计算机导论

中国科学技术大学

University of Science and Technology of China

内容提要

- 图论与图的计算机表示
- 网页排序与Pagerank算法

图论

- 哥尼斯堡七桥问题（1735年）

Leonhard Euler
（1707—1783）

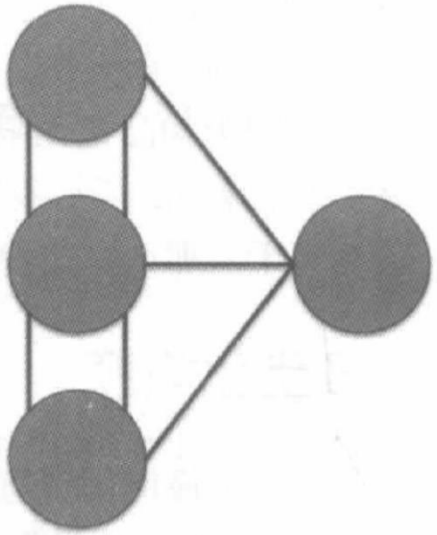
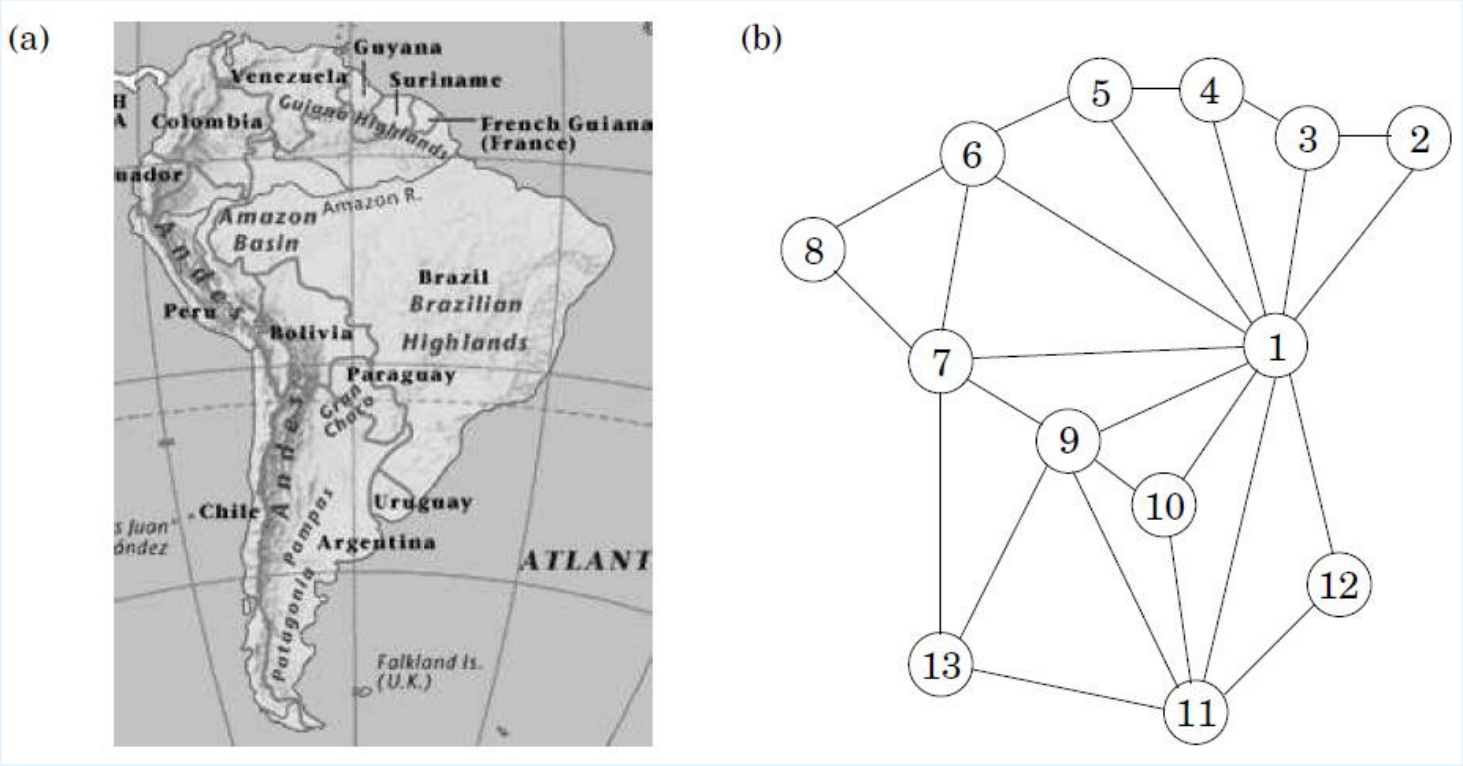


