS
SHORTEST PATH OPTIMALITY CONDITION

- d(i) : shortest distance of the node i from the node s
- cij : the cost of the edge going from node i to node j

$$d(j) \leq d(i) + c_{ij} \qquad \text{for all } (i, j) \in A.$$

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

OPTIMALITY CONDITIONS
NEGATIVE CYCLE OPTIMALITY CONDITION

- A feasible solution x* is an optimal solution of the minimum cost flow problem if and only if it satisfies the negative cycle optimality conditions: namely, the residual network G(x*) contains no negative cost (directed) cycle.
  - Proof. Suppose that x is a feasible flow and that G(x) contains a negative cycle. Then x cannot be an optimal flow, since by augmenting positive flow along the cycle we can improve the objective function value. Therefore, if x* is an optimal flow, then G(x*) cannot contain a negative cycle. Now suppose that x* is a feasible flow and that G(x*) contains no negative cycle. Let x^ be an optimal flow and x* is not equal to x^. The augmenting cycle property shows that we can decompose the difference vector x^ - x* into at most m augmenting cycles with respect to the flow x* and the sum of the costs of flows on these cycles equals cx^ - cx*. Since the lengths of all the cycles in G(x*) are nonnegative, cx^ - cx*>= 0, or cx^ > = cx*. Moreover, since x^ is an optimal flow, cx^ <= cx*. Thus cx^ = cx*, and x* is also an optimal flow. This argument shows that if G(x*) contains no negative cycle, then x* must be optimal, and this conclusion completes the proof of the theorem.•

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

OPTIMALITY CONDITIONS
REDUCED COST OPTIMALITY CONDITION

- This is exactly the reduced cost that was mentioned in the class.

$$c_{ij}^d = c_{ij} + d(i) - d(j) \geq 0 \qquad \text{for all arcs } (i, j) \in A.$$

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

OPTIMALITY CONDITIONS
REDUCED COST OPTIMALITY CONDITION

- A feasible solution x* is an optimal solution of the minimum cost flow problem if and only if some set of node potentials π satisfy the following reduced cost optimality conditions:

$$c_{ij}^{\pi} \geq 0 \qquad \text{for every arc } (i, j) \text{ in } G(x^*).$$

- Proof. We shall prove this result to show that the negative cycle optimality conditions is equivalent to the reduced cost optimality conditions, suppose that the solution x* satisfies the latter conditions. Therefore, $\sum_{(i,j)\in W} c_{ij}^{\pi} \geq 0$ for every directed cycle w in G(x*). Consequently, $\sum_{(i,j)\in W} c_{ij}^{\pi} = \sum_{(i,j)\in W} c_{ij} \geq 0$, so G(x*) contains no negative cycle.
- To show the converse, assume that for the solution x*, G(x*) contains no negative cycle. Let d(.) denote the shortest path distances from node 1 to all other nodes in G(x*). Recall that if the network contains no negative cycle, the distance labels d(·) are well defined and satisfy the conditions d(j) <= d(i) + cij for all (i, j) in G(x*). We can restate these inequalities as cij - (- d(i)) + (- d(j)) >= 0, or $c_{ij}^{\pi} \geq 0$ if we define π = - d. Consequently, the solution x* satisfies the reduced cost optimality conditions.

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

OPTIMALITY CONDITIONS
SLACKNESS OPTIMALITY CONDITIONS

- A feasible solution x* is an optimal solution of the minimum cost flow problem if and only if for some set of node potentials π, the reduced costs and flow values satisfy the following complementary slackness optimality conditions for every arc (i, j) in A:

$$\text{If } c_{ij}^{\pi} > 0, \text{ then } x_{ij}^{*} = 0.$$

$$\text{If } 0 < x_{ij}^{*} < u_{ij}, \text{ then } c_{ij}^{\pi} = 0.$$

$$\text{If } c_{ij}^{\pi} < 0, \text{ then } x_{ij}^{*} = u_{ij}.$$

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

OPTIMALITY CONDITIONS
SLACKNESS OPTIMALITY CONDITIONS

- Proof. We show that the reduced cost optimality conditions are equivalent to this. To establish this result, we first prove that if the node potentials and the flow vector X satisfy the reduced cost optimality conditions, then they must satisfy these. Consider three possibilities for any arc (i, j) in A .

Case 1. If $c_{ij}^{\pi} > 0$, the residual network cannot contain the arc (j, i) because $c_{ji}^{\pi} = -c_{ij}^{\pi} < 0$ for that arc, contradicting (9.7). Therefore, $x_{ij}^{*} = 0$.

Case 2. If $0 < x_{ij}^{*} < u_{ij}$, the residual network contains both the arcs (i, j) and (j, i). The reduced cost optimality conditions imply that $c_{ij}^{\pi} \geq 0$ and $c_{ji}^{\pi} \geq 0$. But since $c_{ji}^{\pi} = -c_{ij}^{\pi}$, these inequalities imply that $c_{ij}^{\pi} = c_{ji}^{\pi} = 0$.

Case 3. If $c_{ij}^{\pi} < 0$, the residual network cannot contain the arc (i, j) because $c_{ij}^{\pi} < 0$ for that arc, contradicting (9.7). Therefore, $x_{ij}^{*} = u_{ij}$.

- We have thus shown that if the node potentials and the flow vector X satisfy the reduced cost optimality conditions, they also satisfy the complementary slackness optimality conditions.

CYCLE CANCELING

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

CYCLE CANCELING

- First and simplest solution to the problem
- This algorithm maintains a feasible solution and at every iteration attempts to improve its objective function value.
- first establishes a feasible flow x in the network by solving a maximum flow problem
- Then it iteratively finds negative cost-directed cycles in the residual network and augments flows on these cycles.
- The algorithm terminates when the residual network contains no negative cost-directed cycle.
- This termination condition is correct due to the Negative Cycle Optimality Condition

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

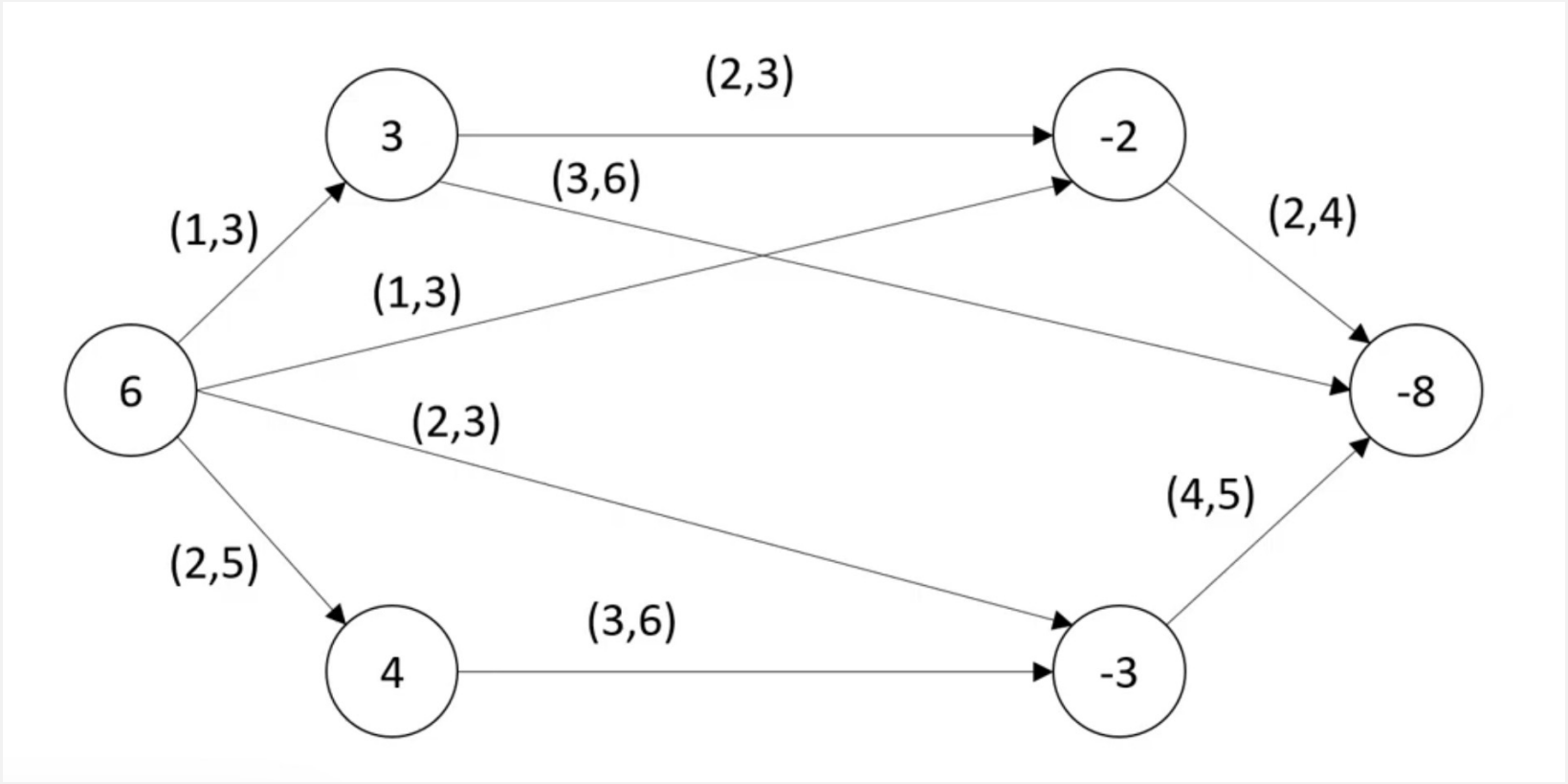
CYCLE CANCELING

```
algorithm cycle-canceling;
begin
    establish a feasible flow x in the network;
    while G(x) contains a negative cycle do
    begin
        use some algorithm to identify a negative cycle W;
        δ := min{r_ij : (i, j) ∈ W};
        augment δ units of flow in the cycle W and update G(x);
    end;
end;
```

By adding a dummy source and sink (in a way that is explained later) and using the ford-Fulkerson algorithm on the new graph

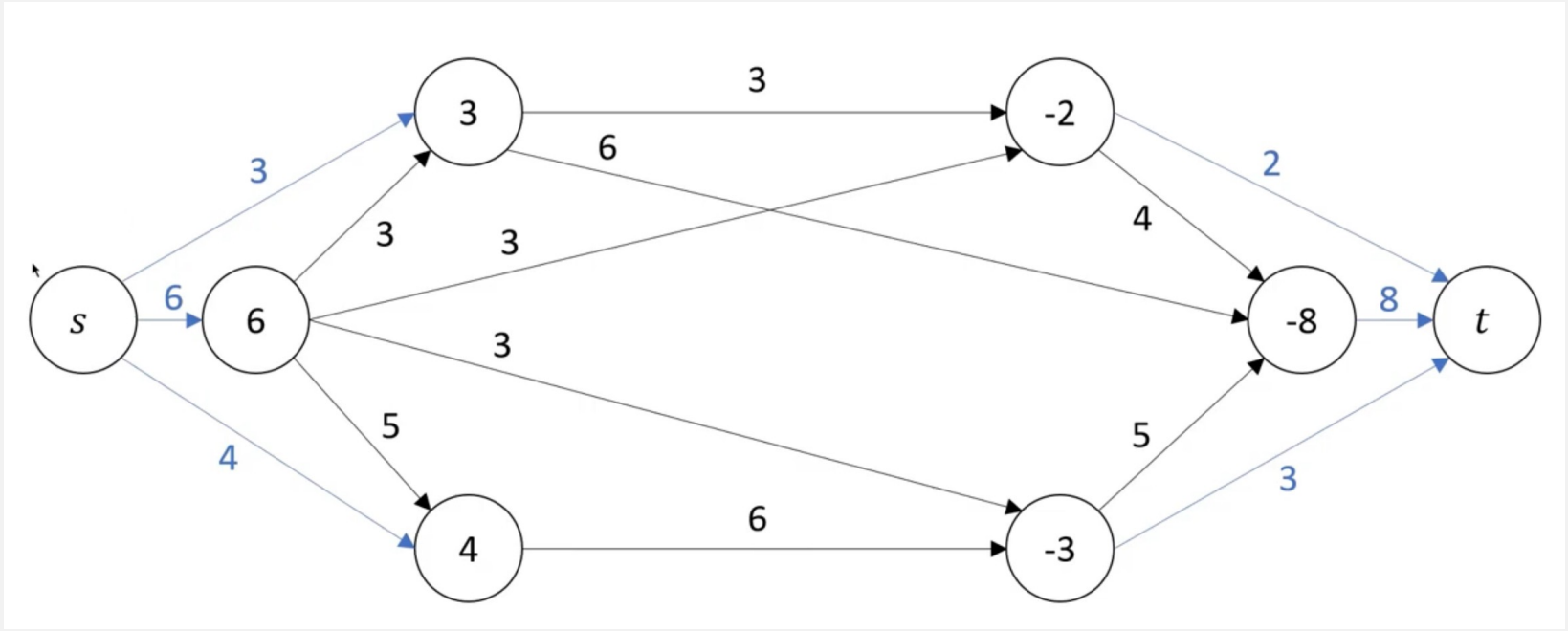
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

