# 第二章 网络中的若干优化问题

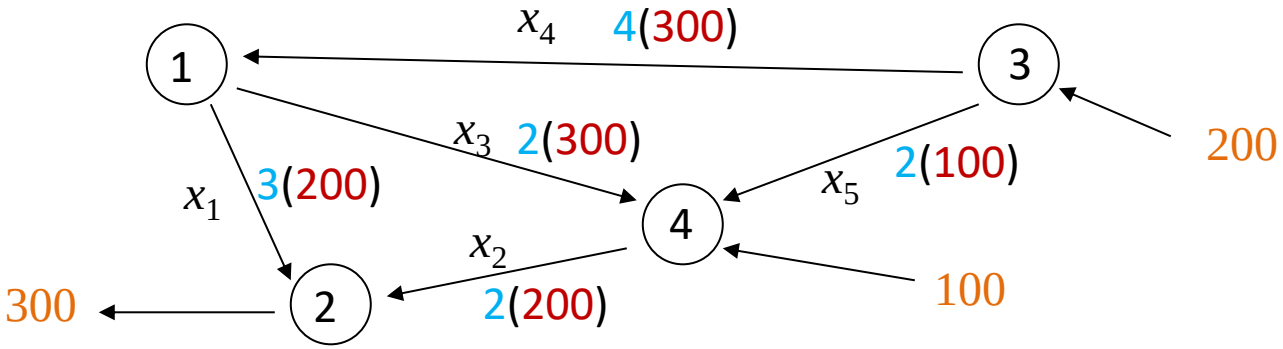
## 第三节 最小费用流问题

# 内容

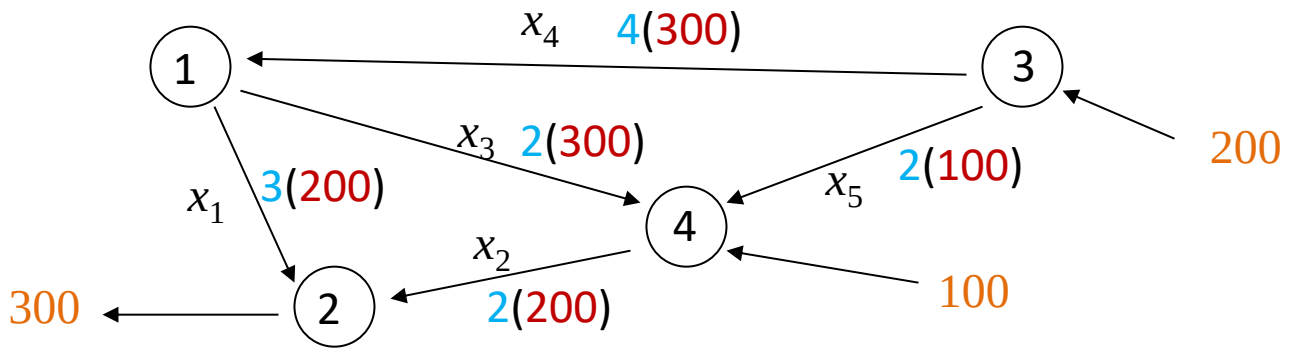
- 最小费用流问题的定义
- 最小费用流问题的最优性条件
- 最小费用流问题的算法
  - Cycle cancelling algorithm
  - Successive shortest path algorithm
  - Primal-Dual algorithm
- 最小费用流问题的应用

# 最小费用流问题的定义

- 最小费用流问题(minimum cost flow problem)解决的是当网络中的弧有不同容量限制和运输成本时，如何以最小的成本来完成给定的运输任务的问题。



# 最小费用流问题的定义




- 最小费用流问题可以写成一个线性数学规划问题
- 目标函数：总的运输成本

$$\min \sum_{ij \in A} c_{ij} x_{ij} = 3x_1 + 2x_2 + 2x_3 + 4x_4 + 2x_5$$

- 约束：

1. 节点流量守恒约束

$$x_4 - x_1 - x_3 = 0$$
$$x_1 + x_2 = 300$$
$$-x_4 - x_5 = -200$$
$$x_5 - x_2 + x_3 = -100$$

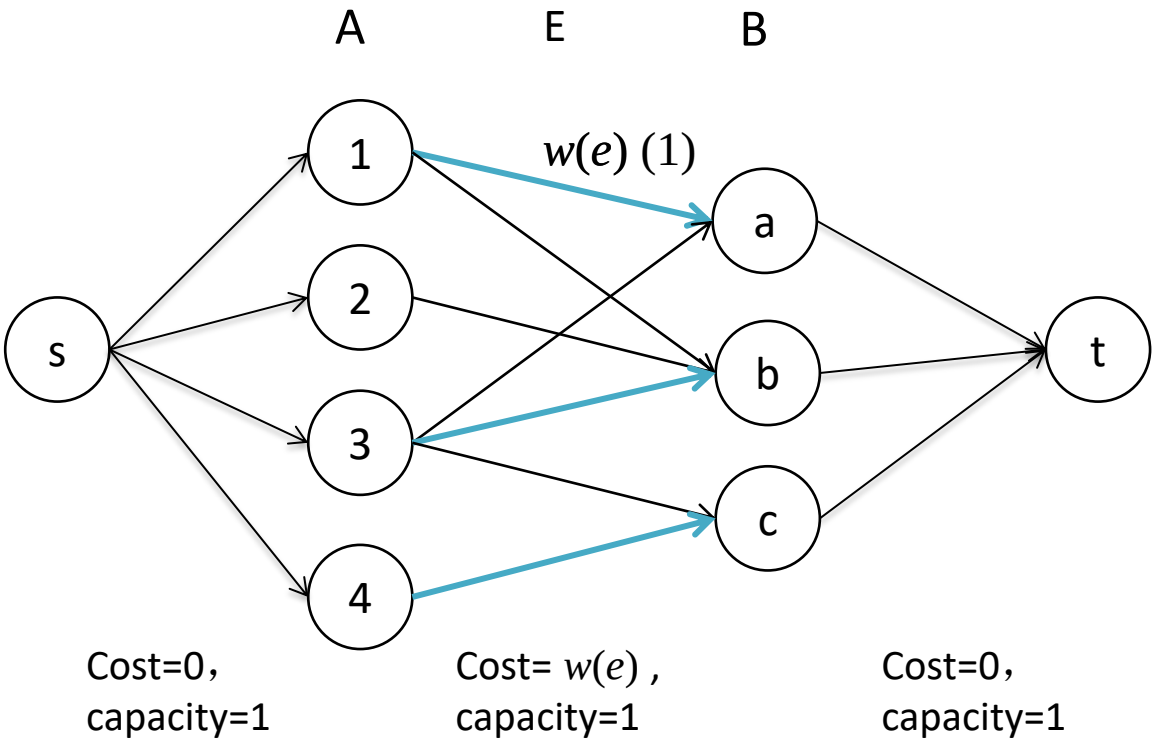
2. 弧上的容量限制

$$0 \le x_1 \le 200$$
$$0 \le x_2 \le 200$$
$$0 \le x_3 \le 300$$
$$0 \le x_4 \le 300$$
$$0 \le x_5 \le 100$$

# 最小费用流问题的应用

## 1. 二分图的最小权重匹配

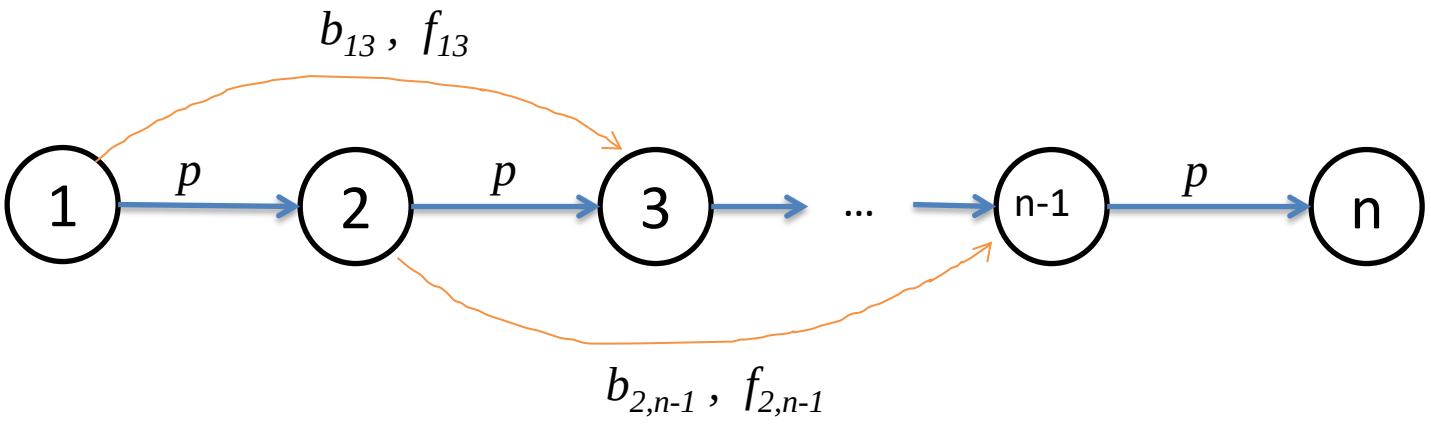
给定一个二分图 $G = (A \cup B, E)$, 命 $w: E \to R$ 为任意匹配边的权重. 找到一个该二分图的匹配方案$M \subseteq E$使得该匹配下的总权重最小。



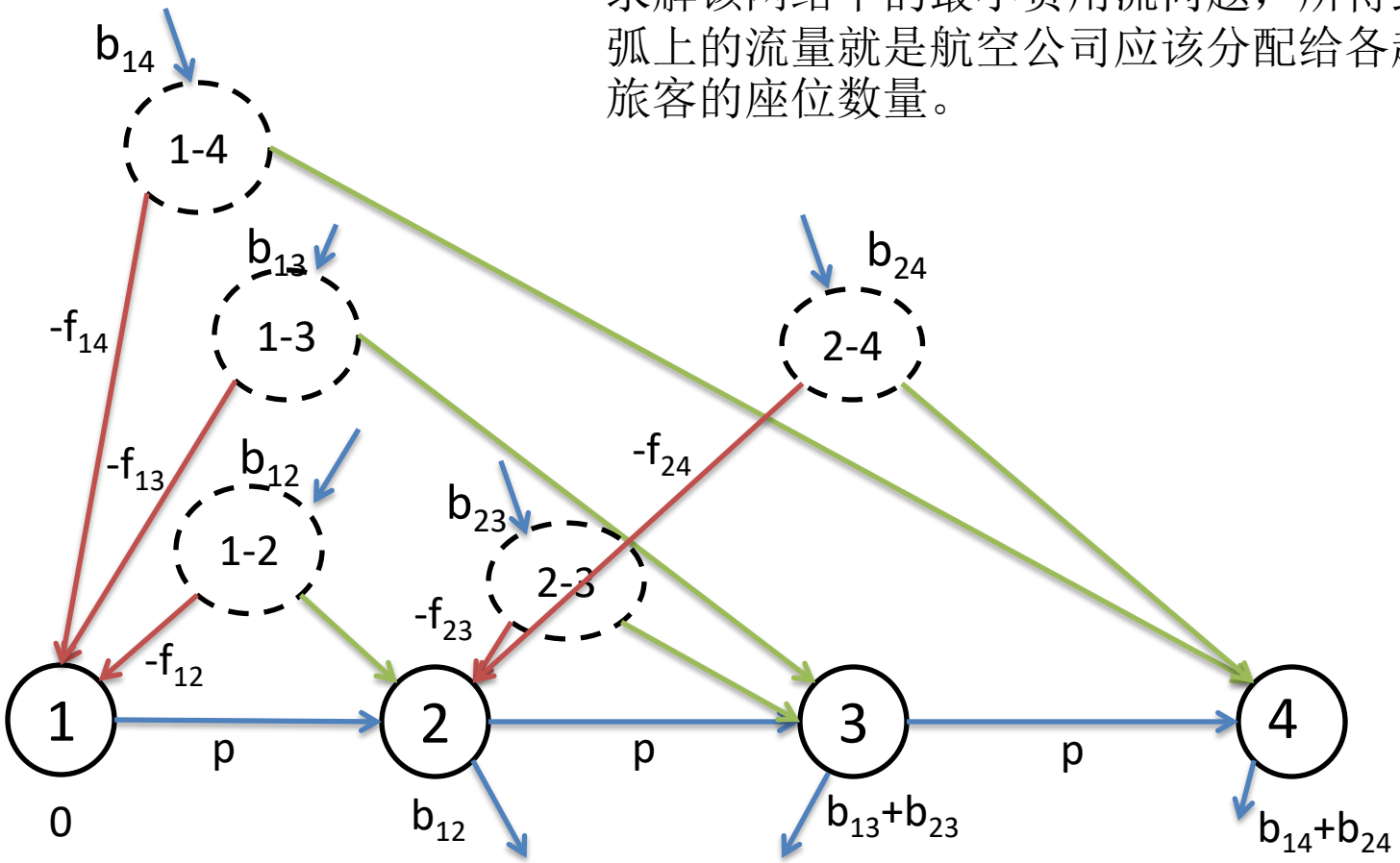
# 最小费用流问题的应用

## 2. 联程航班的最优装载量分配

考虑某航空公司的一架飞机，它以固定的顺序访问一系列的城市 1,2,3…,n，如下图。飞机的容量为$p$。命 $b_{ij}$ 表示往返于$i$和$j$点的乘客数量，$f_{ij}$ 表示该航空公司出售的$i$和$j$点的机票价格。航空公司希望确定它分配给不同起止点间旅客需求的座位数量，使得在不超过飞机容量限制的情况下最大化其收益。



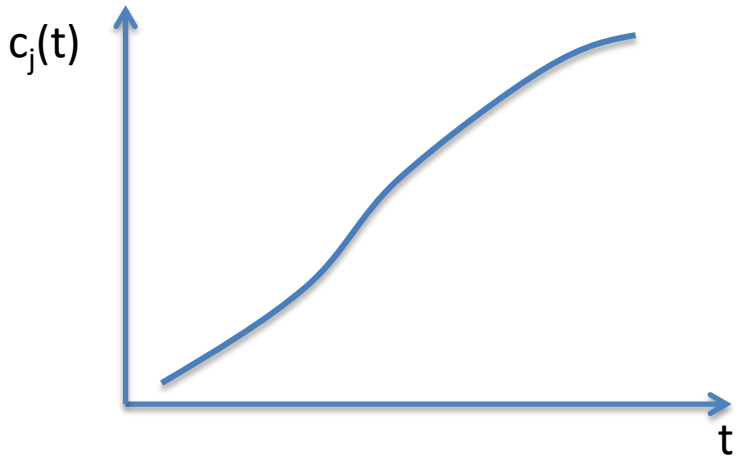
求解该网络下的最小费用流问题，所得到的红色弧上的流量就是航空公司应该分配给各起止点间旅客的座位数量。



# 最小费用流问题的应用

## 3. 考虑延迟成本的排程问题

考虑一个有q台机器的平行机排程问题，共有p个任务需要执行。每个任务的作业时间都是$\alpha$. 每个任务的延迟成本是其完成时间的单调增函数：$c_j(t)$ j=1,…p. 请找到这p个任务的排程方案，使得总延期成本 $\sum_{j=1}^{N} c_j(t)$ 最小.

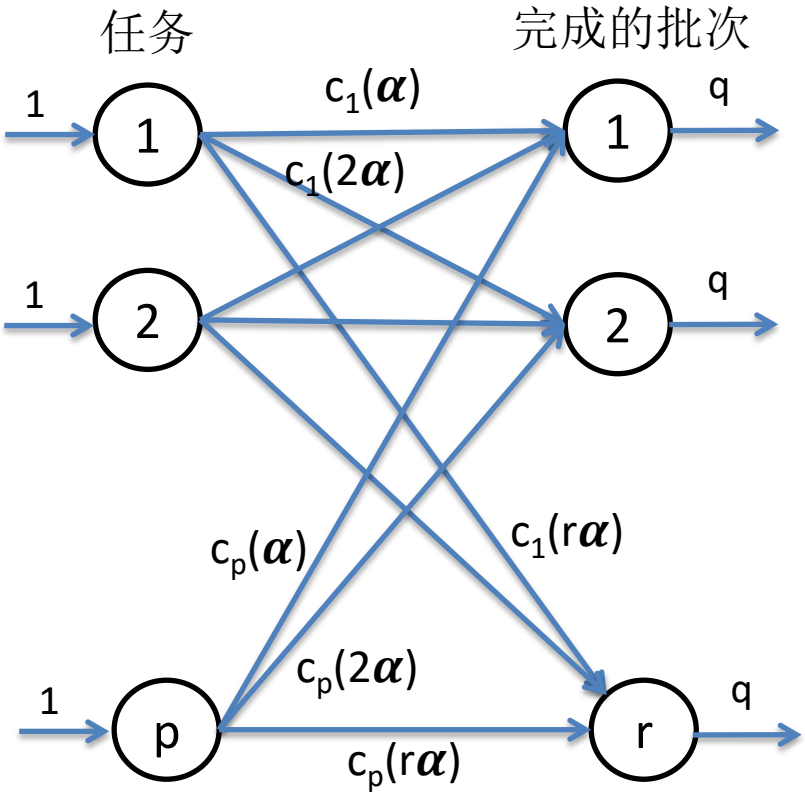



| 机器 | 0 | α | 2α | 3α | 4α |
|---|---|---|---|---|---|
| 1 | 1 | 3 | 2 | 7 | |
| 2 | 8 | 4 | 15 | 13 | |
| … | | | | | |
| q | 6 | 11 | | | |

# 最小费用流问题的应用

## 3. 考虑延迟成本的排程问题

考虑一个有q台机器的平行机排程问题，共有p个任务需要执行。每个任务的作业时间都是$\alpha$. 每个任务的延迟成本是其完成时间的单调增函数：$c_j(t)$ j=1,…p. 请找到这p个任务的排程方案，使得总延期成本 $\sum_{j=1}^{N} c_j(t)$ 最小.