图12-6　哥尼斯堡的七座桥（左）及欧拉的简化地图（右）

图的应用

- 很多问题都可以使用图来描述和刻画
  - 地图染色问题：在地图上将不同的国家涂上不同的颜色，使得相邻的国家被涂上不同的颜色
  - 期末考试安排：大学教务处安排每门课期末考试的时间，要求如果有同学选两门课，则这两门课不能同时进行考试

- 使用“图”来描述问题
  - 地图染色
    - 忽略不相干的信息，比如边界的形状
    - 将国家描述为图的节点，两个国家之间如果有边界，则相应节点之间有边连接。
    - 巴西-节点1
    - 阿根廷-节点11
  - 安排考试
    - 每门课是一个节点
    - 如果有同学选两门课，对应两个点有边连接



- 图的表示
  - 包含点的集合V和边的集合E
  - $V=\{v_1,v_2,\cdots\}$，如果 $|V|=n$，图可以用一个邻接矩阵表示，

$$a_{ij}=\begin{cases}1 & \text{存在从}v_i\text{到}v_j\text{的边}\\ 0 & \text{不存在边}\end{cases}$$

  - 如果图的边有方向（称为有向图），则 $a_{ij}\neq a_{ji}$；如果无方向（称为无向图），则 $a_{ij}=a_{ji}$，矩阵是对称矩阵。
  - 在计算机中存储图，如果采用邻接矩阵，占用空间 $O(n^2)$，这种方法在矩阵密度不高的情况下比较浪费空间
  - 仅仅存储边，比如在每个节点上存储和该节点连接的边，占用空间 $O(|E|)$。称为邻接表

- 如何确定图的存储方式
  - 取决于|V|和|E|之间的关系。
  - 当|E|<|V|-1，图中将出现孤立的点
  - 边数最少的连通的图是树，有|E|=|V|-1
  - |E|最多为$|V|^2$-|V|
  - 当|E|接近|V|时，称图是稀疏的，此时采用邻接表的方式比较节省空间。

  - 例如，如果WWW上每个网页是一个节点，每个超链接是一个边，若有80亿个节点，采用邻接矩阵存储需要几百万T(10^{12})的空间。如果采用邻接表（几百亿超链接），一两个T就够了。

