# 数据分析与算法设计
## 第10章 迭代改进
## （Iterative Improvement）

李旻
百人计划研究员
浙江大学 信息与电子工程学院
Email: min.li@zju.edu.cn

# 迭代改进

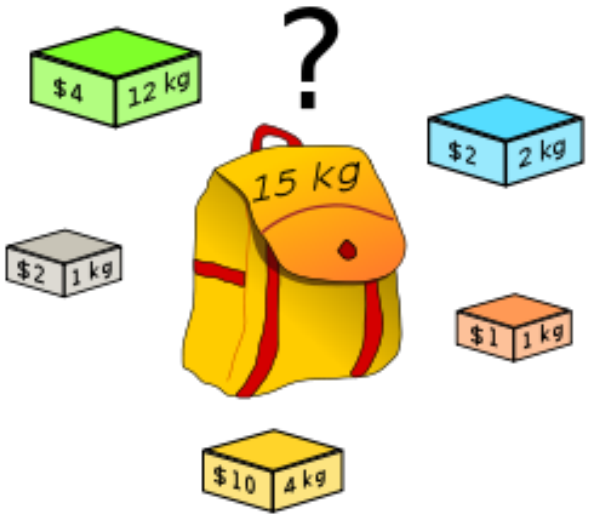
- 基本思想：针对所需要解决的问题，从满足问题约束的某些可行解出发，通过重复应用一些简单的步骤来不断地改进它，从而得到最终解
- 几个关键要素
  - 需要一个初始的可行解（简单解或近似解）
  - 设计合适的优化改进策略
  - 避免陷入局部最优
- 贪婪技术 vs 迭代改进
  - 前者基于部分解迭代扩展得到问题的解，后者基于可行解迭代改进得到问题的解

# 目录

- 线性规划问题：单纯形法
  (The Simplex Method)
- 最大流量问题
  (The Maximum-flow Problem)
- 二分图的最大匹配
  (Maximum Matching in Bipartite Graphs)
- 稳定婚姻问题
  (The Stable Marriage Problem)

# 线性规划(Linear Programming)

- 引例（背包问题的线性规划表示）
  - 给定n个重量为$w_1$、$w_2$、…$w_n$，价值为$v_1$、$v_2$、…$v_n$的物品和一个承重为$W$的背包，要求选择一些物品装入背包，使得背包内物品的价值之和达到最大




$$\text{maximize} \quad \sum_{j=1}^{n} v_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} w_j x_j \le W$$

$$0 \le x_j \le 1 \quad \text{for } j = 1, \ldots, n.$$

连续版本

# 线性规划

- 数学模型三要素
  - 一组待确定的决策变量
  - 一个明确的目标函数：为决策变量的线性函数
  - 一组约束条件：用决策变量的线性函数来表示

$$\text{Max(Min)}\ Z = c_1 \cdot x_1 + c_2 \cdot x_2 + \cdots + c_n \cdot x_n$$

$$\text{s.t.}\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + \cdots + a_{1n} \cdot x_n \leqslant (\geqslant, =)\ b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + \cdots + a_{2n} \cdot x_n \leqslant (\geqslant, =)\ b_2 \\ \vdots \\ a_{m1} \cdot x_1 + a_{m2} \cdot x_2 + \cdots + a_{mn} \cdot x_n \leqslant (\geqslant, =)\ b_m \\ x_j \geqslant 0, \quad j = 1, \cdots, n \end{cases}$$

- 应用：运筹学的一个重要分支，广泛应用于军事作战、经济分析、经营管理和工程技术等方面，为合理地利用有限的人力、物力、财力等资源作出的最优决策，提供科学的依据

# 线性规划

Leonid Kantorovich

Tjalling Koopmans

1975年诺贝尔经济学奖获得者

14 October 1975

The Royal Swedish Academy of Sciences has decided to award the Prize in Economic Science in Memory of Alfred Nobel for 1975 in equal shares to

Professor **Leonid Kantorovich**, USSR,
and
Professor **Tjalling C. Koopmans**, USA,

**for their contributions to the theory of optimum allocation of resources.**

**Optimum Allocation of Resources**
Leonid Kantorovich and Tjalling Koopmans have both done their most important scientific work in the field of normative economic theory, *i.e.*, the theory of the optimum allocation of resources. As the starting point of their work in this field, both have studied the problem – fundamental to all economic activity – of how available productive resources can be used to the greatest advantage in the production of goods and services. This field embraces such questions as what goods should be produced, what methods of production should be used and how much of current production should be consumed, and how much reserved to create new resources for future production and consumption.

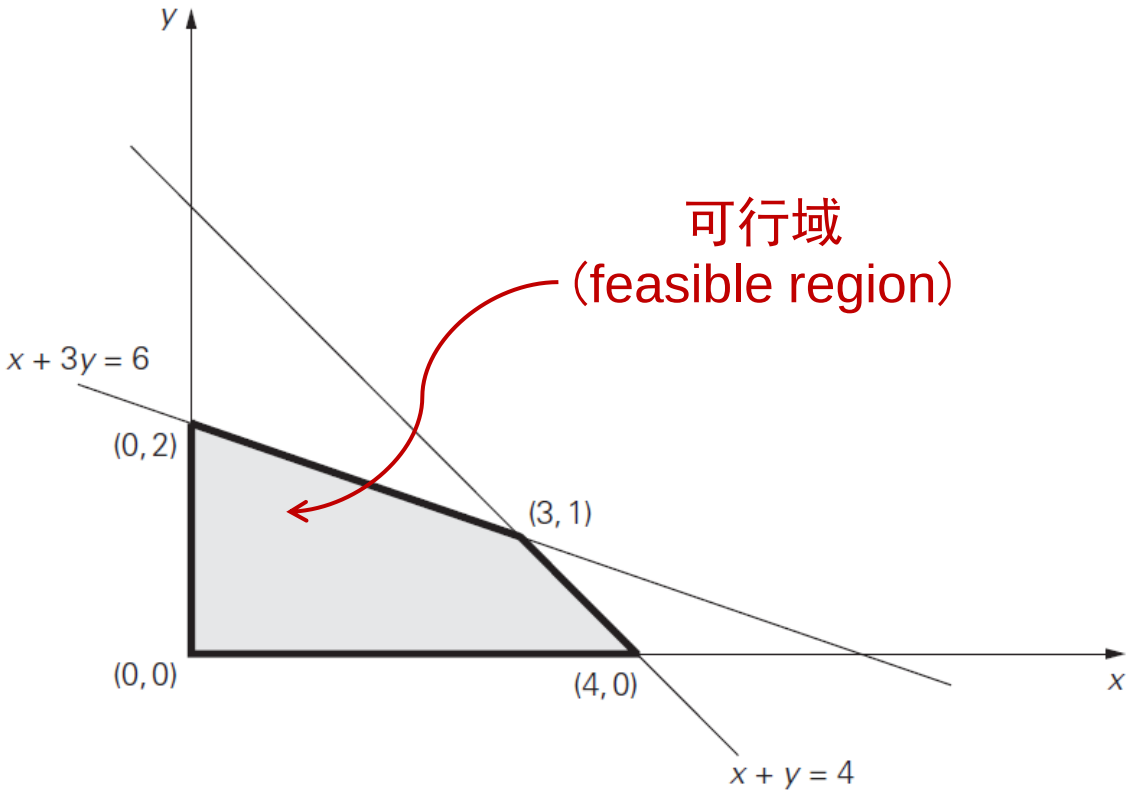
**Improved Economic Planning**
As they have formulated the problems and described the connection between production results and productive inputs in new ways, these two scholars have been able to achieve highly significant results. Early in his research, Professor Kantorovich applied the analytical technique of linear programming to demonstrate how economic planning in his country could be improved. Professor Koopmans, for his part, has shown for instance that on the basis of certain efficiency criteria, it is possible directly to make important deductions concerning optimum price systems.

