随机算法

朱同鑫

本章内容

- 随机算法的基本概念
- 随机数值算法
  ——计算π值、计算定积分
- Las Vegas算法
  ——第k小元素问题
- Sherwood随机化方法
- Monte Carlo算法
  ——字符串相等判断问题
- 子串匹配问题
- 最近点对随机算法
- 素数测试

随机算法的基本概念

- 从一个例子看随机算法的作用
  - 【例】判断函数 $f(x_1, x_2, \cdots, x_n)$在区域 D中是否恒为0， f很复杂，不能数学简化，给出确定性结果很麻烦
    - 若随机产生一个n维坐标$(r_1, r_2, \cdots, r_n) \in D$，代入得$f(r_1, r_2, \cdots, r_n) \neq 0$，则可判定区域$D$内$f$不恒为0
    - 若对很多个随机产生的坐标进行测试，结果次次均为0，则可说$f \neq 0$的概率是非常小（测试数量越多，可信度越高）

结论：有不少问题，目前只有效率很差的确定性求解算法， 但用随机算法去求解，可以很快地获得相当可信的结果。

随机算法的基本概念

- 随机算法
  - 随机算法是一种使用概率和统计方法在其执行过程中对于下一计算步骤作出随机选择的算法
- 随机算法的优越性
  - 对于有些问题：算法简单
  - 对于有些问题：时间复杂性低
  - 对于有些问题：同时兼有简单和时间复杂性低

随机算法的基本概念

- 随机算法的随机性
  - 对于同一实例的多次执行，效果可能完全不同
  - 时间复杂性是一个随机变量
  - 解的正确性和准确性也是随机的
- 随机算法的分类
  - 随机数值算法
  - Monte Carlo算法
  - Las Vegas算法
  - Sherwood算法

随机算法的基本概念

- 随机数值算法
  - 主要用于数值问题求解
  - 算法的输出往往是近似解
  - 近似解的精确度与算法执行时间成正比
- Monte Carlo算法
  - 主要用于求解需要准确解的问题
  - 算法可能给出错误的解
  - 获得正确解的概率与算法执行时间成正比

随机算法的基本概念

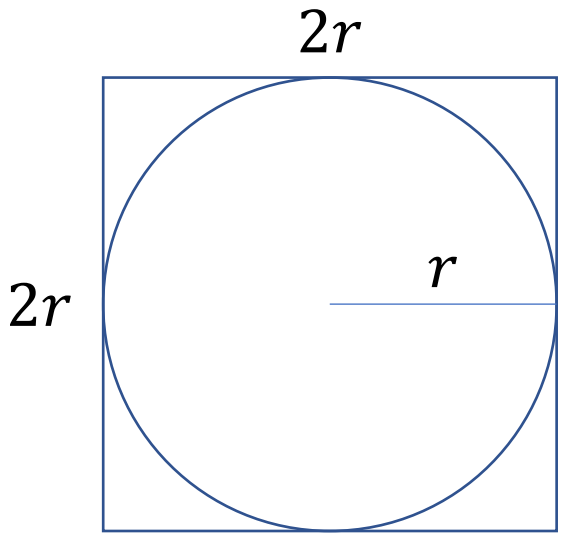
- Las Vegas算法
  - 可能找不到解， 但一旦找到一个解，该解一定是正确的
  - 找到解的概率与算法执行时间成正比
  - 增加对问题反复求解次数, 可使求解无效的概率任意小
- Sherwood算法
  - 一定能够求得一个正确解
  - 确定算法的最坏与平均复杂性差别大时，加入随机性，即得到Sherwood算法
  - 消除最坏行为与特定实例的联系

随机算法的基本概念

- 随机算法分析的特征
  - 仅依赖于随机选择：不依赖于输入的分布
  - 确定算法的平均复杂性分析：依赖于输入的分布
- 随机算法分析的目标
  - 平均时间复杂性：时间复杂性随机变量的期望
  - 获得正确解的概率/获得优化解的概率
  - 解的精确度分析

随机数值算法——计算π值

- 数学基础
  - 设有一个半径为 r 的圆及其外切四边形
  - 向正方形随机地投掷点，投掷点落入圆内的概率为 $\frac{\pi r^2}{4r^2} = \frac{\pi}{4}$
  - 投掷 n 个点，设 k 个点落入圆内
  - 用 k/n 逼近 $\pi/4$，即 $k/n \approx \pi/4$，于是 $\pi \approx 4k/n$



随机数值算法——计算π值

- 随机算法伪代码

Computing π (n)