一些图论问题

- 最短路径
- 最短加权路径
- 最大团
- 最小割

pagerank

网页排名

中国科学技术大学

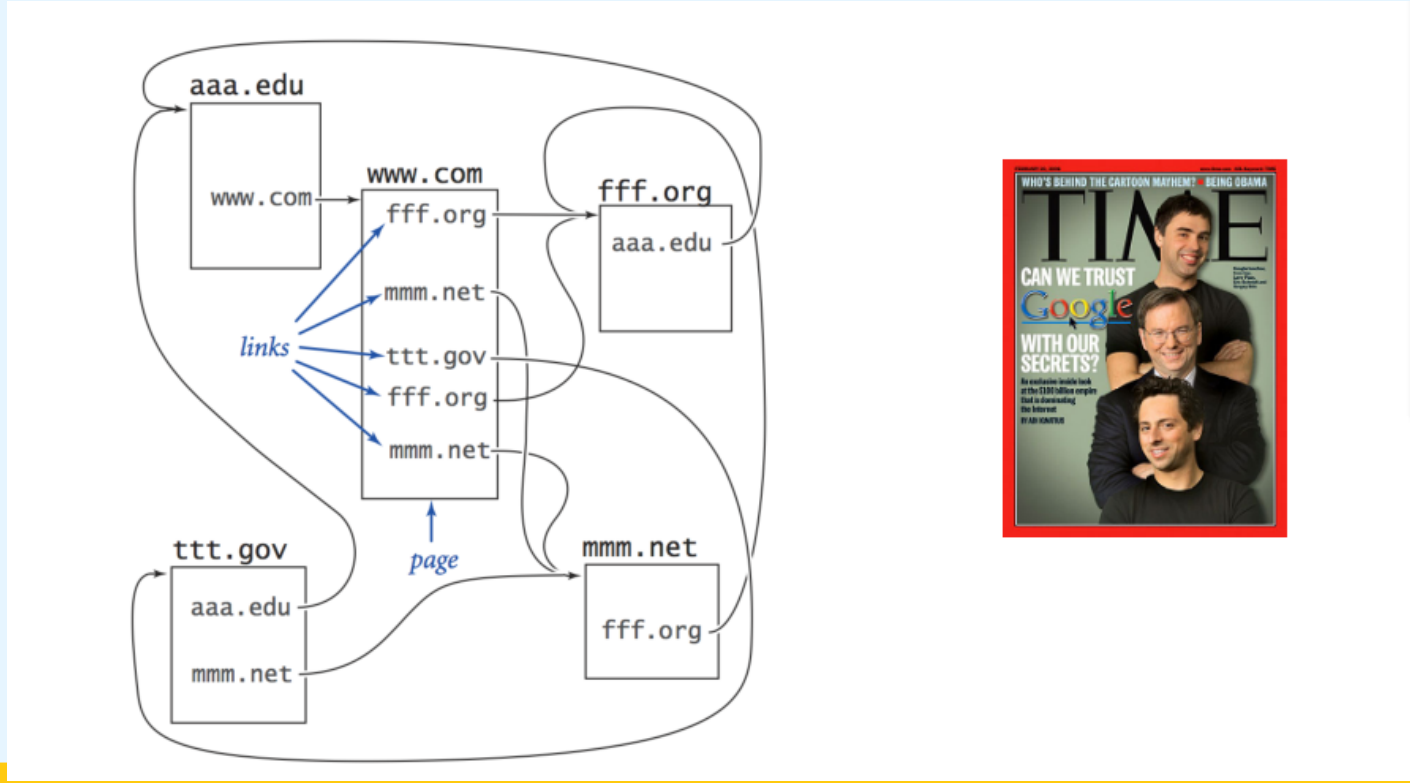
University of Science and Technology of China

网络搜索引擎

- 相关性，搜索内容与结果是否接近
- 价值度，搜索结果对不同的客户是否有用
- 搜索引擎使用的方法
  - 付费广告
  - 手动创建的分类
  - 基于题目，文本，作者的特征提取
  - “流行度”

Google的网页排名算法[Sergey Brin and Larry Page, 1998]

计算基于网页超链接结构的页面热度，使获取世界信息发生了革命性的改变。

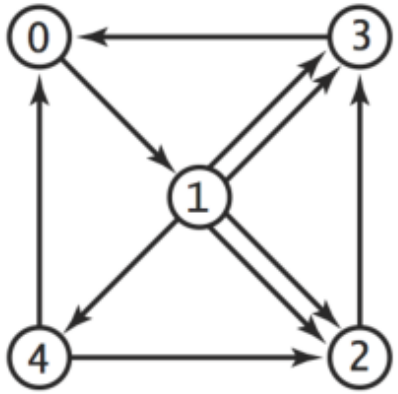


模型

- 定义网页访问行为的参数：
  - 假定90%通过点击，10%通过指定地址来选择跳转到的页面
  - 假定网络页面时被等可能访问的
- 在真实的网络中还有很多需要考虑的因素
  - 选择网页不是等可能的
  - 90-10规则只是一种猜测
  - “返回”按钮不应该考虑在内
  - ……