# 线性规划求解

- 两变量的线性规划问题

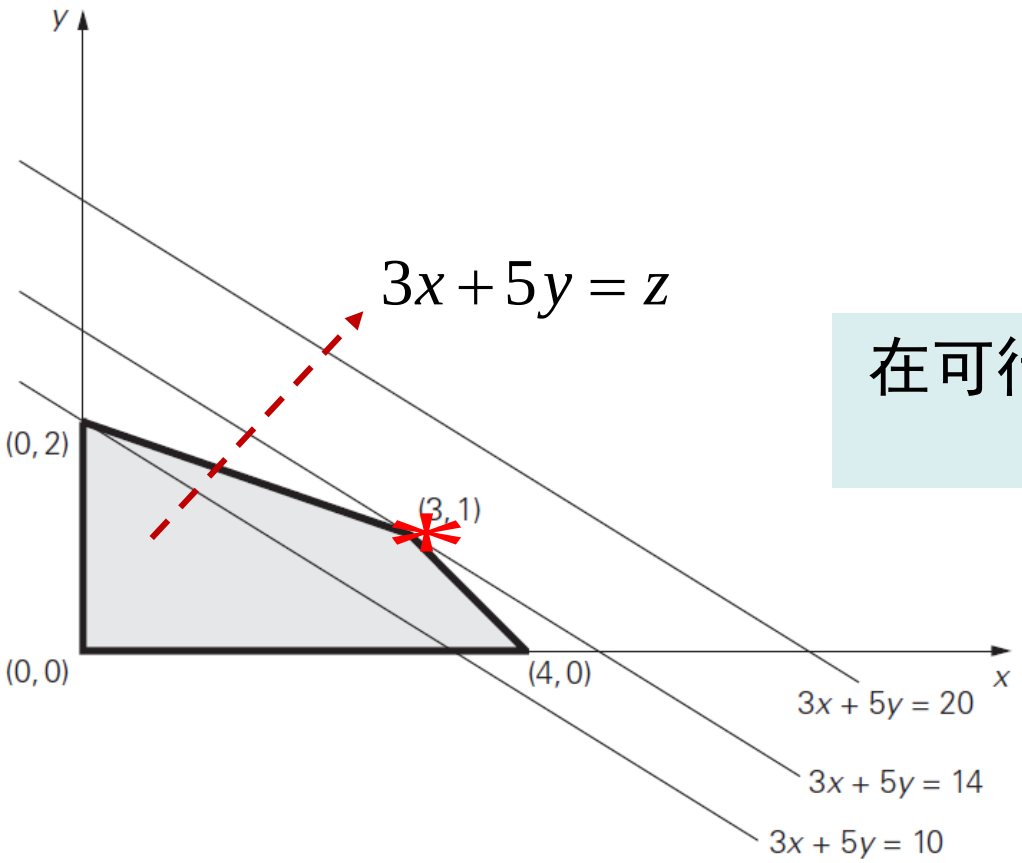
$$\max z = 3x + 5y$$

$$s.t. \begin{cases} x + y \le 4 \\ x + 3y \le 6 \\ x \ge 0, y \ge 0 \end{cases}$$



# 线性规划求解

- 两变量的线性规划问题



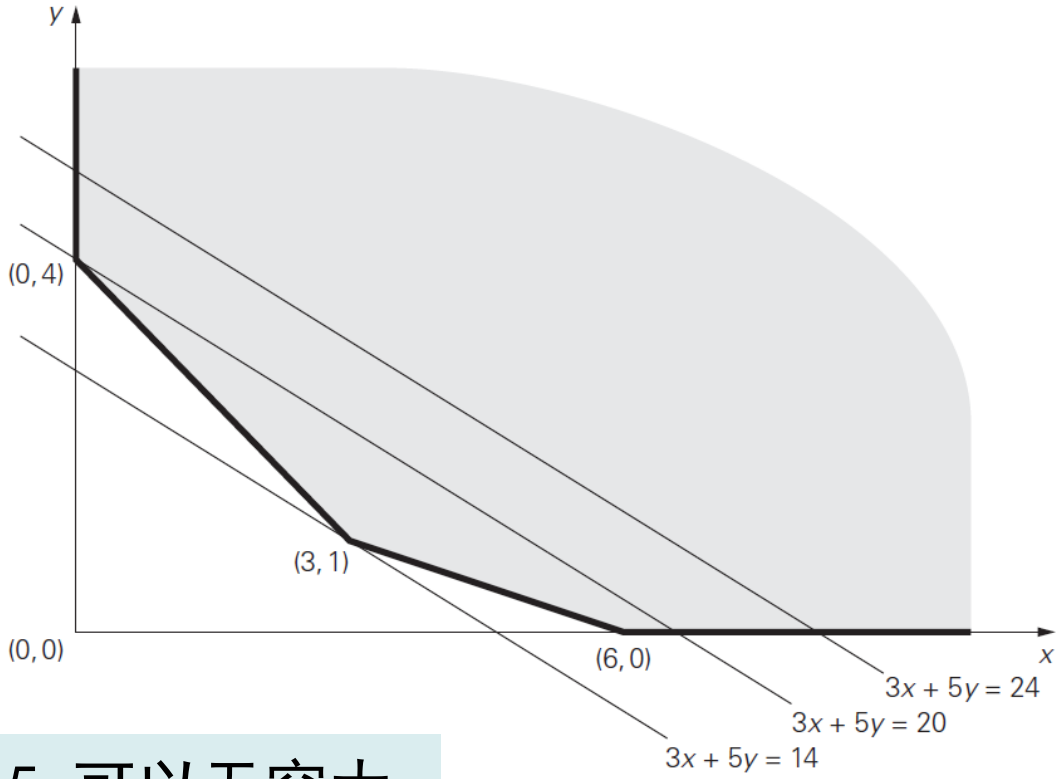

在可行域的一个极点处取得最大值

# 线性规划求解

- 两变量的线性规划问题

$$\max z = 3x + 5y$$

$$s.t.\begin{cases} x + y \geq 4 \\ x + 3y \geq 6 \\ x \geq 0, y \geq 0 \end{cases}$$




无边界的可行域，3x+5y可以无穷大

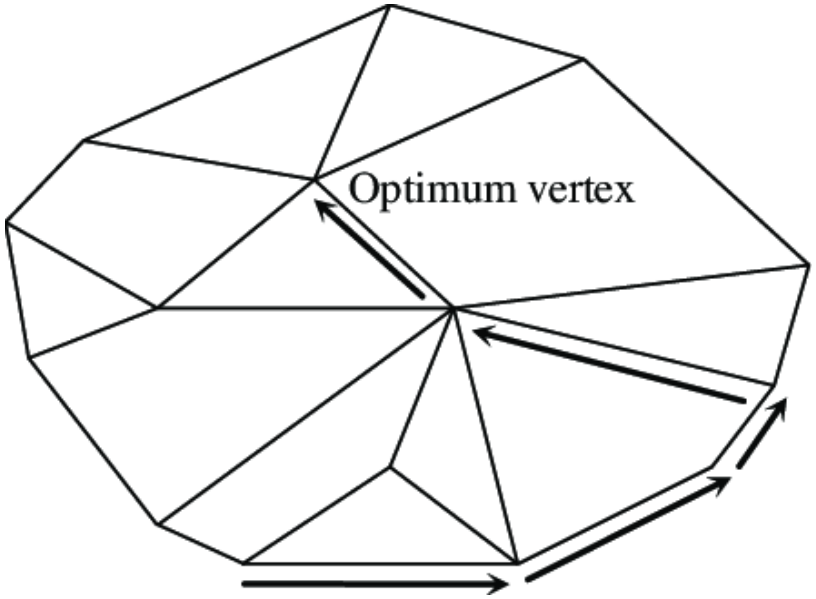
# 线性规划求解

- 极点定理：任何一个可行区域非空有界的线性规划问题都有最优解，而且最优解总是能够在其可行区域的一个极点上找到

- 蛮力法：根据约束条件，生成可行区域的所有极点，从中决定使目标函数最大的一个点
  - 如何有效地确定可行域的极点？
  - 极点的数量随问题规模的增长呈指数级增长

# 单纯形法

- 基本思想：只需检测可行区域极点中的一小部分就可以找到最优点。思路是先在可行域中找到一个极点，然后检查能否在邻接极点取得更佳的函数值，如果不是，则当前极点为最优点，如果是，则重复上述操作



# 单纯形法

- 单纯形法应用于求解线性规划问题的标准形式
  - 必须是一个最大化问题
  - 所有的约束必须用线性方程表示，且右端非负
  - 所有的变量必须要求是非负的

$$\max\ z = \sum_{j=1}^{n} c_j x_j$$

$$s.t. \begin{cases} \sum_{j=1}^{n} a_{ij} x_j = b_i \quad (b_i \geq 0) \\ x_j \geq 0 \end{cases}$$