$r=\lceil p/q \rceil$：每台机器的平均任务处理数

# 最小费用流问题的求解

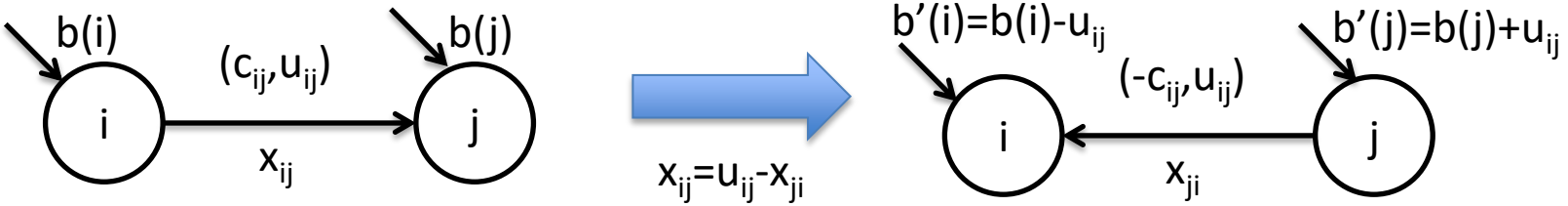
- The algorithm for min-cost network flow problem uses the ideas coming from both the shortest path problem and the maximum flow problem
- Shortest path
  - Optimality conditions: $d(i)+c_{ij} \geq d(j)$ for any $(i,j)$
  - Update $d(j)$ if the condition is violated
- Maximum flow
  - Working on residual network

# Basic Assumptions

- All data are integral
  - Cost, supply/demand, capacity
- The problem is feasible
- There is a direct path between any pair of nodes
  - If not, we can add a dummy arc with a very large cost
- All arc costs are nonnegative
- All lower bounds are zero
  - Non-zero lower bound can be removed (c.f. the maximum flow problem with positive lower bounds)

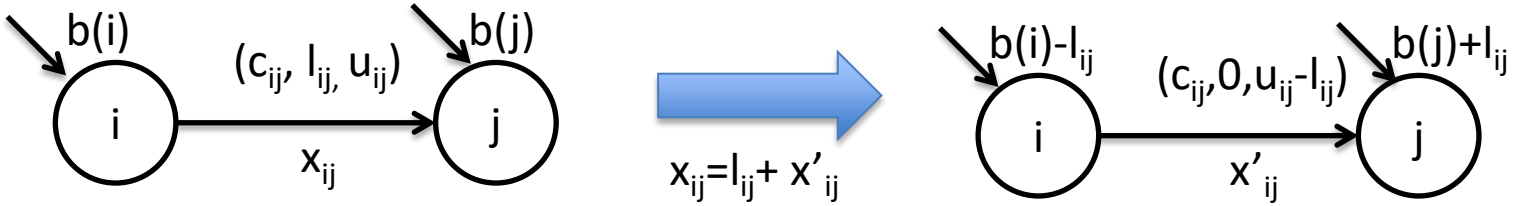
# Network transformations

- **Arc reversal:** typically used to remove arcs with negative costs




Send $u_{ij}$ units of flow on the arc (which decreases b(i) by $u_{ij}$ and increases b(j) by $u_{ij}$ and then replace arc (i,j) by arc (j,i) with cost $-c_{ij}$ . The new flow $x_{ji}$ measures the amount of flow we 'remove' from the 'full capacity' flow of $u_{ij}$ .

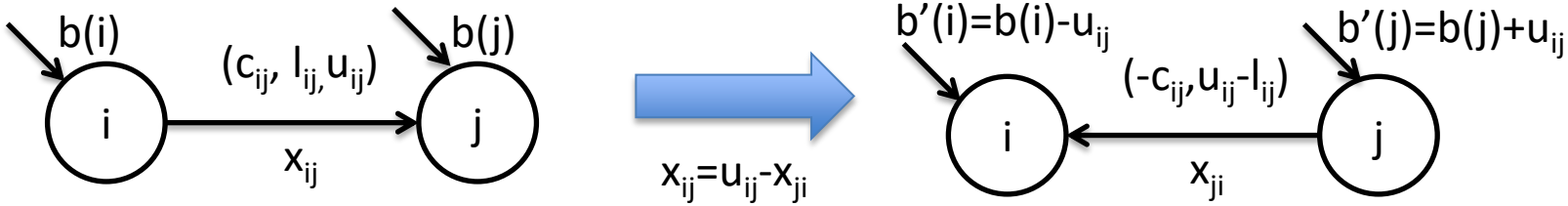
- **Removing non-zero lower bounds** $l_{ij}$




Send $l_{ij}$ units of flow on the arc (which decreases b(i) by $u_{ij}$ and increases b(j) by $u_{ij}$ and measures the incremental flow $x'_{ij}$ on the arc beyond the flow value $l_{ij}$ .

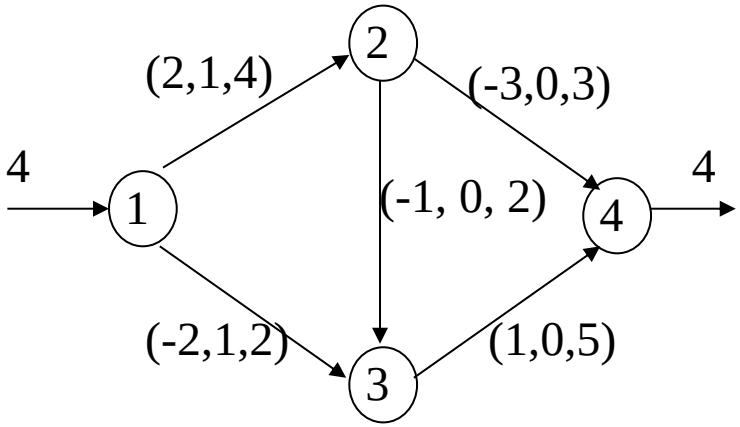
# Network transformations

- **How to transform arcs with both non-zero lower bounds and negative costs?**



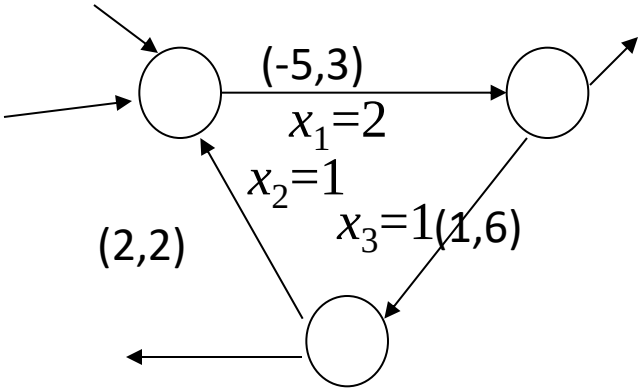
# Example

- Transform the following network to a one with no negative arc and no non-zero lower bound

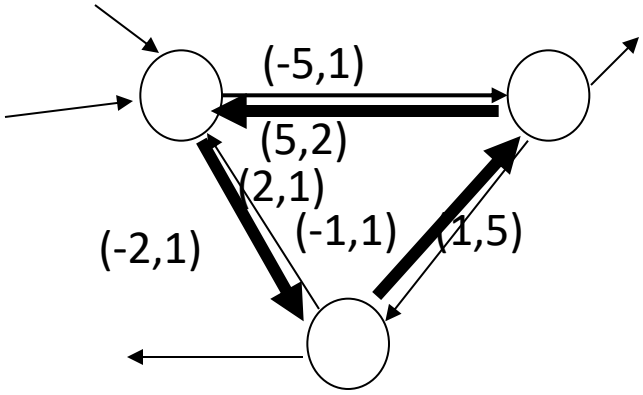


# Residual Network for a Feasible Flow

- For a given flow $x$ on network $G$, we can define a residual network $G(x)$
  - For $x_{ij}$ on each arc $(i,j)$ with $(c_{ij},u_{ij})$ in $G$, we have two arcs in $G(x)$:
    - $(i,j)$ with $(c_{ij}, u_{ij}-x_{ij})$: reduce capacity on the forward arc
    - $(j,i)$ with $(-c_{ij}, x_{ij})$: increase capacity on the backward arc which is with a negative cost




A network with a flow $x$




The residual network $G(x)$

# Algorithms

- Cycle canceling algorithm
- Successive shortest path algorithm
- Primal-Dual algorithm

# Cycle canceling algorithm

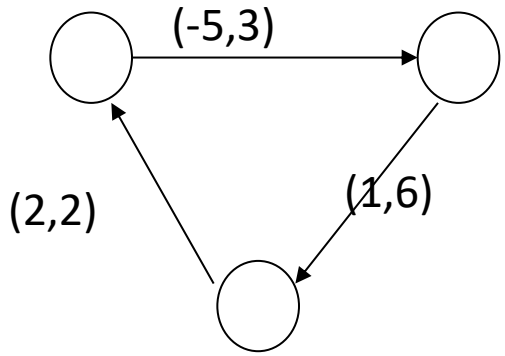
# 最小费用流问题的最优性条件1

## Negative cycle optimality conditions

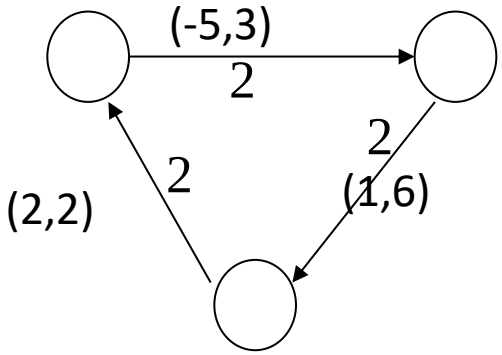
A feasible solution *x* is optimal if and only if the residual network *G*(*x*) contains no negative cost directed cycles.

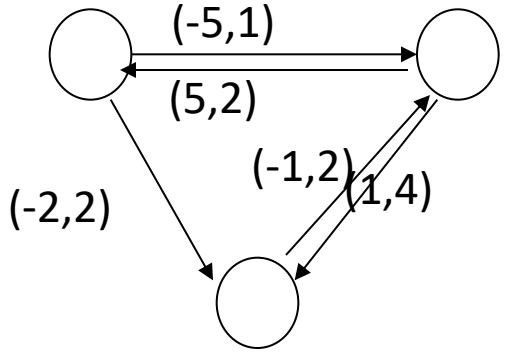
Increase 2 units of flow along the negative cycle

Cost change is (-5+2+1)*2 = -4 < 0




A residual network with a negative cycle




Residual network after flow increase: The negative cycle is cancelled

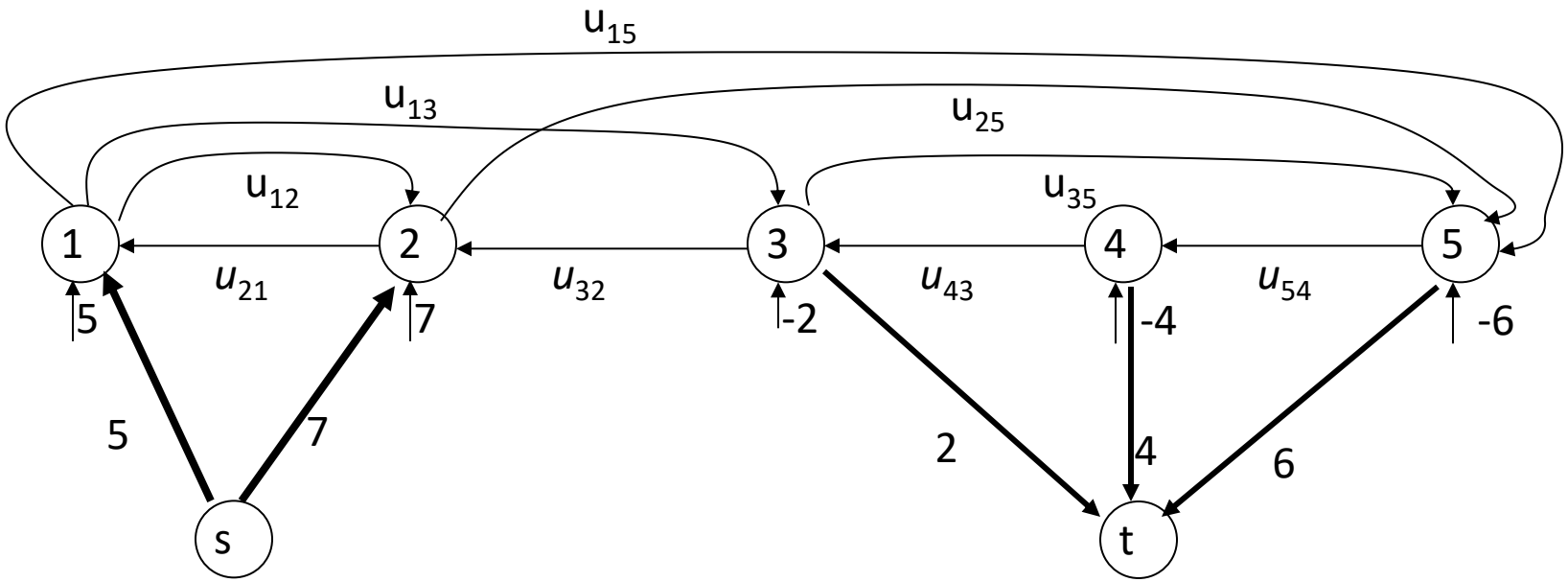
# Cycle Canceling Algorithm

- Find a feasible flow $x$
- **While** $G(x)$ contains a negative cycle $W$ **do**
  - **Begin**
    - Let $\delta = \min\{ r_{ij} \mid (i,j) \text{ on } W\}$
    - Augment $\delta$ units of flow along $W$ and update $G(x)$
  - **End**

How to find an initial feasible flow?

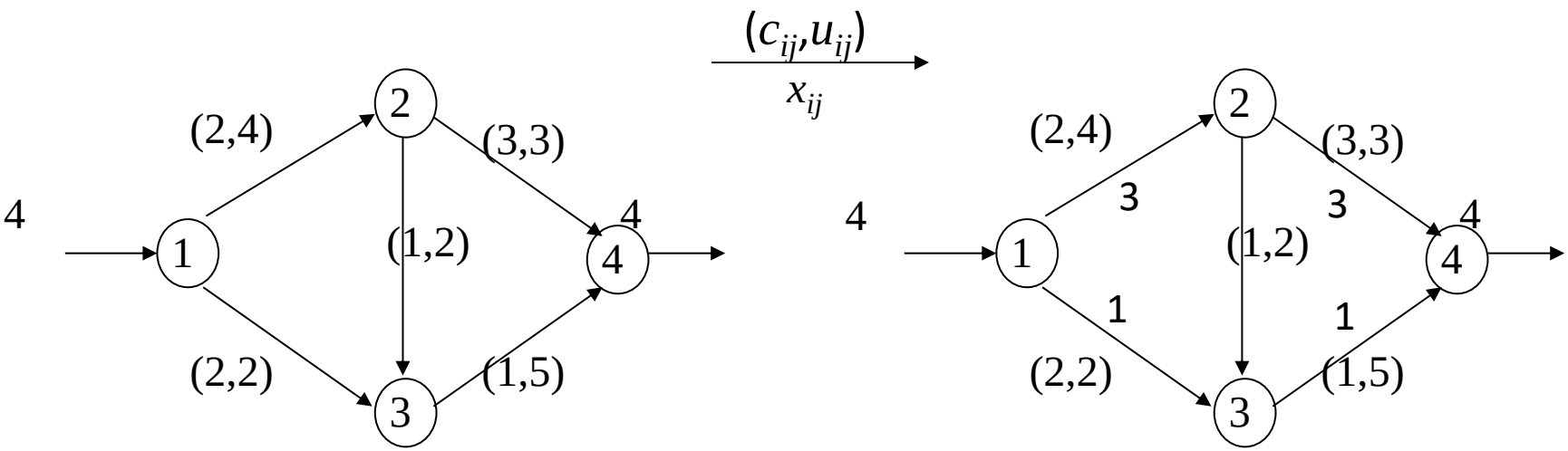
How to detect a negative cycle?

# Initial Feasible Solution



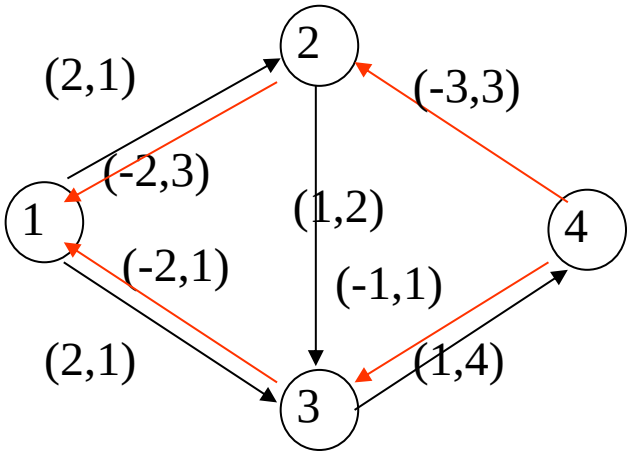

- By solving a maximum flow problem
  - A virtual source node *s* with arcs to each node with a supply
    - Let the arc capacity be the supply quantity
  - A virtual destination node *t* with arcs from each node with a demand (negative supply)
    - Let the arc capacity be the demand quantity
  - Is the maximum flow from *s* to *t* equal to 12 (why 12)?

# Examples for Cycle Canceling




Original network

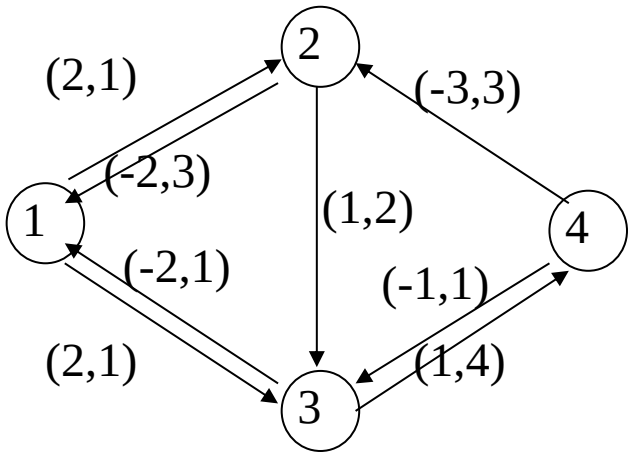
A feasible solution on the original network




Residual network for the feasible solution

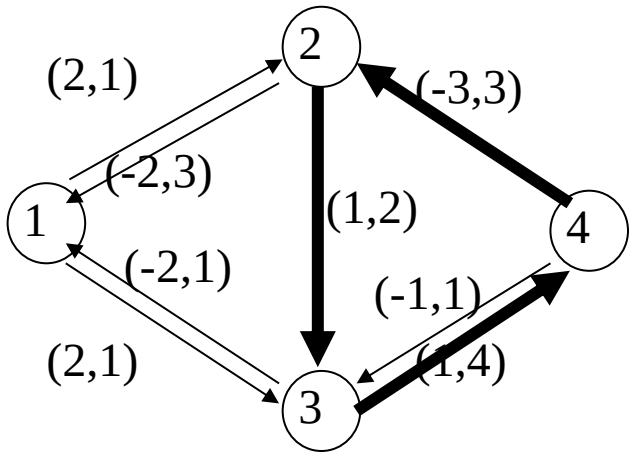
Consider: Given $G$ and $G(x)$, how can we know what is $x$?