CYCLE CANCELING EXAMPLE THE GIVEN GRAPH



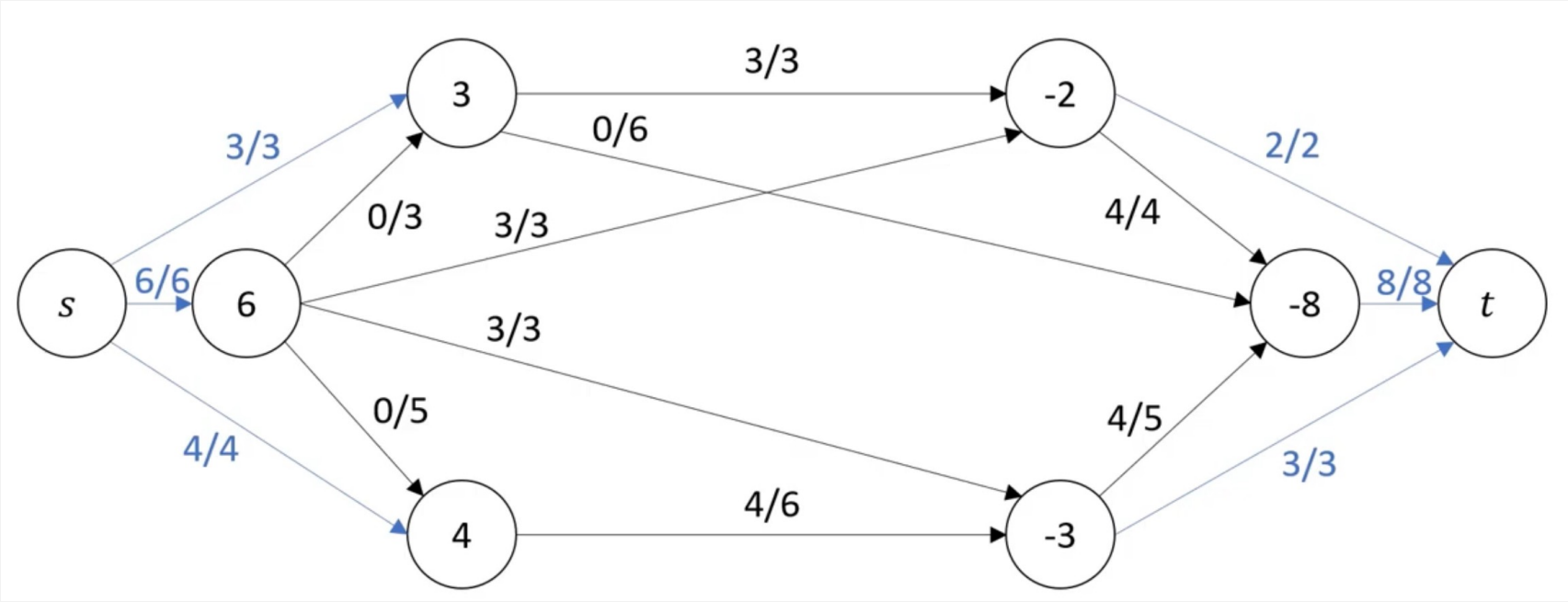
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

CYCLE CANCELING
EXAMPLE
FINDING A FEASIBLE FLOW USING FF



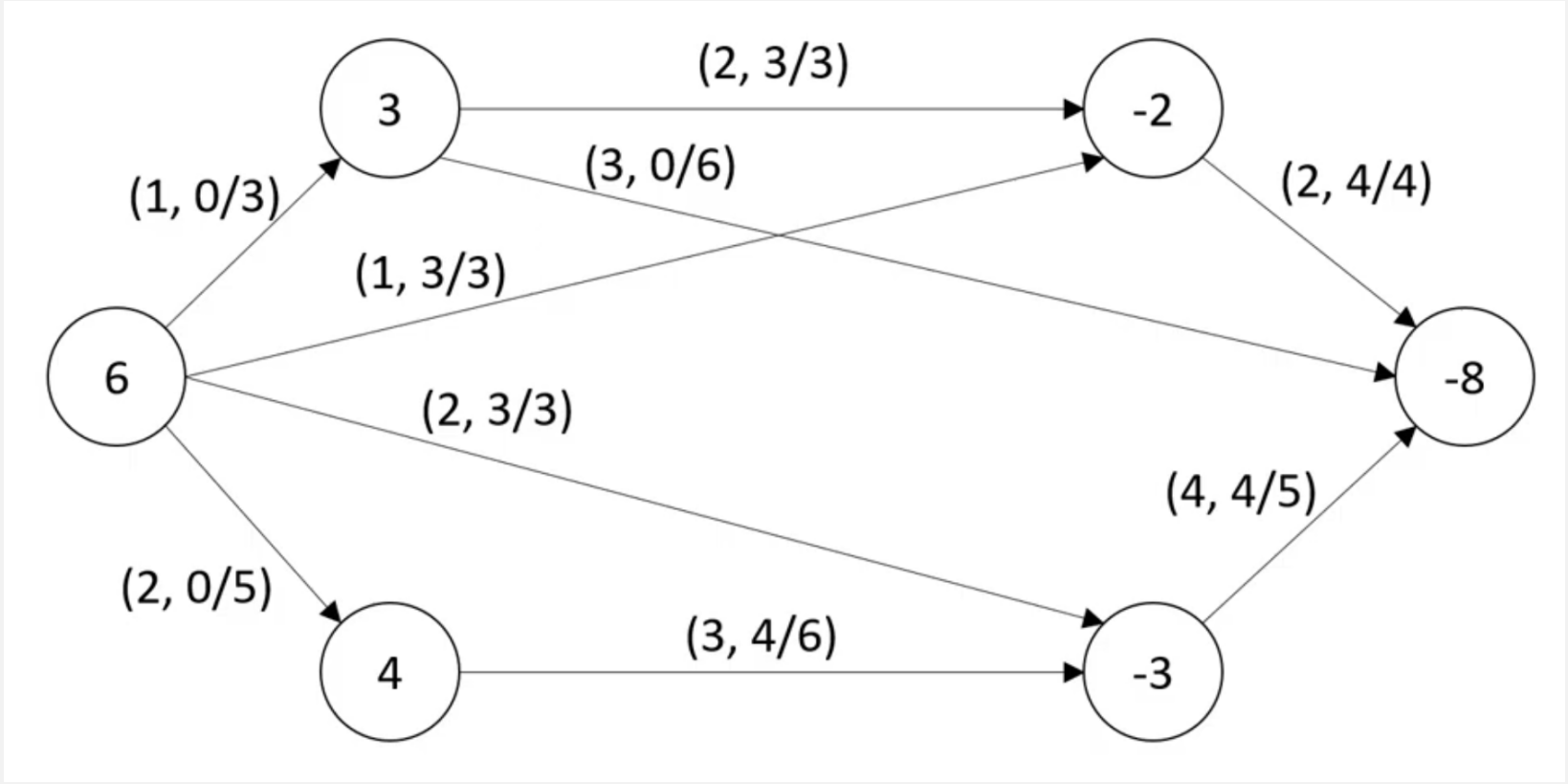
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

CYCLE CANCELING
EXAMPLE
FINDING A FEASIBLE FLOW USING FF



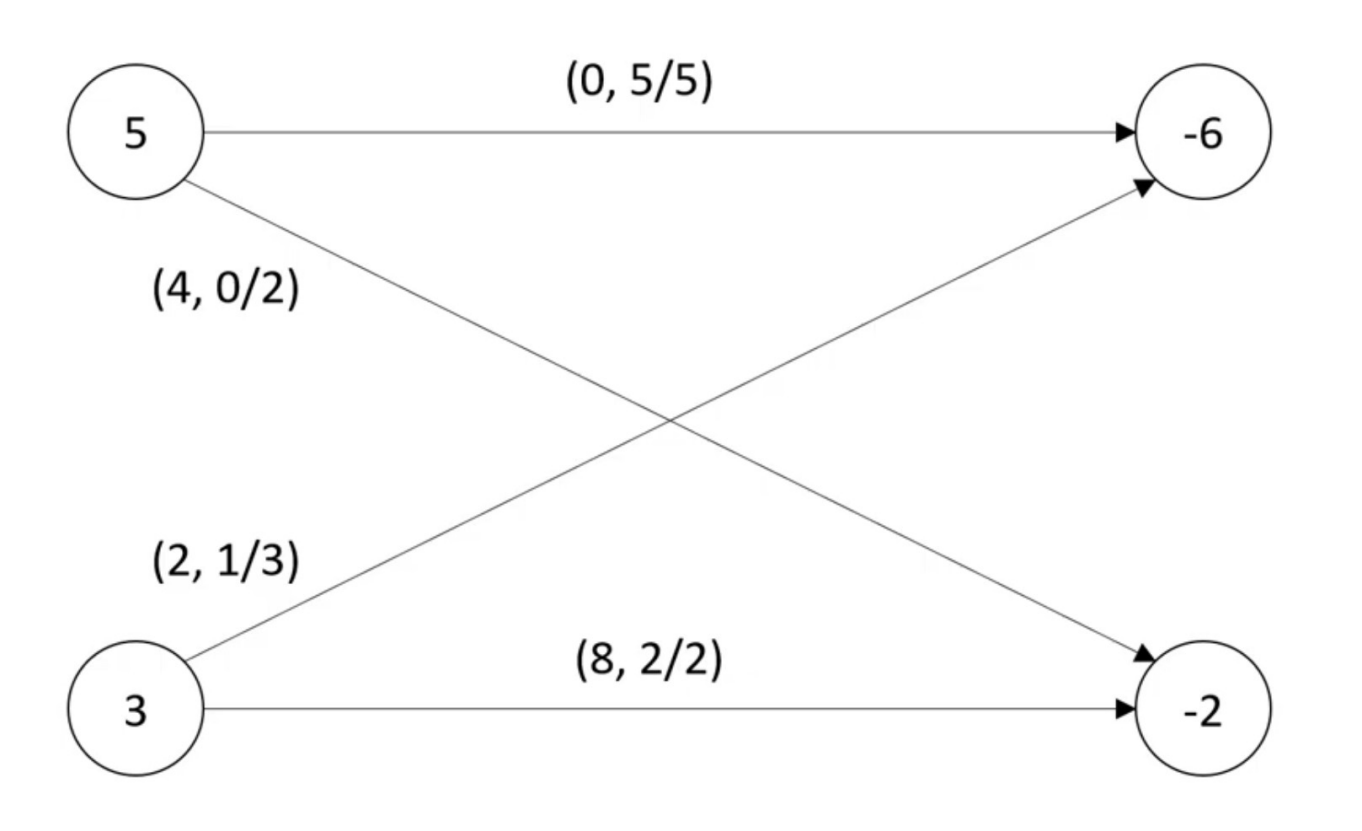
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

