交通地理信息系统

Geographic information system for Transportation (GIS-T)

主讲：徐占东 讲师

zhandong.xu@swjtu.edu.cn

标号法和最短路算法

- 网络表示符号
- 最优化条件
- 标号校验算法
- 标号设置算法

1.网络表示符号

网络表示符号

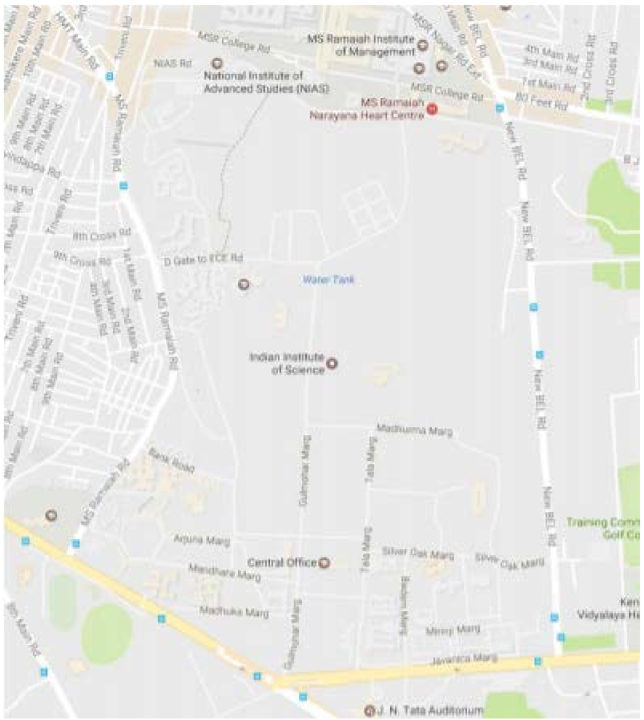
两个基本问题：

- 如何用数学和计算机语言来表示一个物理交通网络?

- 如何有效计算最短路径?

网络表示符号

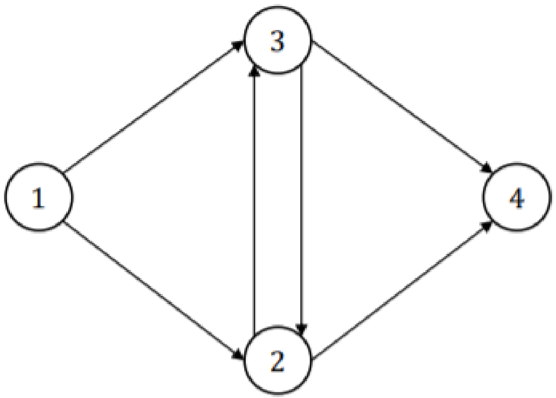
图是节点和弧的集合：



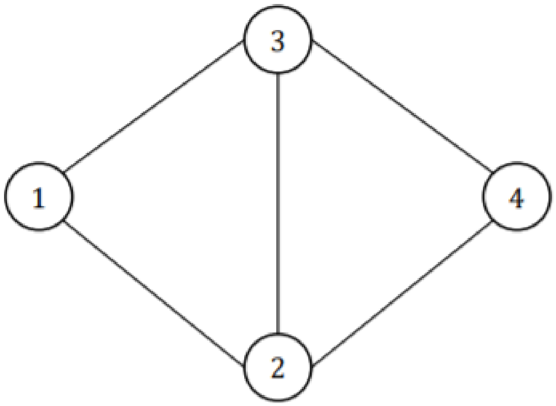
- 交通网络中的节点：道路交叉点或交叉路口
- 交通网络中的弧线：连接相邻路口的道路

网络表示符号

节点有时被称为顶点，弧也称为链接或边
图可以是有向的或无向的



有 向 图



无 向 图

交通网络一般使用有向图来表示，
无向图用于表示其他网络，例如社交网络

网络表示符号

- 我们用 $G=(N,A)$ 表示图， N 代表节点集，A代表弧集，弧 $(i,j)\in A$ 将尾节点 i 连接到头节点 j

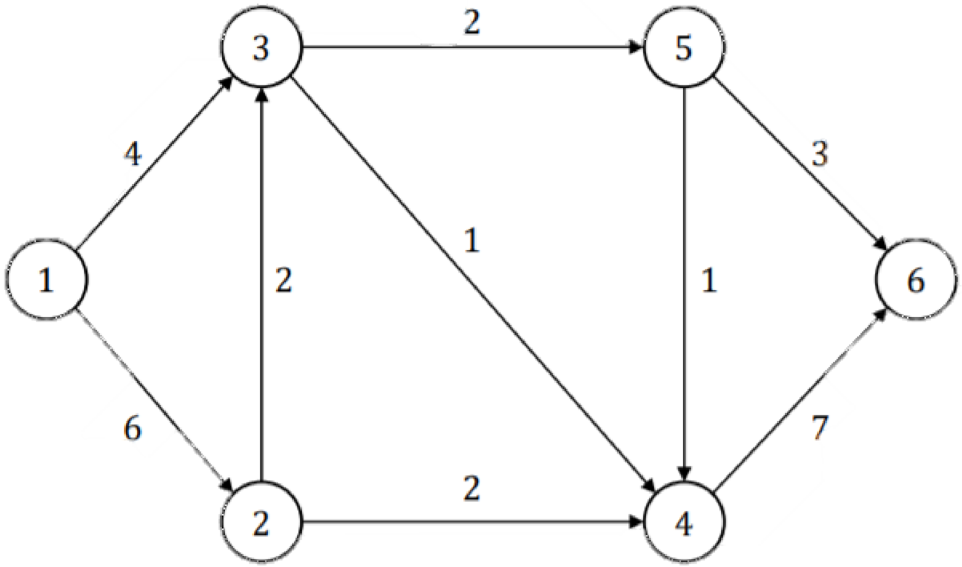
- 用 n 和 m 分别表示节点和弧的数量

- 对于节点 i，节点集合 $\{j:(i,j)\in A\}$ 称为下游节点，连接 i 到它们的弧称为下游弧（可以类似地定义上游节点和弧）

- 此外，图中的节点和弧可以具有如下属性：
  - **费用**
  - **时间**
  - **两点间交通需求**
  - **路口的固定成本（信号延迟）**

网络表示符号

下图有 6 个节点和 9 个弧，假设弧上数值表示行程时间



如何在计算机上存储网络拓扑信息?

网络表示符号

邻接矩阵

节点

$$
\begin{array}{c|cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 0 & 1 & 1 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 1 & 0 & 0 \\
3 & 0 & 0 & 0 & 1 & 1 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 1 \\
5 & 0 & 0 & 0 & 1 & 0 & 1 \\
6 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

节点

旅行时间

$$
\begin{array}{c|cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
1 & 0 & 6 & 4 & 0 & 0 & 0 \\
2 & 0 & 0 & 2 & 2 & 0 & 0 \\
3 & 0 & 0 & 0 & 1 & 2 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 7 \\
5 & 0 & 0 & 0 & 1 & 0 & 3 \\
6 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

优点：弧的连接关系通过**非0元素的行列索引**直接反映出来

缺点：**高稀疏性**（存储内存大）

网络表示符号

关联矩阵

弧

节点

$$
\begin{array}{cc}
 & \begin{array}{ccccccccc} (1,2) & (1,3) & (2,3) & (2,4) & (3,4) & (3,5) & (4,6) & (5,4) & (5,6) \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} & \left(\begin{array}{ccccccccc}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & -1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & -1 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1
\end{array}\right)
\end{array}
$$

旅行时间可以存储在向量中（向量模=弧数）

优点：具有特殊的结构（每列中正好有一个 +1 和 -1），有什么好处?