# Examples for Cycle Canceling




$(c_{ij},u_{ij})$

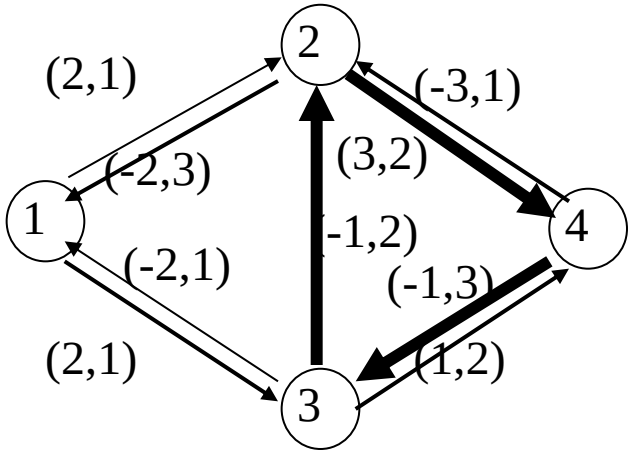
$x_{ij}$




Residual network for the feasible solution

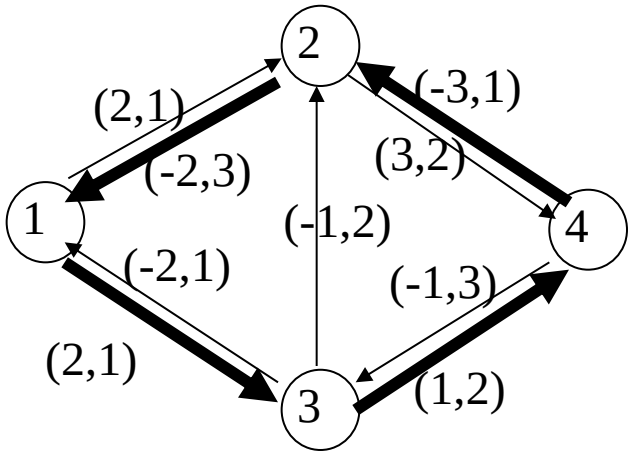
A negative cycle on the residual network

The flow to be sent is min{3,2,4} = 2



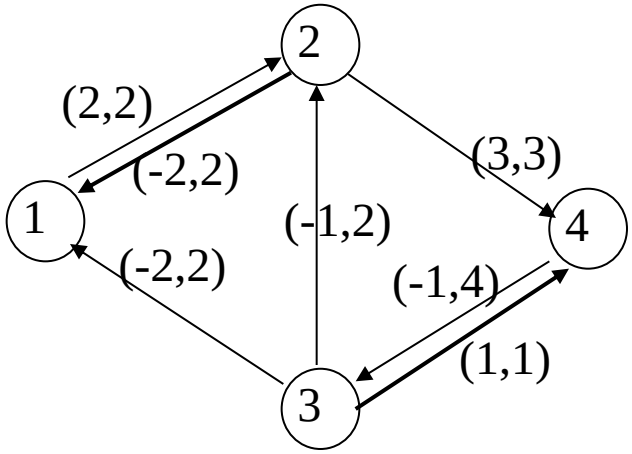

Residual network after sending 2 units of flow

# Examples for Cycle Canceling




$(c_{ij}, u_{ij})$

$x_{ij}$




Another negative cycle on residual network

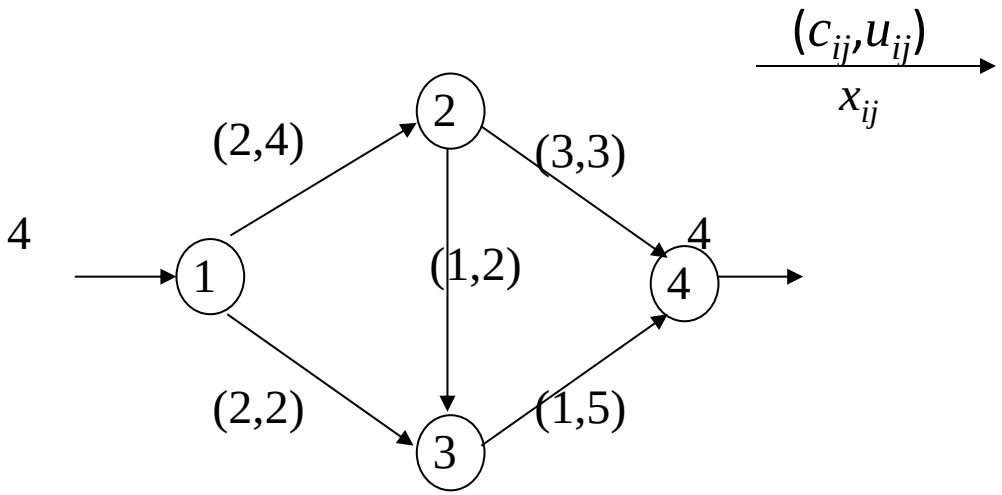
The capacity for the negative cycle is 1

New residual network after sending 1 unit of flow along the negative cycle

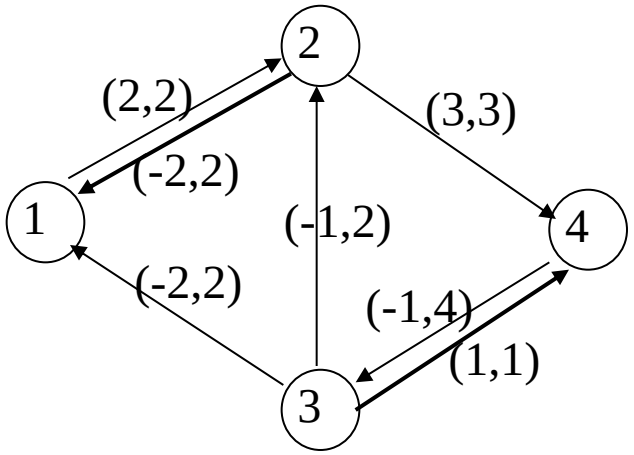
No other negative cycles

The solution cannot be improved

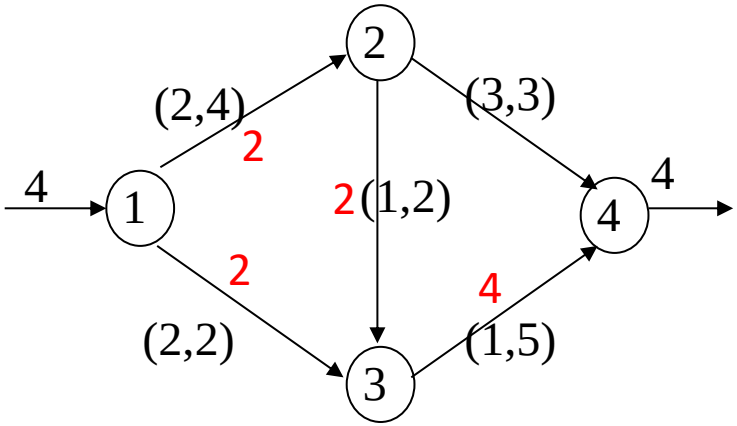
# Examples for Cycle Canceling




Original network




Final residual network

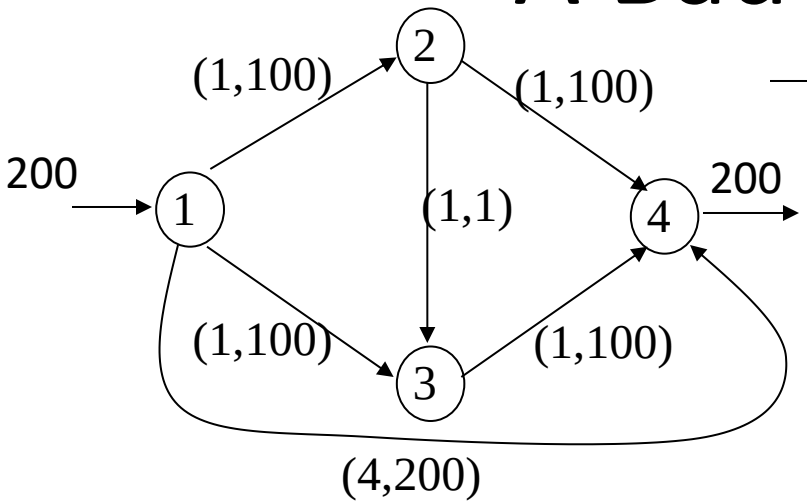



Optimal solution

# Cycle Canceling Algorithm: Complexity

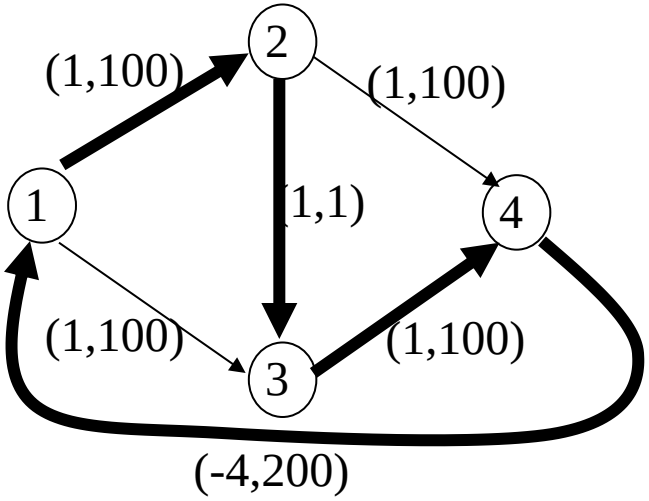
- Number of iterations is in $O(mCU)$
  - C: maximum absolute value of the cost
  - U: maximum capacity
  - mCU is the maximum possible cost for a feasible solution
  - In each iteration the cost is reduced by at least one
- Within each iteration, we need to find a negative cycle, which is in $O(nm)$
  - For example, the FIFO label correcting algorithm
- Overall time complexity is $O(nm^2CU)$
  - Pseudo polynomial time

# A Bad Example



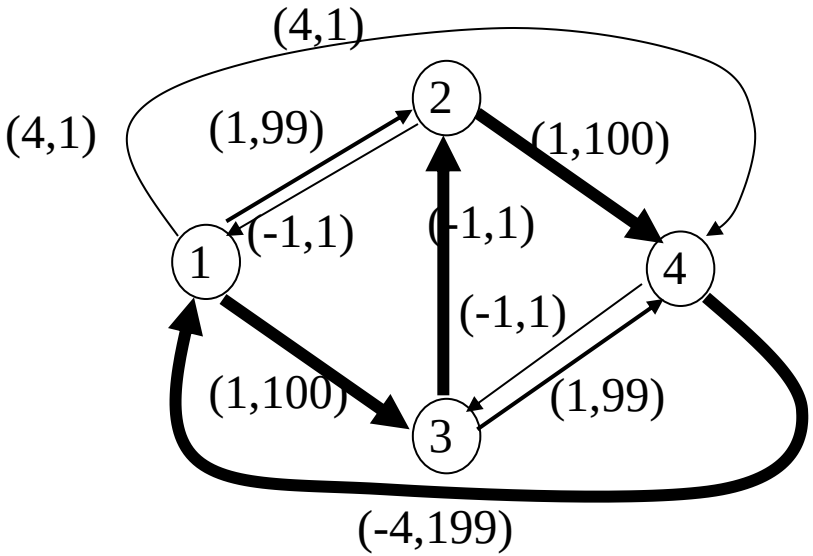

$(c_{ij}, u_{ij})$

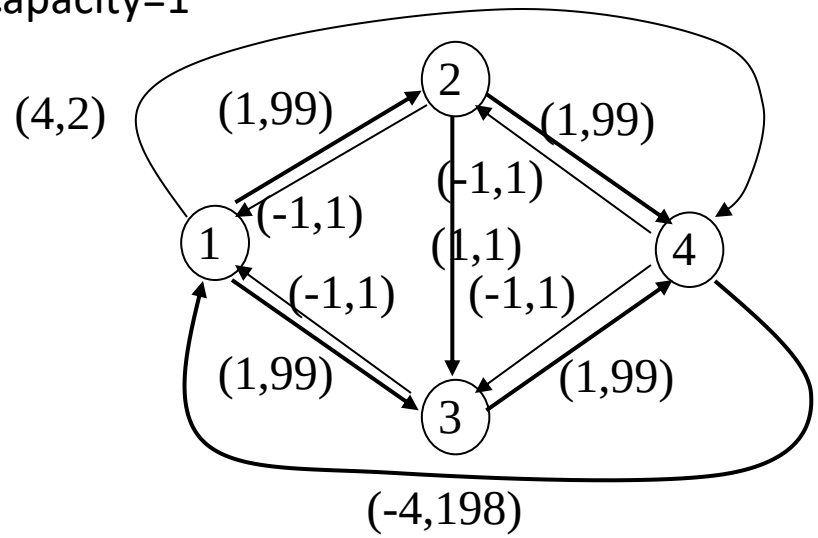
$x_{ij}$




Initial feasible solution chooses sending 200 units flow on arc (1,4)

Residual network & a negative cycle with capacity=1



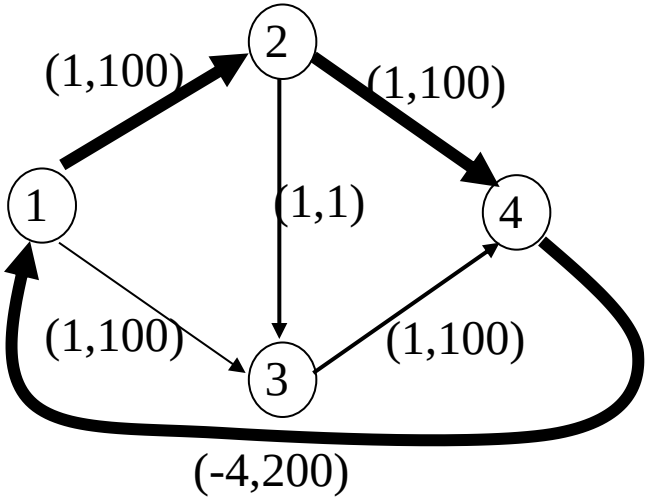




New Residual network & negative cycle

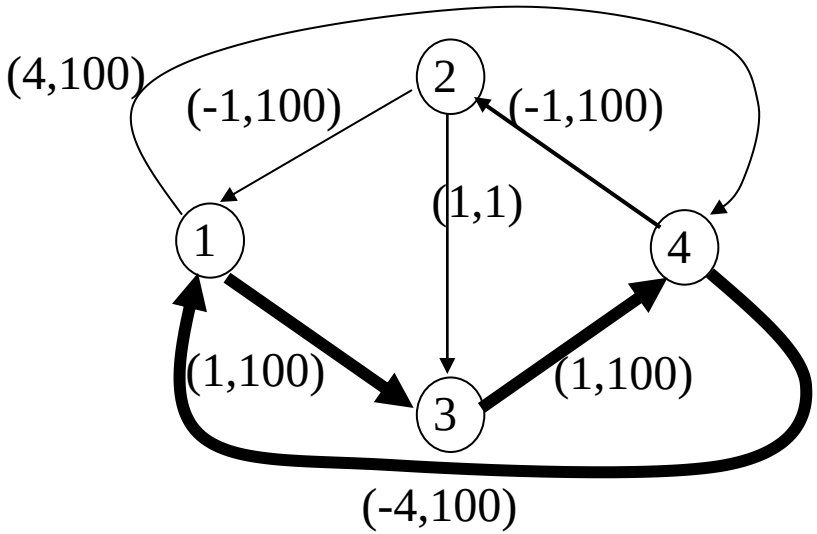
After two iterations

# A Better Case Example

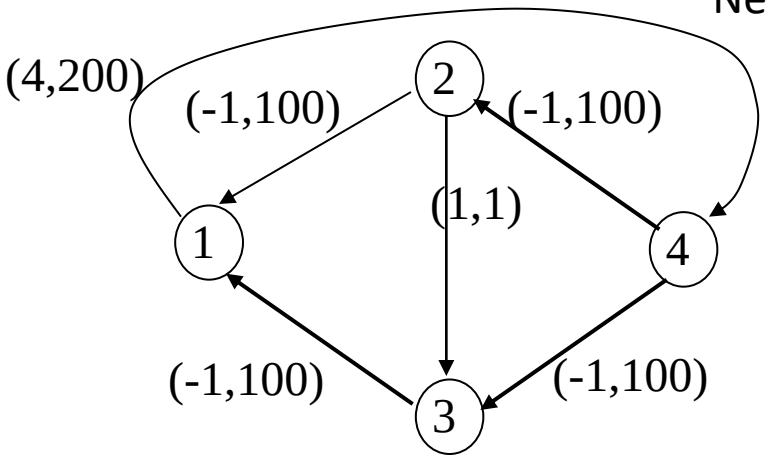
If we repeat the above process, it will take 200 iterations to find the optimal solution. If we choose a different negative cycle, two iterations are enough.




A negative cycle with capacity=100




New Residual network & negative cycle




Final Residual network

Do we have better algorithms?

# Successive shortest path algorithm

# Node Potentials & Link Reduced Cost

- Node potential $\pi$
  - Each node *i* is associated with a real number $\pi(i)$
- Arc reduced cost (under given node potential $\pi$)
  - Defined as $c^{\pi}_{ij} = c_{ij} - \pi(i) + \pi(j)$
  - Applied to both original network and residual network
- Properties
  - For any directed path P from nodes *k* to *l*,

$$\sum_{(i,j)\in P} c^{\pi}_{ij} = \sum_{(i,j)\in P} c_{ij} - \pi(k) + \pi(l)$$

  - For any directed cycle W,

$$\sum_{(i,j)\in W} c^{\pi}_{ij} = \sum_{(i,j)\in W} c_{ij}$$

# 最小费用流问题的最优性条件2

## Reduced Cost Optimality Condition

A feasible solution $x^*$ is optimal if any only if there exist node potentials $\pi$ satisfying

$$c^{\pi}_{ij} \geq 0 \text{ for every arc } (i,j) \text{ in residual network } G(x^*)$$

- Reason: from the negative cycle optimality condition

→If there exist node potential $\pi$ leading to all $c^{\pi}_{ij} \geq 0$, then $G(x^*)$ has no negative cycle (in terms of both link reduced cost and link original cost). So $x^*$ is optimal.

←If $x^*$ is optimal, then $G(x^*)$ has no negative cycle (in terms of original link cost). Shortest path (in terms of original link cost) is well defined on $G(x^*)$ with link original cost.

Let $d(j)$ be the shortest path distance from node 1 to node $j$ on $G(x^*)$. Due to shortest path optimality condition, we have $d(j) \leq d(i)+c_{ij}$ for all $(i,j)$.

Rewrite it as $0 \leq c_{ij} -(-d(i))+(-d(j)) = c^{\pi}_{ij}$ .

So we can find node potentials by $\pi(j)=-d(j)$ for all node $j$, which makes all reduced cost ≥0

# Successive Shortest Path Algorithm

- Ideas
  - Each iteration finds a (possibly infeasible) flow $x$ while maintaining the reduced cost optimality for $G(x)$
    - To guarantee that there exist node potentials $\pi$ making reduced cost $c^{\pi}_{ij} \geq 0$ for every arc $(i,j)$ on $G(x)$
    - Called a pseudo flow
  - Gradually makes the pseudo flow feasible

# Theoretical Preparation

- Consider a pseudo flow $x$ and node potentials $\pi$
  - satisfying the reduced cost optimality condition under $\pi$
- We can have a new node potential $\pi'$ by
  - choosing a node $s$, calculating $d(j)$, the shortest path distance, to every node $j$ on the residual network
  - defining new node potentials $\pi'$ by $\pi'(j)=\pi(j)-d(j)$
  - The pseudo flow x also satisfies the reduced cost optimality condition($c^{\pi'}_{ij}>=0$) with the new node potentials $\pi'(j)$
- Properties for the new reduced cost $c^{\pi'}_{ij}$
  - The new reduced cost $c^{\pi'}_{ij}=0$ if arc $(i,j)$ is on the shortest path from node $s$ to node $j$ ($d(j)=d(i)+c^{\pi}_{ij}$, $c^{\pi}_{ij}= c_{ij}-\pi(i)+\pi(j)$)

# Theoretical Preparation

- Suppose that a pseudo flow (or a flow) x satisfies the reduced cost optimality conditions and we obtain x’ by sending flow along the shortest path from node s to some other node k, then x’ also satisfies the reduced cost optimality conditions.