# 单纯形法

- 如果原问题不是标准形式，则需按下分类进行变换
  - 若原问题是最小化问题$\min z = \sum_{j=1}^{n} c_j x_j$，则定义$z_1 = -z$，转化为最大化问题$\max z_1 = -\sum_{j=1}^{n} c_j x_j$
  - 若某个右端常数$b_i < 0$，则该约束左右端乘以-1
  - 若某约束为“≤”型的不等式约束，则可在左端加上一个非负辅助变量（松弛变量），使得不等式化为等式；若某约束为“≥”型的不等式约束，则可在左端减去一个非负辅助变量（松弛变量），使得不等式化为等式

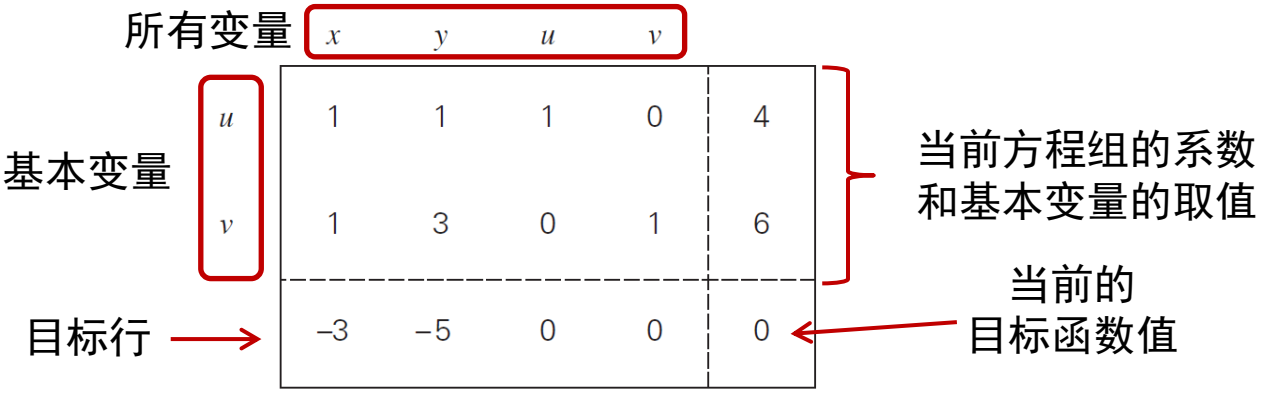
$$\max z = 3x + 5y$$

$$s.t.\begin{cases} x + y \le 4 \\ x + 3y \le 6 \\ x \ge 0, y \ge 0 \end{cases} \Rightarrow \max z = 3x + 5y + 0u + 0v$$

$$s.t.\begin{cases} x + y + u = 4 \\ x + 3y + v = 6 \\ x, y, u, v \ge 0 \end{cases}$$

# 单纯形法

- 假定标准化后的线性规划问题的约束是包含$m$个等式的$n$元方程组（$n \geq m$），则可将其中$n-m$个变量置零得到$m$个等式的$m$元方程组
  - 基本变量：未被置零的变量
  - 非基本变量：被置零的变量
  - 基本解：$m$元方程组具有唯一解时，该解称为基本解
  - 基本可行解：所有坐标为非负的基本解
- 单纯形表（simplex tableau）：根据极点存储基本可行解的信息



# 单纯形法

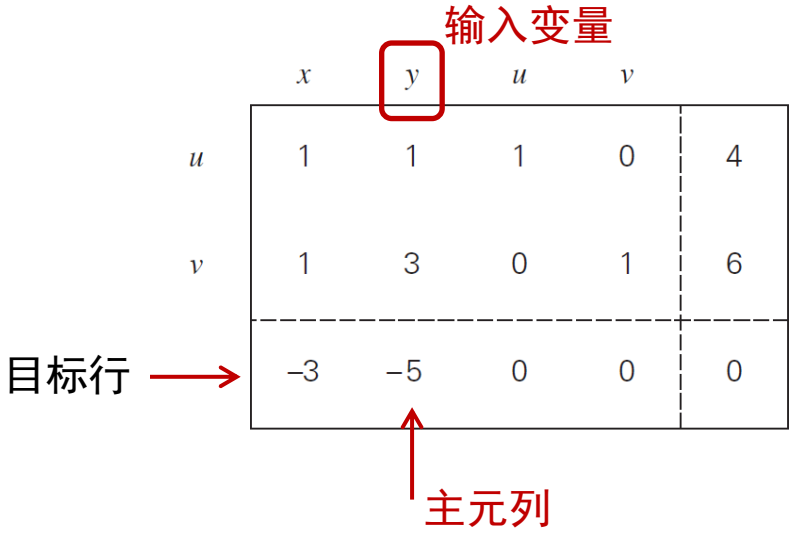
- 第0步：初始化
  - 将给定的线性规划问题转成标准形式，并建立初始单纯形表格，表格的最右列都为非负，剩下的m列确定了基本可行解的基本变量，行用基本变量标识

| | $x$ | $y$ | $u$ | $v$ | |
|---|---|---|---|---|---|
| $u$ | 1 | 1 | 1 | 0 | 4 |
| $v$ | 1 | 3 | 0 | 1 | 6 |
| 目标行 → | −3 | −5 | 0 | 0 | 0 |

- 第1步：最优测试
  - 如果目标行的所有单元格都是非负的，则算法停止，该表格代表一个最优解；否则，通过后续步骤确定当前极点的邻接极点，并按相应规则更新表格

# 单纯形法

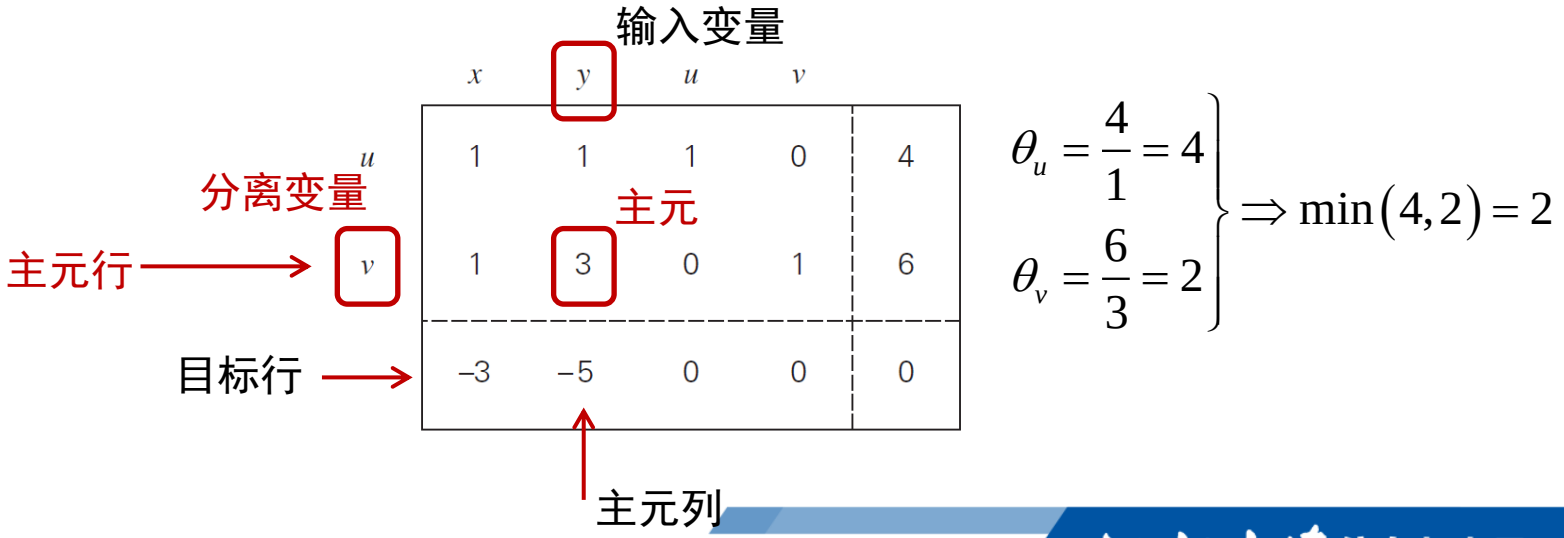
- 第2步：确定输入变量（进基变量）
  - 从目标行中选一个最能影响结果的变量（常用规则是选取绝对值最大的负单元格），即为输入变量，这个变量所在列即为主元列