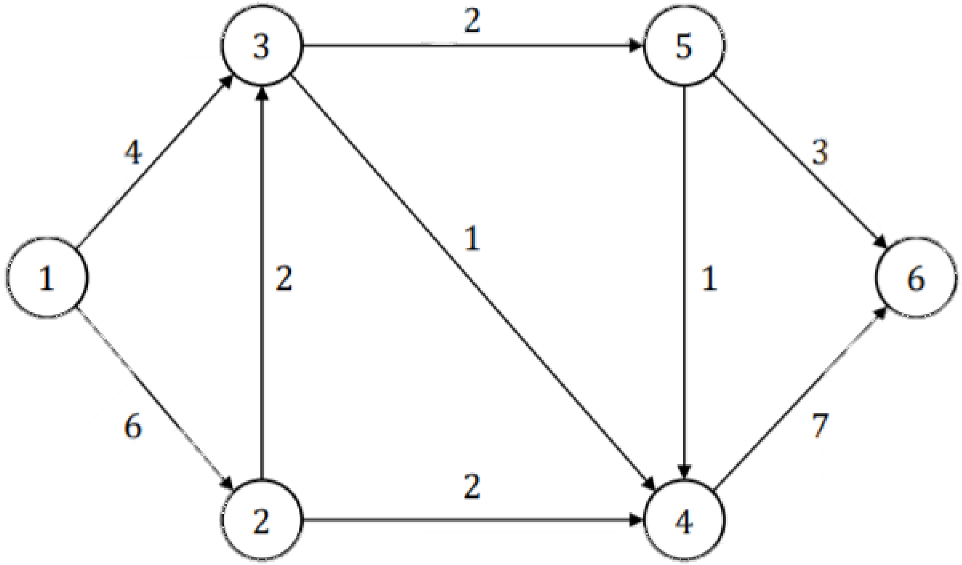
缺点：**高稀疏性**

网络表示符号

邻接表:

1: 2, 3
2: 3, 4
3: 4, 5
4: 6
5: 4, 6
6:

1: (2,6), (3,4)
2: (3,2), (4,2)
3: (4,1), (5,2)
4: (6,7)
5: (4,1), (6,3)
6:



优点：数据紧凑、无冗余

缺点：检索特定弧需要查询邻接表，成本较高

2.最优化条件

最优化条件

最短路问题：

- 最短路径问题是找到从原点 r 到一个节点的最优距离/时间/成本（和路径）。
- 令 μ_i 表示距离标记，它表示从源 r 到节点 i 的路径成本，令$\mu_r = 0$。
- 将$\{\mu_i, \forall i\}$视为决策变量，如何定义最优化的充分必要条件?

标号的充分和必要条件

命题（必要条件）

如果标记为 μ的向量是最短路径距离：

$$\mu_j \le \mu_i + t_{ij} \ \forall\ (i, j) \in A$$

证明

反证法！

标号的充分和必要条件

命题（充分条件）

标号 μ表示从 r 到不同节点的路径长度最短，且满足 $\mu_j \le \mu_i + t_{ij}\ \forall (i,j) \in A$

证明

令 $P = r = i_1 - i_2 - \ldots - i_k = s$ 是从 r 到 s 的任意路径。令它的长度为 μ_s 。由于 μ_s 满足以下不等式，

$$
\begin{aligned}
\mu_{i_k} &\le \mu_{i_{k-1}} + t_{i_{k-1}i_k} \\
\mu_{i_{k-1}} &\le \mu_{i_{k-2}} + t_{i_{k-2}i_{k-1}} \\
&\vdots \\
\mu_{i_2} &\le \mu_{i_1} + t_{i_1 i_2}
\end{aligned}
$$

累加上面的不等式，$\mu_s \le \mu_r + \Sigma_{(i,j)\in P}\, t_{ij} = \Sigma_{(i,j)\in P}\, t_{ij}$。因此，$\mu_s$是从 r 到 s 的路径成本的下限。

路径的充分和必要条件

- 充分必要条件通常也称为**贝尔曼最优条件**
- 前面的命题处理距离标号的最优性，路径的最优条件是什么?

路径的充分和必要条件

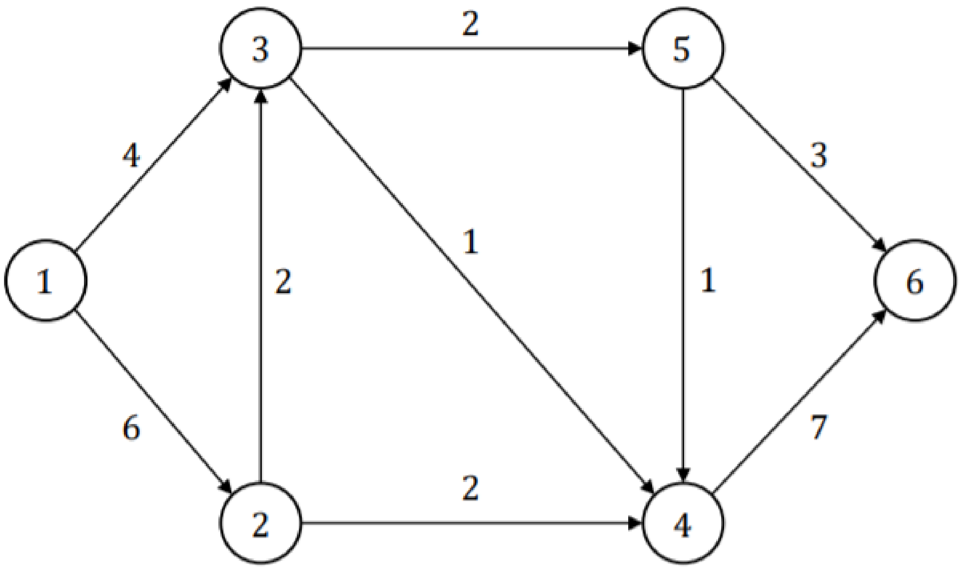
命题（充分和必要条件）

令 μ 表示最短路径距离的向量，当且仅当 $\mu_j = \mu_i + t_{ij} \forall (i,j) \in P$时，从 r 到 s 的路径 P 是最优的

证明非常相似，且使用了子路径最优性属性

线性规划公式

尝试使用优化模型建立节点 1 和 6 之间的最短路径



三要素：目标、决策变量和约束条件

线性规划公式

目标

最小化旅行时间

决策变量

二进制变量 x_{ij}：如果链接 (i, j) 属于最短路径，则为 1，否则为 0

约束条件

流量守恒：单位流量进入节点 1 并离开节点 6；当它通过中间节点时，**“进去多少=出来多少”**

线性规划公式

目标函数：$\min 4x_{13} + 6x_{12} + 2x_{23} + \ldots + 7x_{46} + 3x_{56}$

流量约束条件：

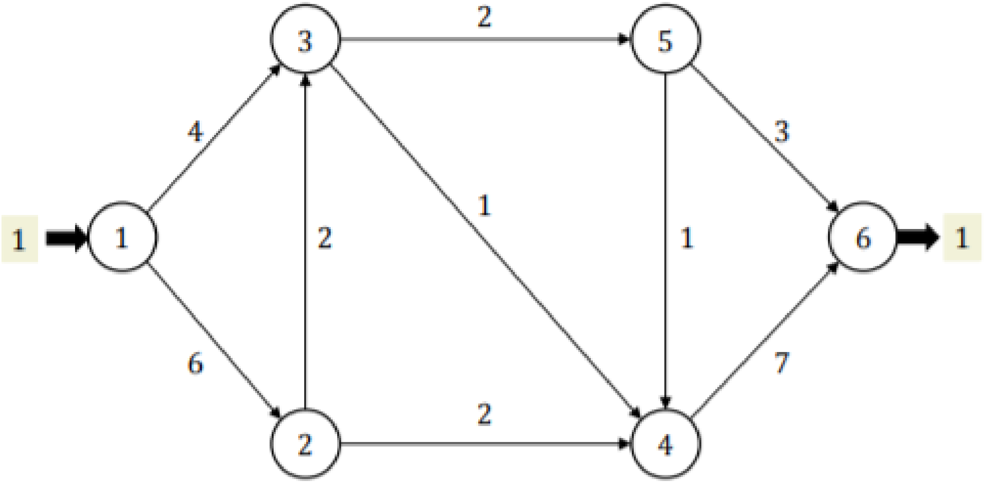
节点 1: $x_{12} + x_{13} = 1$

节点 2: $x_{12} = x_{23} + x_{24}$

节点 3: $x_{13} + x_{23} = x_{34} + x_{35}$

节点4和5？？

节点 6: $x_{46} + x_{56} = 1$



0、1变量约束：$x_{ij} \in \{0,1\} \, \forall \, (i,j) \in A.$

线性规划公式

等式约束**标准化**：

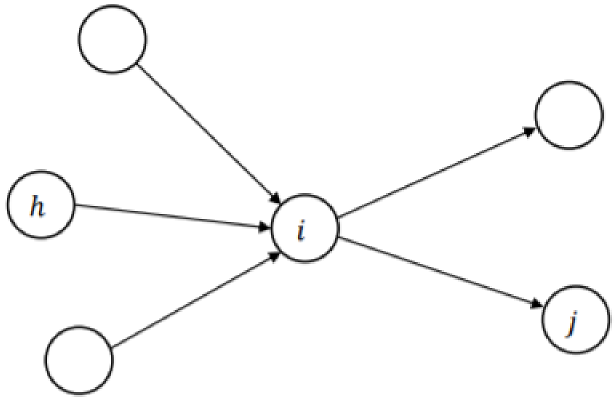
$$
\begin{aligned}
x_{12} + x_{13} &&&&&&&&& = 1 \\
-x_{12} && +x_{23} + x_{24} &&&&&&& = 0 \\
- x_{13} && -x_{23} && +x_{34} + x_{35} &&&&& = 0 \\
&& - x_{24} && -x_{34} && +x_{46} - x_{54} &&& = 0 \\
&&&& - x_{35} && + x_{54} && +x_{56} & = 0 \\
&&&&&& -x_{46} && -x_{56} & = -1
\end{aligned}
$$