数据输入

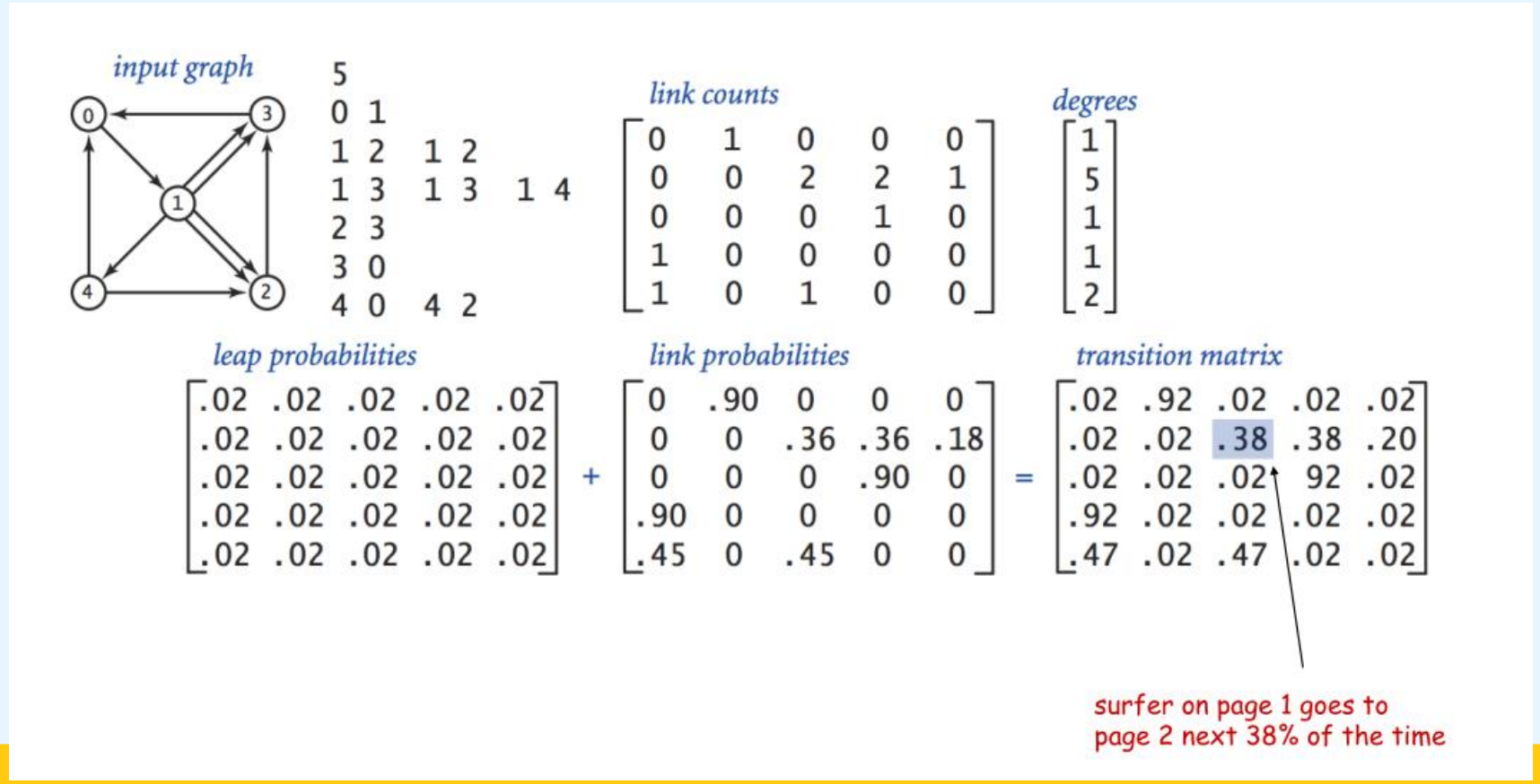
- N个页面，编号0~N-1
- 用一对整数表示一个链接




```
% more tiny.txt
5          ← N
0 1
1 2   1 2
1 3   1 3   1 4
2 3
3 0                  ← links
4 0   4 2
```