目标函数中y对应的系数为5，大于x对应的系数3，因此选择增加y会比较大幅度地增加目标函数，但是具体的增量还要受限于约束条件

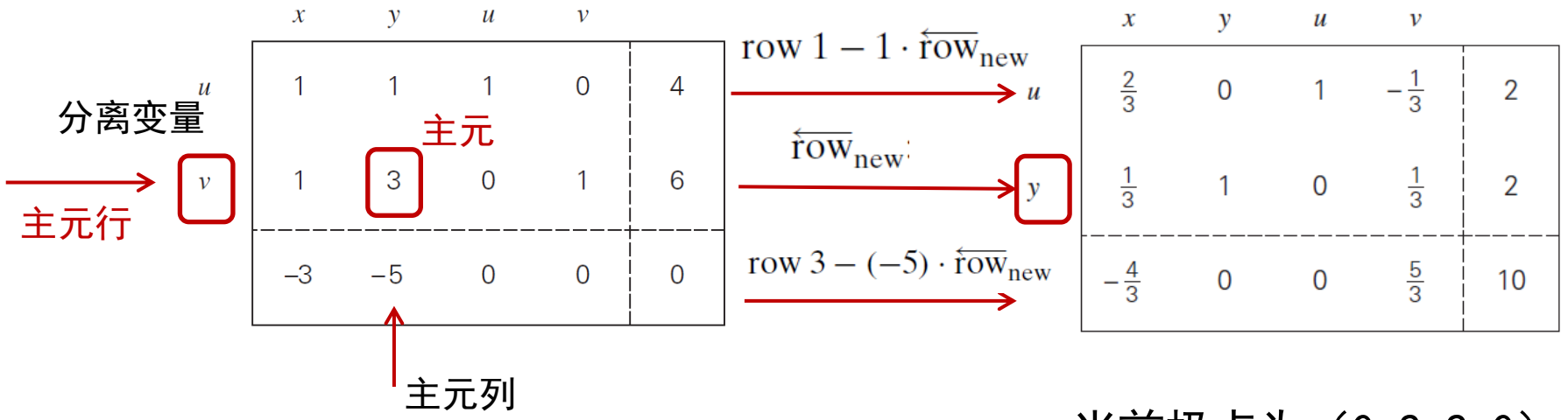
# 单纯形法

- 第3步：确定分离变量（离基变量）
  - 目标：把输入变量变得尽可能大的同时，需要把一个原有的基本变量变成0，并保证所有其它变量的非负性，从而找到一个使目标函数更大的邻接极点
  - 规则：对于主元列上的每个正单元格，其所在行最右单元格除以主元列的单元格，得到一个$\theta$比率，该比率最小的行确定了分离变变量和主元行。若主元列单元格都为负或0，问题无界，算法终止



# 单纯形法

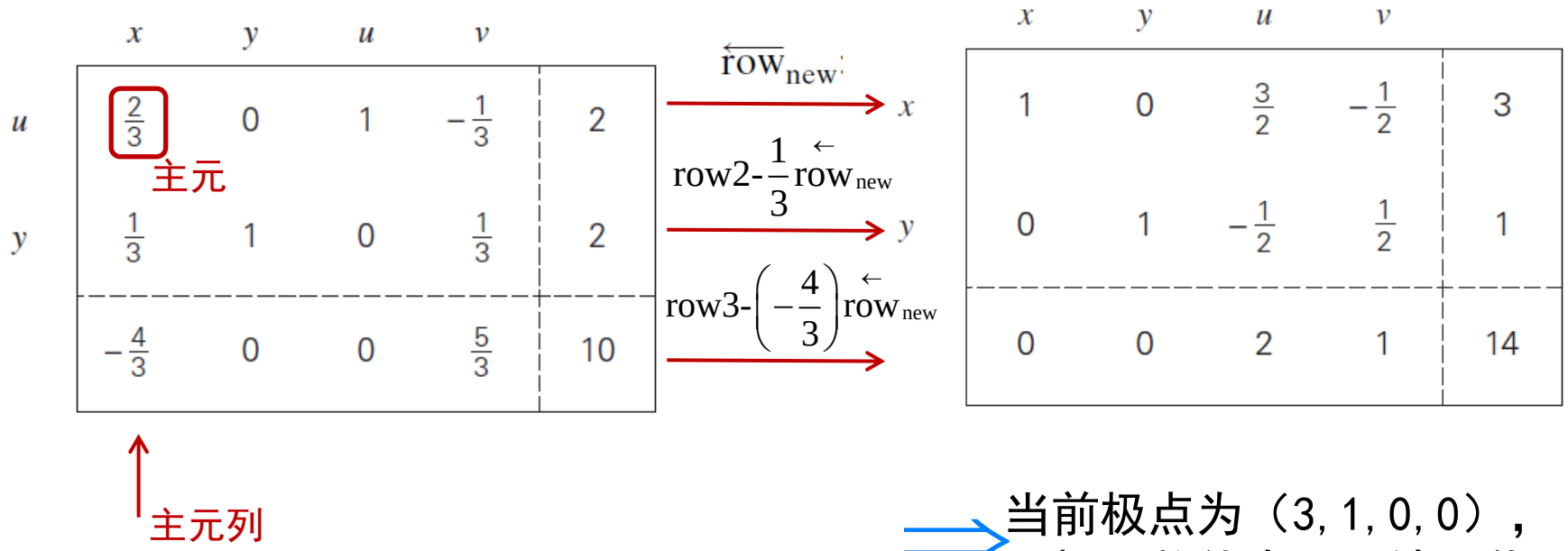
- 第4步：建立下一张表格
  - 将主元行的所有单元格除以主元得到新主元行；将包括目标行在内的其它行，减去该行主元列单元格和新主元的乘积，更新该行；另外，把主元行前的标识用主元列的变量名替换，返回第1步




⟹ 当前极点为（0, 2, 2, 0），目标函数值为10

# 单纯形法

- 返回第一步，重复相关步骤，直到最优测试成功




当前极点为（3, 1, 0, 0），
目标函数值为14，达最优

# 单纯形法

- 其它要点
  - Bland法则：如果一个或多个基本变量等于0，则该实例为退化实例，基本的单线形法会重复处理某些极点，陷入“环路”。为此，需针对第2步、第3步做相应改进

**修改后的第 2 步：** 在目标行的负单元格中，选择下标最小的列。

**修改后的第 3 步：** 在 $\theta$ 比率最小的行中，选择下标最小的基本变量所标识的行。

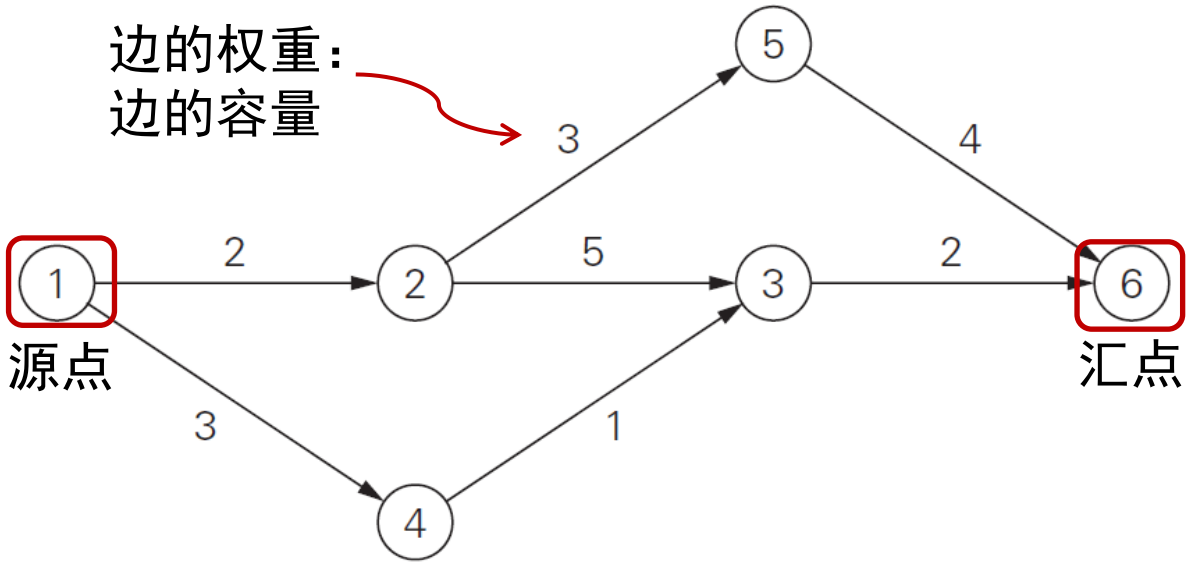
  - 算法效率：典型应用中，如果$m$和$n$分别是约束等式和变量的数量，则算法的迭代次数在$m$和$3m$之间，每次迭代操作的次数和$mn$成正比
  - 其它算法：椭球法、Karmarkar算法、内点法等