CYCLE CANCELING
EXAMPLE
GOING BACK TO THE MAIN GRAPH



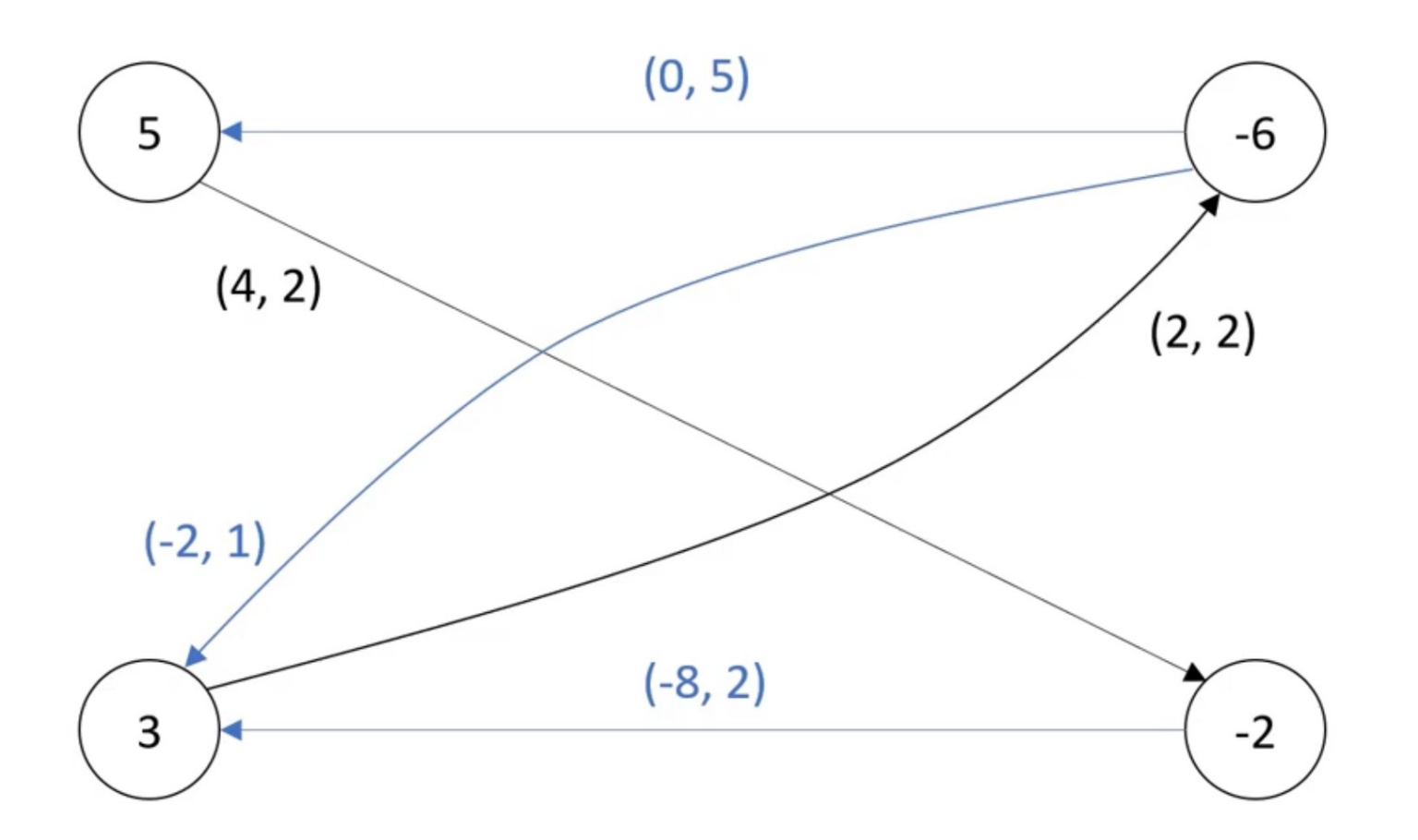
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

CYCLE CANCELING
EXAMPLE
FINDING A NEGATIVE CYCLE



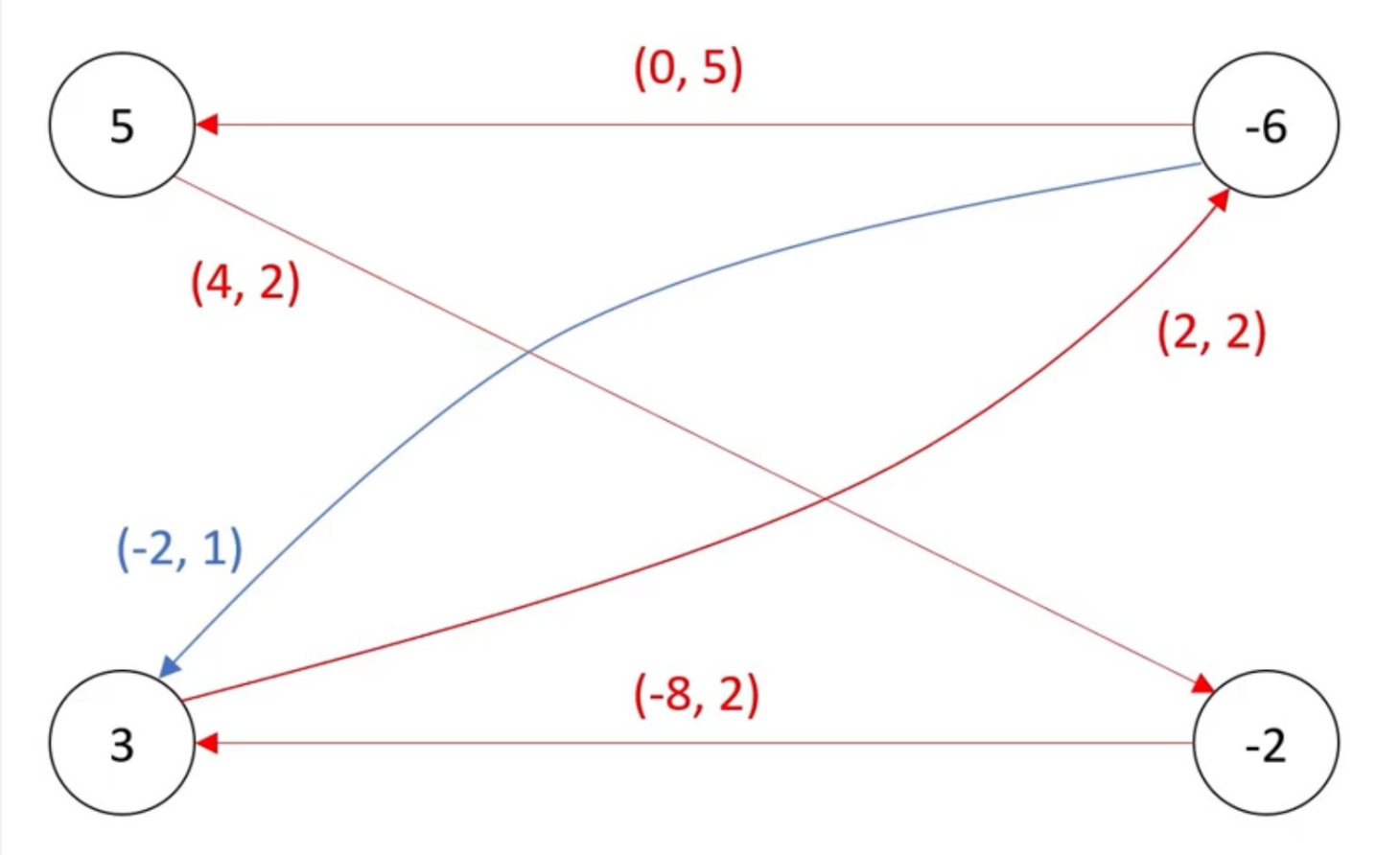
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

CYCLE CANCELING
EXAMPLE
FINDING A NEGATIVE CYCLE (RESIDUAL)



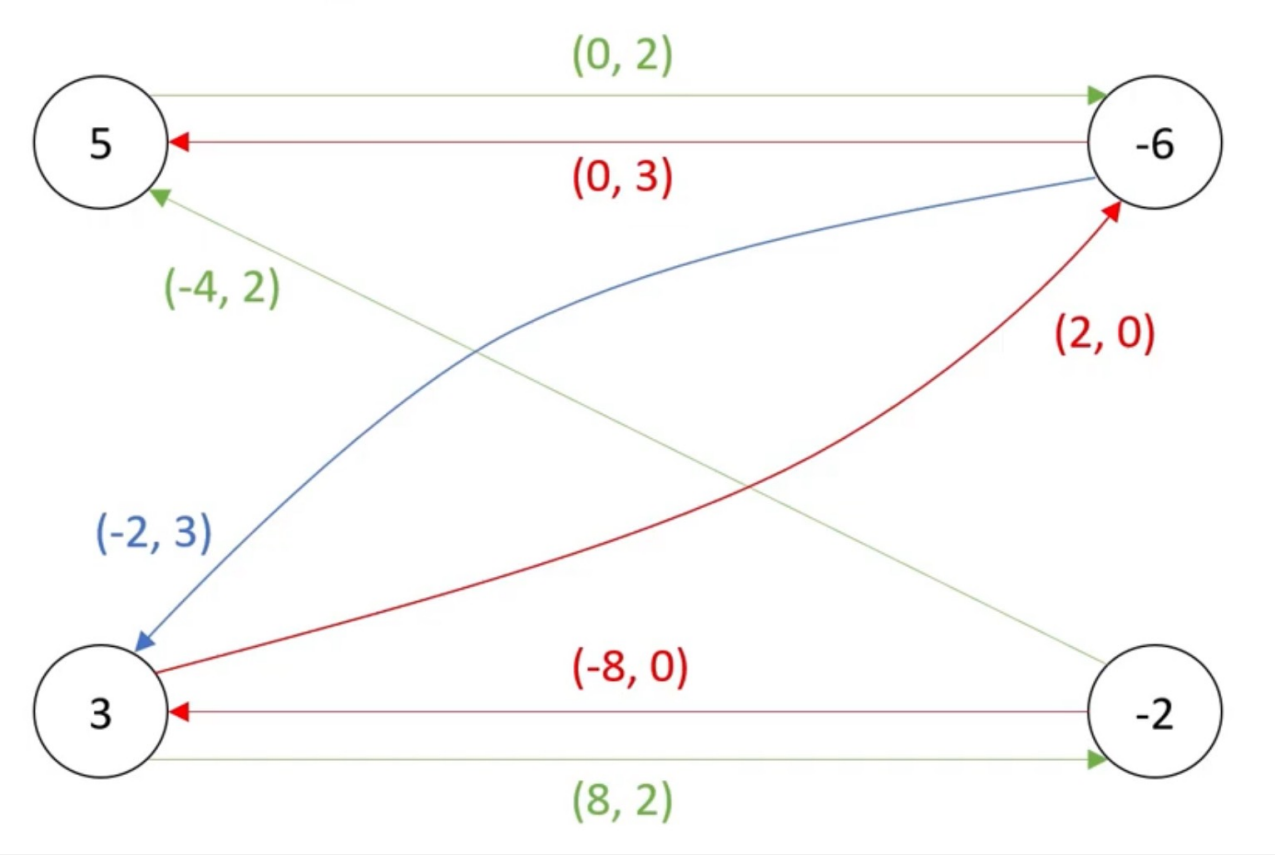
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

CYCLE CANCELING
EXAMPLE
FINDING A NEGATIVE CYCLE (RESIDUAL)



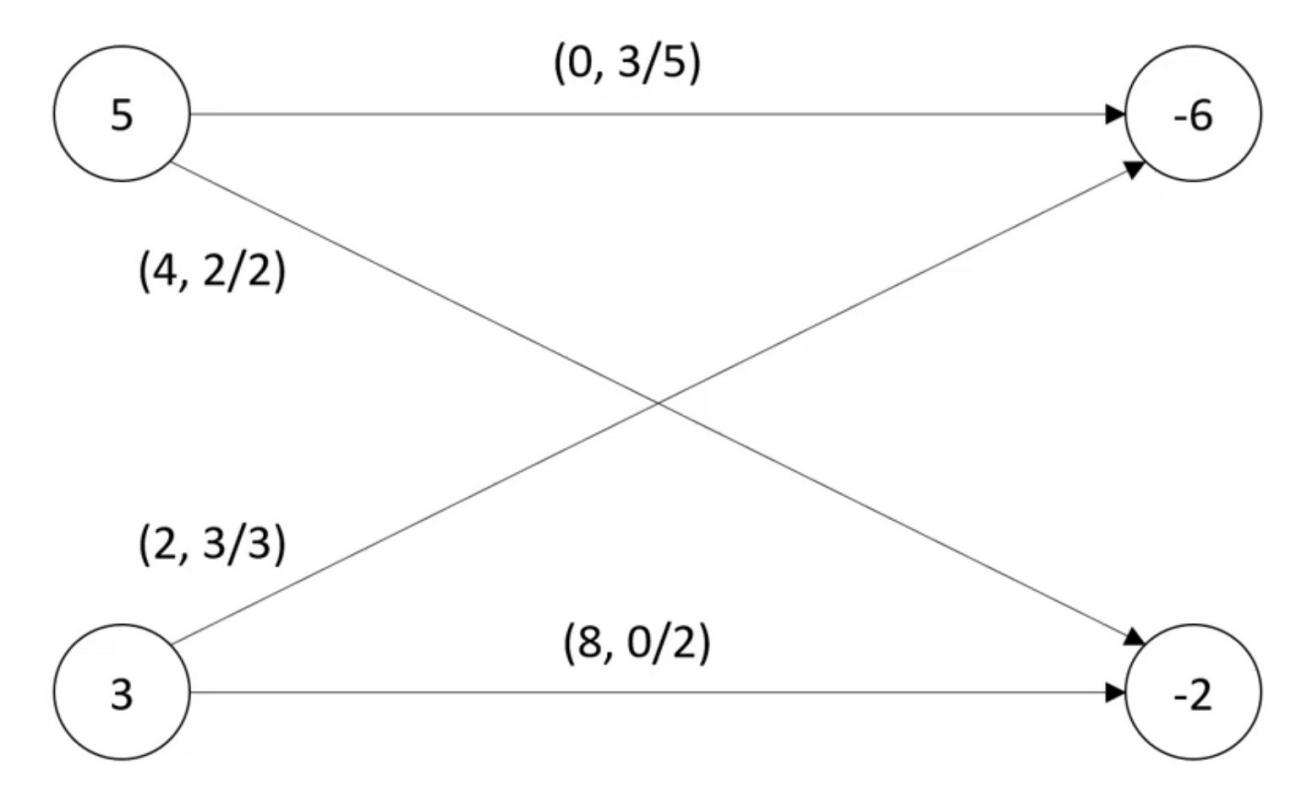
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

CYCLE CANCELING EXAMPLE AUGMENTING $\delta\ (=2)$ UNITS OF FLOW



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

CYCLE CANCELING
EXAMPLE
UPDATING THE MAIN GRAPH



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

CYCLE CANCELING SUMMERY

- **To check feasibility:**

  First check that supplies and demands are balanced by summing b(i).