约束条件的结构（**关联矩阵**）

$$
\begin{pmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & -1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & -1 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1
\end{pmatrix}
\begin{pmatrix}
x_{12} \\ x_{13} \\ x_{23} \\ x_{24} \\ x_{34} \\ x_{35} \\ x_{46} \\ x_{54} \\ x_{56}
\end{pmatrix}
$$

线性规划公式

广义表达形式：



等式约束：

$$\sum_{j:(i,j)\in A} x_{ij} - \sum_{h:(h,i)\in A} x_{hi} = \begin{cases} 1 & \text{若} i = r \\ -1 & \text{若} i = s \\ 0 & \text{其他情况} \end{cases}$$

线性规划公式

最短路径问题可表达为**线性整数规划模型**:

$$\min \sum_{(i,j)\in A} t_{ij}x_{ij}$$

$$\text{s.t.} \quad \sum_{j:(i,j)\in A} x_{ij} - \sum_{h:(h,i)\in A} x_{hi} = \begin{cases} 1 & \text{if } i = r \\ -1 & \text{if } i = s \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} \in \{0,1\} \, \forall \, (i,j) \in A$$

回想一下，等式约束可以写成 $Ax = b$，其中 A 是 $n \times m$ 矩阵，x 是 $m \times 1$ 向量，b 是 $n \times 1$ 向量

线性规划公式

- 由于节点弧关联矩阵 A的每列中恰好有一个 +1 和一个 -1，因此满足称为总单模性的属性
- 如果约束矩阵具有这个性质，可以用不等式代替整数约束，将问题作为一般**线性规划**求解，得到整数最优解

$$\min \sum_{(i,j)\in A} t_{ij}x_{ij}$$

$$\text{s.t.} \quad \sum_{j:(i,j)\in A} x_{ij} - \sum_{h:(h,i)\in A} x_{hi} = \begin{cases} 1 & \text{if } i = r \\ -1 & \text{if } i = s \\ 0 & \text{otherwise} \end{cases}$$

$$0 \le x_{ij} \le 1 \, \forall (i,j) \in A$$

请注意，上界 $x_{ij} \le 1 \, \forall (i,j) \in A$ 是多余的

线性规划公式

将线性规划模型转换为标准形式，并写出最短路径问题的Karush-Kuhn-Tucker (KKT)最优化条件！

线性规划公式

构建拉格朗日函数:

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \sum_{(i,j)\in A} t_{ij}x_{ij} + \sum_{(i,j)\in A} \lambda_{ij}(-x_{ij}) + \sum_{i\in N} \mu_i \left(\sum_{j:(i,j)\in A} x_{ij} - \sum_{h:(h,i)\in A} x_{hi} - b_i \right)$$

$$\frac{\partial \mathcal{L}}{\partial x_{ij}} = t_{ij} - \lambda_{ij} + \mu_i - \mu_j = 0$$
$$\Rightarrow \lambda_{ij} = t_{ij} + \mu_i - \mu_j$$

λ_{ij} 也称为降低成本（reduced cost）

原变量约束:

$$\boldsymbol{Ax} = \boldsymbol{b}$$

$$x_{ij} \geq 1 \ \forall \ (i,j) \in A$$

乘子变量约束:

$$\lambda_{ij} \geq 0 \ \forall \ (i,j) \in A$$

互补松弛约束:

$$\lambda_{ij} x_{ij} = 0 \ \forall \ (i,j) \in A$$

拉格朗日梯度为零约束:

$$\lambda_{ij} = t_{ij} + \mu_i - \mu_j \ \forall, (i,j) \in A$$

根据上述条件，将 μ_i 解释为距离标号，贝尔曼条件:

$\mu_j \leq t_{ij} + \mu_i$，如果$x_{ij} = 1 \Rightarrow \mu_j = \mu_i + t_{ij}$

线性规划公式

有许多有效的算法来求解一般的 LP（如经典的Simplex单纯性法）

对于网络问题，如何设计更有效的算法?

3.标号检验算法（Label correcting）

标号检验算法

回顾最短路径距离的最优条件：

$$\mu_j \leq tij + \mu_i \forall (i,j) \in A$$

最短路径算法的一般思路：

1.将除原点以外的所有节点的标号初始化为∞，原点的标号设置为 0

2.标号是最短距离的上限，通过迭代使得标号下降，直到满足上述最优条件

标号检验算法

- 在大多数算法中，在寻找从原点 r 到网络中某个节点的最短距离时，可以获得到所有其他节点的最短距离，被称为一对多算法；
- 无须存储路径，在终止时，利用前任标号回溯最优路径（最小生成树）
- 类似的，可以设计多对一算法，找到从所有节点到目的地 s 的最短路径

标号检验算法

在最简单的标号校正算法版本中：

- 扫描网络中的每条弧，如果违反最优性条件，则更新头节点的标号
- 重复直到没有弧违反最优性条件

通用标号校正(G,r)

STEP1.初始化

$\mu_r = 0, \pi_r = r$

$\mu_i = \infty, \pi_i = -1 \ \forall i \in N\{r\}$

STEP 2.而某些弧(i, j)满足$\mu_j > \mu_i + t_{ij}$时，

$\mu_j = \mu_i + t_{ij}$

$\pi_j = i$

结束循环

标号检验算法

可以证明通用标号校正方法有效

- **证明它终止**
- **当它终止时，证明满足最优性条件**

找到违反最优性条件的弧的一种选择是以**固定顺序扫描**所有弧
但这种方式并不“智能”

如何做的更好?

标号检验算法

与其在每次迭代中扫描所有弧，不如保留一个**可能违反最优性**的下游弧节点列表，将此称为扫描合格列表 (SEL)

如果不满足最优条件，则从该列表中选择一个节点 i 并更新其下游节点 j 的标号：$(i,j) \in A$。当 μ_j 减小时，会发生什么情况？ i 的

- **j 下游节点的标号**
- **j 上游节点的标号**

因此，如果有任何更新，将 j 添加到 SEL（总是这样？）

标号检验算法

修正标号校正(G,r)

Step 1: 初始化

$\mu_r = 0, \pi_r = r$

$\mu_i = \infty, \pi_i = -1 \ \forall\, i \in N\backslash\{r\}$

SEL = {r}

Step 2:

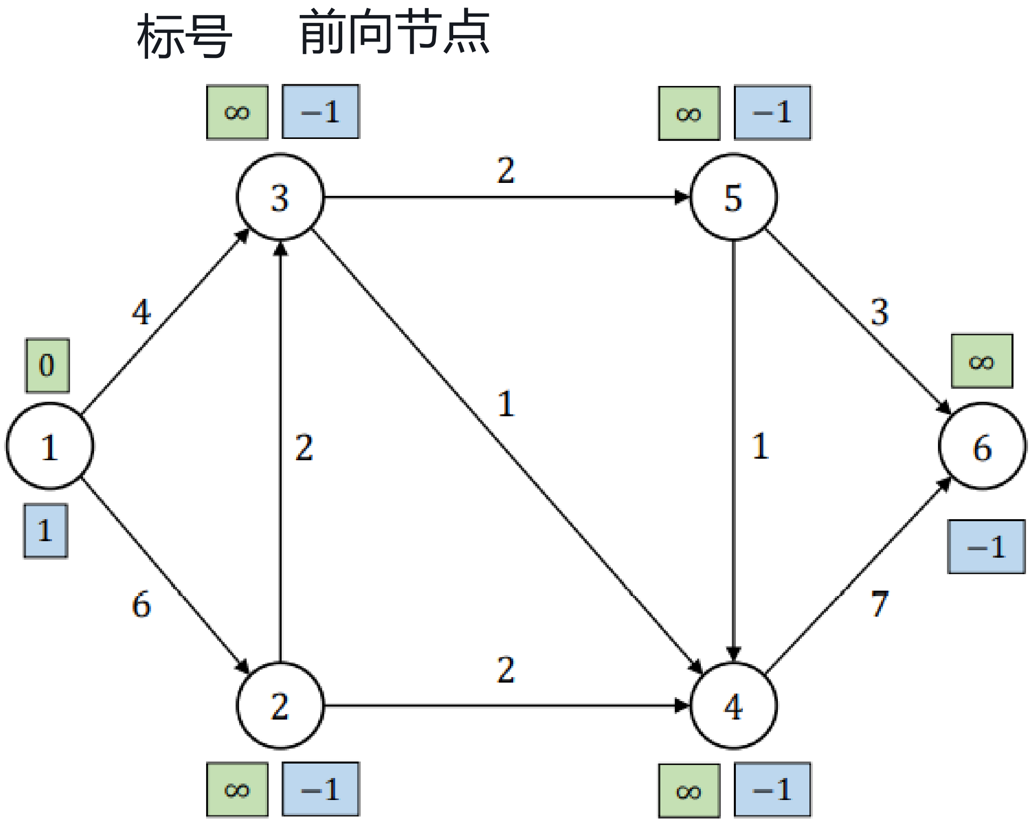
while SEL$\neq \emptyset$ **do**
  Remove i from SEL
  for $j : (i,j) \in A$ **do**
    if $\mu_j > \mu_i + t_{ij}$ **then**
      $\mu_j = \mu_i + t_{ij}$
      $\pi_j = i$
      if $j \notin$ SEL **then** add j to SEL
    end if
  end for
end while