# 目录

- 单纯形法
  (The Simplex Method)
- 最大流量问题
  (The Maximum-flow Problem)
- 二分图的最大匹配
  (Maximum Matching in Bipartite Graphs)
- 稳定婚姻问题
  (The Stable Marriage Problem)

# 最大流量问题

- 问题：如何使传输网络（管道系统、通信系统、配电系统等）上的物质流最大化？
- 建模：流量网络




节点的流量守恒：
输入流量 = 输出流量

$$\sum_{j:(j,i)\in E} x_{ji} = \sum_{j:(i,j)\in E} x_{ij}$$

源点的总输入流 =
汇点的总输出流

$$\sum_{j:(1,j)\in E} x_{1j} = \sum_{j:(j,n)\in E} x_{jn}$$

# 最大流量问题

- 建模：优化问题

$$\text{maximize} \quad v = \sum_{j:(1,j)\in E} x_{1j}$$

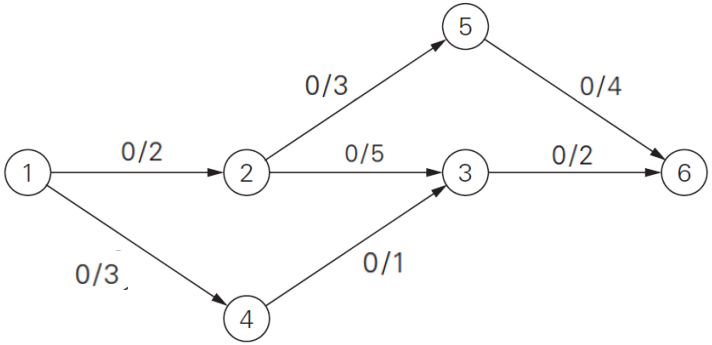
$$\text{subject to} \quad \sum_{j:(j,i)\in E} x_{ji} - \sum_{j:(i,j)\in E} x_{ij} = 0 \quad \text{for } i = 2, 3, \ldots, n-1$$

$$0 \le x_{ij} \le u_{ij} \quad \text{for every edge } (i, j) \in E.$$

  - 线性规划问题，可转成标准形式，并采用单纯形法解。但可以利用问题结构，设计出更快的迭代改进算法

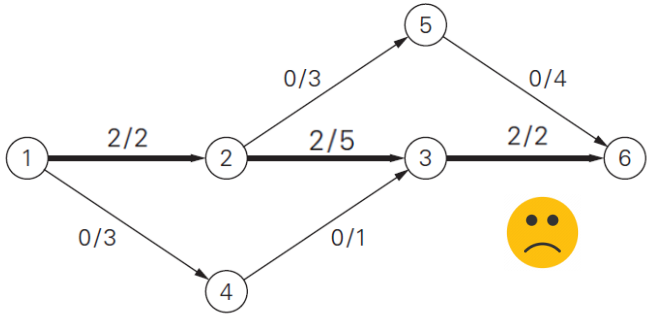
# 增益路径法（Ford-Fulkerson法）

- 基本思想：从流量0开始，试着找一条可以传输更多流量的源点到汇点的路径（称为流量增益路径）；如果找到，则沿着该路径调整边上的流量，以得到更大的流量值，并试着为新的流量找到一条新的增益路径

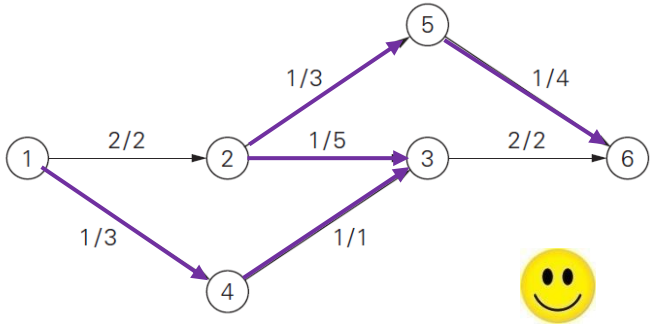



初始化：流量为0

考虑增益路径
1->2->3->6




考虑增益路径
1->4->3<-2->5->6



# 增益路径法

- 为了找到流量增益路径，需要考虑两类边：
  - 前向边：从i连向j，有正的未使用容量$r_{ij} = u_{ij} - x_{ij}$
  - 后向边：从j连向i，具有正的流量$x_{ij}$
- 假设$r$是路径中所有前向边的未使用容量$r_{ij}$和所有后向边的流量$x_{ij}$的最小值
- 则在每条前向边的当前流量上增加$r$，在每条后向边的流量上减去$r$，就会得到一个可行的更大流量

$\xrightarrow{+r} i \xrightarrow{+r}$，　$\xrightarrow{+r} i \xleftarrow{-r}$，　$\xleftarrow{-r} i \xrightarrow{+r}$，　$\xleftarrow{-r} i \xleftarrow{-r}$.

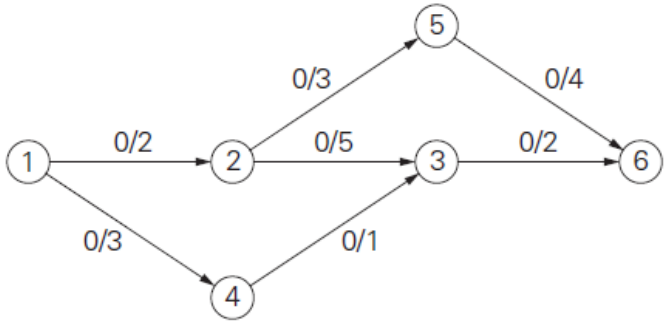
  - $r$为整数时，每次迭代中，流量的值至少增加1

# 增益路径法

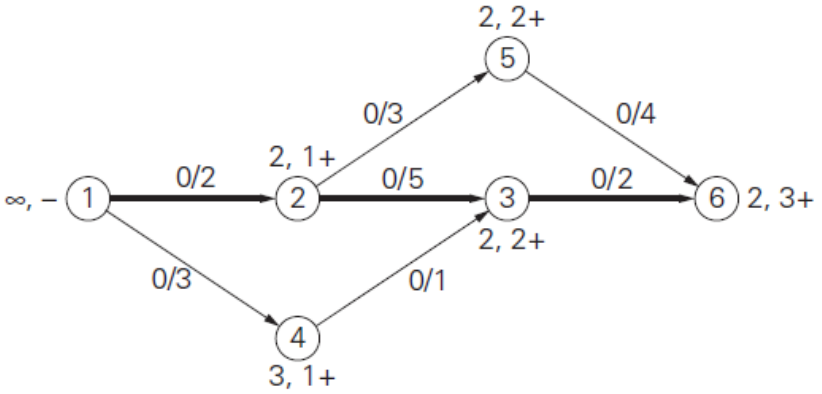
- 如何高效地生成流量增益路径？
  - 利用广度优先查找，用数量最少的边来生成路径
- 最短增益路径法（先标记先扫描算法）
  - 顶点标记方式：用两个记号来标记一个新的顶点
    - 第一个标记指出从源点到被标记顶点还可增加的流量$l_j$
      - 如果被标记顶点j是从i到j的有向边，则$l_j = \min\{l_i, r_{ij}\}$
      - 如果被标记顶点j是从j到i的有向边，则$l_j = \min\{l_i, x_{ji}\}$
    - 第二个标记记录其前序顶点，并加上“+”或“–”号来表示该前序顶点到本顶点是前向边还是后向边
  - 用队列来存取标记的顶点