$k = 0;$

for $i = 1$ ***to*** n ***do***

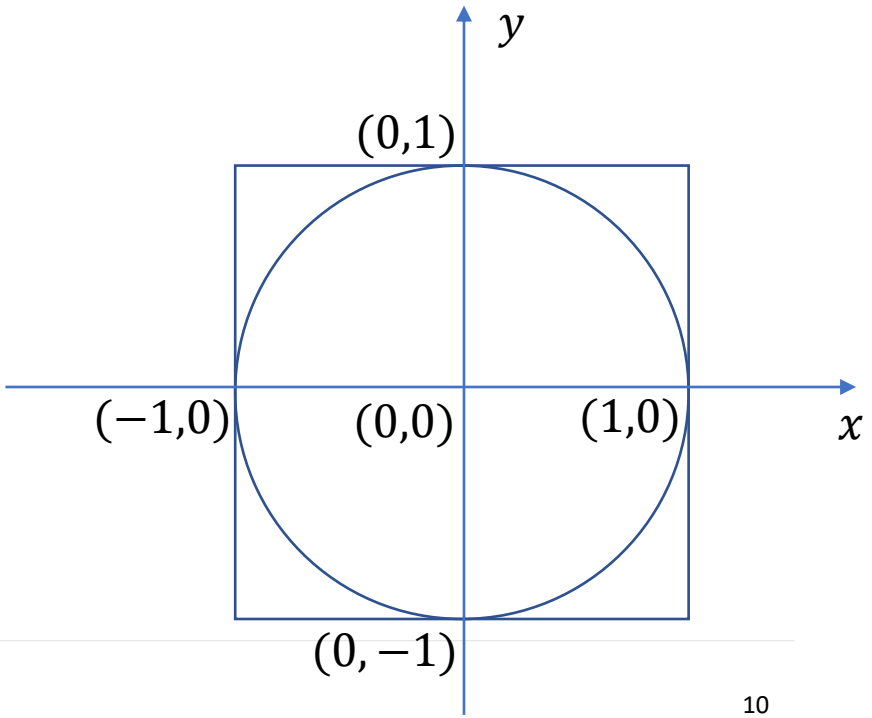
随机地产生正方形中的一点(x, y);

if $x^2 + y^2 \leq 1$ ***then***

$k = k + 1;$

return $4k/n;$

解的精确度随着随机样本大小n增加而增加



时间复杂度$O(n)$

n不是输入的大小，而是随机样本的大小

随机数值算法——计算定积分

- 问题：计算积分$\int_a^b g(x)dx$
- 数学基础
  - 强大数定律：假定$\{s(x)\}$是相互独立同分布的随机变量序列，如果它们有有限的数学期望$E(s(x)) = a$，则

$$\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} s(x_i) = a = \mathrm{E}\big(\mathrm{s}(\mathrm{x})\big)$$

随机数值算法——计算定积分

- 计算积分
  - 令$f(x)=\frac{1}{b-a}$为区间$[a,b]$上的一组独立、同分布的随机变量$\{s(x)\}$的概率密度函数
    - 当$\{s(x)\}$为离散随机变量时，期望$E\big(s(x)\big)=\sum s(x)f(x)$
    - 当$\{s(x)\}$为连续随机变量时，期望$E\big(s(x)\big)=\int_a^b s(x)f(x)\,dx$
  - 令$g(x)=s(x)f(x)$，那么期望$E\big(s(x)\big)=\int_a^b g(x)\,dx$
  - 则，$\int_a^b g(x)\,dx=E\big(s(x)\big)=\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}s(x_i)=\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}g(x_i)/f(x_i)$

$$=\lim_{n\to\infty}\frac{b-a}{n}\sum_{i=1}^{n}g(x_i)$$

强大数定律：$\lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^{n}s(x_i)=a=\mathrm{E}\big(\mathrm{s(x)}\big)$

随机数值算法——计算定积分

- 随机算法伪代码

Computing-Definite-Integration (a, b, n)

$R = 0;$

for $i = 1$ ***to*** n ***do***

　　随机地产生$[a, b]$ 中的点X;

　　$R = R + g(x);$

return $(b - a)R/n;$

$$\int_a^b g(x)\,dx = \lim_{n\to\infty} \frac{b-a}{n} \sum_{i=1}^{n} g(x_i)$$

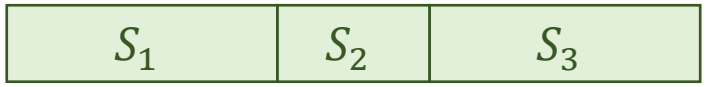
解的精确度随着随机样本大小n增加而增加

时间复杂度$O(n)$

n不是输入的大小，而是随机样本的大小

Las Vegas算法——第k小元素问题

- 问题定义
  - 输入：$S=\{x_1,x_2,\cdots,x_n\}$，整数k，$1\le k\le n$
  - 输出：S中第k小元素
- 随机算法
  - 在n个数中随机找一个数x_i，然后将剩余$n-1$个数与x_i比价，分别放入三个数组S_1 (元素均$<x_i$)、 S_2 (元素均$=x_i$)、 S_3 (元素均$>x_i$)中
    - 当 $k\le|S_1|$ 时，调用 $Select(S_1,k)$
    - 当 $|S_1|<k\le|S_1|+|S_2|$ 时，返回x_i
    - 当 $k>|S_1|+|S_2|$ 时，调用 $Select(S_3,k-|S_1|-|S_2|)$



Las Vegas算法——第k小元素问题

- 性能分析

定理.