矩阵表示

- p[i][j]表示从网页i跳转到网页j的可能性



矩阵表示

```
import stdio
import stdarray

# Read links from standard input and write the corresponding
# transition matrix to standard output. First, process the input
# to count the outlinks from each page. Then apply the 90-10 rule to
# compute the transition matrix. Assume that there are no pages that
# have no outlinks in the input.

n = stdio.readInt()

linkCounts = stdarray.create2D(n, n, 0)
outDegrees = stdarray.create1D(n, 0)

while not stdio.isEmpty():
    # Accumulate link counts.
    i = stdio.readInt()
    j = stdio.readInt()
    outDegrees[i] += 1
    linkCounts[i][j] += 1

stdio.writeln(str(n) + ' ' + str(n))

for i in range(n):
    # Print probability distribution for row i.
    for j in range(n):
        # Print probability for column j.
        p = (.90 * linkCounts[i][j] / outDegrees[i]) + (.10 / n)
        stdio.writef('%8.5f', p)
    stdio.writeln()
```

Monte Carlo模拟

- 从网页0开始
- 根据转化矩阵不断选择下一个页面
- 计算访问每个页面的频率

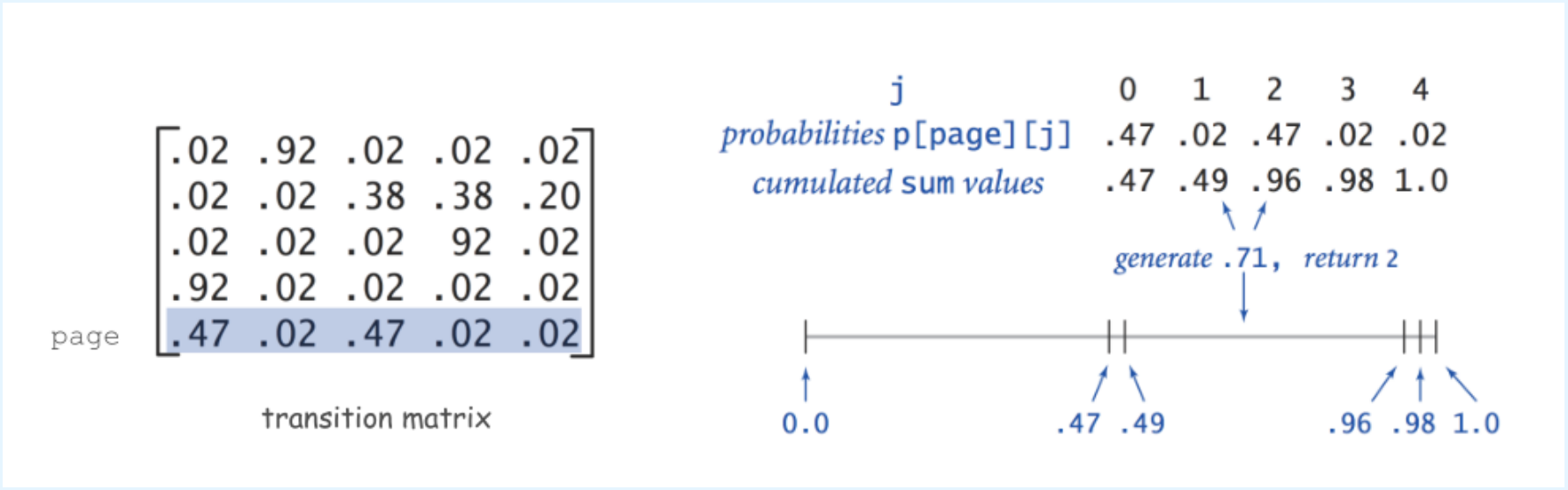
page

$$\begin{bmatrix} .02 & .92 & .02 & .02 & .02 \\ .02 & .02 & .38 & .38 & .20 \\ .02 & .02 & .02 & 92 & .02 \\ .92 & .02 & .02 & .02 & .02 \\ .47 & .02 & .47 & .02 & .02 \end{bmatrix}$$