# 增益路径法

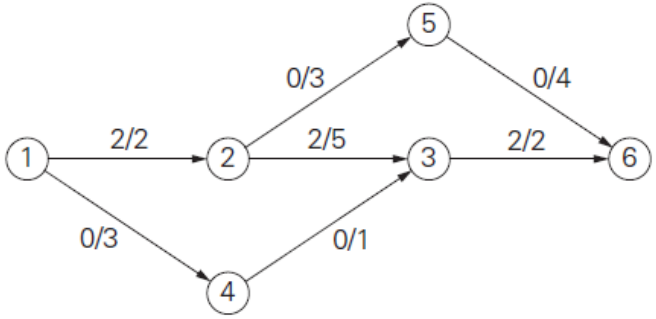
- 最短增益路径法（先标记先扫描算法）
  - 建立一个空队列，用于存放每次标记的节点。第一次将源点放入此队列中，以后取节点操作都是对队列进行
  - 前向边扫描：每次都先取队列的第一个节点，然后查看此节点指向哪些节点，若被指向的节点未被标记，且相应的正向边未使用容量为正，则将被指向的节点进行标记后放入队列。循环做此步直至当前节点所指向的点全部标记且入队列
  - 后向边扫描：当前节点的前向边遍历结束后，进行反向边遍历，将有正流量的反向边节点进行标记后放入队列
  - 当前节点前向、后向遍历结束后，检查汇点是否被标记，若没被标记则继续取队列的下一个节点；若被标记则表示一条增广路径已被找到，更新各边的流量，然后将所有节点的标记都初始化、将队列清空，将源点入队，开始下一条增益路径的寻找



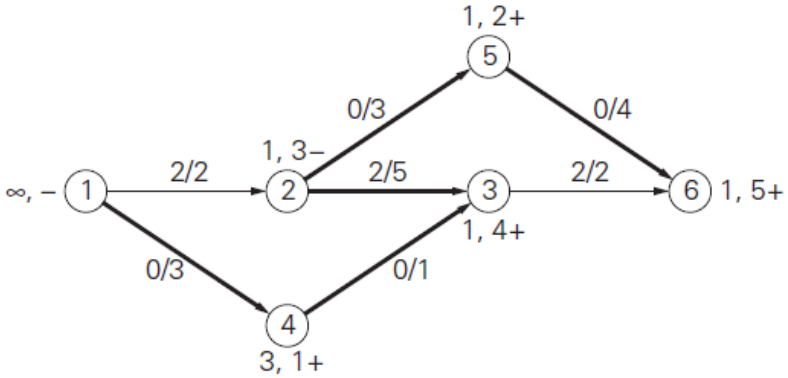

Queue: 1 2 4 3 5 6
↑ ↑ ↑ ↑



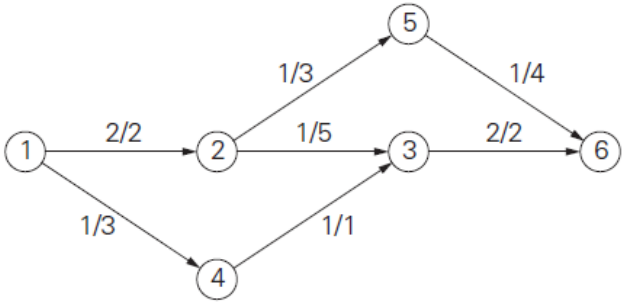

Augment the flow by 2 (the sink's first label) along the path $1 \rightarrow 2 \rightarrow 3 \rightarrow 6$.



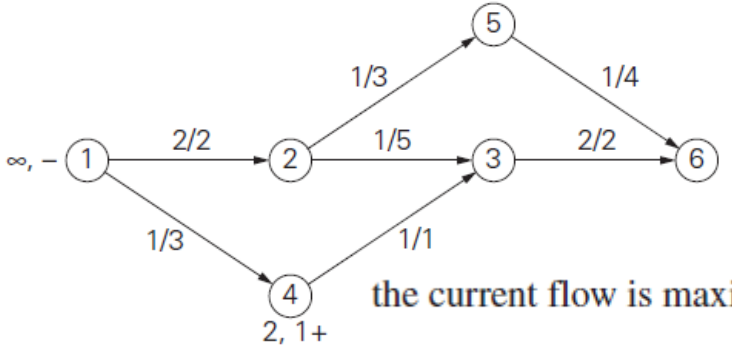

Queue: 1 4 3 2 5 6
↑ ↑ ↑ ↑ ↑




Augment the flow by 1 (the sink's first label) along the path $1 \rightarrow 4 \rightarrow 3 \leftarrow 2 \rightarrow 5 \rightarrow 6$.




Queue: 1 4
↑ ↑




the current flow is maximal.

**算法** ShortestAugmentingPath($G$)

//最短增量路径算法的实现

//输入：网络 $G$，具有一个源点 1 和一个汇点 $n$，每条边 $(i,j)$ 的容量都是正整数 $u_{ij}$

//输出：最大流量 $x$

对网络中的每条边 $(i,j)$，设 $x_{ij}=0$

把源点标记为 $\infty,-$，再把源点加入到空队列 $Q$ 中

**while not** Empty($Q$) **do**

  $i \leftarrow$ Front($Q$); Dequeue($Q$)

  **for** 从 $i$ 到 $j$ 的每条边 **do** //前向边

    **if** $j$ 没有被标记

      $r_{ij} \leftarrow u_{ij} - x_{ij}$

      **if** $r_{ij} > 0$

        $l_j \leftarrow \min\{l_i, r_{ij}\}$；用 $l_j, i^+$ 来标记 $j$

        Enqueue($Q, j$)

  **for** 从 $j$ 到 $i$ 的每条边 **do** //后向边

    **if** $j$ 没有被标记

      **if** $x_{ji} > 0$

        $l_j \leftarrow \min\{l_i, x_{ji}\}$；用 $l_j, i^-$ 来标记 $j$

        Enqueue($Q, j$)

  **if** 汇点被标记了

    //沿着找到的增益路径进行增益

    $j \leftarrow n$ //从汇点开始，用第二个标记反向移动

    **while** $j \neq 1$ //没有到达源点

      **if** 顶点 $j$ 的第二个标记是 $i^+$

        $x_{ij} \leftarrow x_{ij} + l_n$

      **else** //顶点 $j$ 的第二个标记是 $i^-$

        $x_{ji} \leftarrow x_{ji} - l_n$

      $j \leftarrow i$；$i \leftarrow i$ 的第二个标记指出的顶点

    除了源点，擦去所有顶点的标记

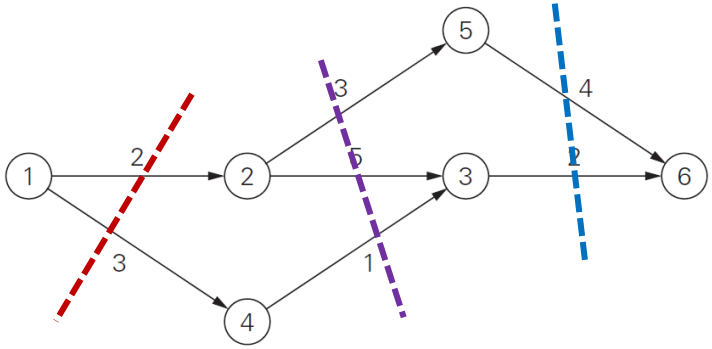
    用源点对 $Q$ 重新初始化

**return** $x$ //当前的流量是最大的

# 最大流-最小割定理

- 增益路径法得到的最终流量为什么是最优的？
  - 通过分析网络的流量和网络的割(cut)的关系来证明

- 割：网络中的一个边集，如果该边集中所有的边从网络中删除，则网络中不存在从源点到汇点的有向路径
  - 把网络节点分为两个子集：X、$\overline{X}$，其中X包含源点，$\overline{X}$为X的补集、包含汇点，则它们相对应的割记为$C(X,\overline{X})$




$X=\{1\}$, $\bar{X}=\{2,3,4,5,6\}$ $\Rightarrow$ $C(X,\bar{X})=\{(1,2),(1,4)\}$

$X=\{1,2,3,4,5\}$, $\bar{X}=\{6\}$ $\Rightarrow$ $C(X,\bar{X})=\{(3,6),(5,6)\}$

$X=\{1,2,4\}$, $\bar{X}=\{3,5,6\}$ $\Rightarrow$ $C(X,\bar{X})=\{(2,3),(2,5),(4,3)\}$

割的容量：割的边的容量和

# 最大流-最小割定理

- 定理：网络中的最大流量值等于它的最小割的容量
- 证明
  - 假设$C(X,\bar{X})$是容量为$c$的割，$v$为通过该割的流量，则

$$v = \sum_{i\in X,\, j\in\bar{X}} x_{ij} - \sum_{j\in\bar{X},\, i\in X} x_{ji} \le \sum_{i\in X,\, j\in\bar{X}} x_{ij} \le \sum_{i\in X,\, j\in\bar{X}} u_{ij}$$