# Successive Shortest Path Algorithm

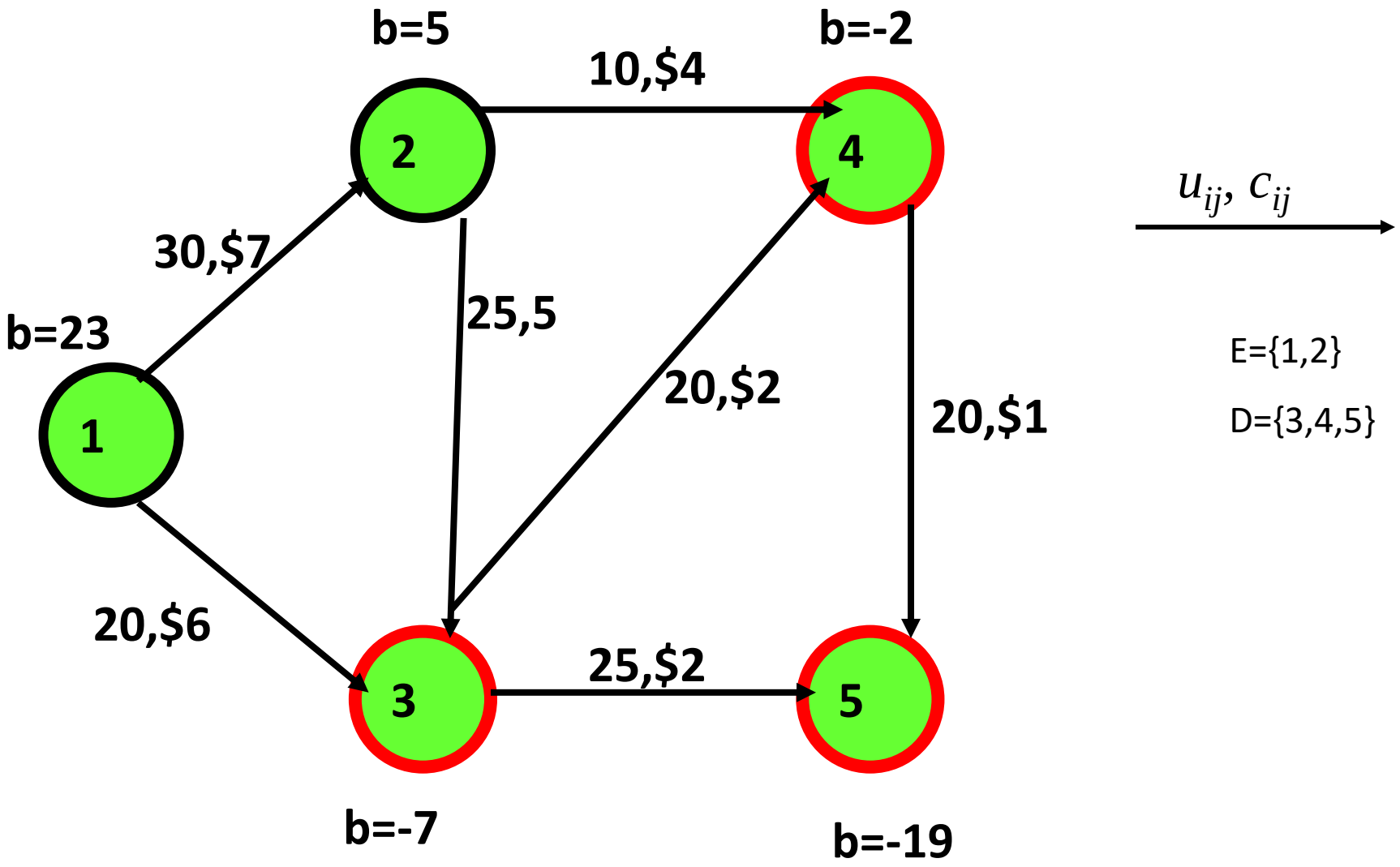
Gradually makes the pseudo flow feasible

- In a given pseudo flow $x$,
  - For each node $i$, let $e(i)=b(i)$ + total inflow – total outflow
  - Excess if $e(i)>0$, deficit if $e(i)<0$, balanced if $e(i)=0$
  - Sending flow from a node $i$ with $e(i)>0$ to a node $j$ with $e(j)<0$ will reduce the "degree of infeasibility"
    - We will send flow along the shortest path to maintain $c^{\pi}_{ij} \geq 0$ for every arc $(i,j)$

# Successive Shortest Path Algorithm

- Preprocess
  - Let $x$=0, π=0, e=b
- While there are nodes with e(i)>0 or e(i)<0
  - Select a node $k$ with e($k$)>0 and a node $l$ with e($l$)<0
  - Determine the shortest path $d(j)$ from node $k$ to all other nodes $j$ with respect to reduced cost on G($x$)
  - Update $\pi(j)=\pi(j)-d(j)$ for all node $j$
  - Update reduced costs for each arc $c^{\pi'}_{ij} = c^{\pi}_{ij} + d(i) - d(j)$
  - Send flow from $k$ to $l$ along the shortest path （$c^{\pi}_{ij} = 0$）
  - Update $x$, G($x$), and e(i)

# The Original network

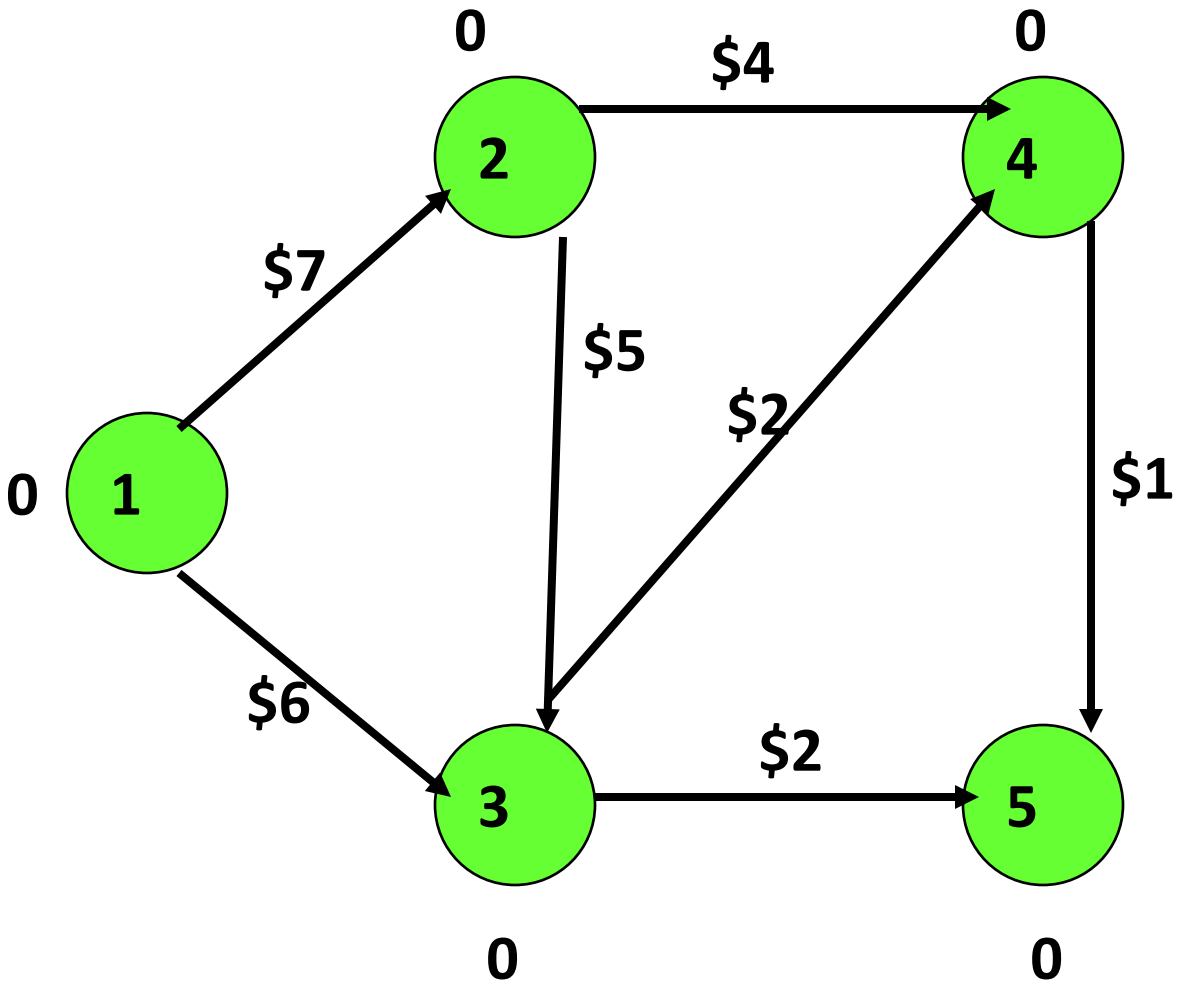



$u_{ij}, c_{ij}$

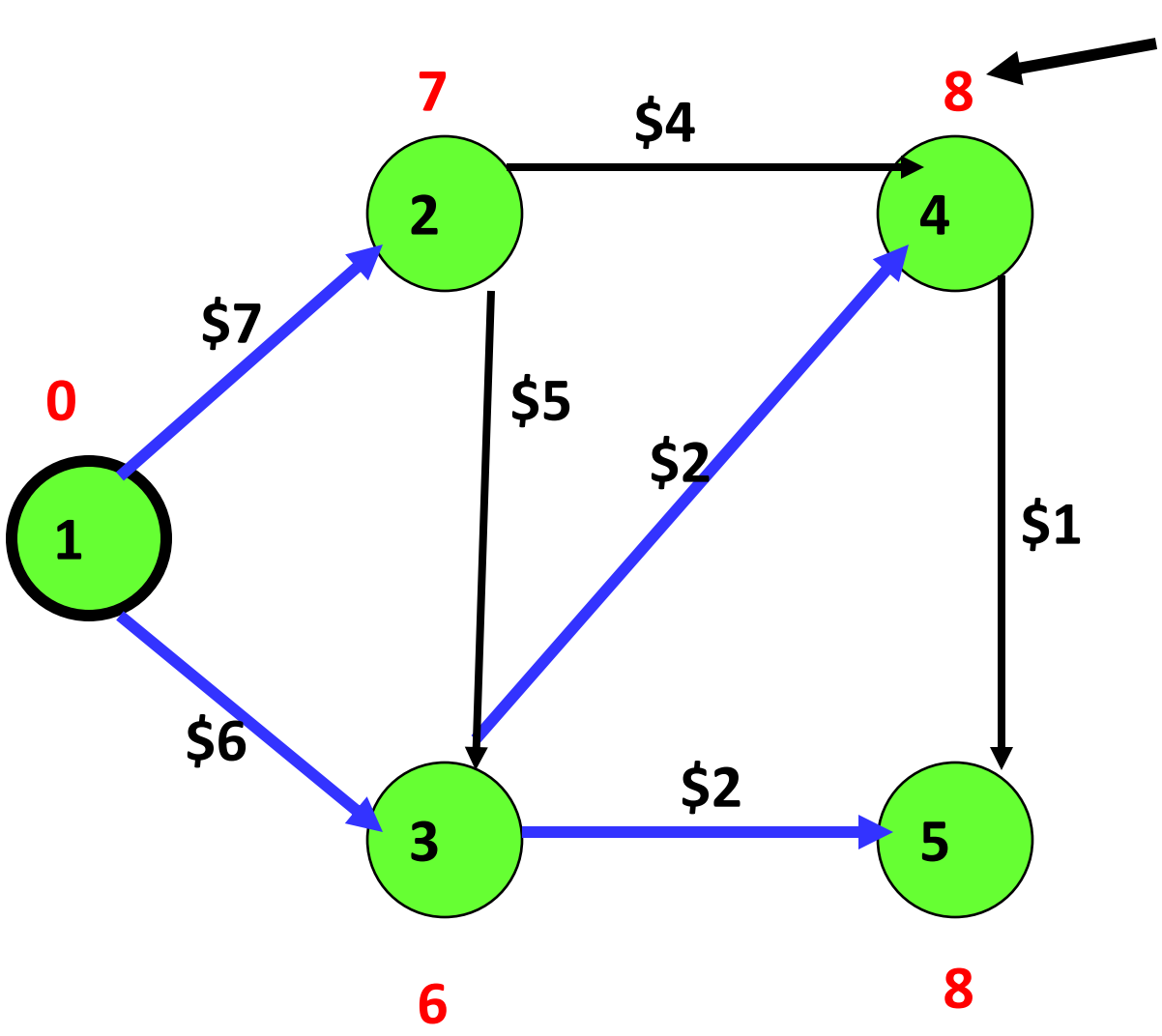
E={1,2}

D={3,4,5}

# Preprocess: Node Potentials and Reduced Costs

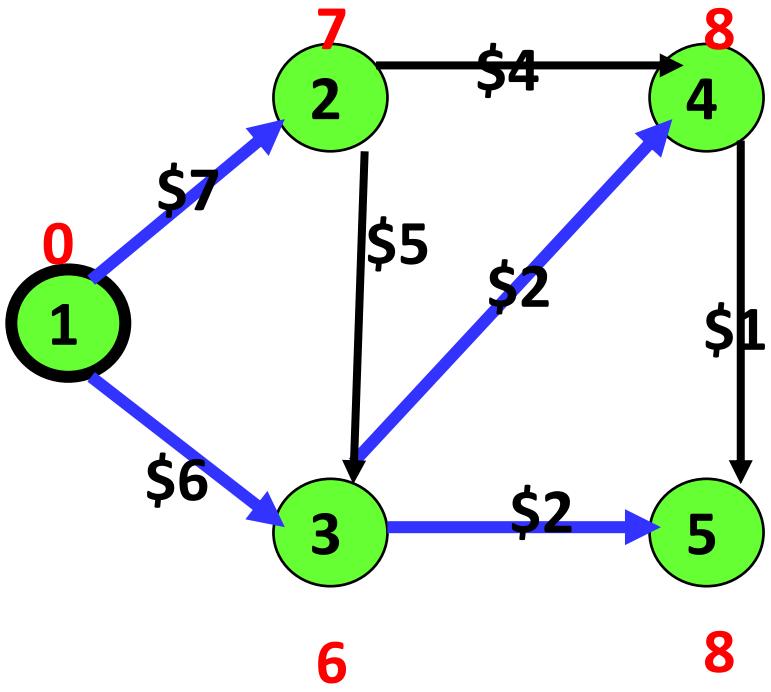


# Select a supply node and find the shortest paths



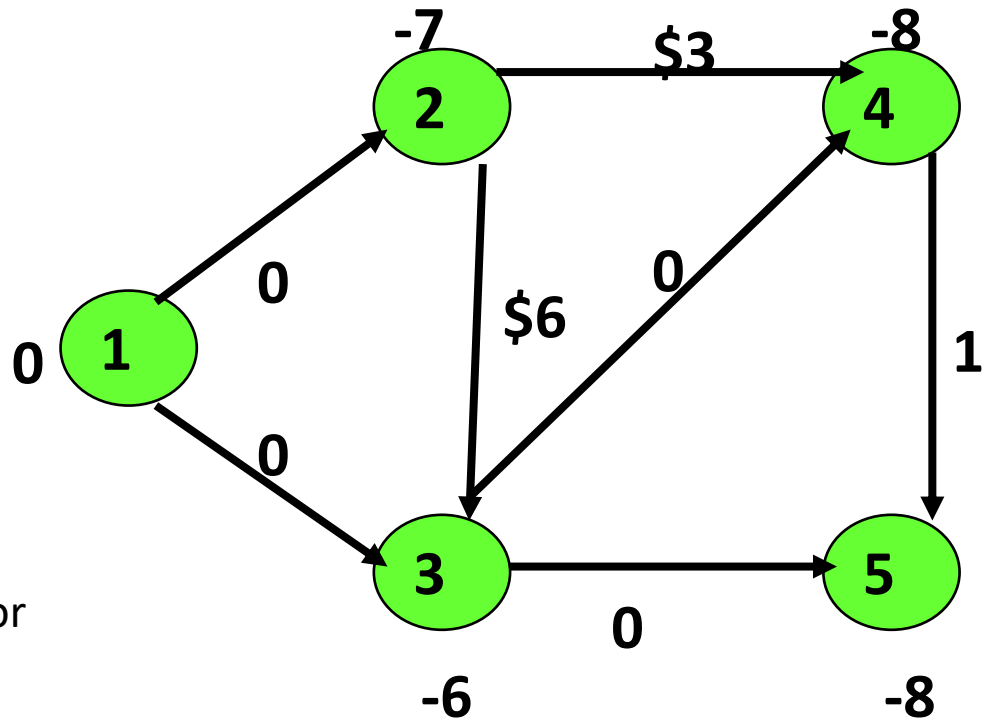

**The shortest path tree is marked in bold and blue.**

# Update the Node Potentials and the Reduced Costs



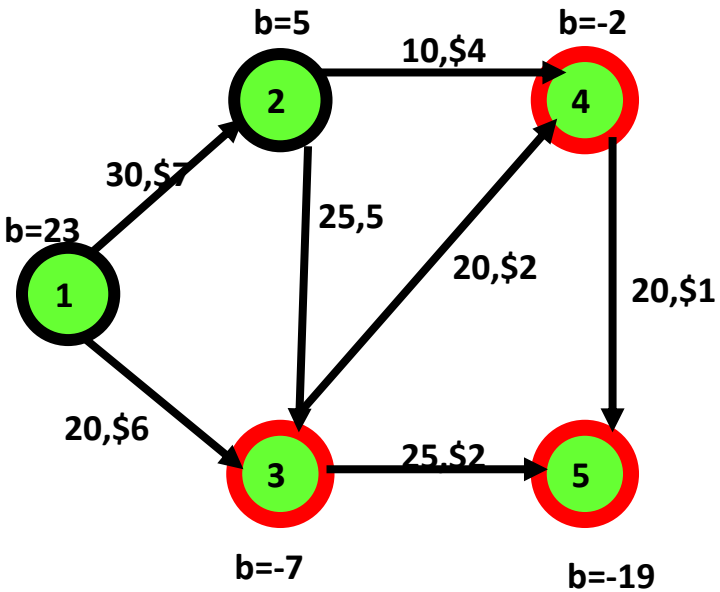

*Update node potentials:* $\pi(j)=\pi(j)-d(j)$

*Update reduced cost:* $c^{\pi}_{ij}=c_{ij}-\pi(i)+\pi(j)$




Observation: reduced cost is 0 for an arc on the shortest path tree

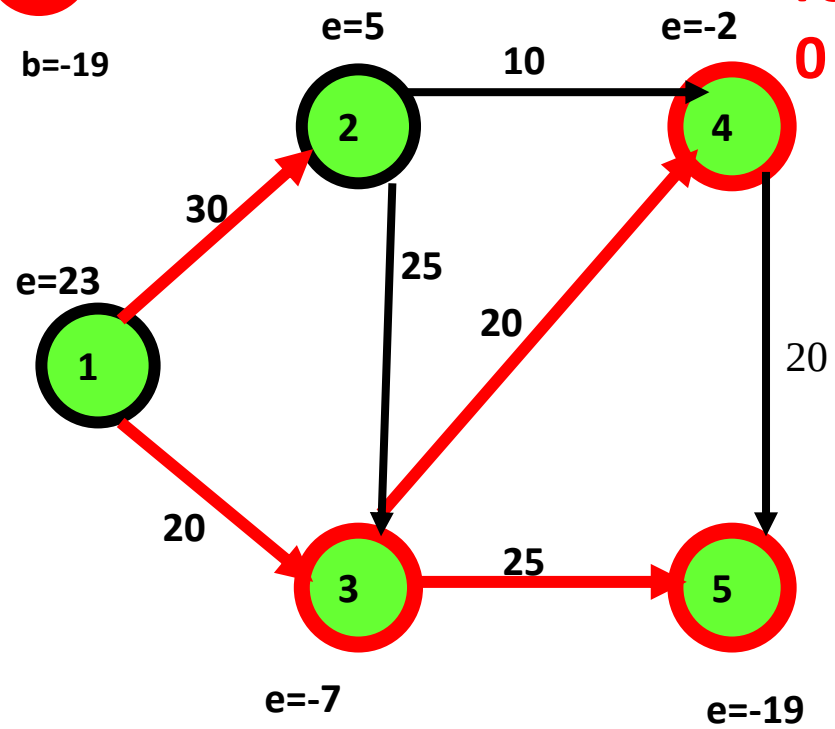
# Send Flow to a Demand Node Along Shortest Path