transition matrix

Monte Carlo模拟

- 随机访问，如何选择下一个页面？
  - 当前页面号对应的行给出访问其他页面号的概率
  - 计算行概率之和
  - 生成0.0~1.0之间的一个随机数r
  - 选择r所在区间号j作为下一个页面



Monte Carlo模拟

- 随机访问代码实现

```
# Make one random move.
r = random.random()
total = 0.0
for j in range(0, n):
    # Find interval containing r.
    total += p[page][j]
    if r < total:
        page = j
        break
```

Monte Carlo模拟

- 代码实现part1

```
import stdio
import stdarray
import sys
import random

# Accept an integer moves as a command-line argument. Read a
# transition matrix from standard input. Perform moves moves as
# prescribed by the transition matrix, and write to standard output
# the relative frequency of hitting each page.

moves = int(sys.argv[1])

n = stdio.readInt()
stdio.readInt() # Discard the second int of standard input.

# Read the transition matrix from standard input.
# p[i][j] is the probability that the surfer moves from
# page i to page j.
p = stdarray.create2D(n, n, 0.0)
for i in range(n):
    for j in range(n):
        p[i][j] = stdio.readFloat()
```

Monte Carlo模拟

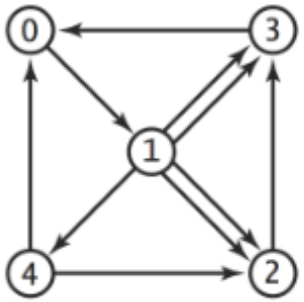
- 代码实现part2

```
# Perform the simulation, thus computing the hits array.
# hits[i] is the number of times the surfer hits page i.
hits = stdarray.create1D(n, 0)
page = 0  # Start at page 0.
for i in range(moves):
    # Make one random move.
    r = random.random()
    total = 0.0
    for j in range(0, n):
        # Find interval containing r.
        total += p[page][j]
        if r < total:
            page = j
            break
    hits[page] += 1

# Write the page ranks.
for v in hits:
    stdio.writef("%8.5f", 1.0 * v / moves)
stdio.writeln()
```