标号检验算法

现在，如何识别：

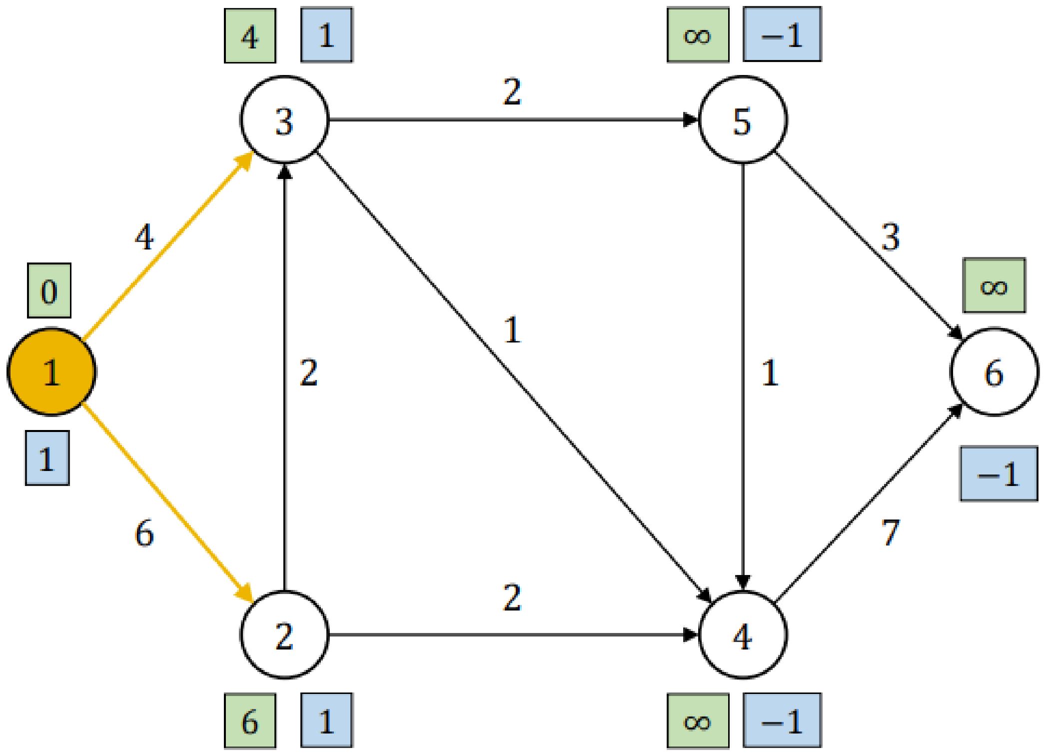
- 最佳路径
- 不可达的节点

标号检验算法案例



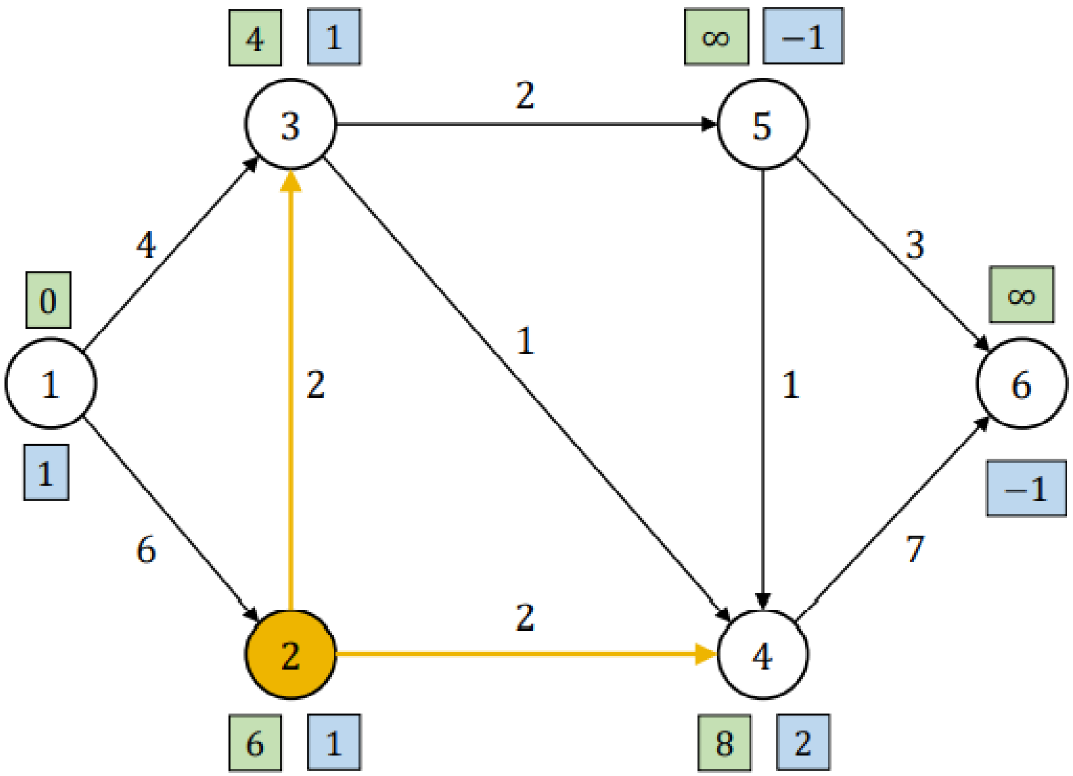
$$SEL = \{1\}$$

标号检验算法案例



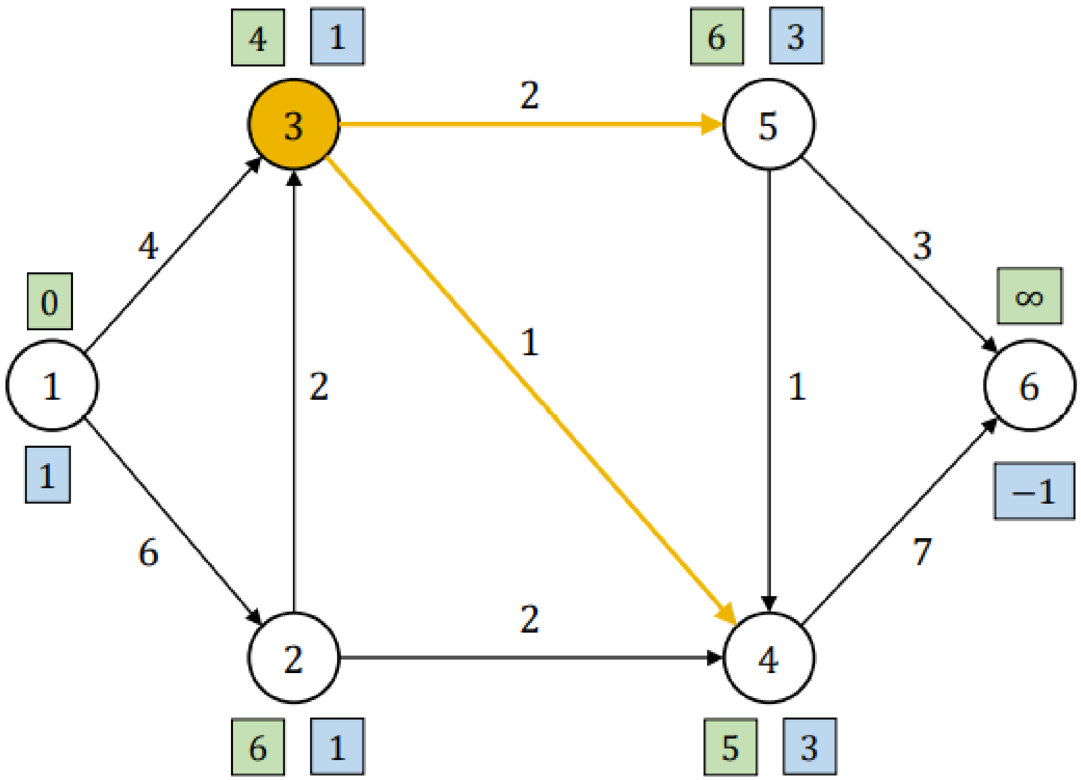
$$SEL = \{2, 3\}$$

标号检验算法案例



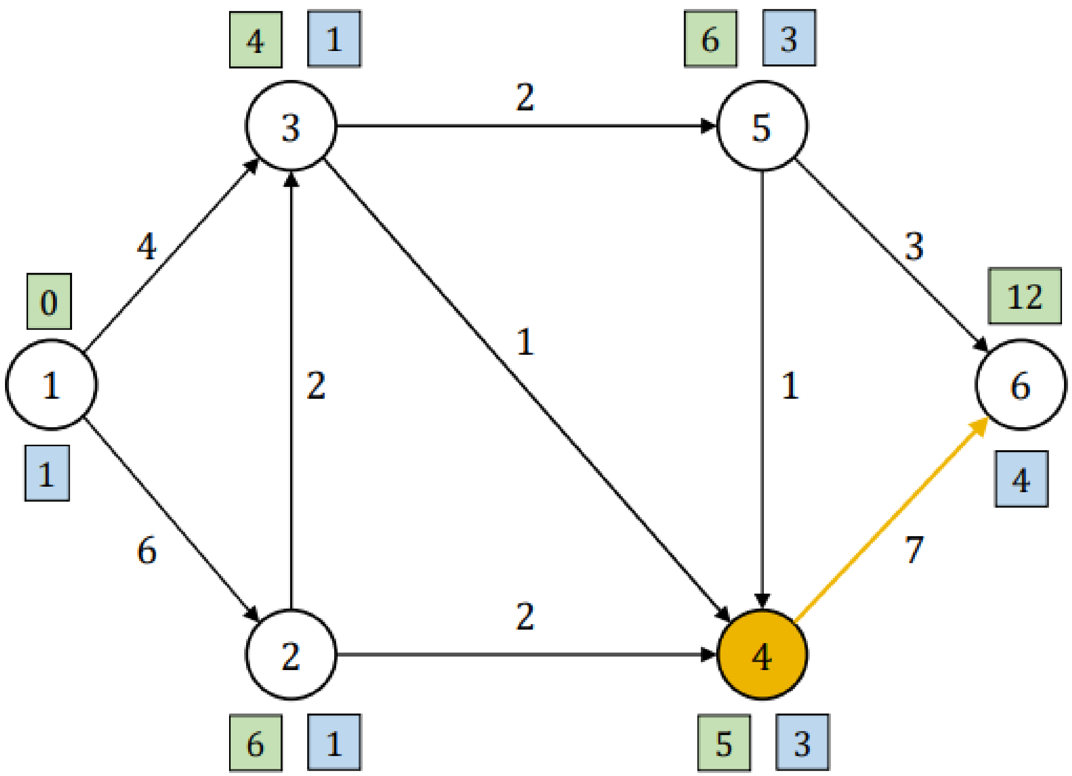
$$SEL = \{3, 4\}$$

标号检验算法案例



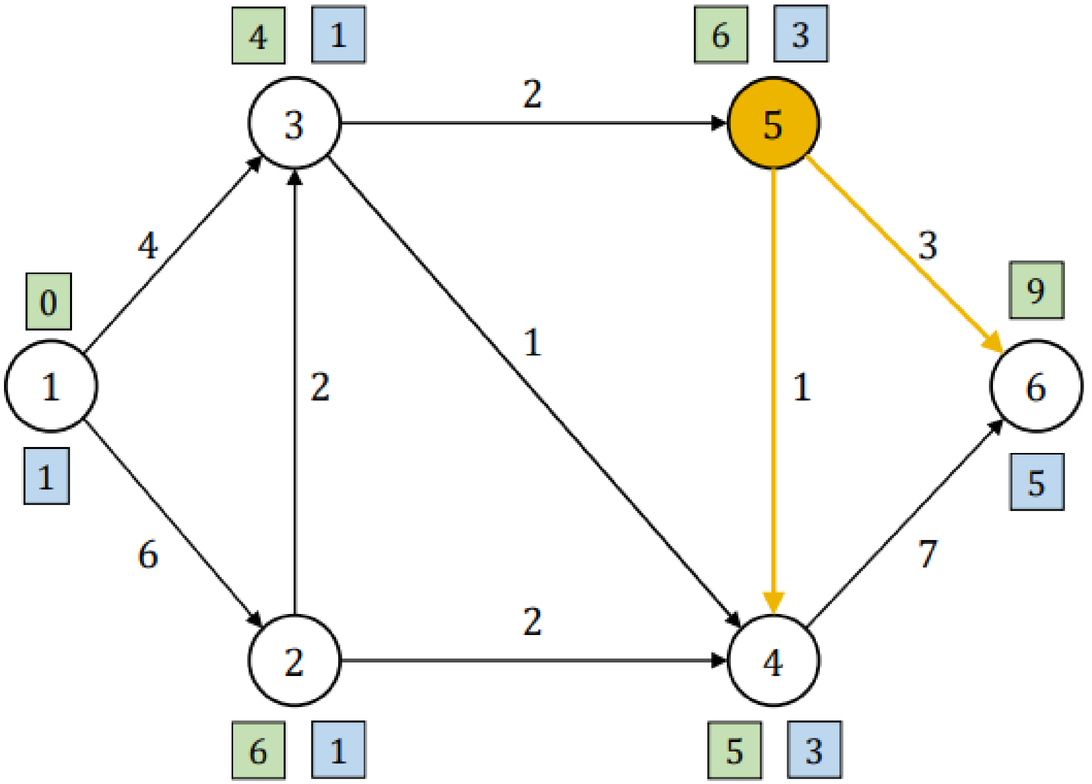
$$SEL = \{4, 5\}$$

标号检验算法案例



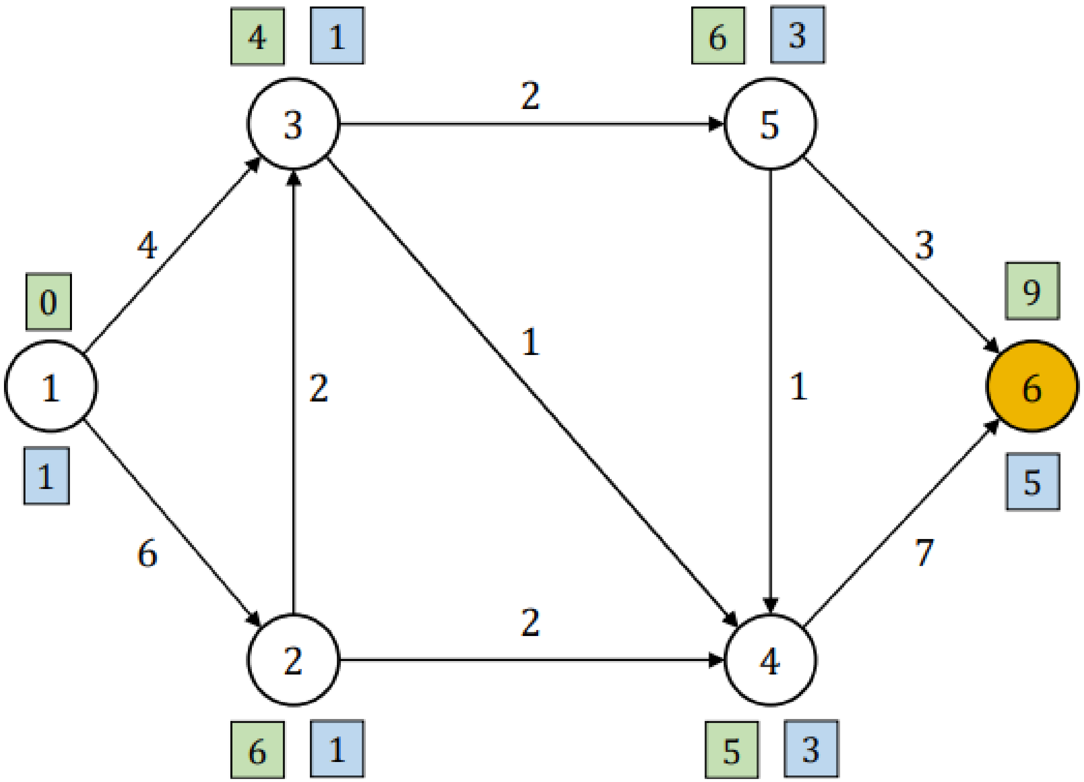
$$SEL = \{5, 6\}$$

标号检验算法案例



$$SEL = \{6\}$$

标号检验算法案例



$$SEL = \emptyset$$

标号检验算法案例

- 这种使用 SEL 的方法比通用标号校正算法快得多

- 还能做得更好吗?

4.标号设置算法（Label setting）

标号设置算法

算法介绍

当弧成本为非负时，可以以更短的方式找到最短路径

在每次迭代时根据顺序来更新一个节点的标号（最优的）

然而，标号检验方法保证只有在**终止后才能给出最佳标号**

标号设置算法

Dijkstra算法