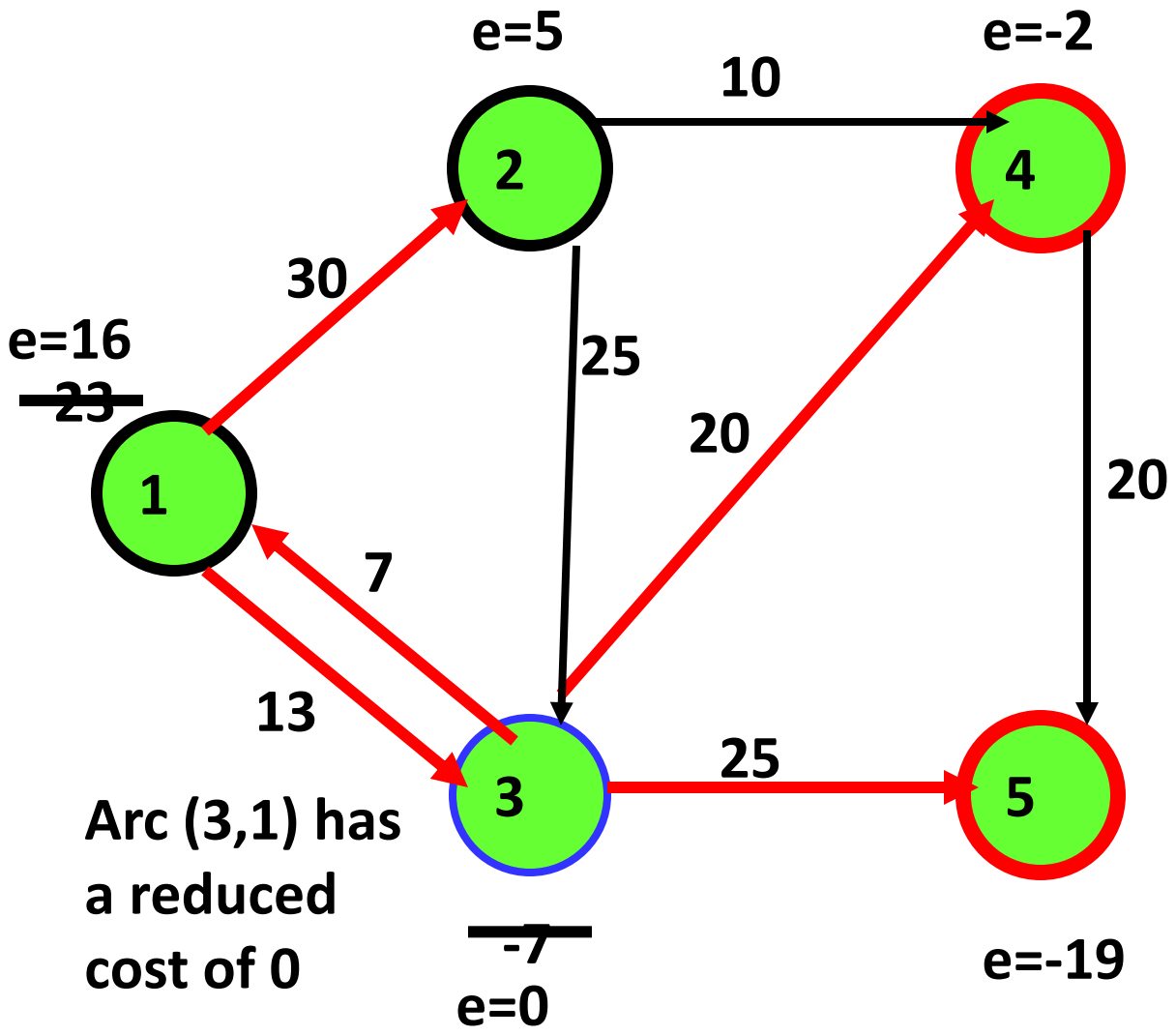

send 7 units from node 1 to node 3

**Arc numbers are residual capacities.**

**Red arcs have a reduced cost of 0**



# Update the Residual Network




If an arc is added to G(x), then it has a reduced cost of 0, and it is red.

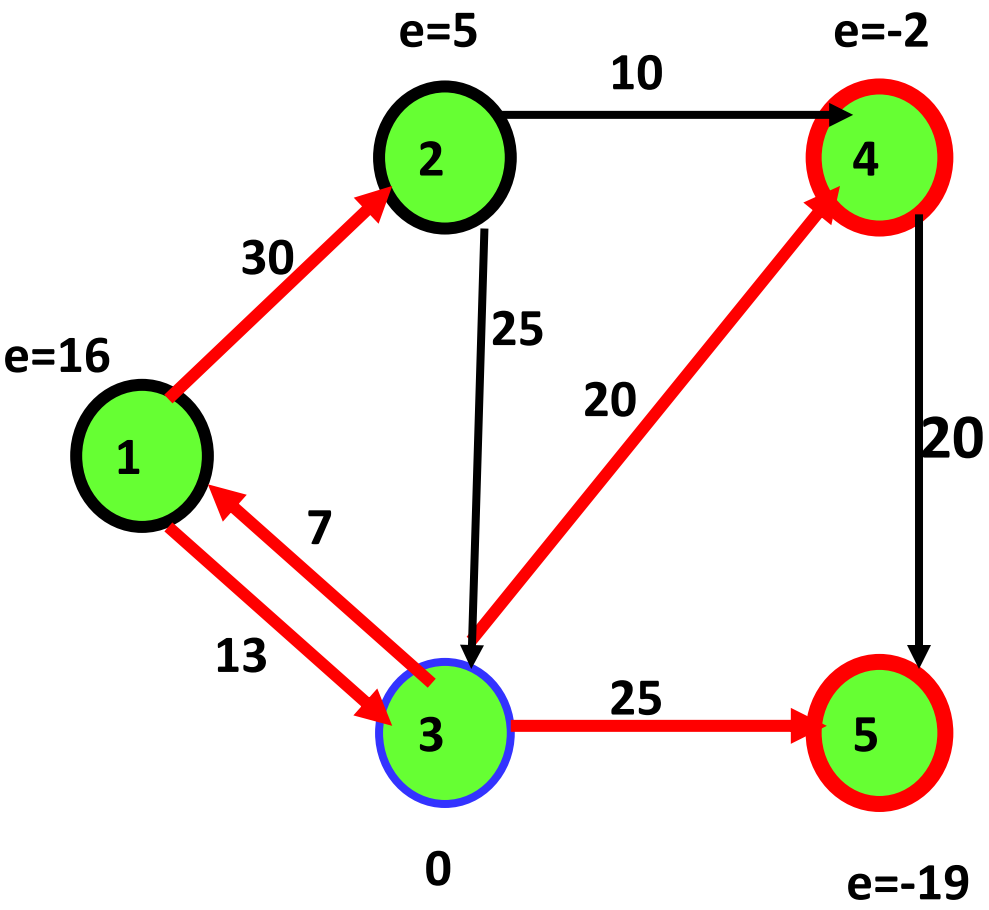
Arcs in the residual network will always have a non-negative reduced cost

# A comment

- At this point, one would choose a source node, and then find the shortest path from the source node to all other nodes, and then update the residual network.

- However, there are still paths of 0 reduced cost in the residual network, and it makes sense to use them. This heuristic is quite useful in practice.

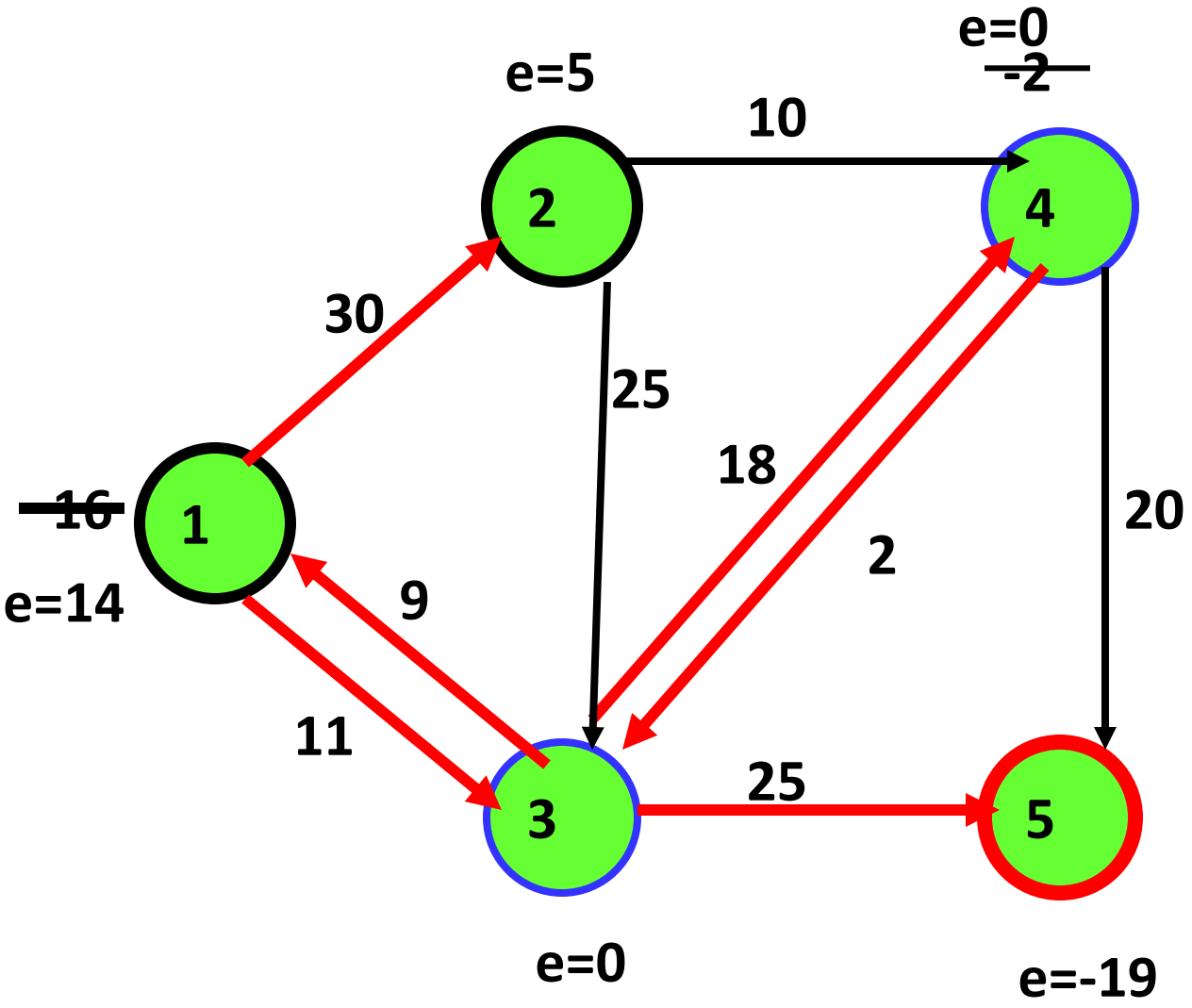
# Send Flow to another Demand Node




**Recall that red arcs have a reduced cost of 0**

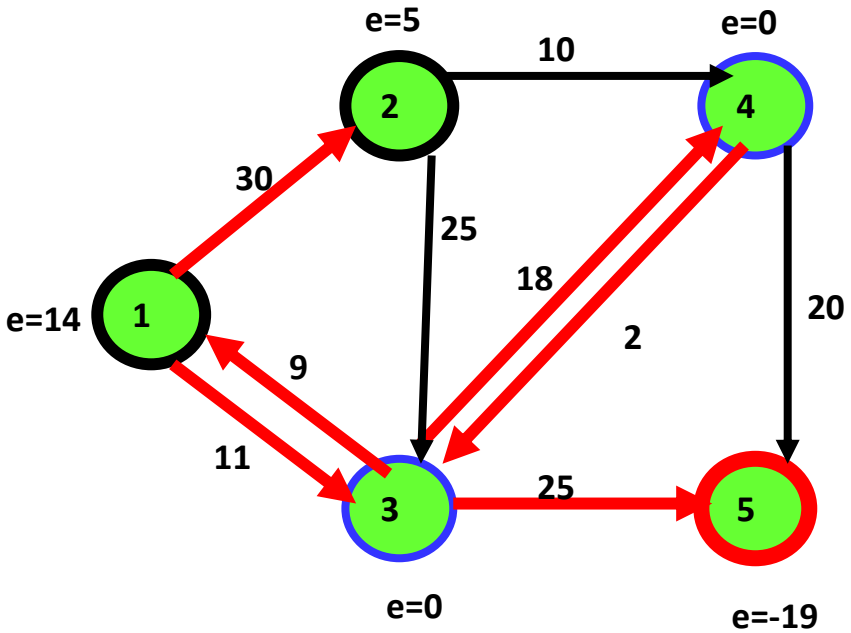
**Send 2 units of flow from node 1 to node 4**

# Update the Residual Network

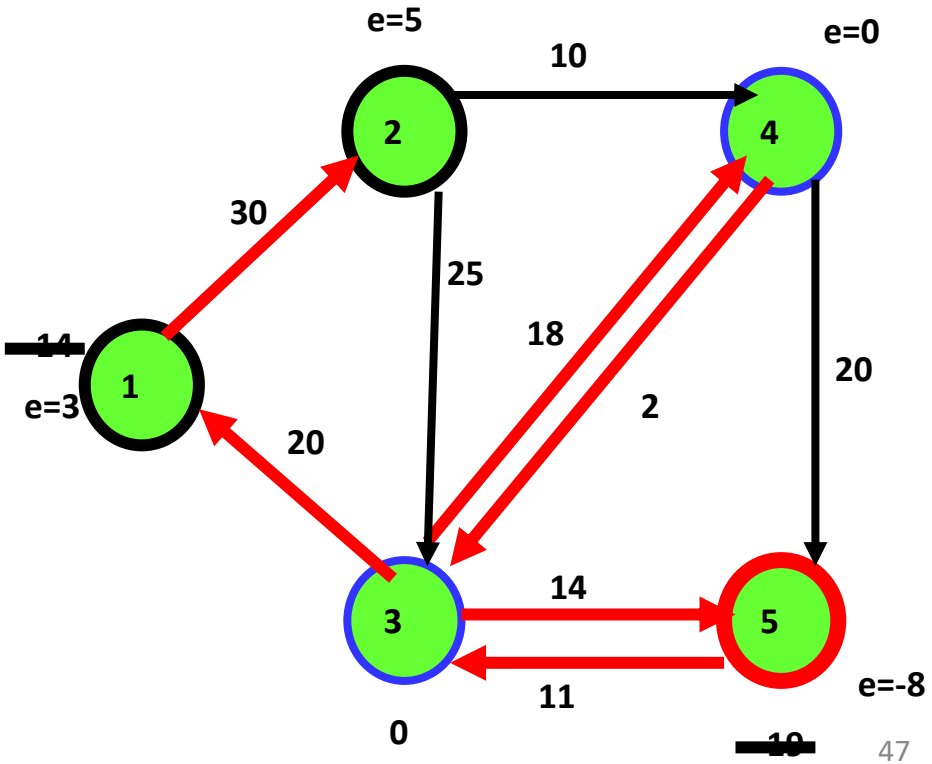



**2 units of flow were sent from node 1 to node 4 on 1-3-4**

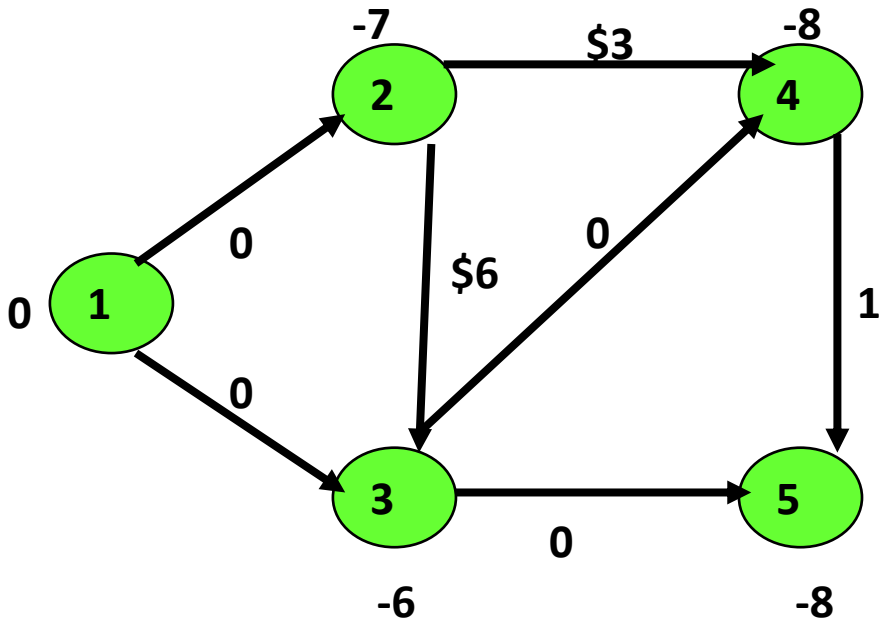
# Send Flow to another Demand Node




**Send 11 flow from node 1 to node 5**



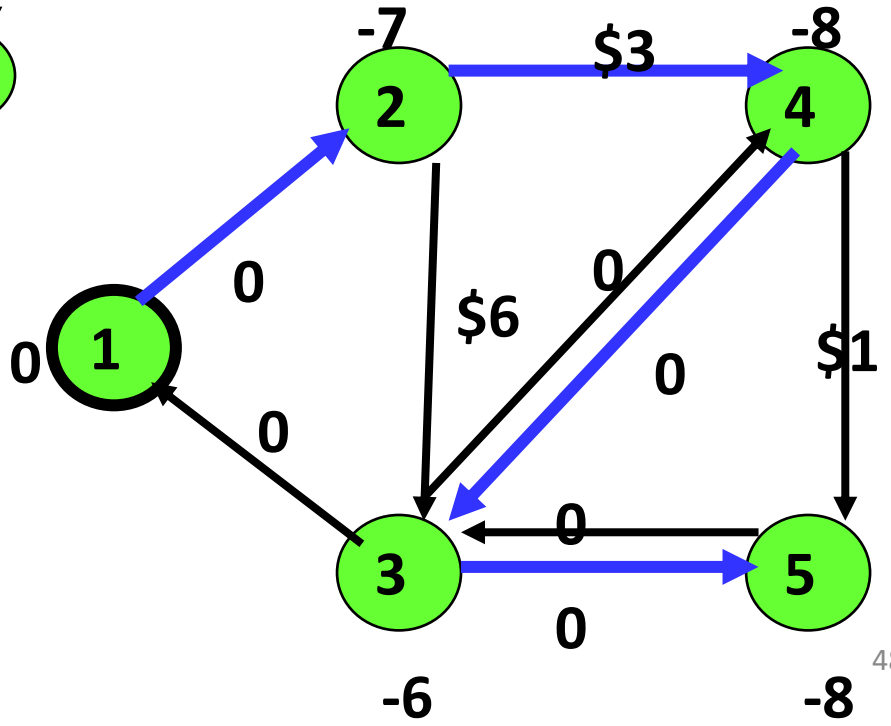
# Select a supply node and find the shortest paths




The values on the nodes are the current node potentials.