  Then, convert to Max Flow to further ensure that the problem is feasible.

- **To solve:**

  Find initial feasible flow using the Max Flow solution found above.

  Repeatedly:

  Build a residual graph.

  Check for negative cost cycles. If there is such a cycle C then let δ be argmin{c(e), e in C}, increase flow on forward arcs in the cycle by δ and decrease flow on backward arcs in the cycle by δ.

  If there are no more negative cost cycles, terminate the algorithm.

INTEGRALITY PROPERTY

- If all arc capacities and supplies/demands of nodes are integer, the minimum cost flow problem always has an integer minimum cost flow.
- Proof. We show this result by performing induction on the number of iterations. The algorithm first establishes a feasible flow in the network by solving a maximum flow problem. By Theorem 6.5 the problem has an integer feasible flow and we assume that the maximum flow algorithm finds an integer solution since all arc capacities in the network are integer and the initial residual capacities are also integer. The flow augmented by the cycle-canceling algorithm in any iteration equals the minimum residual capacity in the cycle canceled, which by the inductive hypothesis is integer. Therefore the modified residual capacities in the next iteration will again be integer. This conclusion implies the assertion of the theorem.

TIME COMPLEXITY

- For the minimum cost flow problem, mCU is an upper bound on the initial flow cost [since cij <= C and xij <= U for all (i, j) in A] and - mCU is a lower bound on the optimal flow cost [since cij >= - C and xij <= U for all (i, j) in A].
- Each iteration of the cycle-canceling algorithm changes the objective function value by an amount $(\sum_{(i,j)\in W} c_{ij})\delta$, which is strictly negative.
- Since we are assuming that all the data of the problem are integral, the algorithm terminates within O(mCU) iterations and runs in O(nm^2 CU) time.

SUCCESSIVE SHORTEST PATH

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

SUCCESSIVE SHORTEST PATH

- The successive shortest path algorithm maintains optimality of the solution (as defined in Reduced Cost Optimality Condition) at every step and strives to attain feasibility.
- It maintains a solution x that satisfies the nonnegativity and capacity constraints, but violates the mass balance constraints of the nodes
- At each step, the algorithm selects a node s with excess supply and a node t with unfulfilled demand and sends flow from s to t along a shortest path in the residual network.
- The algorithm terminates when the current solution satisfies all the mass balance constraints.
- The node potentials play a very important role in this algorithm. Besides using them to prove the correctness of the algorithm, we use them to maintain nonnegative arc lengths so that we can solve the shortest path problem more efficiently.

PSEUDOFLOW

- A pseudoflow is a function x: A => R^+ satisfying only the capacity and nonnegativity constraints; it need not satisfy the mass balance constraints. For any pseudoflow x, we define the imbalance of node i as:

$$e(i) = b(i) + \sum_{\{j:(j,i)\in A\}} x_{ji} - \sum_{\{j:(i,j)\in A\}} x_{ij} \qquad \text{for all } i \in N.$$

- If e(i) > 0 for some node i, we refer to e(i) as the excess of node i; if e(i) < 0, we call -e(i) the node's deficit. We refer to a node i with e(i) = 0 as balanced.
  - Let E and D denote the sets of excess and deficit nodes in the network.

$$\sum_{i\in N} e(i) = \sum_{i\in N} b(i) = 0, \text{ and hence } \sum_{i\in E} e(i) = -\sum_{i\in D} e(i)$$

- Therefore if the network contains an excess node, it must also contain a deficit node.

LEMMA

Lemma 9.11. *Suppose that a pseudoflow (or a flow) x satisfies the reduced cost optimality conditions with respect to some node potentials π. Let the vector d represent the shortest path distances from some node s to all other nodes in the residual network $G(x)$ with c_{ij}^{π} as the length of an arc (i, j). Then the following properties are valid:*

(a) *The pseudoflow x also satisfies the reduced cost optimality conditions with respect to the node potentials $\pi' = \pi - d$.*
(b) *The reduced costs $c_{ij}^{\pi'}$ are zero for all arcs (i, j) in a shortest path from node s to every other node.*

Proof. Since x satisfies the reduced cost optimality conditions with respect to π, $c_{ij}^{\pi} \geq 0$ for every arc (i, j) in $G(x)$. Furthermore, since the vector d represents shortest path distances with c_{ij}^{π} as arc lengths, it satisfies the shortest path optimality conditions, that is,

$$d(j) \leq d(i) + c_{ij}^{\pi} \qquad \text{for all } (i, j) \text{ in } G(x). \tag{9.21}$$

Substituting $c_{ij}^{\pi} = c_{ij} - \pi(i) + \pi(j)$ in (9.21), we obtain $d(j) \leq d(i) + c_{ij} - \pi(i) + \pi(j)$. Alternatively, $c_{ij} - (\pi(i) - d(i)) + (\pi(j) - d(j)) \geq 0$, or $c_{ij}^{\pi'} \geq 0$. This conclusion establishes part (a) of the lemma.

Consider next a shortest path from node s to some node l. For each arc (i, j) in this path, $d(j) = d(i) + c_{ij}^{\pi}$. Substituting $c_{ij}^{\pi} = c_{ij} - \pi(i) + \pi(j)$ in this equation, we obtain $c_{ij}^{\pi'} = 0$. This conclusion establishes part (b) of the lemma. ◆

LEMMA

Lemma 9.12. *Suppose that a pseudoflow (or a flow) x satisfies the reduced cost optimality conditions and we obtain x' from x by sending flow along a shortest path from node s to some other node k; then x' also satisfies the reduced cost optimality conditions.*

Proof. Define the potentials π and π' as in Lemma 9.11. The proof of Lemma 9.11 implies that for every arc (i, j) in the shortest path P from node s to the node k, $c_{ij}^{\pi'} = 0$. Augmenting flow on any such arc might add its reversal (j, i) to the residual network. But since $c_{ij}^{\pi'} = 0$ for each arc $(i, j) \in P$, $c_{ji}^{\pi'} = 0$ and the arc (j, i) also satisfies the reduced cost optimality conditions. These results establish the lemma. ◆

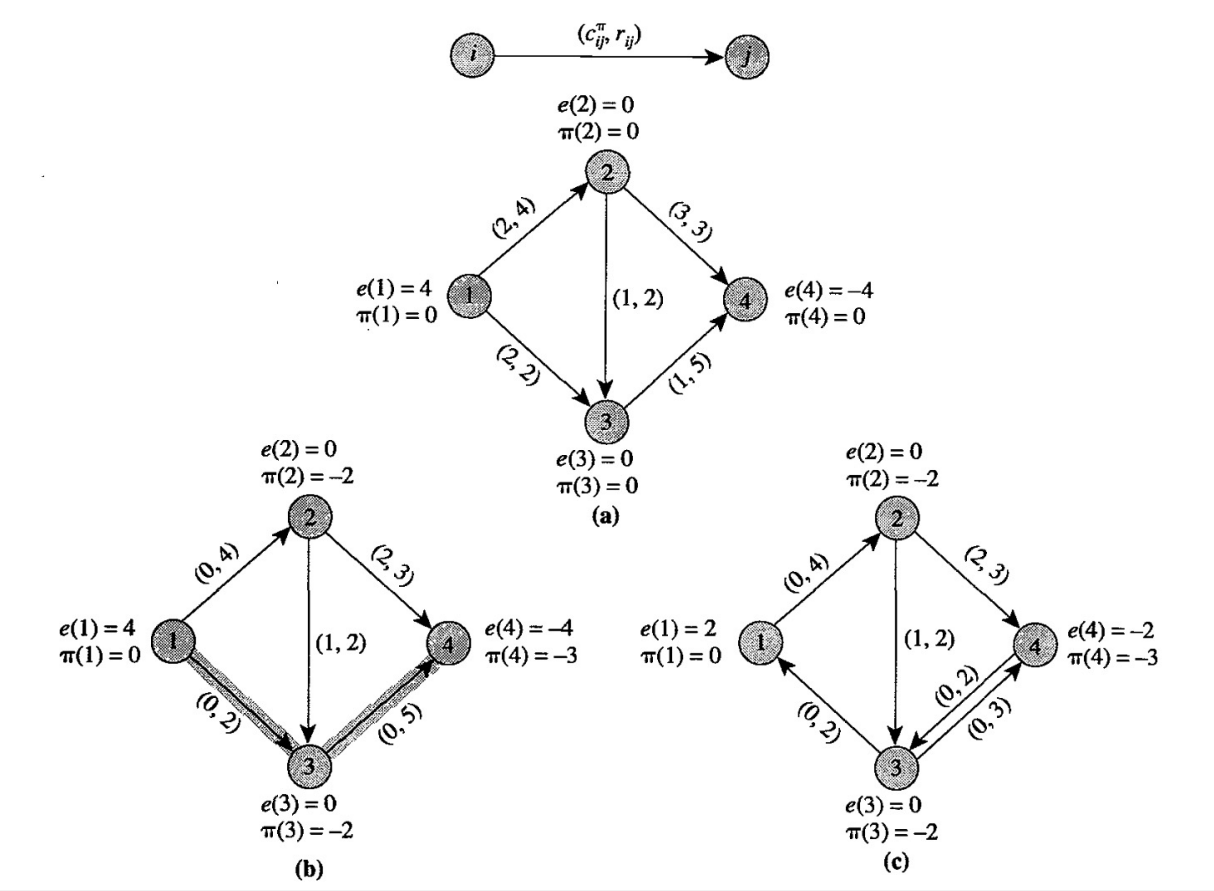
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

PSEUDO CODE

```
algorithm successive shortest path;
begin
    x := 0 and π := 0;
    e(i) := b(i) for all i ∈ N;
    initialize the sets E := {i : e(i) > 0} and D := {i : e(i) < 0};
    while E ≠ ∅ do
    begin
        select a node k ∈ E and a node l ∈ D;
        determine shortest path distances d(j) from node s to all
            other nodes in G(x) with respect to the reduced costs c_ij^π;
        let P denote a shortest path from node k to node l;
        update π := π − d;
        δ := min[e(k), − e(l), min{r_ij : (i, j) ∈ P}];
        augment δ units of flow along the path P;
        update x, G(x), E, D, and the reduced costs;
    end;
end;
```