Edsger W. Dijkstra

标号设置算法

Dijkstra算法

该组节点可以分为两组，S 用于表示设置了标号的节点，其余节点用 $\bar{S}$ 表示

S 中的节点标号是最优的，而 $\bar{S}$ 中的节点标签是暂存上限值

在每次迭代中，选择 $\bar{S}$ 中具有最小标号的节点并将其移动到 S 并扫描其下游弧

标号设置算法

Dijkstra算法(G,r)

Step 1: 初始化

$S = \emptyset,\ \bar{S} = N$

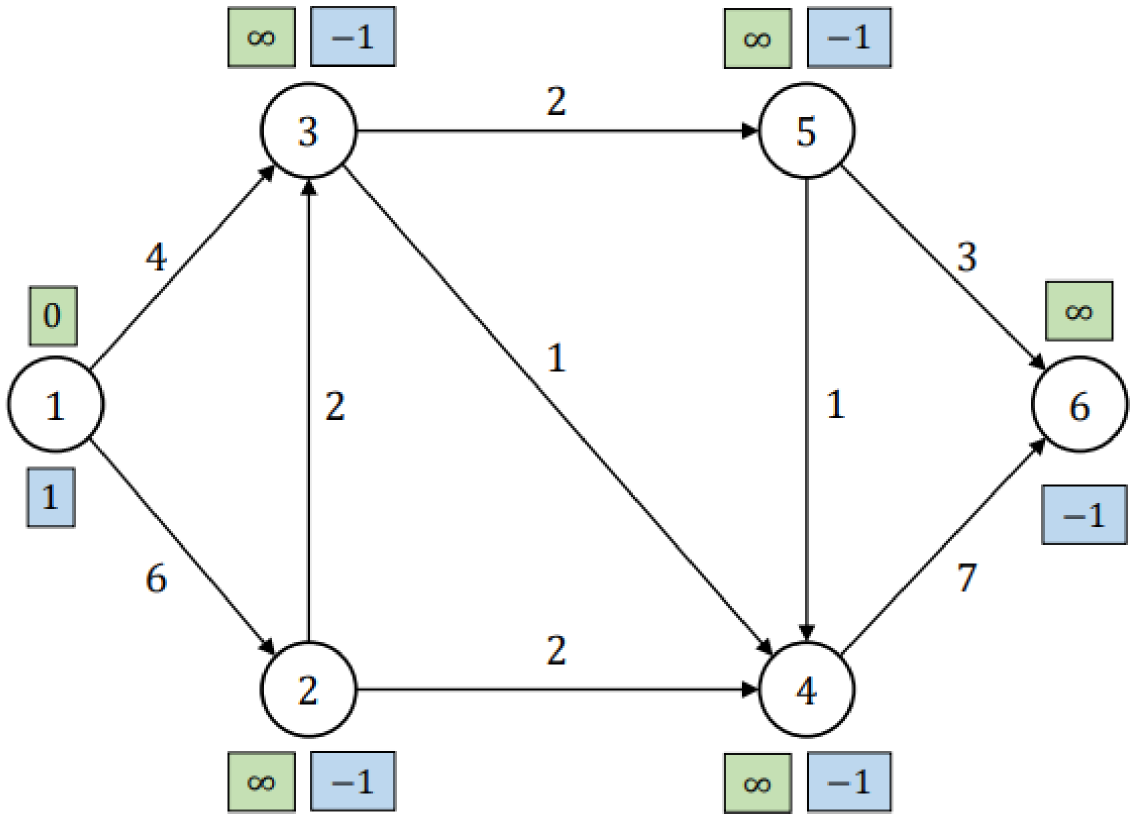
$\mu_r = 0, \pi_r = r$

$\mu_i = \infty, \pi_i = -1\ \forall\, i \in N \backslash \{r\}$

Step 2:

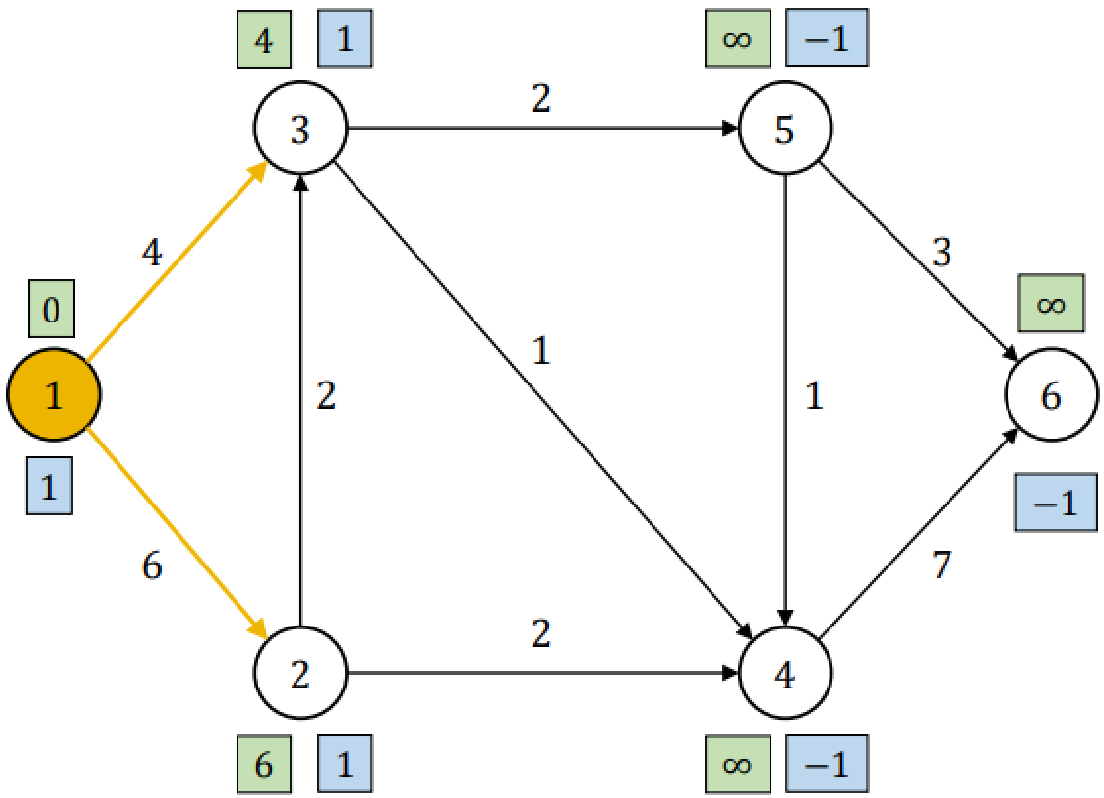
while $\bar{S} \neq \emptyset$ **do**
  $i = \arg\min_{j \in \bar{S}} \mu_j$
  $S = S \cup \{i\},\ \bar{S} = \bar{S} \backslash \{i\}$
  for $j : (i,j) \in A$ **do**
    if $\mu_j > \mu_i + t_{ij}$ **then**
      $\mu_j = \mu_i + t_{ij}$
      $\pi_j = i$
    end if
  end for
end while