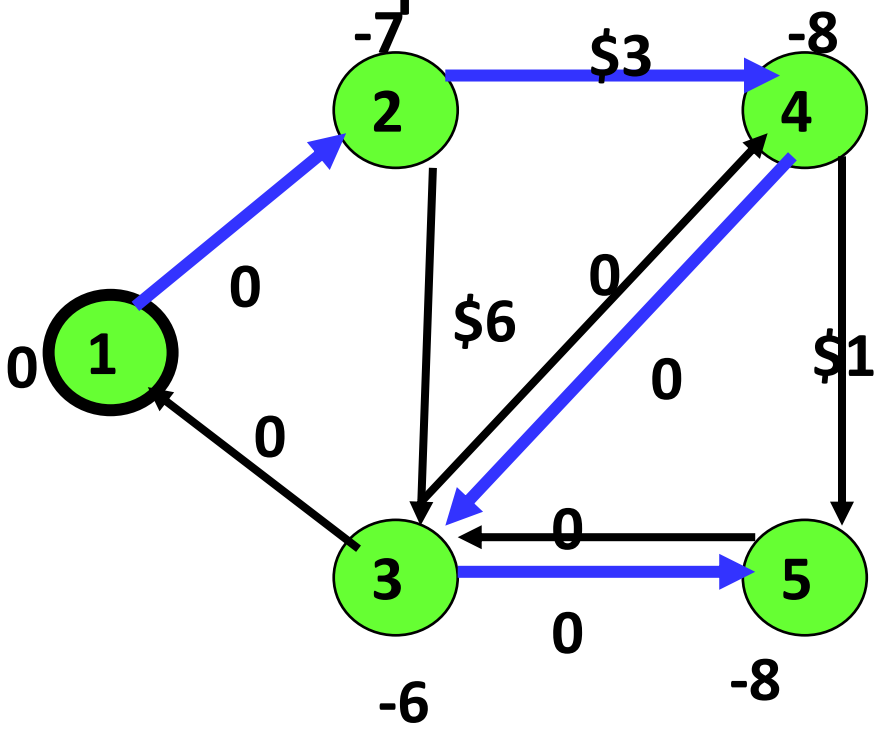
The direction of arcs follows the updated residual network.

The reduced cost does not change after flow sending.




The shortest path tree is marked in **bold and blue** (still from node 1).

# Update the node potentials

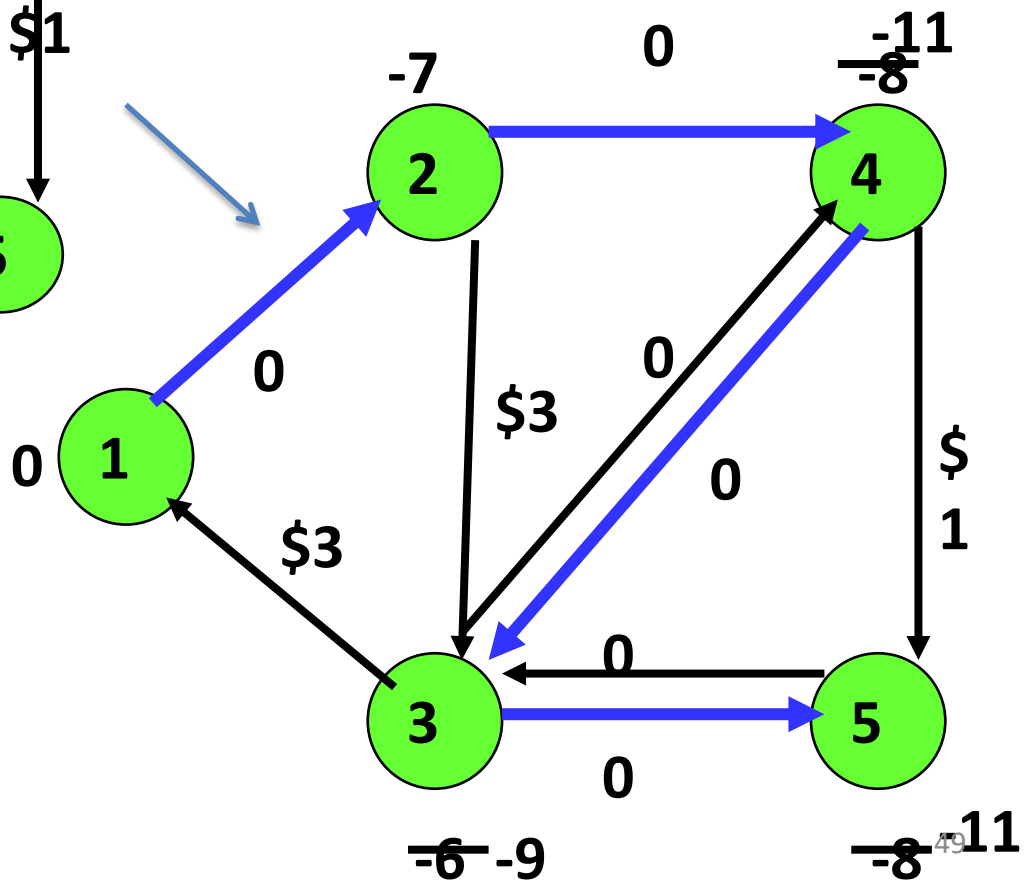



*Update node potentials:*

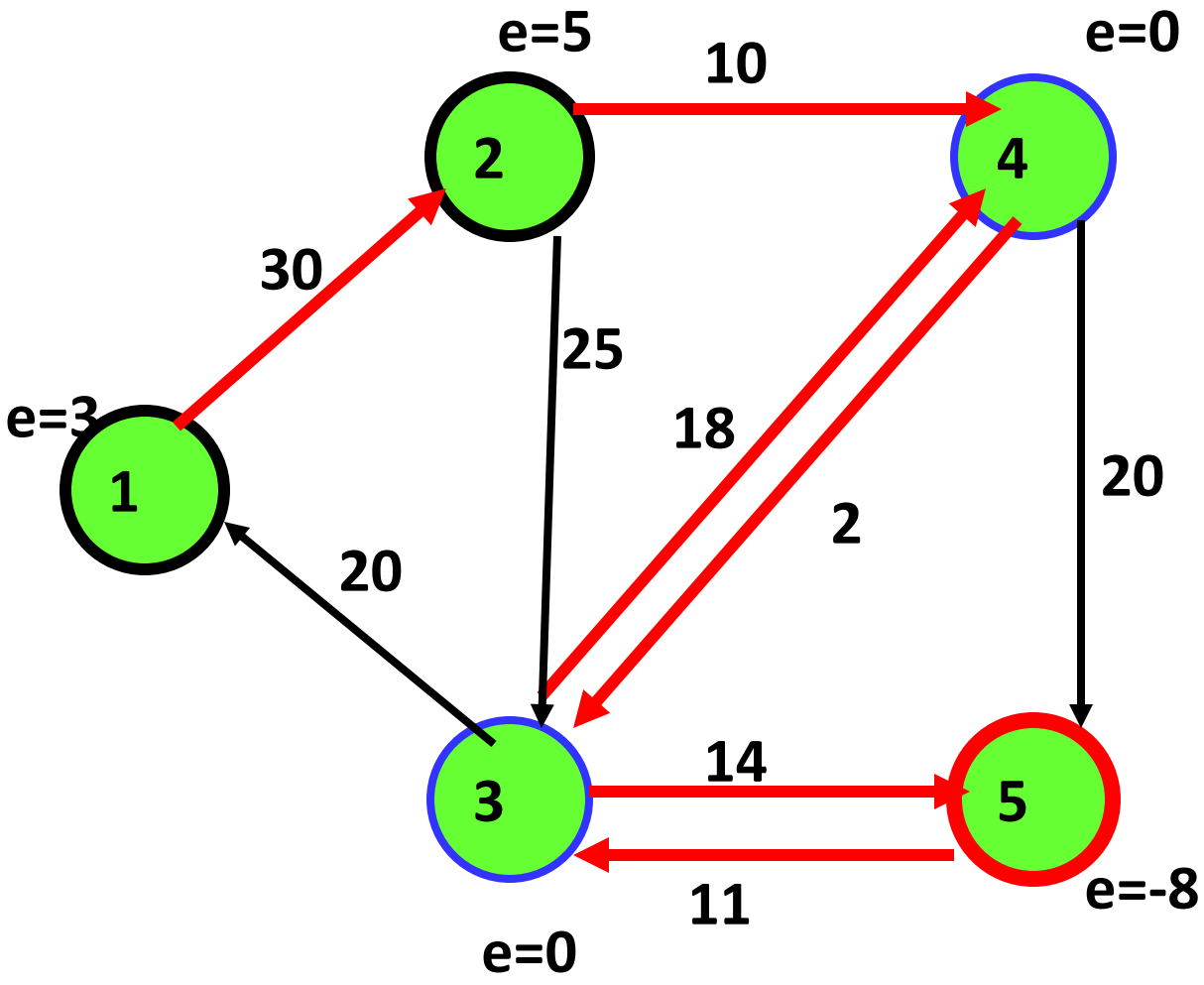
$\pi(j)=\pi(j)-d(j)$

*Update arc reduced cost*

$c^{\pi'}_{ij} = c^{\pi}_{ij} + d(i) - d(j)$



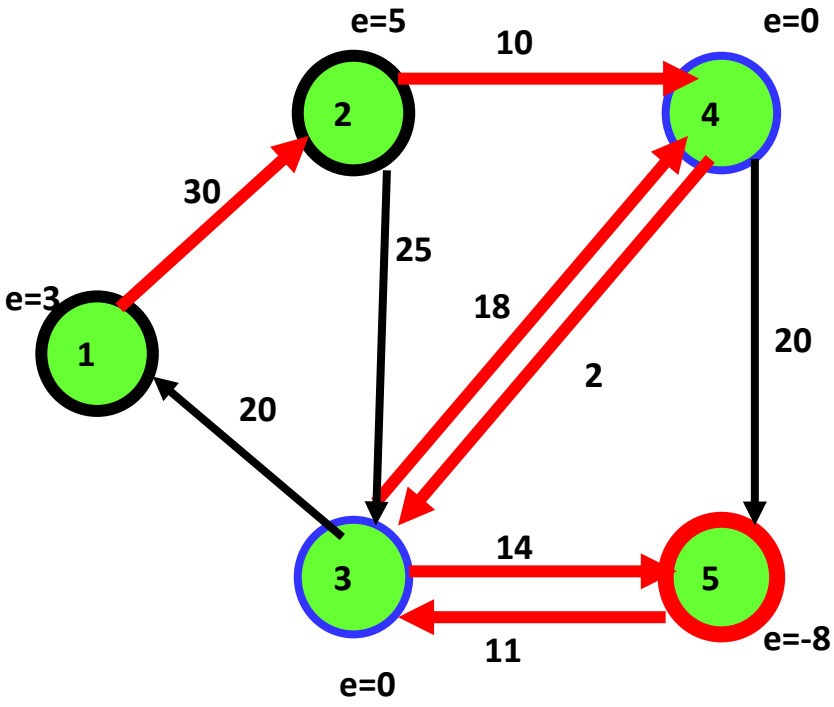
# Send Flow to a Demand Node along shortest path