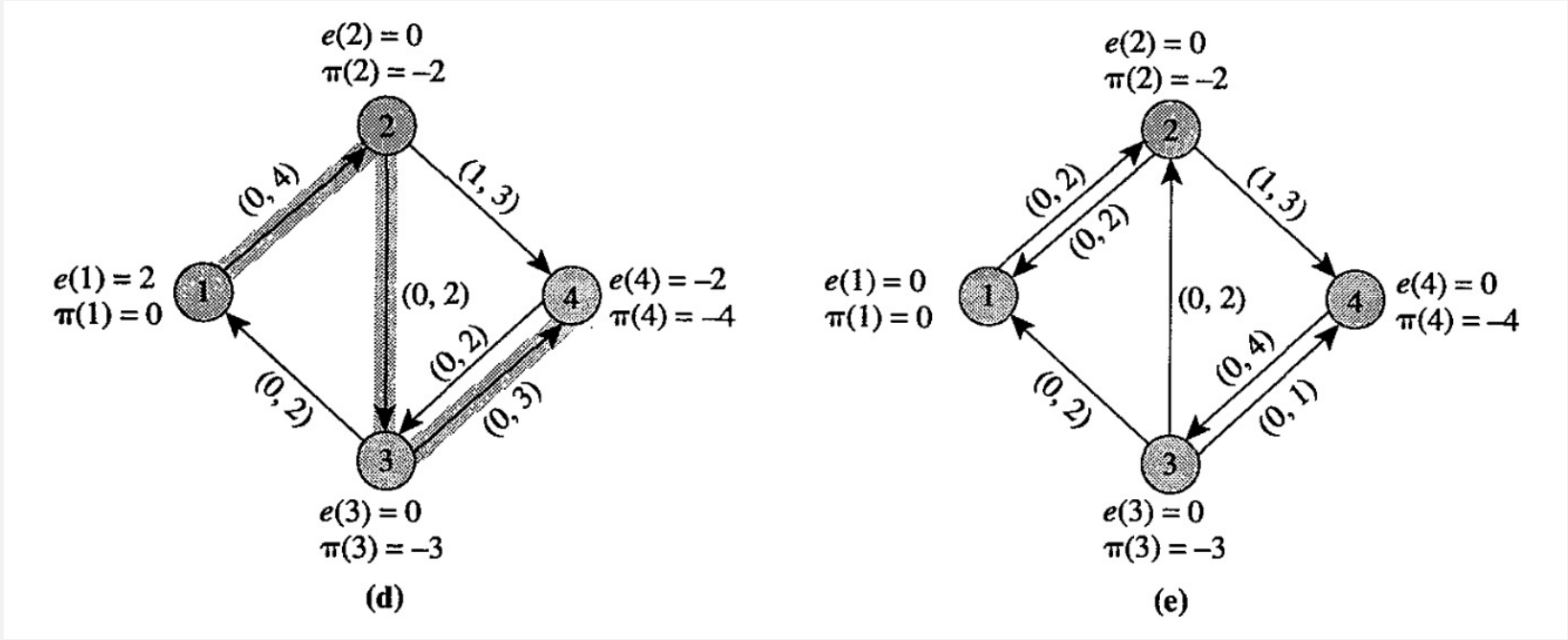
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

EXAMPLE



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

EXAMPLE



CORRECTNESS

- set x = 0, which is a feasible pseudoflow, G(x) = G (with π = 0 satisfies the reduced cost optimality conditions because $c_{ij}^{\pi} = c_{ij} \geq 0$ for every arc (i,j) in the residual network G(x) (recall all arc costs are nonnegative)).
- As long as any node has a nonzero imbalance, both E and D must be nonempty. Thus until all nodes are balanced, the algorithm always succeeds in identifying an excess node k and a deficit node l
- Assumption 4 => residual network contains a directed path from node k to every other node, including node l. => the shortest path distances d(·) are well defined.
- Each iteration of the algorithm solves a shortest path problem with nonnegative arc lengths and strictly decreases the excess of some node (and, also, the deficit of some other node).

COMPLEXITY

- Consequently, if U is an upper bound on the largest supply of any node, the algorithm would terminate in at most nU iterations.
- If S(n, m, C) denotes the time taken to solve a shortest path problem with nonnegative arc lengths, the overall complexity of this algorithm is O(nUS(n, m, nC)). [Note that we have used nC rather than C in this expression, since the costs in the residual network are bounded by nC.]
- The successive shortest path algorithm requires pseudopolynomial time to solve the minimum cost flow problem since it is polynomial in n, m and the largest supply U. This algorithm is, however, polynomial time for the assignment problem, a special case of the minimum cost flow problem, for which U = 1.

CAPACITY SCALING

CAPACITY SCALING

- An inherent drawback of the successive shortest path algorithm is that its augmentations might carry relatively small amounts of flow, resulting in a fairly large number of augmentations in the worst case.
- a scaling technique, guarantees that each augmentation carries sufficiently large flow and thereby reduces the number of augmentations substantially.
- Improve the worst-case algorithmic performance from O(nU (S(N, m, nC)) to O(m log U . S(n, m, nC)).
- The reason that the running time involves S(n, m, nC) rather than S(n, m, C) is that the costs in the residual network are reduced costs, and the reduced cost of an arc could be as large as nC

CAPACITY SCALING

- It is related to the successive shortest path algorithm, just as the capacity scaling algorithm for the maximum flow problem is related to the labeling algorithm.
- Ensures that each shortest path augmentation carries a sufficiently large amount of flow
- This modification to the algorithm reduces the number of successive shortest path iterations from O(nU) to O(m log U)
- Improves on the algorithmic performance
- Illustrates how small changes in an algorithm can produce significant algorithmic improvements (at least in the worst case).

CAPACITY SCALING

- The algorithm maintains a pseudoflow satisfying the reduced cost optimality condition and gradually converts this pseudoflow into a flow by identifying shortest paths from nodes with excesses to nodes with deficits and augmenting flows along these paths.
  - performs a number of scaling phases for different values of a parameter Δ
  - refer to a scaling phase with a specific value of Δ as the Δ -scaling phase
    - Initially, $\Delta = 2^{\lfloor logU \rfloor}$
- The algorithm ensures that in the Δ -scaling phase each augmentation carries exactly Δ units of flow.
  - When it is not possible to do so because no node has an excess of at least Δ, or no node has a deficit of at least Δ, the algorithm reduces the value of Δ by a factor of 2 and repeats the process.
    - Eventually, $\Delta = 1$ and at the end of this scaling phase, the solution becomes a flow
  - This flow must be an optimal flow because it satisfies the reduced cost optimality condition.

CAPACITY SCALING

- In the Δ -scaling phase, each augmentation must start at a node in S(Δ) and end at a node in T(Δ)
- the augmentation take place on a path along which every arc has residual capacity of at least Δ
- Δ -residual network G(x, Δ) is defined as the subgraph of G(x) consisting of those arcs whose residual capacity is at least Δ
- In the Δ -scaling phase, the algorithm augments flow from a node in S(Δ) to a node in T(Δ) along a shortest path in G(x, Δ)

$$S(\Delta) = \{i : e(i) \geq \Delta\},$$
$$T(\Delta) = \{i : e(i) \leq -\Delta\}.$$

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

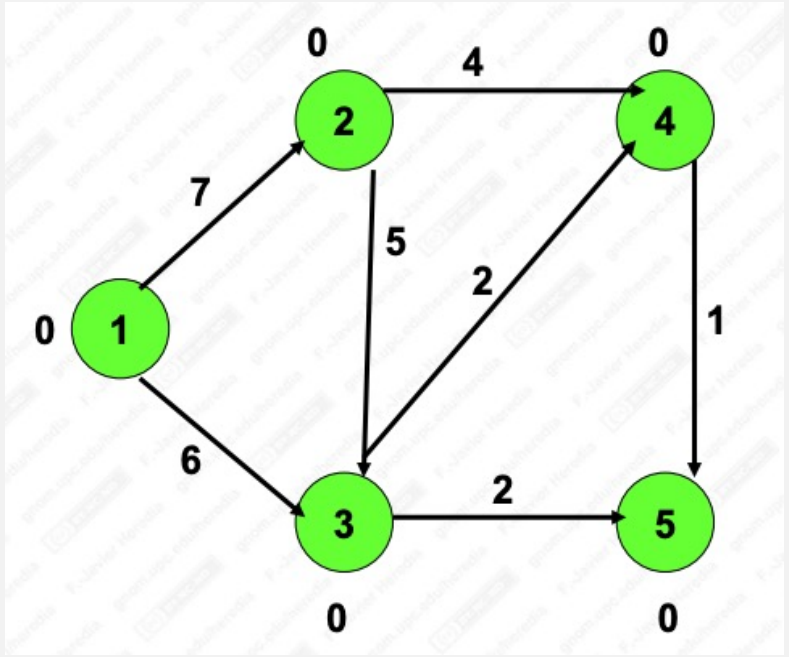
PSEUDO CODE

```
algorithm capacity scaling;
begin
    x := 0 and π := 0;
    Δ := 2^⌊log U⌋;
    while Δ ≥ 1
    begin {Δ-scaling phase}
        for every arc (i, j) in the residual network G(x) do
            if r_ij ≥ Δ and c^π_ij < 0 then send r_ij units of flow along arc (i, j),
              update x and the imbalances e(·);
        S(Δ) := {i ∈ N : e(i) ≥ Δ};
        T(Δ) := {i ∈ N : e(i) ≤ −Δ};
        while S(Δ) ≠ ∅ and T(Δ) ≠ ∅ do
        begin
            select a node k ∈ S(Δ) and a node l ∈ T(Δ);
            determine shortest path distances d(·) from node k to all other nodes in the
                Δ-residual network G(x, Δ) with respect to the reduced costs c^π_ij;
            let P denote shortest path from node k to node l in G(x, Δ);
            update π := π − d;
            augment Δ units of flow along the path P;
            update x, S(Δ), T(Δ), and G(x, Δ);
        end;
        Δ := Δ/2;
    end;
end;
```

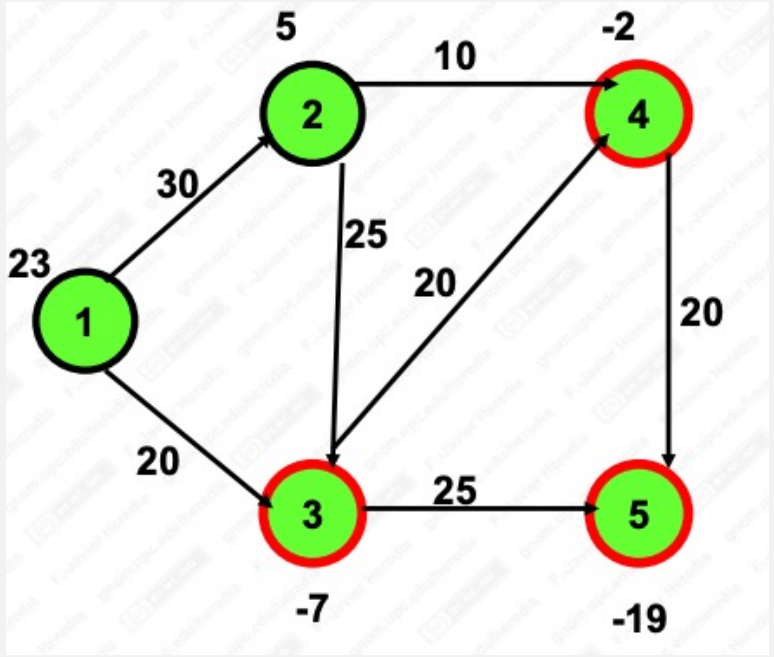
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

EXAMPLE
THE ORIGINAL GRAPH

The original costs and potentials

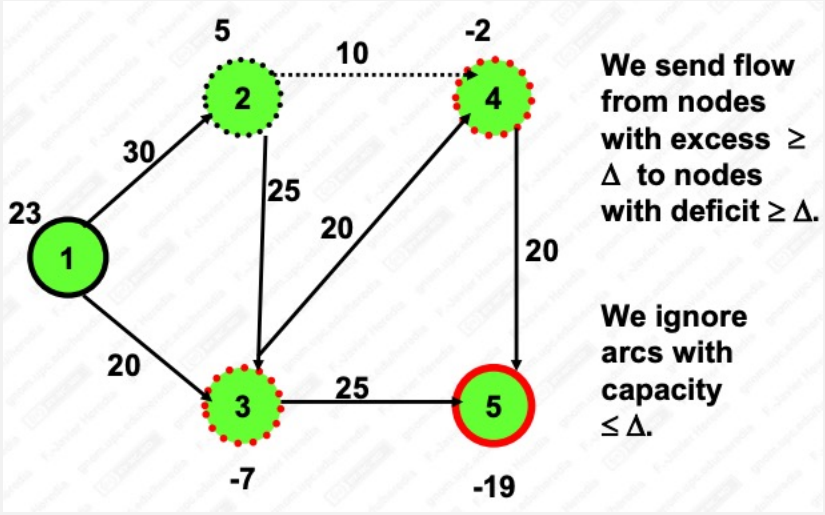


The original capacities and supply/demand



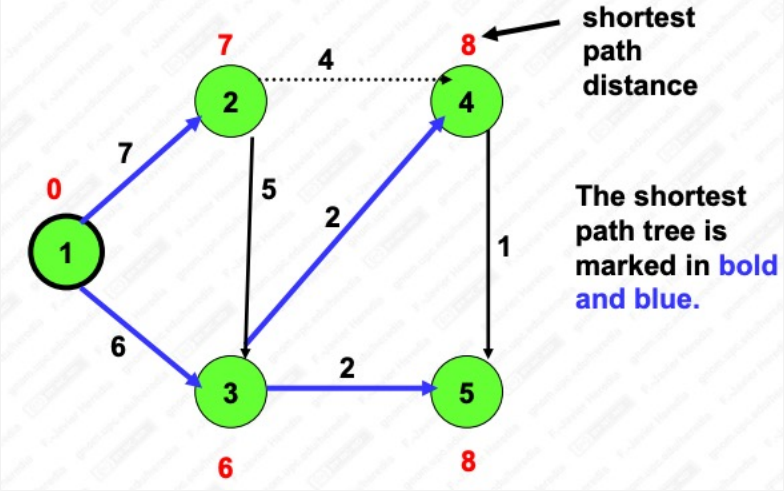
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