标号设置算法案例



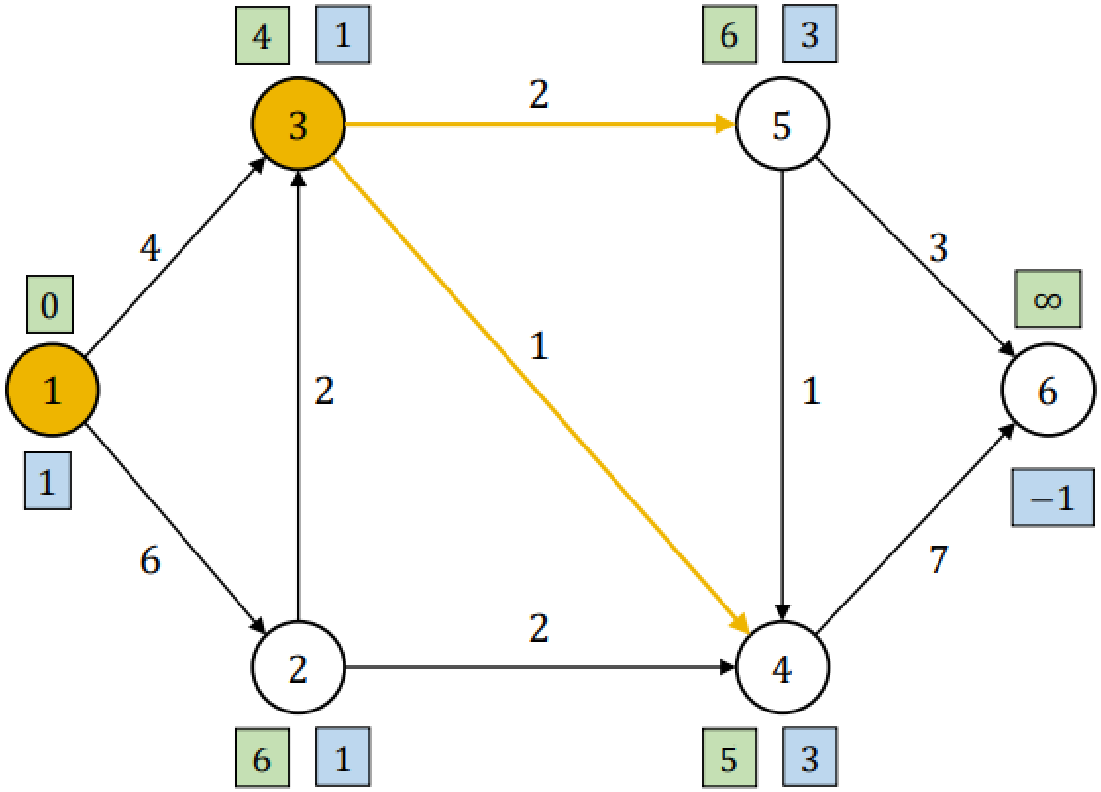
$$S = \emptyset,\ \bar{S} = \{1, 2, \ldots, 6\}$$

标号设置算法案例



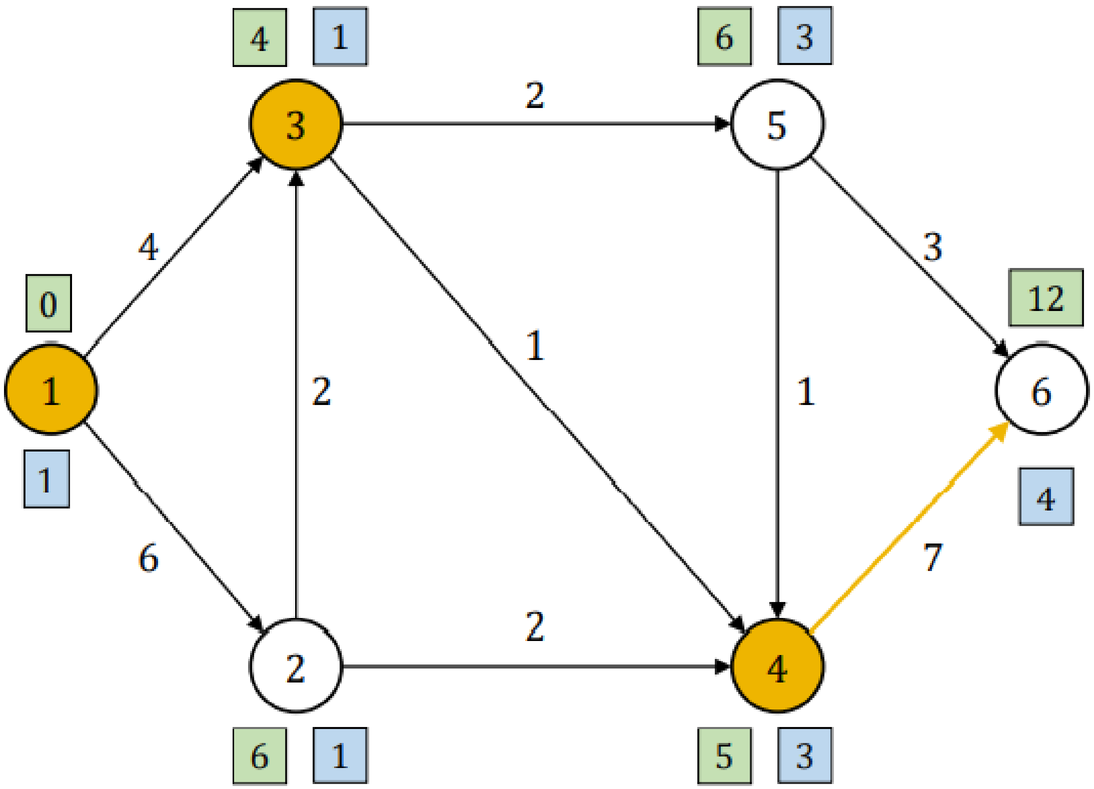
$$S = \{1\},\ \bar{S} = \{2, 3, 4, 5, 6\}$$