Monte Carlo模拟

- 可视化结果
  - 访问频率收敛于页面排名



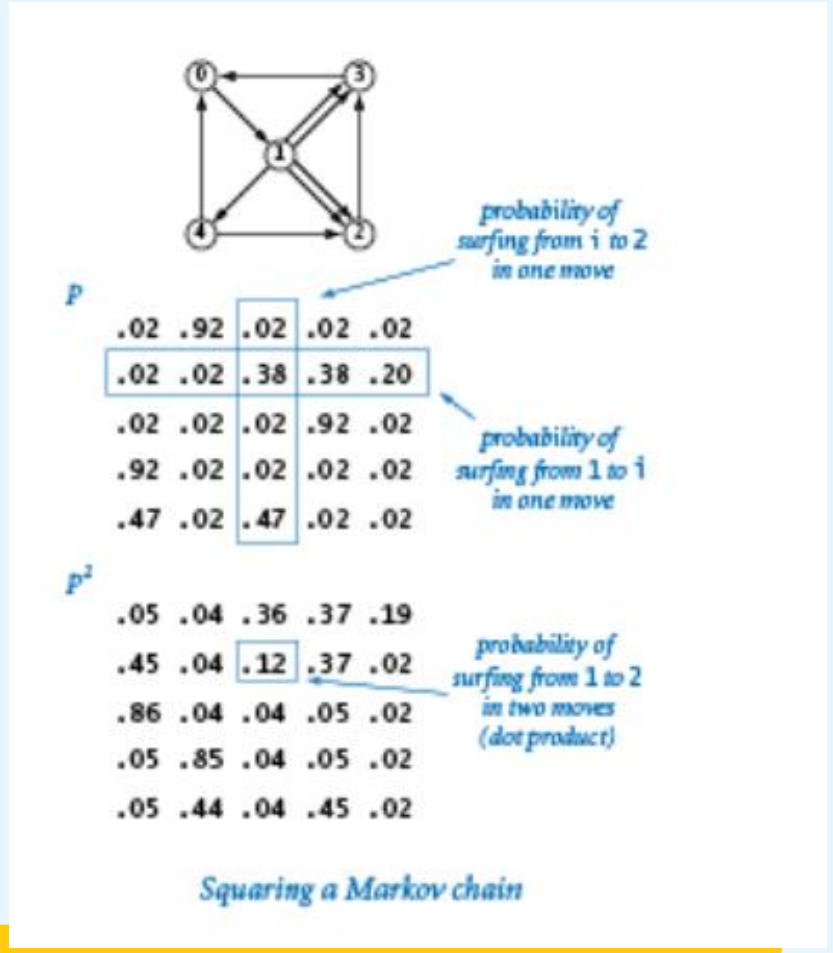
0 .27
1 .27
3 .25
2 .15
4 .07

$$\left[\frac{428{,}671}{1{,}570{,}055}, \frac{417{,}205}{1{,}570{,}055}, \frac{229{,}519}{1{,}570{,}055}, \frac{388{,}162}{1{,}570{,}055}, \frac{106{,}498}{1{,}570{,}055}\right]$$

混合马尔科夫模型

模拟用户随机浏览网页的行为耗时大，可以用线性代数的方法更有效率地求解

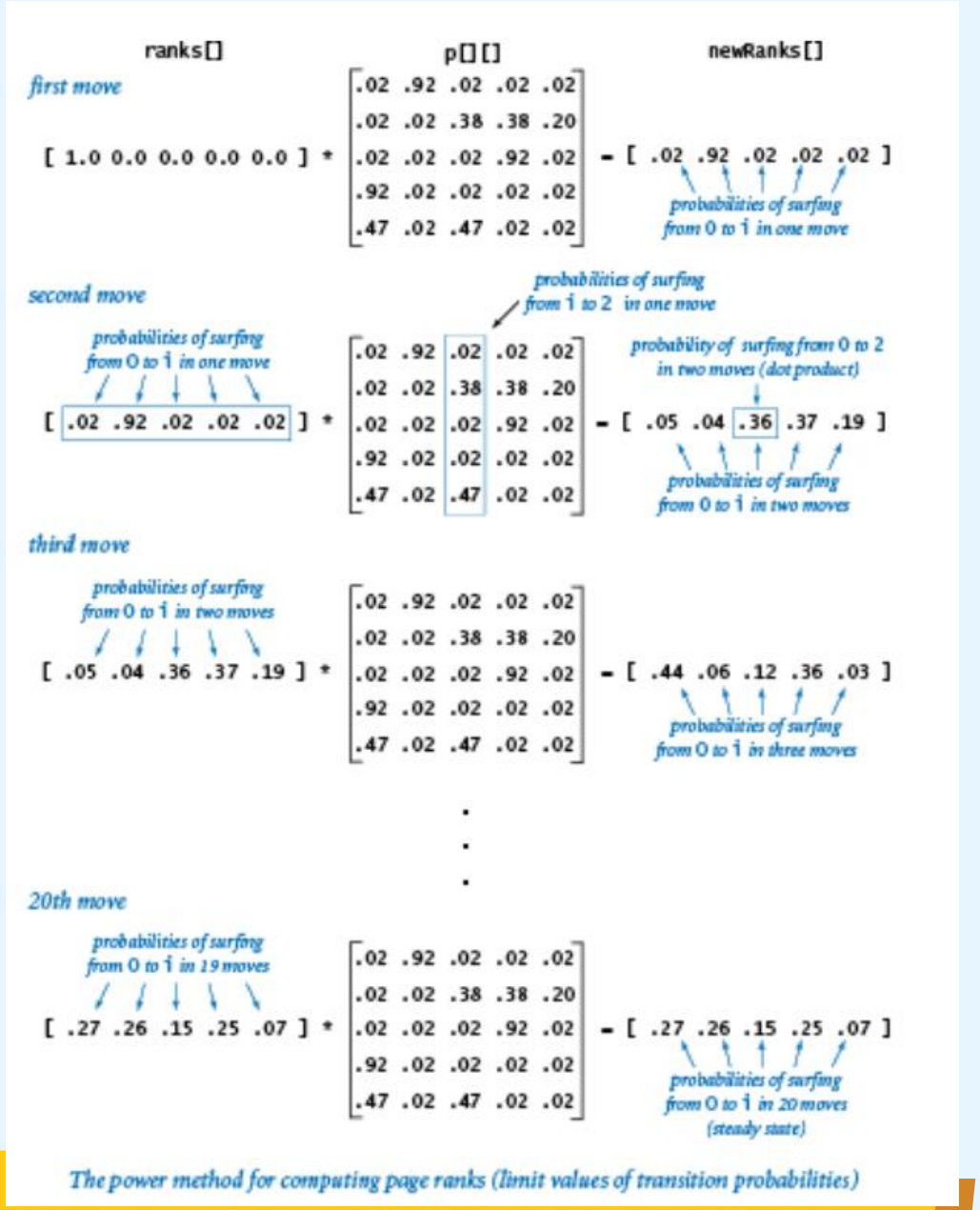
- 马尔科夫链求平方
  - 计算经过两步从页面i到页面j的概率，即计算i到k，再从k到j，这样相当于对矩阵求平方