EXAMPLE
$\Delta = 16$

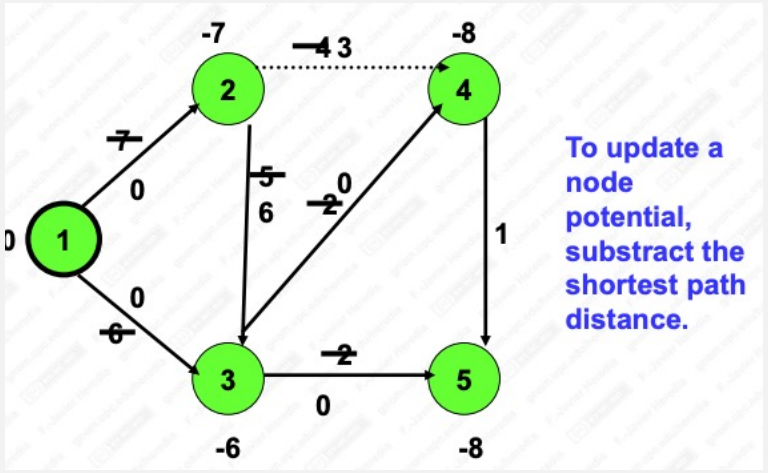


دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

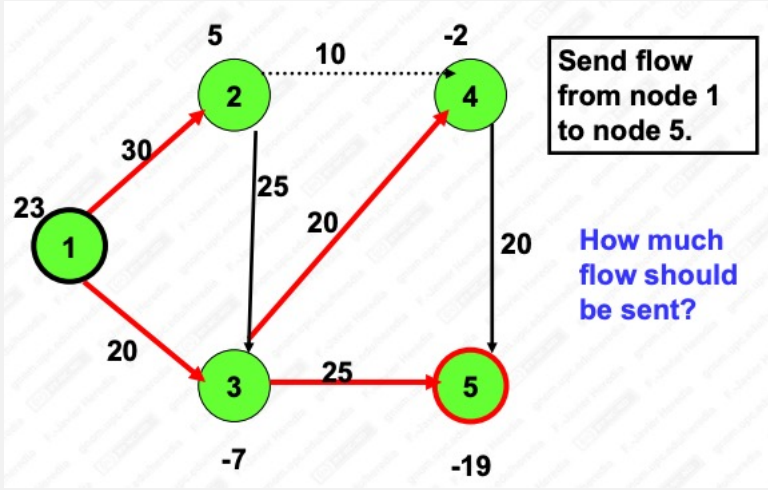
EXAMPLE
SELECT A SUPPLY NODE AND FIND THE SHORTEST PATHS



EXAMPLE
UPDATE THE POTENTIALS AND REDUCED COSTS

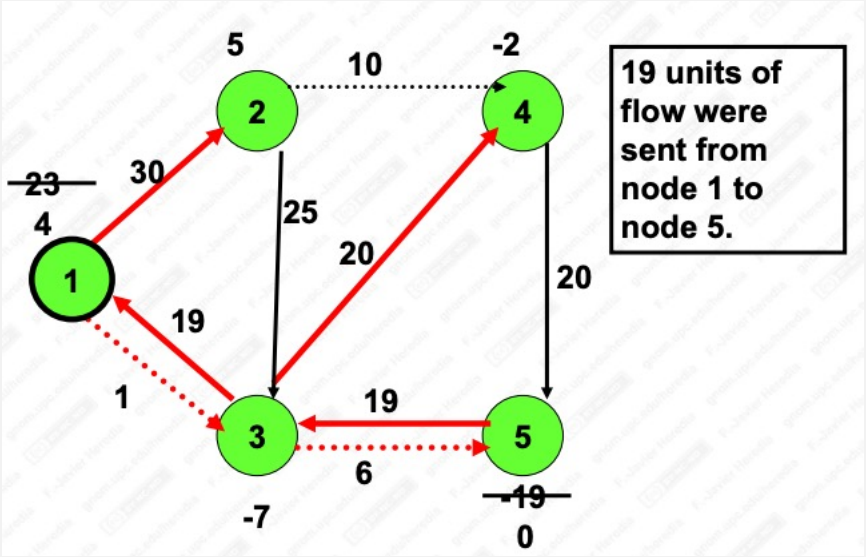


EXAMPLE
SELECT A FLOW ALONG A SHORTEST PATH IN G(X, 16)