标号设置算法案例



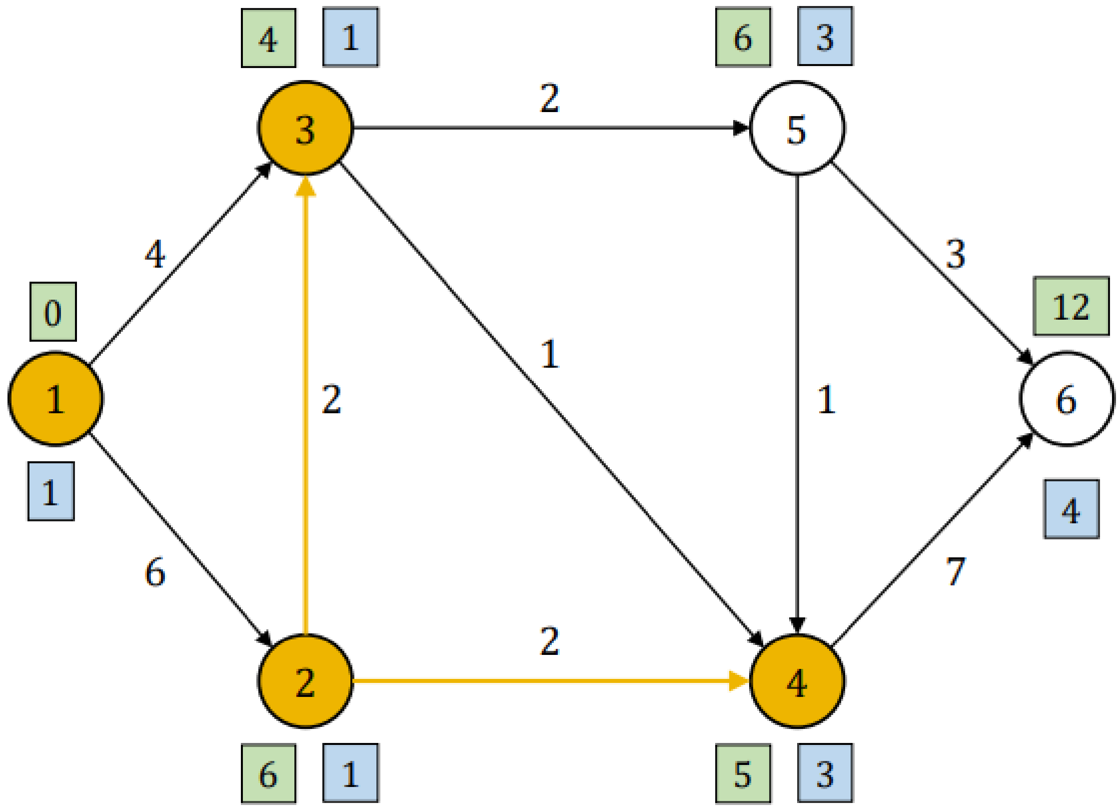
$$S = \{1, 3\},\ \bar{S} = \{2, 4, 5, 6\}$$

标号设置算法案例



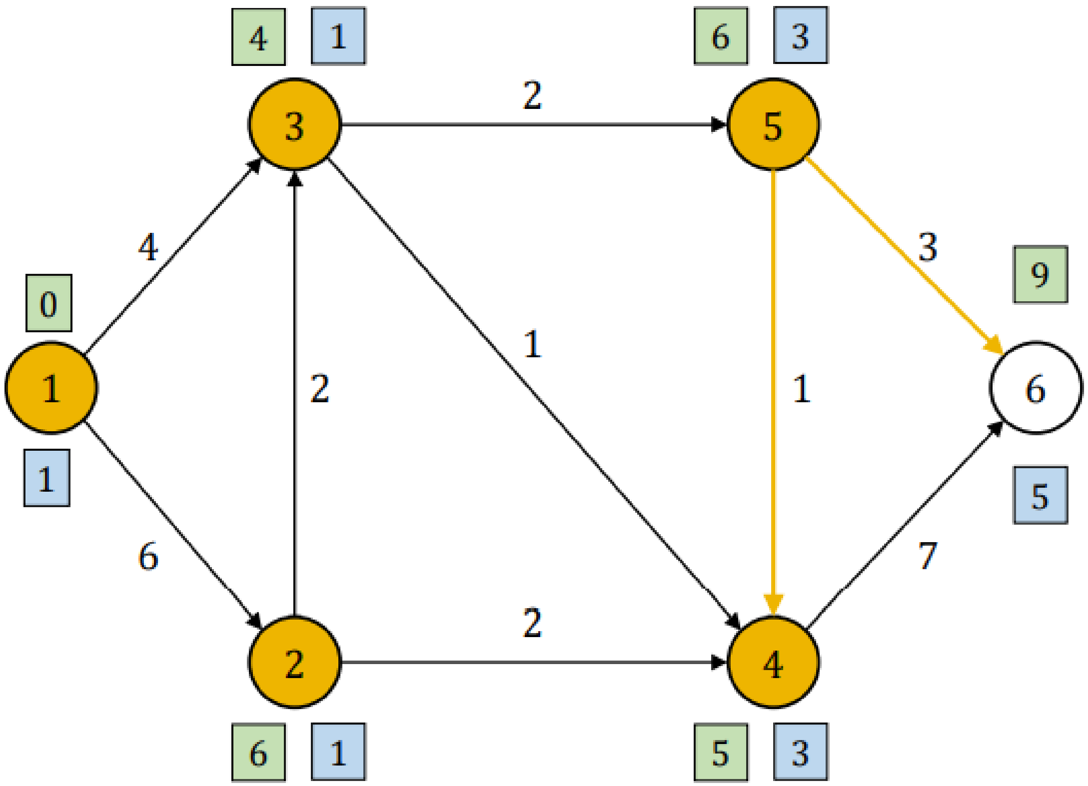
$$S = \{1, 3, 4\},\ \bar{S} = \{2, 5, 6\}$$

标号设置算法案例



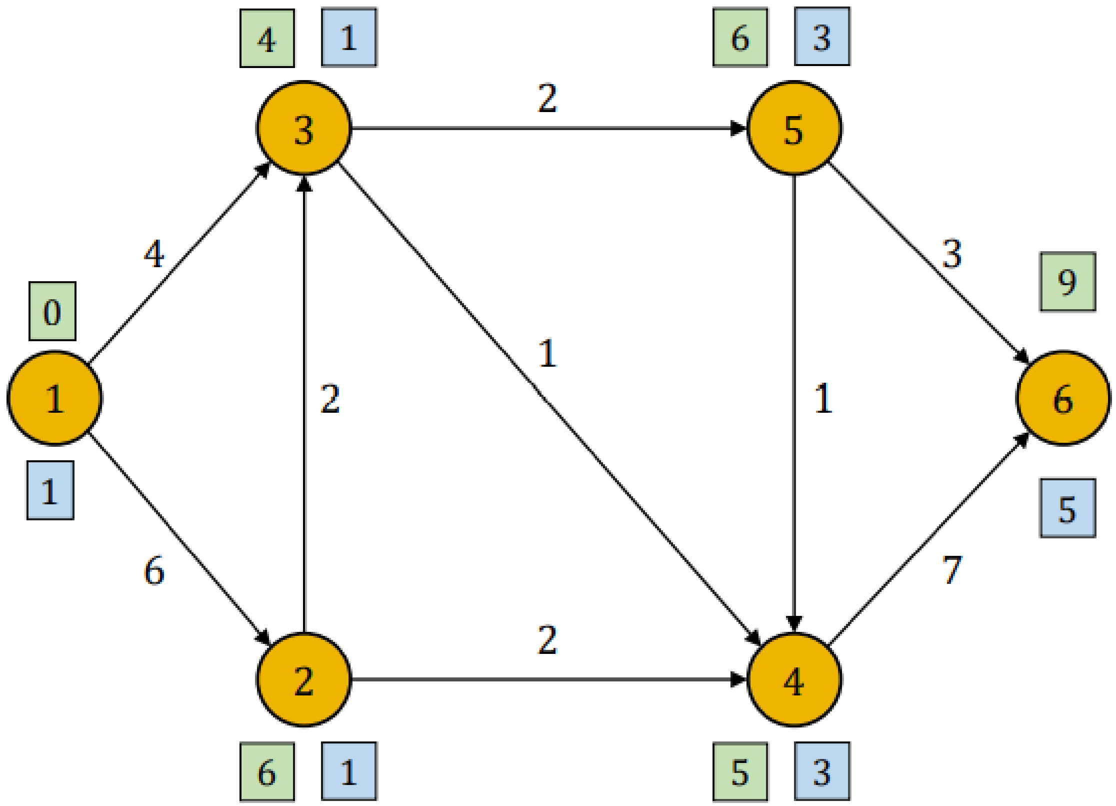
$$S = \{1, 3, 4, 2\},\ \bar{S} = \{5, 6\}$$

标号设置算法案例



$$S = \{1, 3, 4, 2, 5\},\ \bar{S} = \{6\}$$

标号设置算法案例



$$S = \{1, 3, 4, 2, 5, 6\},\ \bar{S} = \emptyset$$

计算性能

- 标号设置算法在 n 次迭代后终止（假设所有节点都可到达），因为在每一步中，将一个节点从 S^- 移动到 S
- 标号校正算法可能需要更多次迭代，因为节点可以重新进入 SEL。但在每次迭代中，标号设置算法需要更多时间（为什么？）
- 为不同实现的两种算法推导最坏情况复杂性

补充阅读

- **Boyles**, Chapter 2
- Ahuja, R. K., Magnanti, T. L., & Orlin, J. B. (1993).

  Network flows: theory, algorithms, and applications.