Squaring a Markov chain

混合马尔科夫模型

- 矩阵幂计算
  - 矩阵幂运算代价很高，可以转化为只计算向量的乘积



The power method for computing page ranks (limit values of transition probabilities)

Monte Carlo method（蒙特卡罗模拟）

Stan Ulam, Feynman and von Neumann

Pascal问题

“连续掷一对骰子24次得到两个6”，是否可以下注？

1 - (35/36)^24 = 0.49

```
#Figure 16.1
def rollDie():
    return random.choice([1,2,3,4,5,6])

def checkPascal(numTrials):
    """Assumes numTrials an int > 0
       Prints an estimate of the probability of winning"""
    numWins = 0
    for i in range(numTrials):
        for j in range(24):
            d1 = rollDie()
            d2 = rollDie()
            if d1 == 6 and d2 == 6:
                numWins += 1
                break
    print('Probability of winning =', numWins/numTrials)
```

Blaise Pascal (1623-1662)

Pierre de Fermat(1601-1665)

双骰子（Double Dice）博弈

- Craps游戏用的骰子就是我们常见的六面骰子，玩家丢出两个骰子，骰子滚动停止之后，朝上的点数相加之后，决定输赢。
  - 过线投注：投出7或11，赢；投出2，3，12，输；投出其他点（4，5，6，8，9，10），继续投，投出与第一次相同的点赢，如先投出7则输。
  - 不过线：投出2或3，赢；投出7或11，输；投出12则平均；投出其他点（4，5，6，8，9，10），继续投，先投出与第一次相同的点则输，如先投出7则赢。

搜索引擎优化

- SEO（Search Engine Optimization）
- 利用搜索引擎的规则提高网站在有关搜索引擎内的自然排名。目的是让其在行业内占据领先地位，获得品牌收益。很大程度上是网站经营者的一种商业行为，将自己或自己公司的排名前移。

- 黑帽（black hat）：通过作弊手法欺骗搜索引擎和访问者，会遭到搜索引擎惩罚的手段
- 白帽（white hat）：通过正规技术和方式，且被搜索引擎所接受的SEO技术

感谢关注~！

中国科学技术大学

University of Science and Technology of China