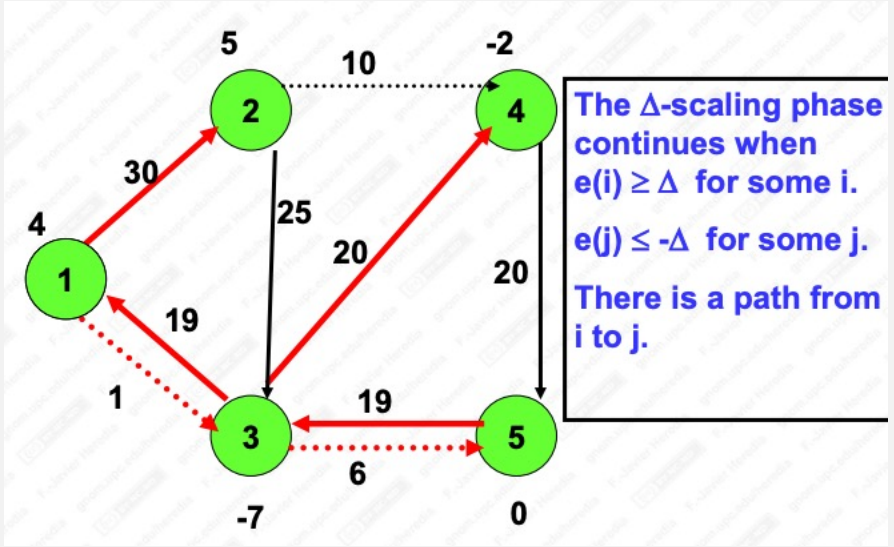


دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

EXAMPLE
UPDATE THE RESIDUAL NETWORK

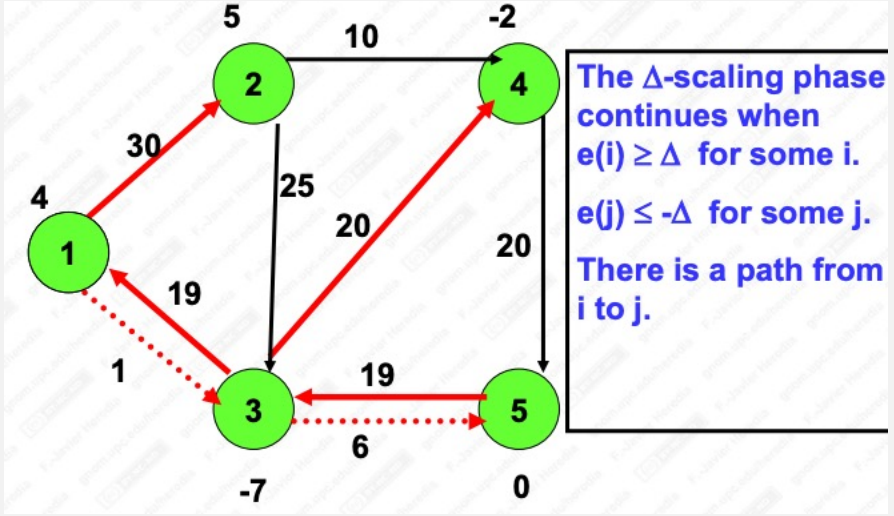


EXAMPLE
ENDS THE 16-SCALING PHASE

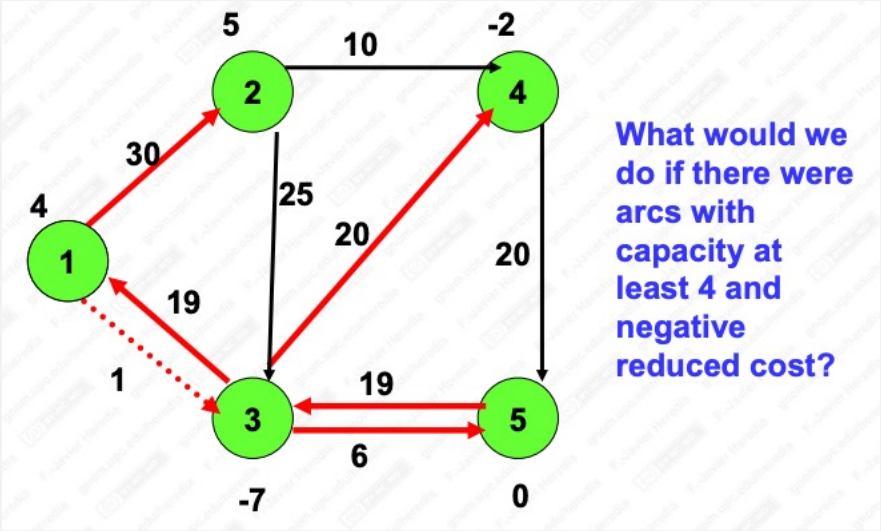


دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

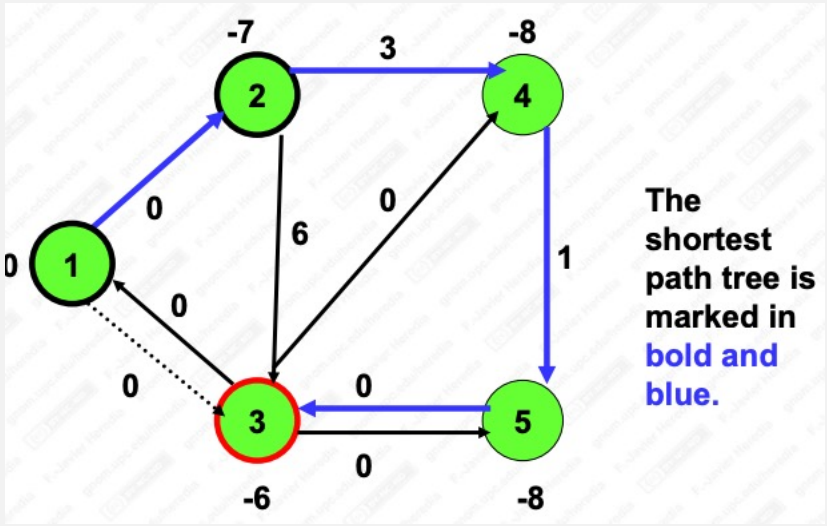
EXAMPLE
BEGINS AND ENDS THE 8-SCALING PHASE



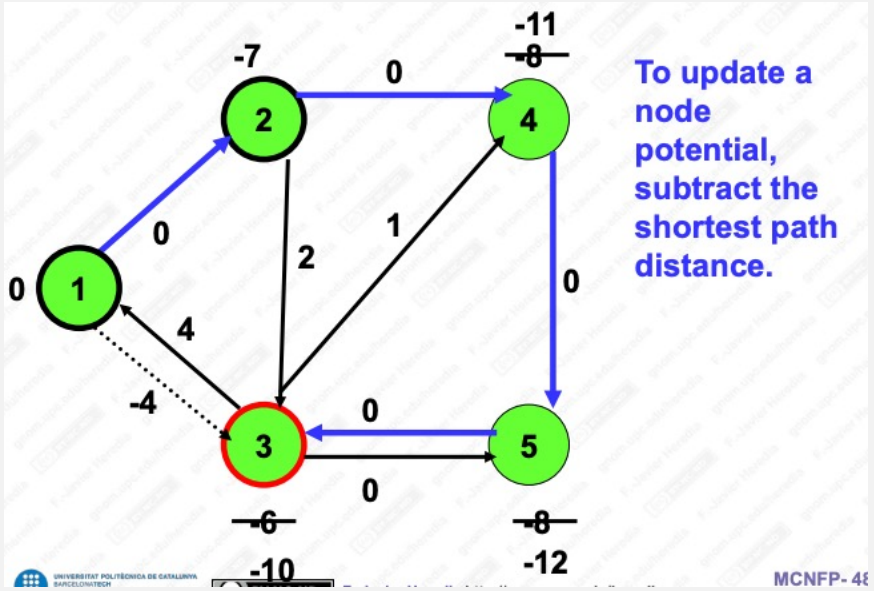
EXAMPLE
BEGINS 4-SCALING PHASE



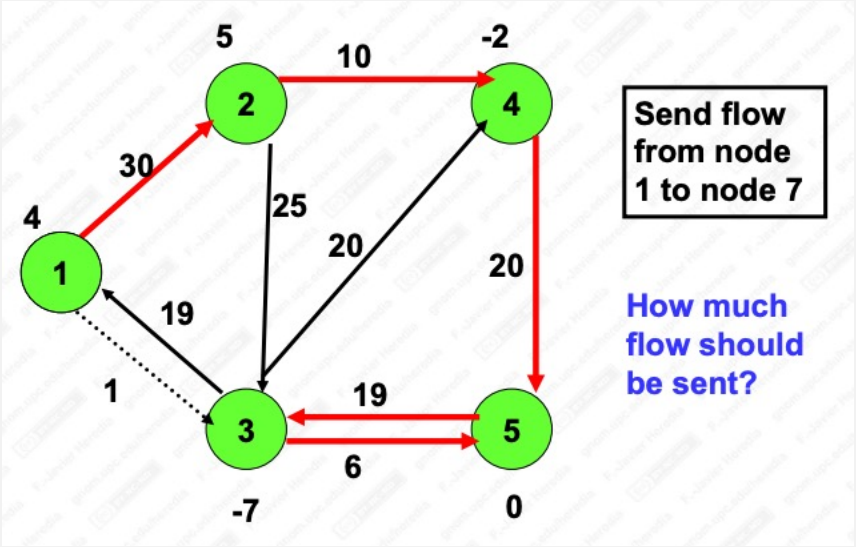
EXAMPLE
SELECT A LARGE EXCESS POINT AND FIND THE SHORTEST PATH



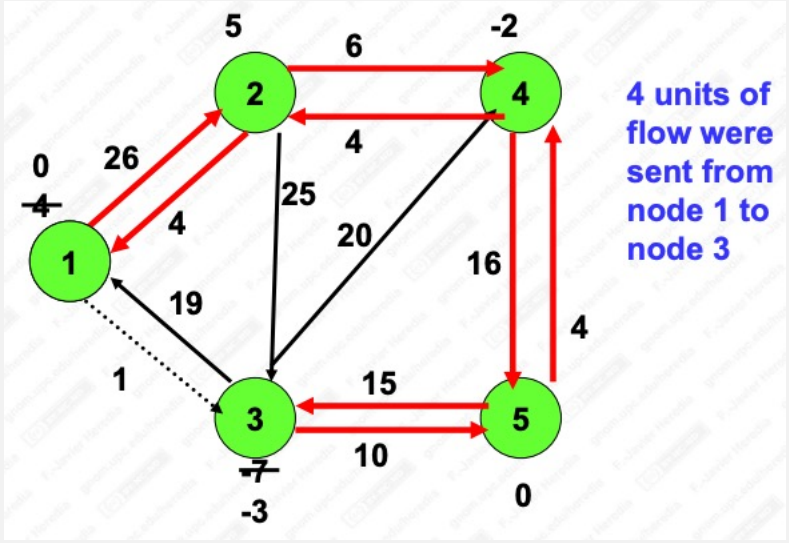
EXAMPLE
UPDATE THE POTENTIALS AND REDUCED COSTS



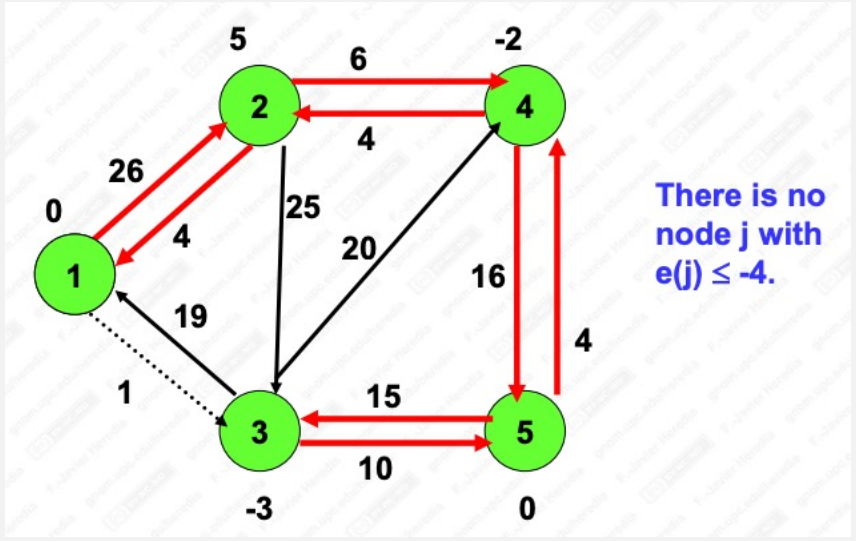
EXAMPLE
SEND FLOW ALONG SHORTEST PATH IN G(X,4)



EXAMPLE
UPDATE THE RESIDUAL NETWORK

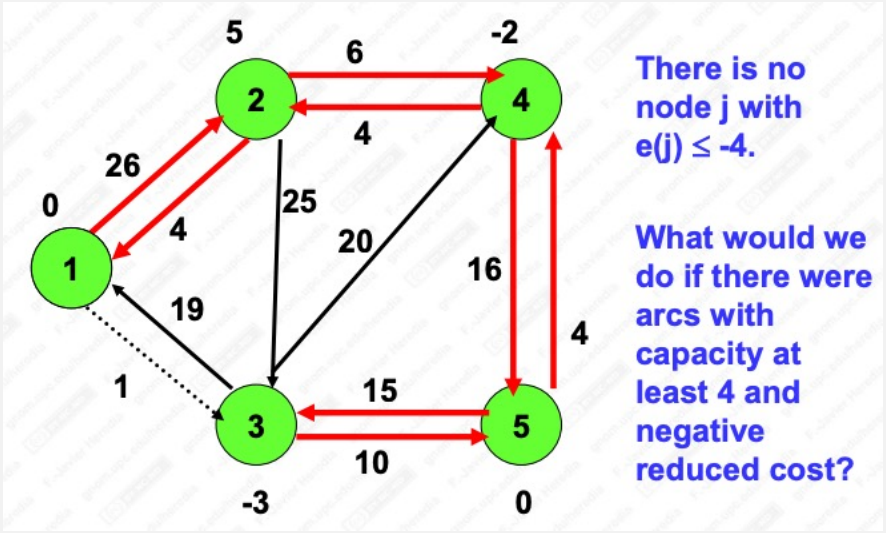


EXAMPLE
THIS ENDS 4-SCALING PHASE

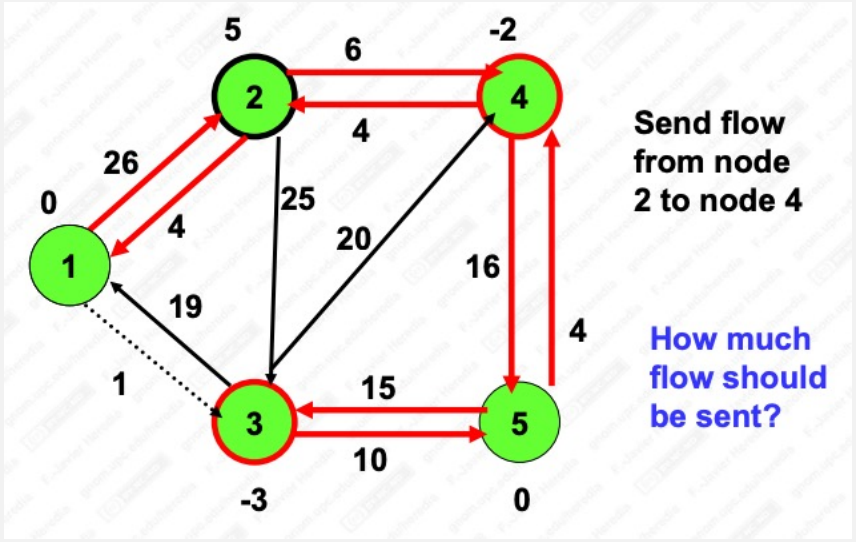


دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

EXAMPLE
BEGIN THE 2-SCALING PHASE

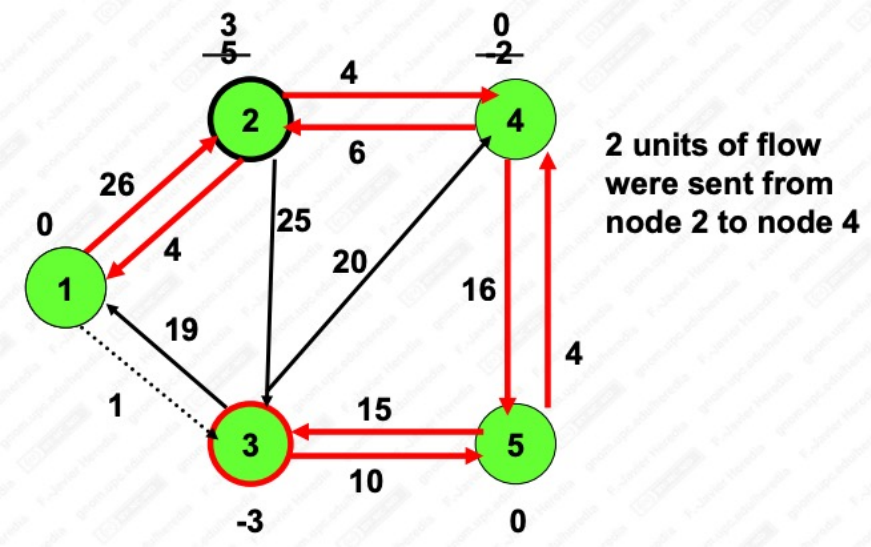


EXAMPLE
SEND FLOW ALONG A SHORTEST PATH



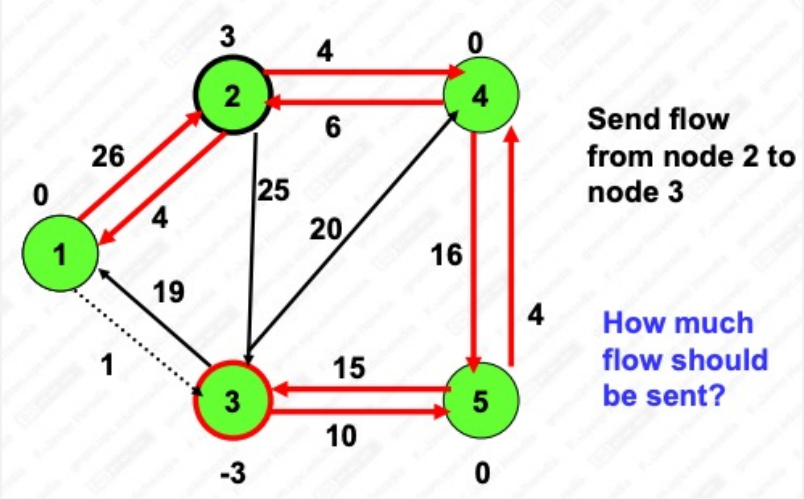
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