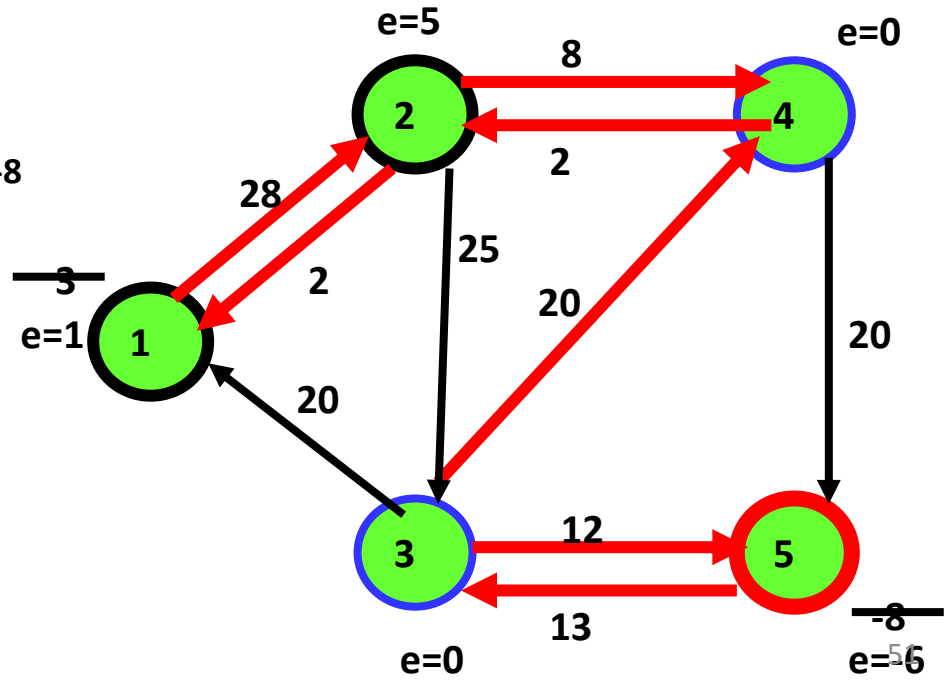

**Send flow from node 1 to node 5**

**How much flow will be sent? 2**

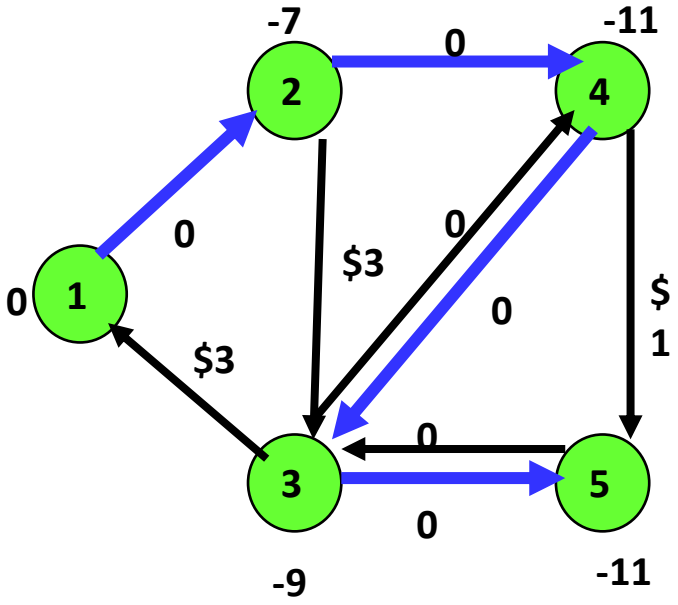
# Update the Residual Network

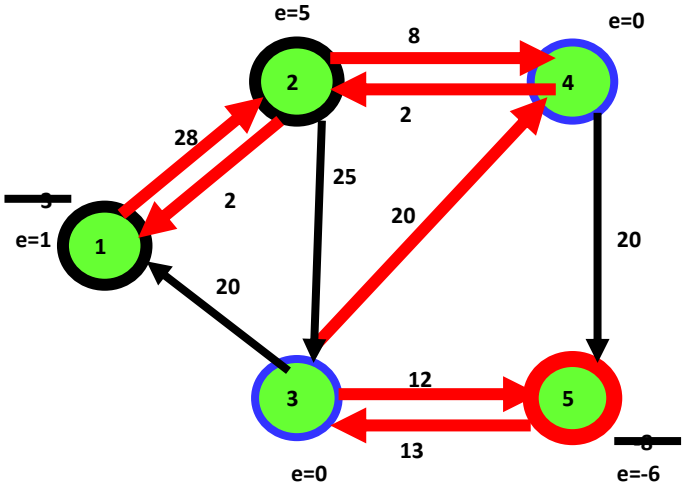



**2 units of flow were sent from node 1 to node 5**

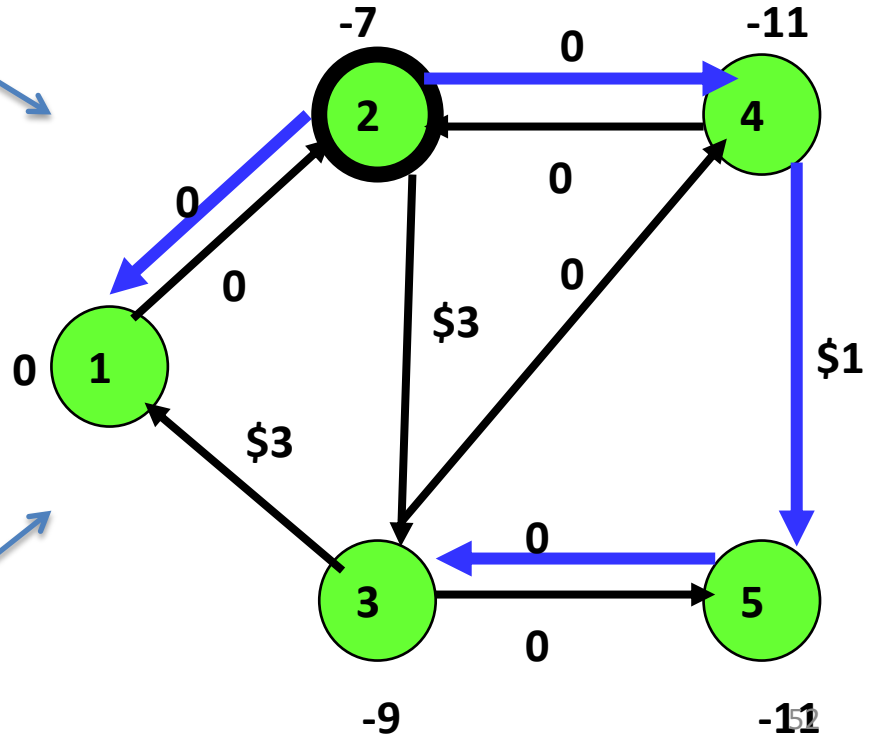


# Select a supply node and find the shortest paths

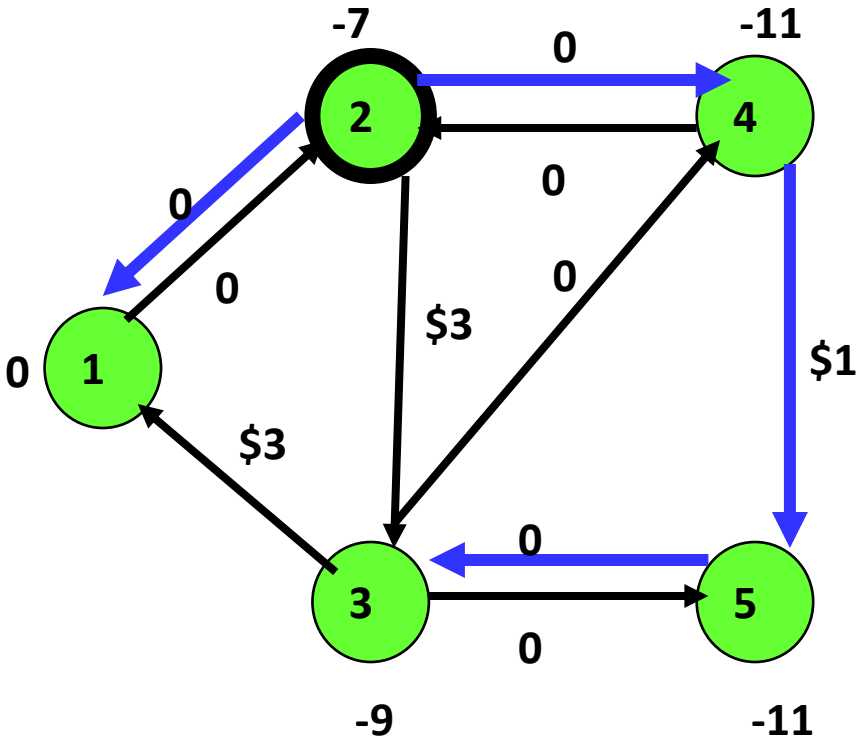



The shortest path tree (from node 2) is marked in **bold and blue.**



# Update the Node Potentials and the Reduced Costs

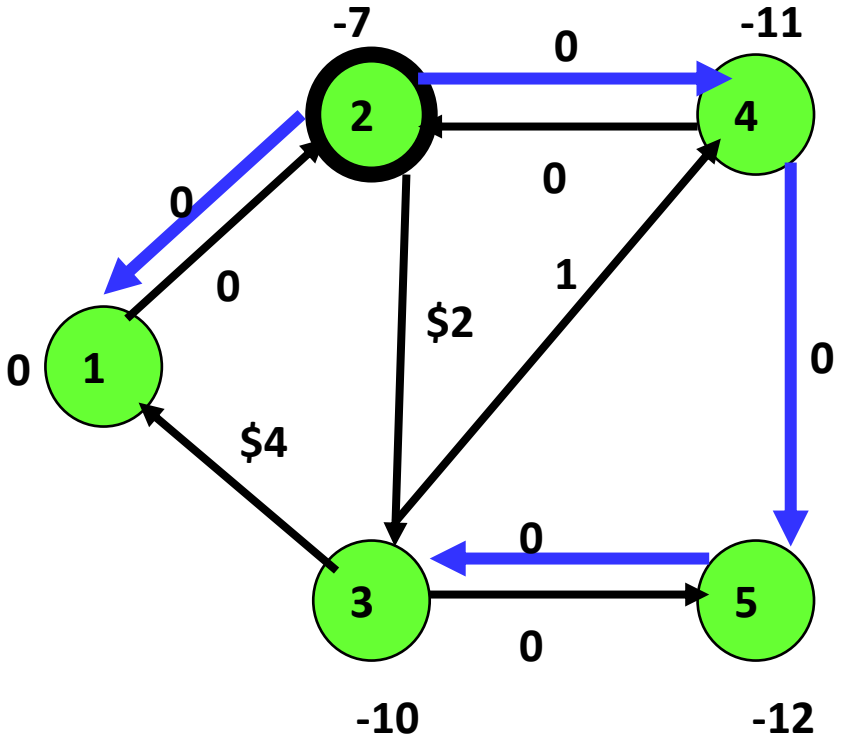



*Update node potentials:*

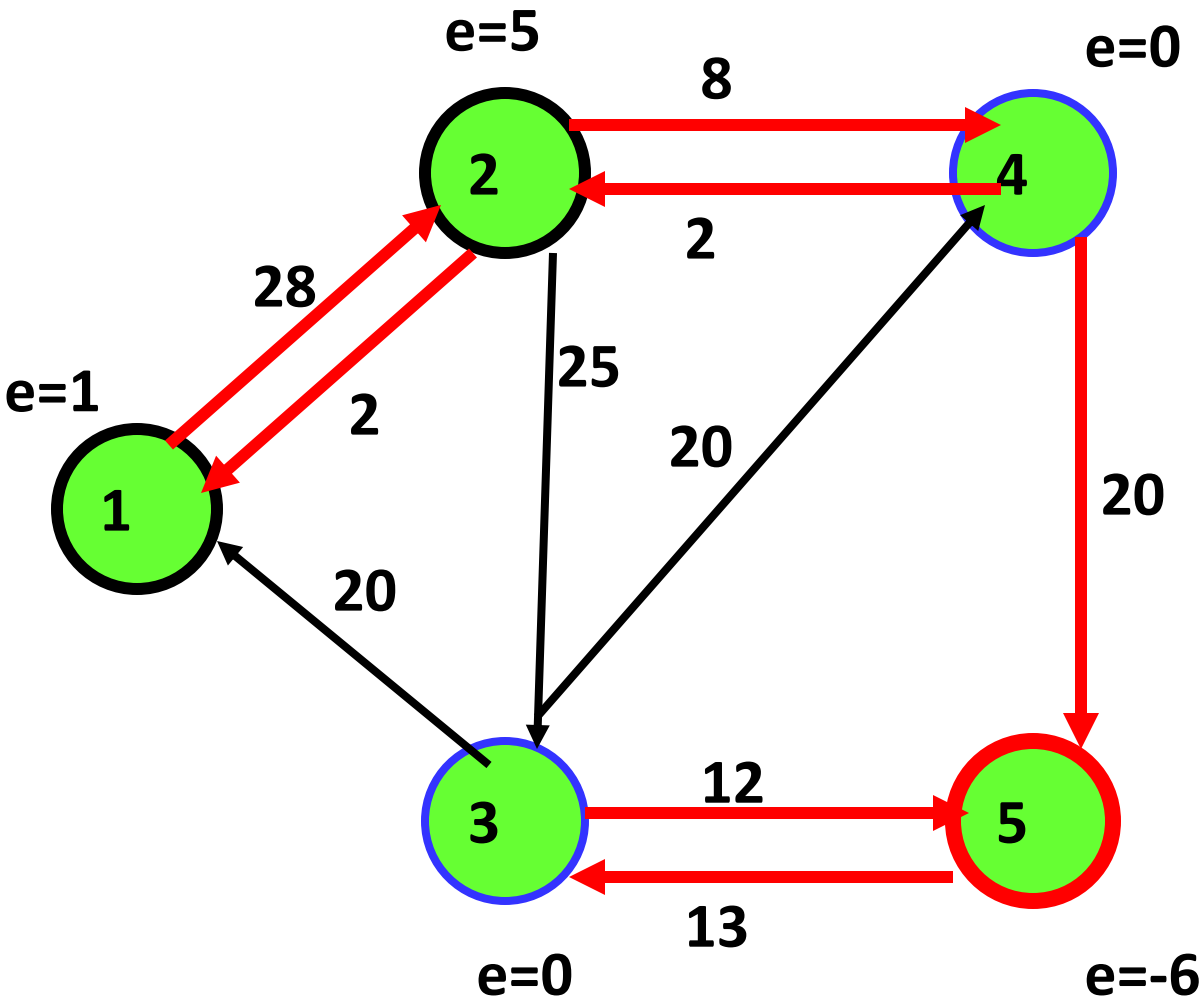
$\pi(j)=\pi(j)-d(j)$

*Update arc reduced cost*

$c^{\pi'}_{ij} = c^{\pi}_{ij} + d(i) - d(j)$



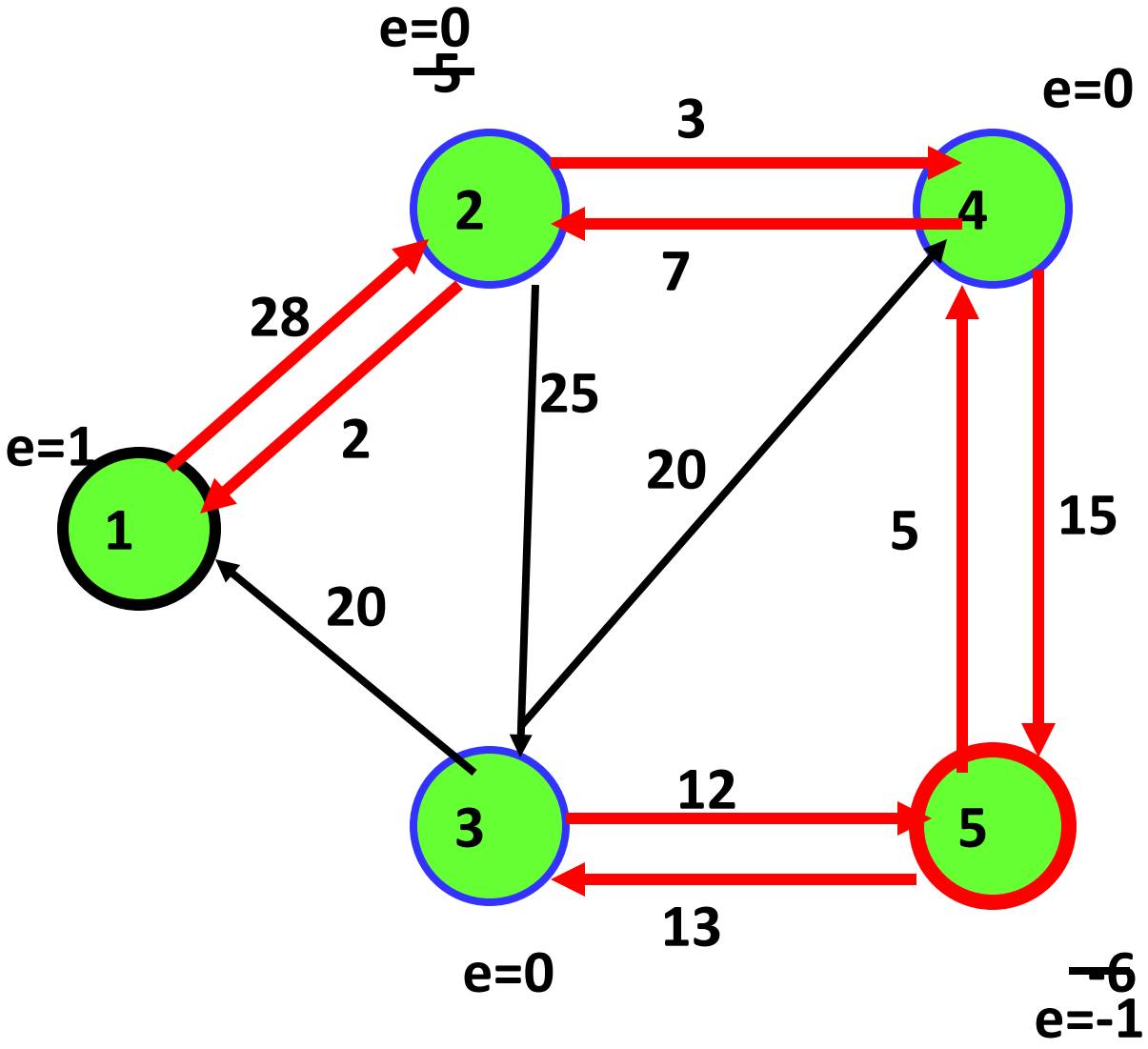
# Send Flow to a Demand Node




**Send flow from node 2 to node 5**

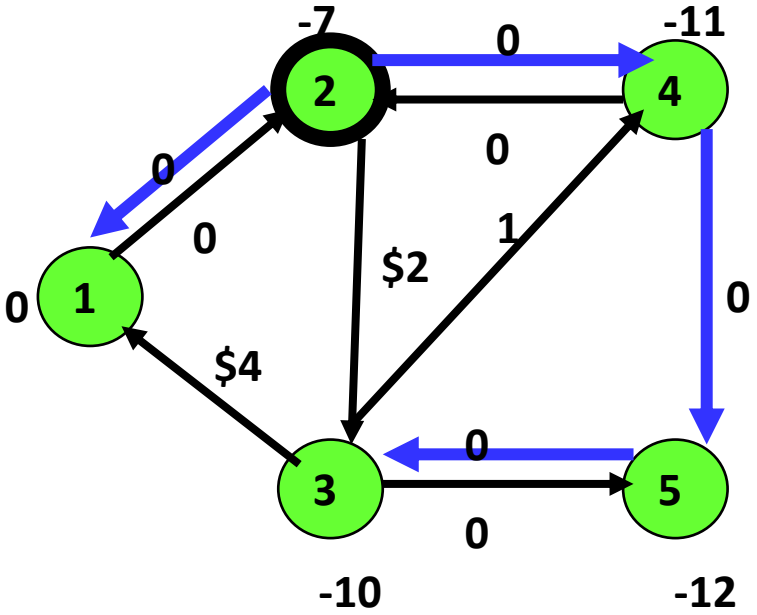
**How much flow can be sent?**

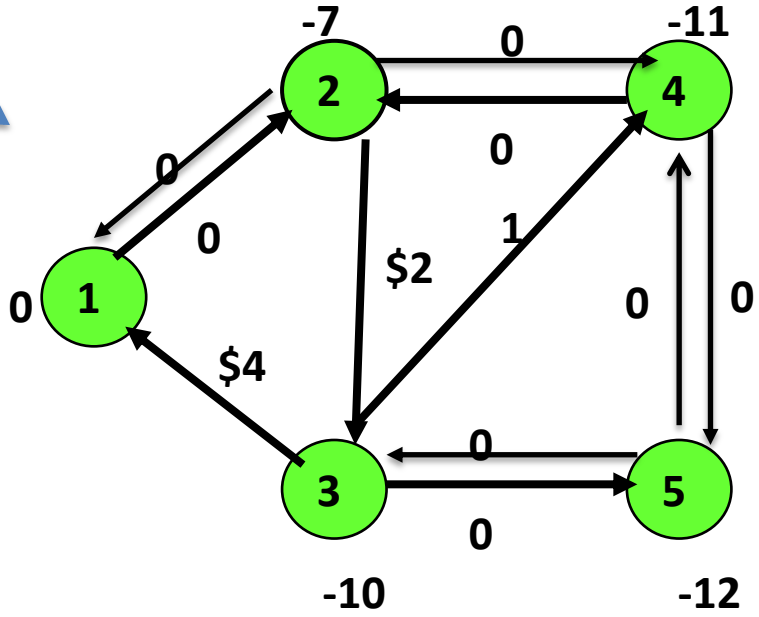
# Update the Residual Network

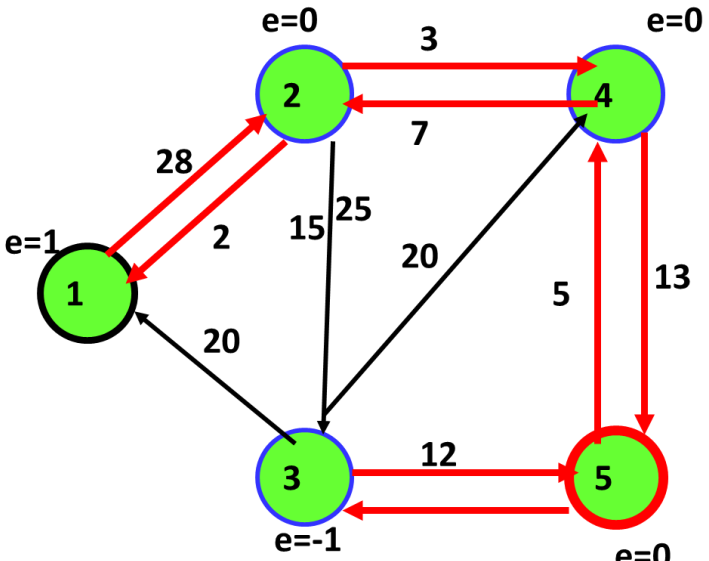



**5 units of flow were sent from node 2 to node 6.**

# Update the Node Potentials and the Reduced Costs

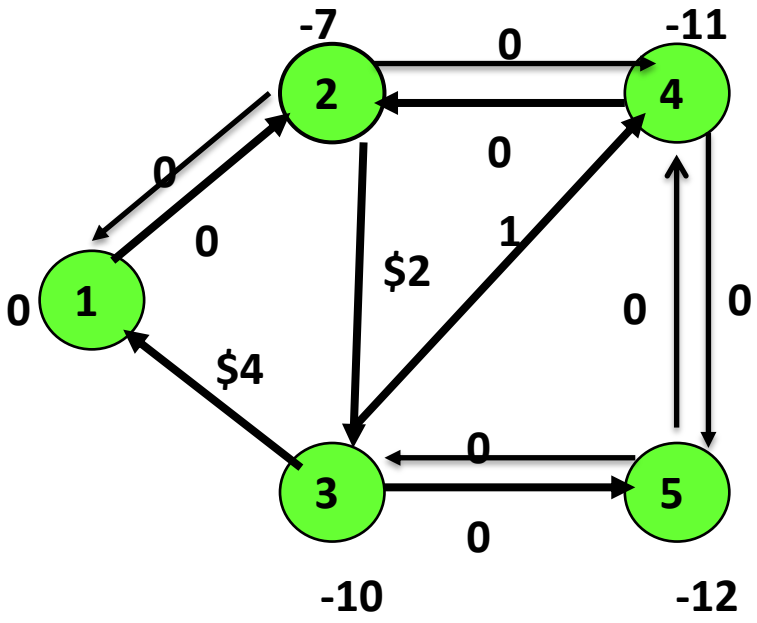



Select node 1

# Update the Node Potentials and the Reduced Costs

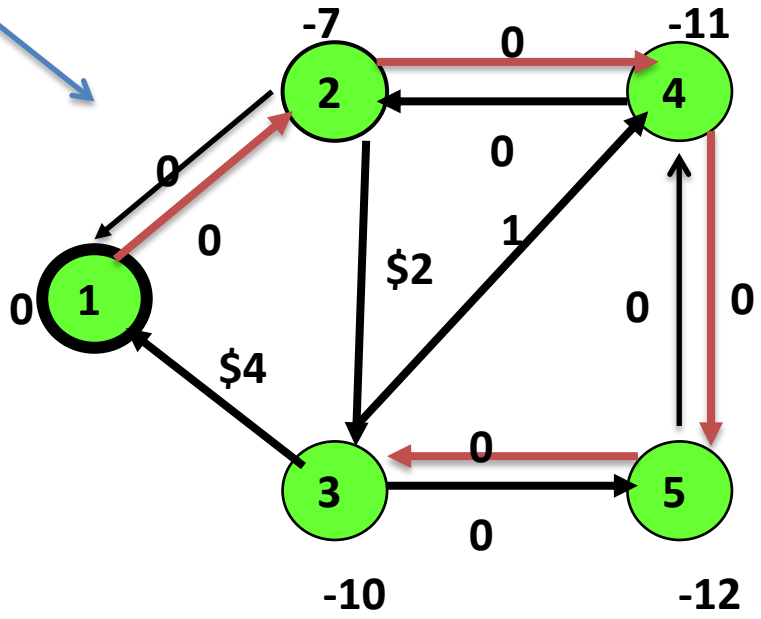



*Update node potentials:*

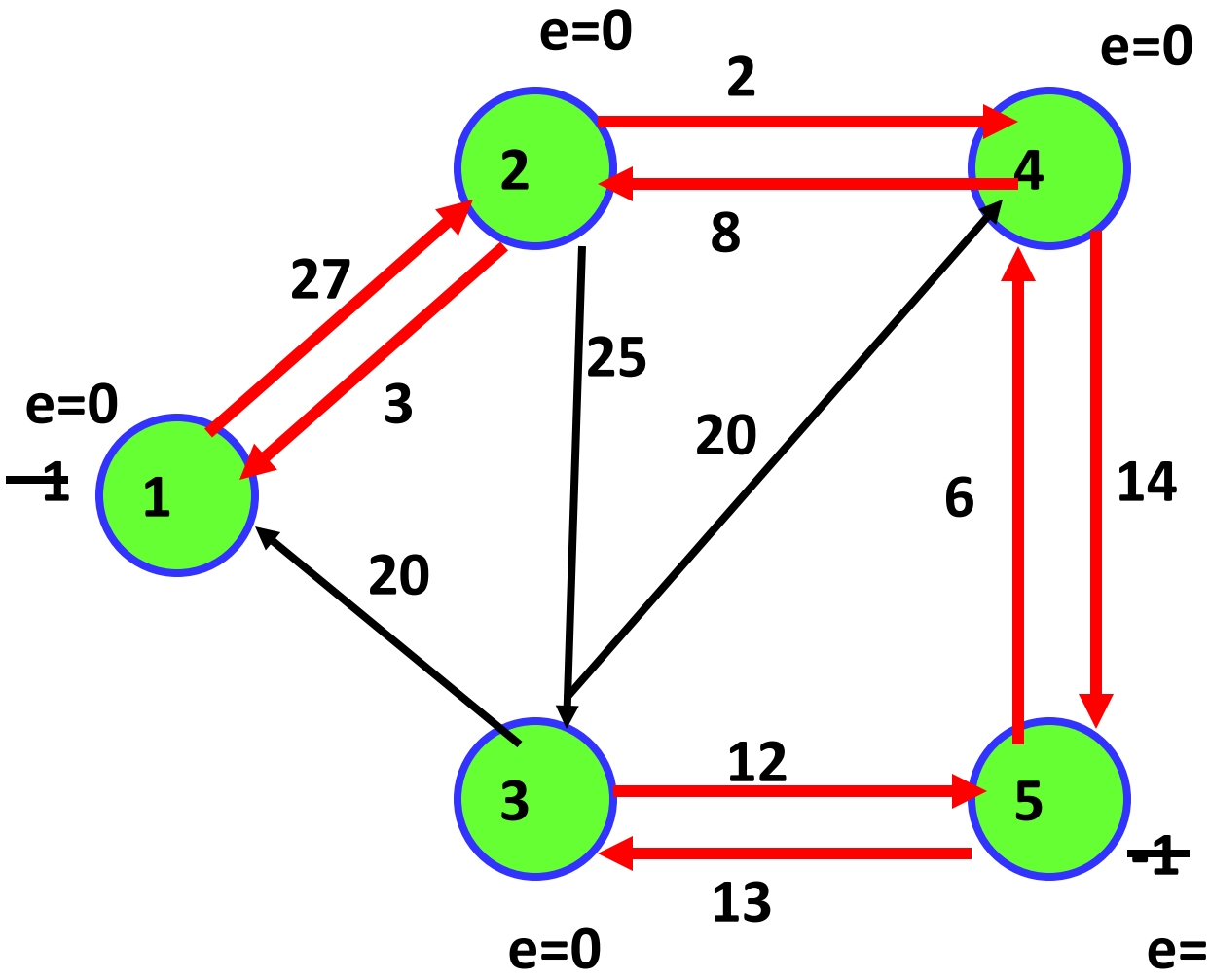
$\pi(j)=\pi(j)-d(j)$

*Update arc reduced cost*

$c^{\pi'}_{ij} = c^{\pi}_{ij} + d(i) - d(j)$



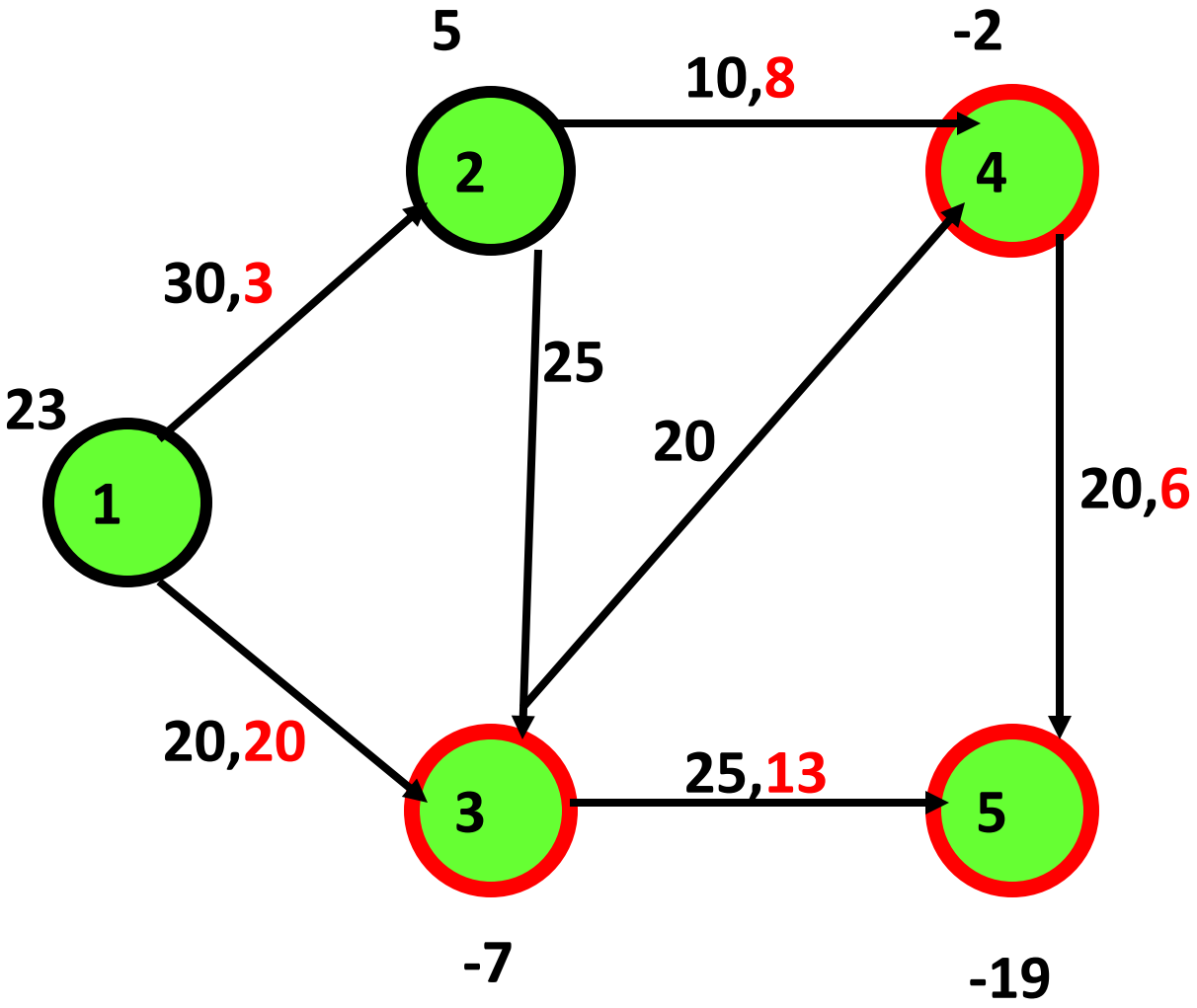
# Update the Residual Network



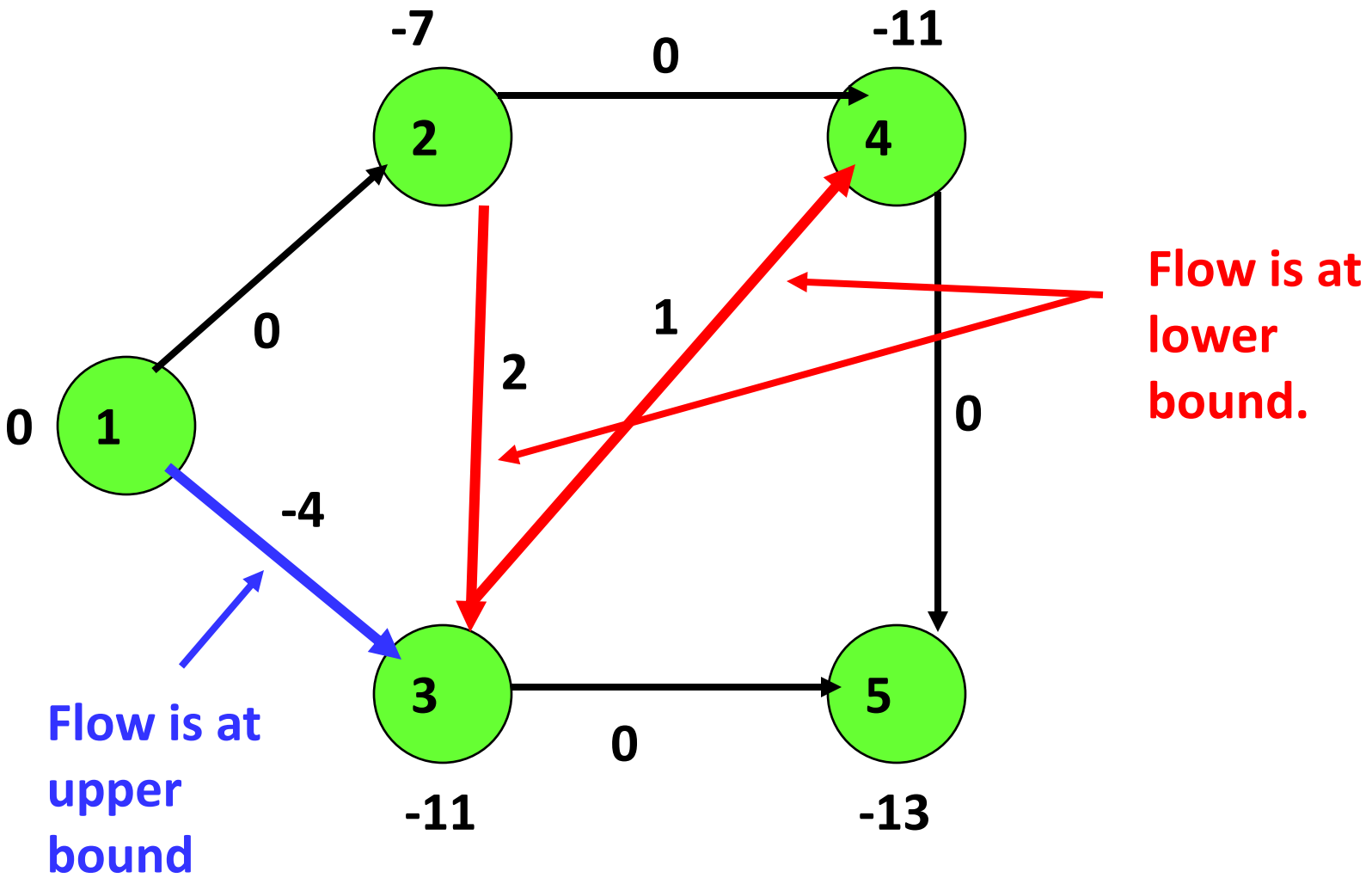

1 unit of flow was sent from node 1 to node 5.

The resulting flow is feasible, and also optimal.

# The Final Optimal Flow



# The Final Optimal Node Potentials and the Reduced Costs



# Complexity Analysis

- The successive shortest path algorithm will terminate in finite steps
  - because each iteration reduce a positive e(i) for some node *i* by at least 1
  - at most $O(nU)$ iterations
- Each iteration needs to solve a shortest path problem in time $S(n,m,nC)$
- Overall time complexity in $O(nU \cdot S(n,m,nC))$
  - Pseudo polynomial time

# Comparing of two algorithms

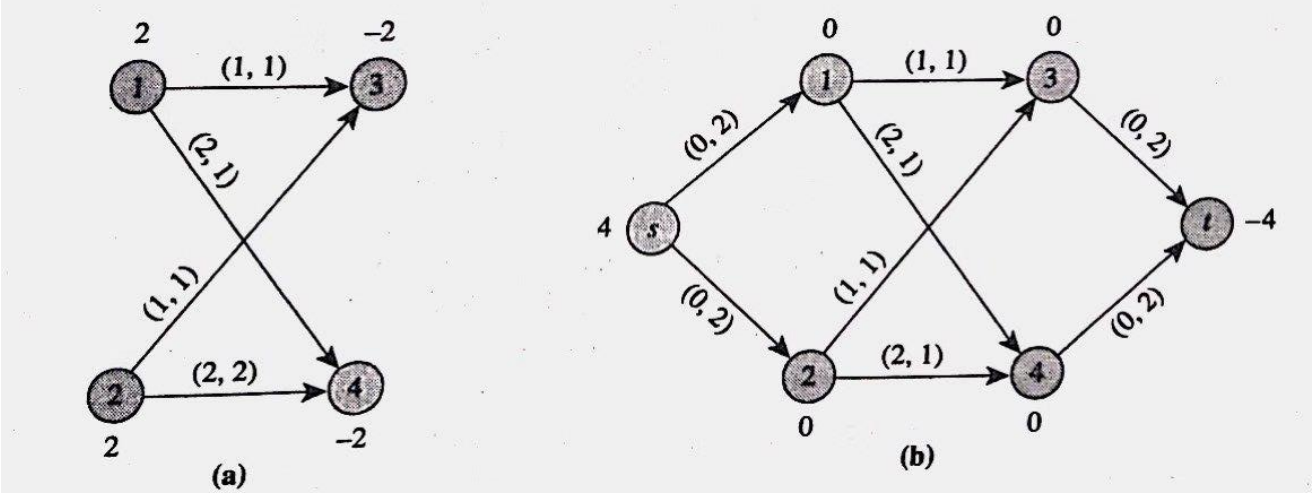
- Cycle canceling algorithm
  - Always maintain feasibility, approach optimality
  - $O(nm^2CU)$