    - 网络中任意可行流量值不可能超过任意割的容量
    - 最大流量值为所有可行流中最大的，等于最小割的容量

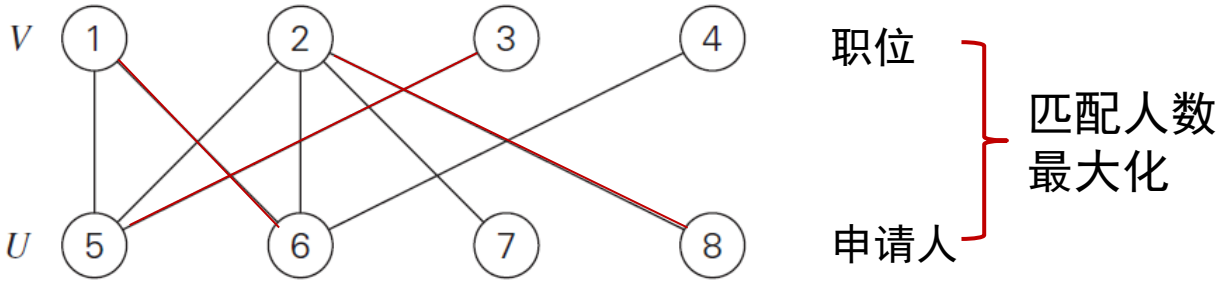
- 采用最短增益路径算法中的标记方法，最后一次迭代，所有从已标记顶点到未标记顶点的边就构成一个最小割

# 目录

- 单纯形法
  (The Simplex Method)
- 最大流量问题
  (The Maximum-flow Problem)
- 二分图的最大匹配
  (Maximum Matching in Bipartite Graphs)
- 稳定婚姻问题
  (The Stable Marriage Problem)

# 二分图的最大匹配

- 二分图
  - 顶点分为两个不相交的集合V和U
  - 图的每条边连接两个集合中各一个顶点
  - 匹配：边的子集，其中任意两条边无相同顶点
  - 最大匹配：包含最多边的匹配



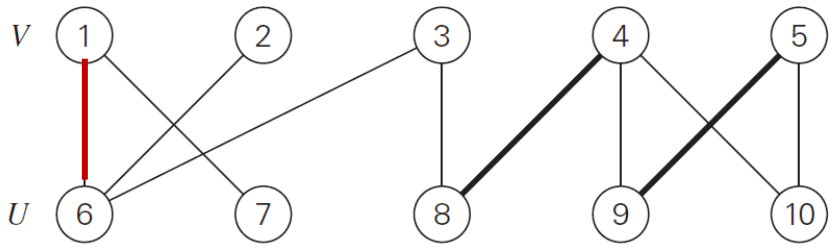

二分图及匹配示意

# 二分图的最大匹配

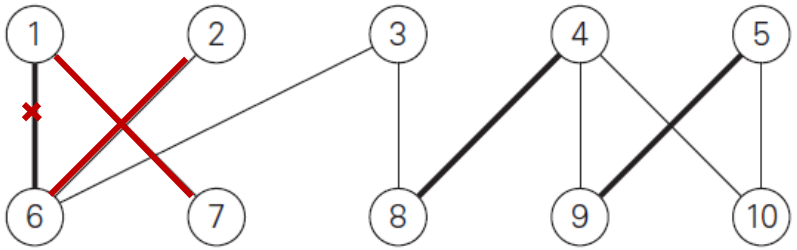
- 自由顶点：集合中未匹配的顶点
- 增益路径：
  - 直接连接V和U中的自由顶点
  - 一头连接V中的自由顶点，另一头连接U中的自由顶点，路径上的边交替出现在E-M和M中（M为某个匹配）；也就是说，路径的第一条边不属于M，第二条边属于M，以此类推，直到最后不属于M的边
- 增益路径的长度总是为奇数，因此，可把位置为奇数的边加入M，把位置为偶数的边从M中删除，就可以生成一个新的匹配，该匹配比M多一条边（即增益）

# 二分图的最大匹配

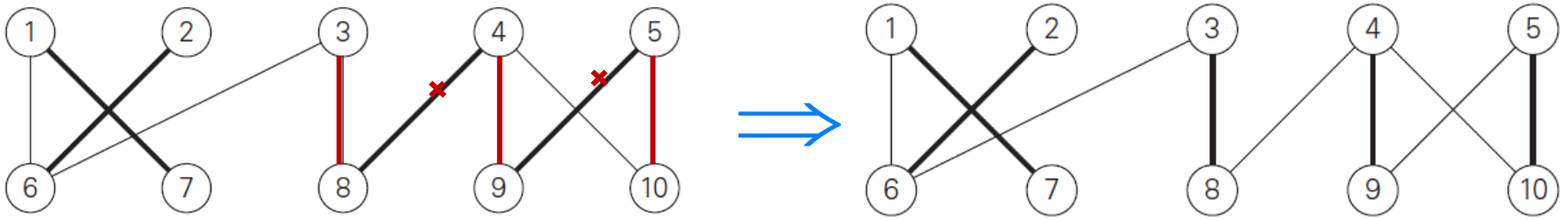
- 增益路径举例




增益路径：1, 6

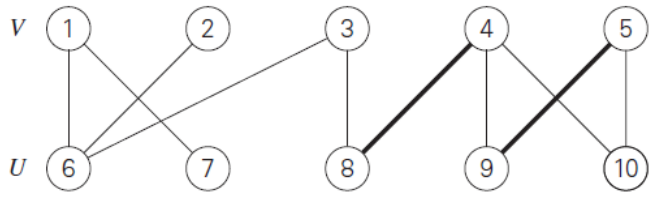



增益路径：2, 6, 1, 7

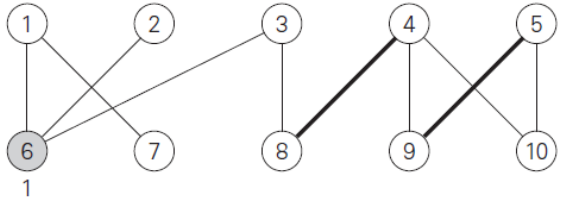



增益路径：3, 8, 4, 9, 5, 10

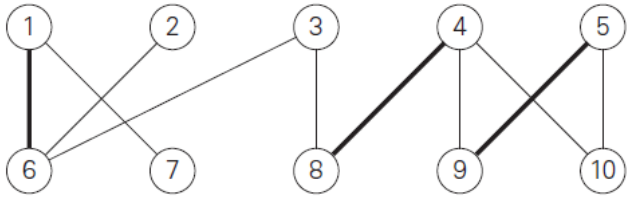
当且仅当M不存在增益路径时，M是最大匹配



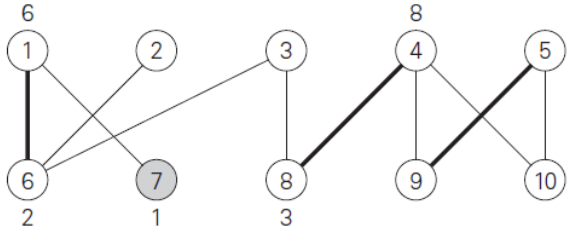

Queue: 1 2 3



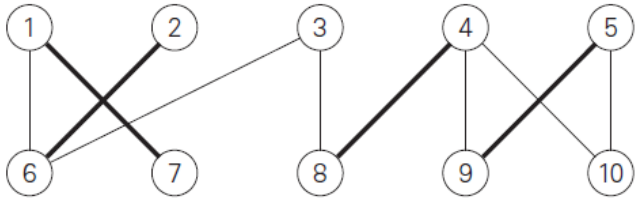

Queue: 1 2 3
↑
Augment from 6



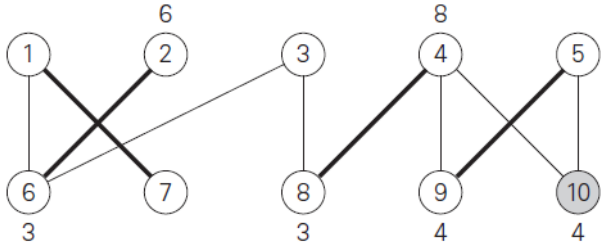

Queue: 2 3



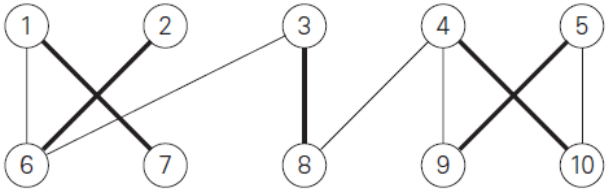

Queue: 2 3 6 8 1 4
↑ ↑ ↑ ↑ ↑
Augment from 7




Queue: 3




Queue: 3 6 8 2 4 9
↑ ↑ ↑ ↑ ↑
Augment from 10




Queue: empty ⇒ maximum matching

算法　MaximumBipartiteMatching($G$)