EXAMPLE
UPDATE THE RESIDUAL NETWORK

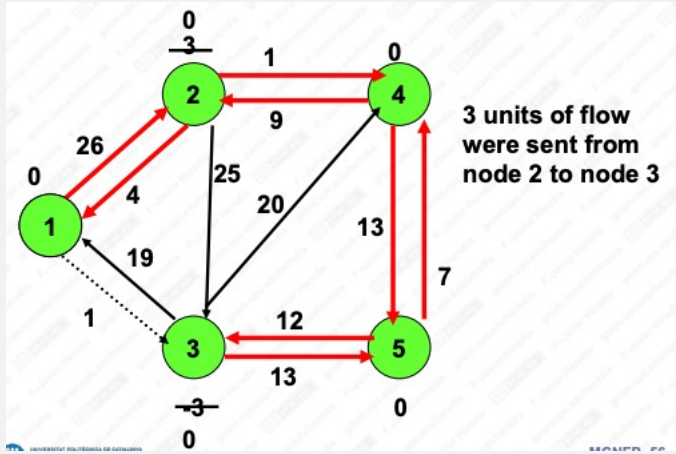


دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

EXAMPLE
SEND FLOW ALONG A SHORTEST PATH

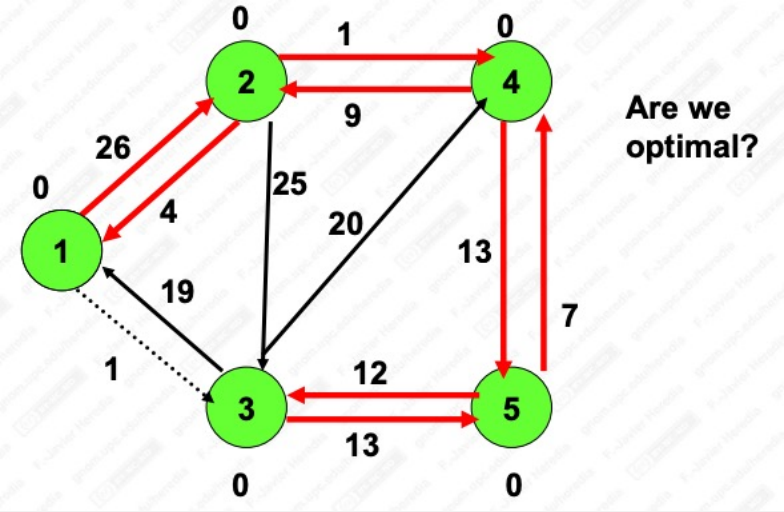


EXAMPLE
UPDATE THE RESIDUAL NETWORK

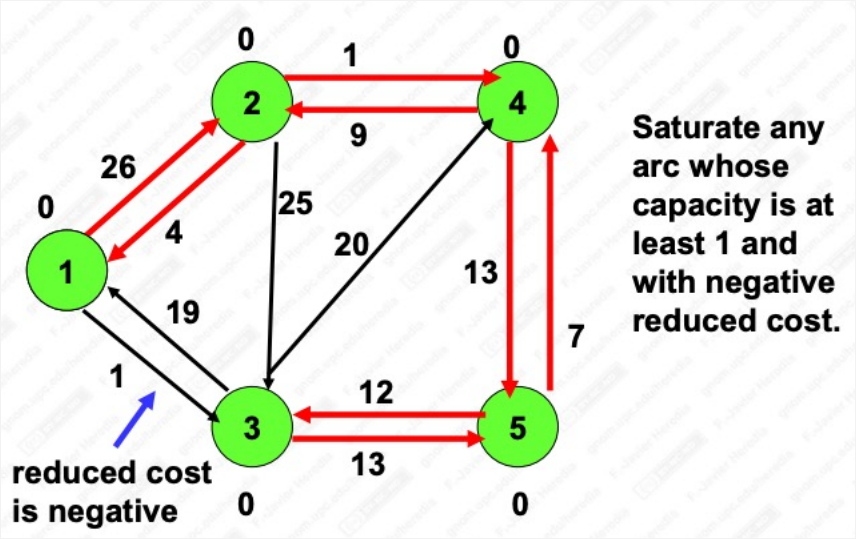


دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

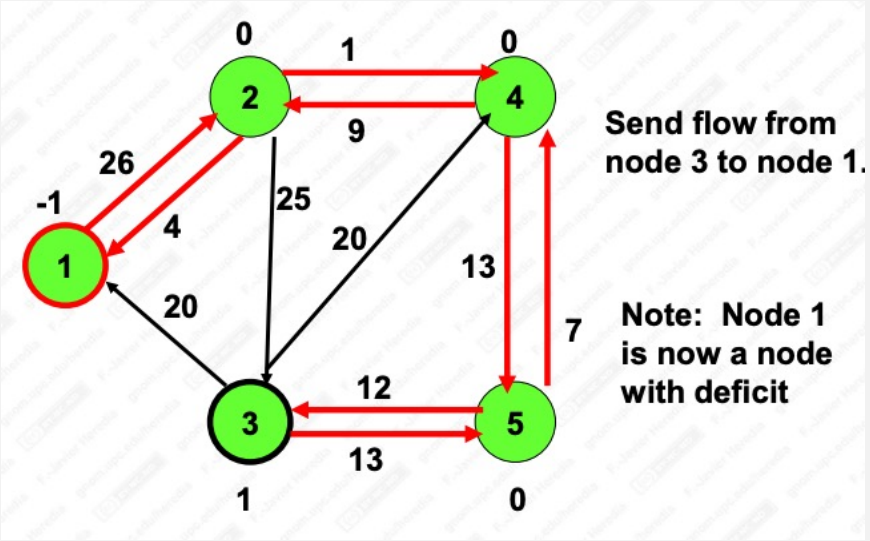
EXAMPLE
THIS ENDS THE 2-SCALING PHASE



EXAMPLE
BEGIN THE 1-SCALING PHASE

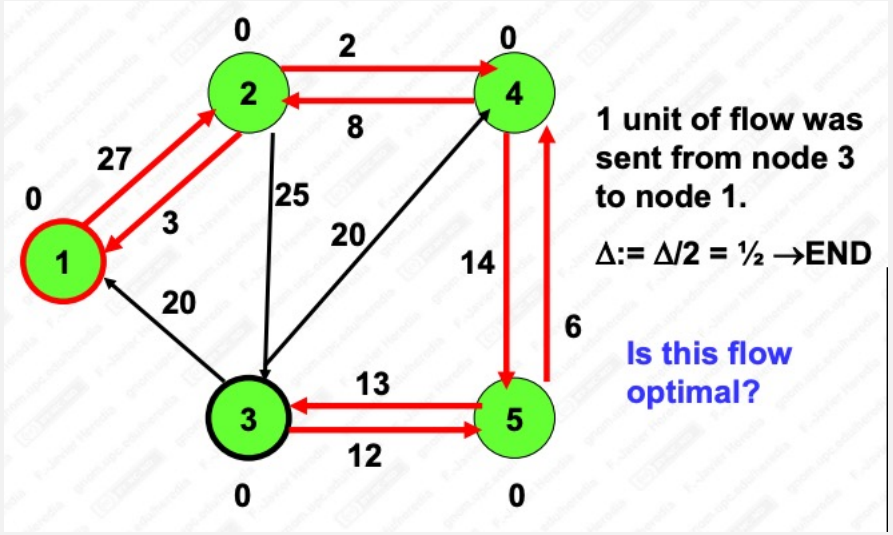


EXAMPLE
UPDATE THE RESIDUAL NETWORK



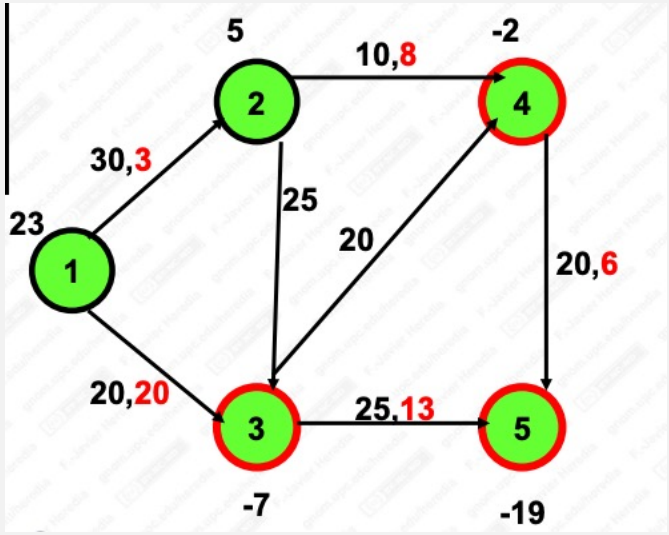
دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

EXAMPLE
UPDATE THE RESIDUAL NETWORK

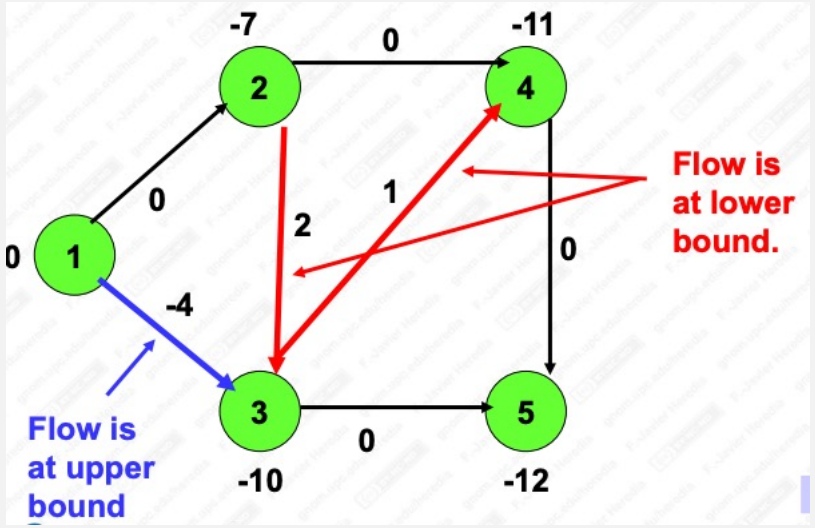


دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

EXAMPLE
THE FINAL OPTIMAL FLOW



EXAMPLE
THE FINAL OPTIMAL FLOW, POTENTIALS AND REDUCED COSTS



دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

CORRECTNESS

- 2 Δ -scaling phase ends when S(2 Δ) = ∅ or T(2 Δ) = ∅
- At that point, either e(i) < 2Δ for all i or e(i) > - 2 Δ for all i.
- These conditions imply that the sum of the excesses (whose magnitude equals the sum of deficits) is bounded by 2nΔ .
- At the beginning of the Δ -scaling phase, the algorithm first checks whether every arc (i, j) in Δ -residual network satisfies the reduced cost optimality condition $c_{ij}^{\pi} \geq 0$.
- in the Δ-scaling phase the sum of the excesses is bounded by 2(n + m)Δ
- Since each augmentation in this phase carries at least Δ units of flow from a node in S(Δ) to a node in T(Δ), each augmentation reduces the sum of the excesses by at least Δ units. Therefore, a scaling phase can perform at most 2(n + m) augmentations
- The capacity scaling algorithm solves the minimum cost flow problem in O(m logU S(n, m, nC)) time.

COMPARISON

COMPARISON

Algorithm	Number of iterations	Features
Cycle-canceling algorithm	$O(mCU)$	1. Maintains a feasible flow x at every iteration and augments flows along negative cycles in $G(x)$. 2. At each iteration, solves a shortest path problem with arbitrary arc lengths to identify a negative cycle. 3. Very flexible: some rules for selecting negative cycles leads to polynomial-time algorithms.
Successive shortest path algorithm	$O(nU)$	1. Maintains a pseudoflow x satisfying the optimality conditions and augments flow along shortest paths from excess nodes to deficit nodes in $G(x)$. 2. At each iteration, solves a shortest path problem with nonnegative arc lengths. 3. Very flexible: by selecting augmentations carefully, we can obtain several polynomial-time algorithms.

Capacity scaling

FINAL WORDS

REFERENCES

1. Network flows by Ahuja, Magnanti, Orlin
2. https://cp-algorithms.com/graph/min_cost_flow.html
3. https://www.topcoder.com/thrive/articles/Minimum%20Cost%20Flow%20Part%20Two:%20Algorithms
4. https://sboyles.github.io/teaching/ce377k/class14.pdf
5. https://upcommons.upc.edu/bitstream/handle/2117/191052/nf06_minimum_cost-4967.pdf
6. https://www-di.inf.puc-rio.br/~laber/MinCostFlow.pdf
7. https://www.columbia.edu/~cs2035/courses/ieor6614.S12/mcf-sp.pdf
8. https://homes.di.unimi.it/righini/Didattica/OttimizzazioneCombinatoria/MaterialeOC/9a%20-%20MinCostFlow.pdf
9. https://webpages.iust.ac.ir/yaghini/Courses/Network_891/05_1_Introduction.pdf
10. https://www.columbia.edu/~cs2035/courses/ieor6614.S16/mcf.pdf

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

THANKS FOR YOUR ATTENTION.
CONTACT INFO:

sarinaheshmatii@gmail.com