- Successive shortest path algorithm
  - Always maintain reduced cost optimality, approach feasibility
  - $O(nU \cdot S(n,m,nC))$

# Primal-Dual Algorithm

- The primal-dual algorithm is similar to the successive shortest path algorithm in the sense that
  - It also maintain a pseudo flow that satisfies the reduced cost optimality conditions;
  - It also gradually converts the pseudo flow into a flow by augmenting flows along shortest paths.
- But the primal-dual algorithm generally transforms the minimum cost flow problem into a maximal flow problem with a single excess node and a single deficit node.

# Transformed network

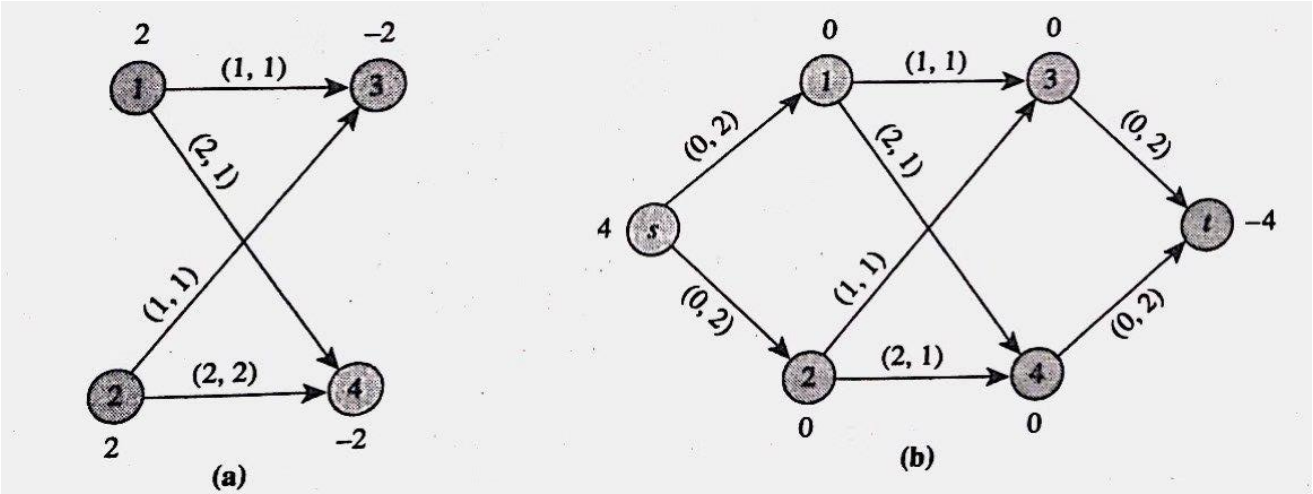
- Introduce a dummy source node s and a dummy sink node t:
  - For each node i with b(i)>0, add a zero cost arc (s,i) with capacity b(i)
  - For each node i with b(i)<0, add a zero cost arc (i,t) with capacity –b(i).
  - Set b(s)=$\sum_{\{i:b(i)>0\}} b(i)$, b(t)=-b(s), and b(i)=0 for all i∈ $N$

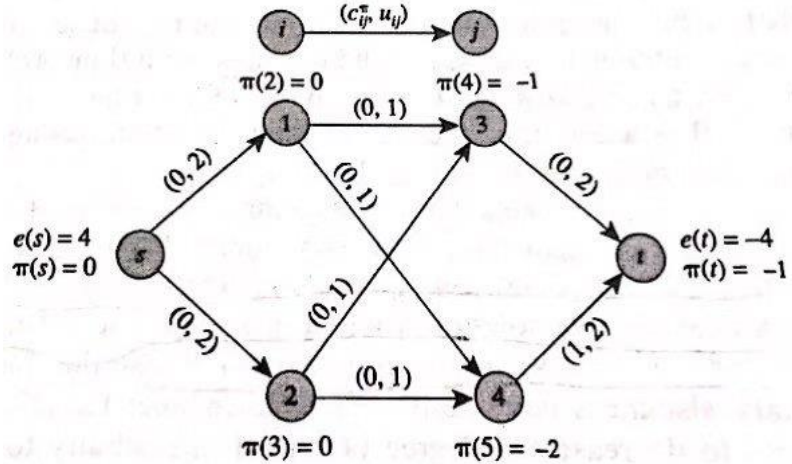


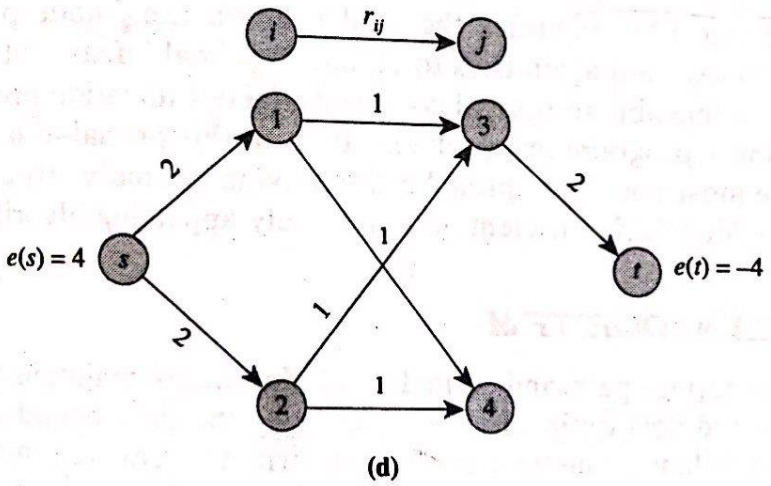
# Primal-Dual Algorithm

- Preprocess
  - Let $x$=0, π=0, e(s)=b(s) and e(t)=b(t)
- While e(s)>0 do
  - Determine the shortest path $d(j)$ from node s to all other nodes $j$ with respect to reduced cost on G($x$)
  - Update π($j$)=π($j$)-$d(j)$ for all node $j$
  - Define the admissible network $G^o$(x): it only contain arcs in G(x) with zero reduced cost, the capacity of each arc in $G^o$(x) is the same as that in G(x)
  - Establish a maximum flow from node s to node t in $G^o$(x)
  - Update $x$, G($x$), and e(s),e(t)

# Example


2
−2
1
3
(1, 1)
(2, 1)
(1, 1)
2
(2, 2)
4
2
−2
(a)
0
0
1
3
(1, 1)
(0, 2)
(2, 1)
(0, 2)
4
s
t
−4
(0, 2)
(1, 1)
(0, 2)
2
(2, 1)
4
0
0
(b)


$(c_{ij}^{\pi}, u_{ij})$
i
j
π(2) = 0
π(4) = −1
1
(0, 1)
3
(0, 2)
(0, 2)
(0, 1)
e(s) = 4
π(s) = 0
s
t
e(t) = −4
π(t) = −1
(0, 2)
(0, 1)
(1, 2)
2
(0, 1)
4
π(3) = 0
π(5) = −2


i
$r_{ij}$
j
1
1
3
2
2
e(s) = 4
s
1
t
e(t) = −4
2
1
2
1
2
4
(d)

# discussion

- The primal dual algorithm guarantees that the excess of node s strictly decreases at each iteration.
- If *S(n, m,* C) and *M(n, m, U)* denote the solution times of shortest path and the maximum flow algorithms, the primal-dual algorithm has an overall complexity of $O(\min\{nU, nC\} \cdot \{S(n, m, nC) + M(n, m, U)\})$.