//用类似广度优先查找的遍历来求二分图的一个最大匹配

//输入：二分图 $G=\langle V,U,E\rangle$

//输出：输入图中一个最大基数匹配

初始化

初始的边集合 $M$ 包含某些合法的匹配(例如空集合)

初始队列 $Q$ 包含 $V$ 中的所有自由顶点(任意序)

**while not** Empty($Q$) **do**

情况1：队列的第一个顶点w在集合V中 →

$w\leftarrow$Front($Q$); Dequeue($Q$)

**if** $w\in V$

**for** 邻接 $w$ 的每个顶点 $u$ **do**

**if** $u$ 是自由顶点

//增益

$M\leftarrow M\cup(w,u)$

$v\leftarrow w$

**while** $v$ 已经被标记 **do**

$u\leftarrow$ 以 $v$ 标记的顶点；　$M\leftarrow M-(v,u)$

$v\leftarrow$ 以 $u$ 标记的顶点；　$M\leftarrow M\cup(v,u)$

删去所有顶点的标记

用 $V$ 中所有的自由顶点重新初始化 $Q$

**break**　　// 退出 **for** 循环

**else**　// $u$ 已经匹配

**if**　$(w,u)\notin M$　**and** $u$ 未标记

用 $w$ 标记 $u$

Enqueue($Q,u$)

情况2：队列的第一个顶点w在集合U中 →

**else**　// $w\in U$ (而且匹配)

用 $w$ 来标记 $w$ 的对偶 $v$

Enqueue($Q,v$)

**return** $M$　//当前匹配是最大匹配

# 目录

- 单纯形法
  (The Simplex Method)
- 最大流量问题
  (The Maximum-flow Problem)
- 二分图的最大匹配
  (Maximum Matching in Bipartite Graphs)
- 稳定婚姻问题
  (The Stable Marriage Problem)

# 稳定婚姻问题

- 问题描述
  - 场景：n个男士的集合Y = $\{m_1, m_2, \ldots, m_n\}$；n个女士的集合X = $\{w_1, w_2, \ldots, w_n\}$；男士（女士）按潜在结婚对象对女士（男士）有一个优先级排序表
  - 婚姻匹配：M为Y和X之间的一个最大匹配，即包含n对(m, w)

男士的优先选择

| | 1st | 2nd | 3rd |
|---|---|---|---|
| Bob: | Lea | Ann | Sue |
| Jim: | Lea | Sue | Ann |
| Tom: | Sue | Lea | Ann |

女士的优先选择

| | 1st | 2nd | 3rd |
|---|---|---|---|
| Ann: | Jim | Tom | Bob |
| Lea: | Tom | Bob | Jim |
| Sue: | Jim | Tom | Bob |

# 稳定婚姻问题

- 问题描述
  - 受阻对(m, w)：在M中，男士m和女士w没有配对，但他们都更倾向于对方而不是M中的伴侣
  - 稳定婚姻匹配：不存在受阻对的匹配

等级矩阵

| | Ann | Lea | Sue |
|---|---|---|---|
| Bob | 2,3 | 1,2 | 3,3 |
| Jim | 3,1 | 1,3 | 2,1 |
| Tom | 3,2 | 2,1 | 1,2 |

考虑匹配M={(Bob, Ann), (Jim, Lea), (Tom, Sue)}来说，(Bob, Lea)是受阻对

# 稳定婚姻问题

**稳定婚姻算法**（Gale-Shapley算法）

输入：有一个 $n$ 个男士的集合和一个 $n$ 个女士的集合，以及每个男士选择女士的优先级和每个女士选择男士的优先级。

输出：一个稳定的婚姻匹配。

**第 0 步：**一开始所有的男士和女士都是自由的。

**第 1 步：**如果有自由男士，从中任选一个然后执行以下步骤：

- **求婚** 选中的自由男士 $m$ 向 $w$ 求婚，$w$ 是他优先列表上的下一个女士(即优先级最高而且之前没有拒绝过他)。
- **回应** 如果 $w$ 是自由的，她接受求婚和 $m$ 配对。如果她不是自由的，她把 $m$ 和她当前的配偶做比较。如果她更喜欢 $m$，她接受 $m$ 的求婚，她的前配偶就变成自由人。否则，她拒绝 $m$ 的求婚，$m$ 还是自由的。

**第 2 步：**返回 $n$ 个匹配对的集合。

**定理** 稳定婚姻算法不超过 $n^2$ 迭代就会终止，并会输出一个稳定的婚姻匹配。

自由男士：
Bob, Jim, Tom

| | Ann | Lea | Sue |
|---|---|---|---|
| Bob | 2, 3 | [1,2] | 3, 3 |
| Jim | 3, 1 | 1, 3 | 2, 1 |
| Tom | 3, 2 | 2, 1 | 1, 2 |

Bob 向 Lea 求婚
Lea 接受了

自由男士：
Jim, Tom

| | Ann | Lea | Sue |
|---|---|---|---|
| Bob | 2, 3 | [1,2] | 3, 3 |
| Jim | 3, 1 | 1, 3 | 2, 1 |
| Tom | 3, 2 | 2, 1 | 1, 2 |

Jim 向 Lea 求婚
Lea 拒绝了

自由男士：
Jim, Tom

| | Ann | Lea | Sue |
|---|---|---|---|
| Bob | 2, 3 | [1,2] | 3, 3 |
| Jim | 3, 1 | 1, 3 | [2,1] |
| Tom | 3, 2 | 2, 1 | 1, 2 |

Jim 向 Sue 求婚
Sue 接受了

自由男士：
Tom

| | Ann | Lea | Sue |
|---|---|---|---|
| Bob | 2, 3 | [1,2] | 3, 3 |
| Jim | 3, 1 | 1, 3 | [2,1] |
| Tom | 3, 2 | 2, 1 | 1, 2 |

Tom 向 Sue 求婚
Sue 拒绝了

自由男士：
Tom

| | Ann | Lea | Sue |
|---|---|---|---|
| Bob | 2, 3 | 1, 2 | 3, 3 |
| Jim | 3, 1 | 1, 3 | [2,1] |
| Tom | 3, 2 | [2,1] | 1, 2 |

Tom 向 Lea 求婚
Lea 用 Tom 替换掉 Bob

自由男士：
Bob

| | Ann | Lea | Sue |
|---|---|---|---|
| Bob | [2,3] | 1, 2 | 3, 3 |
| Jim | 3, 1 | 1, 3 | [2,1] |
| Tom | 3, 2 | [2,1] | 1, 2 |

Bob 向 Ann 求婚
Ann 接受了

# 课后作业

| 章<br>X | 节<br>X. Y | 课后作业题<br>Z | 思考题<br>Z |
|---|---|---|---|
| 10 | 10.1 | 2,5 | 8,10 |
| | 10.2 | 2 | 4,10 |
| | 10.3 | 2 | 6,10 |
| | 10.4 | 4 | 2,10 |
| | 算法设计题<br>（参见下一页PPT） | | |

注：只需上交“课后作业题”（包括算法设计题目）；以“学号姓名_chX.pdf ”规范命名，提交到“学在浙大”指定文件夹。DDL：2024年5月7日

# 课后作业：算法设计题

- 问题描述：假设有来自$n$家不同机构的代表参加一个国际会议。每家机构的代表人数分别为$r_i(i = 1,2,\dots,n)$。会议餐厅共有$m$张餐桌，每张餐桌可容纳$c_i\ (i = 1,2,\dots,m)$个代表就餐。为了使代表们充分交流，希望从同一机构来的代表不在同一个餐桌就餐
  ① 请设计一个高效的算法(用伪代码或图的形式描述)，用于求解满足要求的代表就餐方案
  ② 请将算法应用于如下实例：4家机构，代表人数分别为4, 5, 3, 5；5张餐桌，可容纳人数分别为3, 5, 2, 6, 4。求出就餐方案