若以等概率方法在n个数中随机取数，则该算法用到的比较次数的期望值不超过$4n$。

证明.

设$C(n)$是输入规模为n时，算法比较次数的期望值，并设取到任意数的概率相同，假设取到第j小的数。

- 若$j > k$，需要调用$Select(S_1, k)$，而$|S_1| = j - 1$，比较次数期望为$C(j-1)$
- 若$j = k$，无需继续进行比较，直接返回第$j = k$个元素
- 若$j < k$，需要调用$Select(S_3, k - |S_1| - |S_2|)$，而$|S_3| = n - j$，比较次数期望为$C(n-j)$

Las Vegas算法——第k小元素问题

- 性能分析

定理.

若以等概率方法在n个数中随机取数，则该算法用到的比较次数的期望值不超过$4n$。

证明.

$$C(n) = n + \frac{1}{n}\left(\sum_{j=k+1}^{n} C(j-1) + \sum_{j=1}^{k-1} C(n-j)\right)$$

$$= n + \frac{1}{n}\left(\sum_{i=k}^{n-1} C(i) + \left(C(n-1) + C(n-2) + \cdots + C(n-k+1)\right)\right)$$

$$= n + \frac{1}{n}\left(\sum_{i=k}^{n-1} C(i) + \sum_{i=n-k+1}^{n-1} C(i)\right)$$

$$\leq n + \frac{1}{n}\left(\sum_{i=n-\frac{n}{2}+1}^{n-1} C(i) + \sum_{i=\frac{n}{2}}^{n-1} C(i)\right)$$

由于$C(i)$是非减函数，即$i < j$时，总有$C(i) \leq C(j)$，$C(n)$在$k = \lceil n/2 \rceil$时取得最大值

心求至善 **归纳法证明$C(n) \leq 4n$**

Las Vegas算法——第k小元素问题

- 性能分析

定理.

若以等概率方法在n个数中随机取数，则该算法用到的比较次数的期望值不超过$4n$。

证明.

归纳法证明$C(n) \le n + \frac{1}{n}\left(\sum_{i=n-\frac{n}{2}+1}^{n-1} C(i) + \sum_{i=\frac{n}{2}}^{n-1} C(i)\right) \le 4n$

1. 当$n = 1$时，$C(1) \le 4$显然成立
2. 假设当$n \le m-1$时（$m \ge 2$），$C(n) \le 4n$成立，往证当$n = m$时，$C(n) \le 4n$成立

$$C(m) \le m + \frac{1}{m}\left(\sum_{i=m-\frac{m-1}{2}+1}^{m-1} 4i + \sum_{i=\frac{m}{2}}^{m-1} 4i\right) \le m + \frac{1}{m}\left(8\sum_{i=\frac{m}{2}}^{m-1} i\right) \le 4m$$

Sherwood随机化方法

- 一般过程
  - 若问题已经有平均性质较好的确定性算法，但是该算法在最坏情况下效率不高
  - 引入一个随机数发生器（通常服从均匀分布），将一个确定性算法改成一快个随机算法
  - 【例】速排序
    - 每次选择一个基准数，比它小的放左边，大的放右边，如此递归
    - 平均效率很好，但是最坏情况下，每次选择的基准若都是最小的，或是最大的，算法效率不高
    - 可随机预处理(洗牌)，使输入均匀分布，再运行算法

Monte Carlo算法——字符串相等判断

- 问题描述：
  - 设A处有一个长字符串X（e.g. 长度为10^6），B处也有一个长字符串Y，A将X发给B，由B判断是否有$X = Y$
- 判断方法：
  - 首先，A将X的长度发给B，若B判断该长度与Y的长度不等，则$X \neq Y$
  - 若长度相等，采用“取指纹”的判断方法
    - A对X进行处理，取出X的“指纹”，将指纹发给B
    - 由B检查X的“指纹”是否等于Y的“指纹”
    - 若取k次指纹(每次指纹不同)，每次两者结果均相同，则认为$X = Y$。
    - 随着k的增大，误判率可趋于0

Monte Carlo算法——字符串相等判断

- 字符串取“指纹”方法
  - 令字符串X的二进制编码长度为n，则 $X < 2^n$，取$f(X) \equiv X(mod\ p)$作为X的指纹，其中p是小于$2n^2$的素数，
  - p的二进制长度：$\log_2 p \le \lfloor \log_2 2n^2 \rfloor + 1 = O(\log_2 n)$，$f(X)$的二进制长度≤ $\log_2 p$，即传输长度可大大缩短
  - 【例】X的二进制是10^6位，即$n = 10^6$，则$p < 2 \times 10^{12} \approx 2^{40.8631}$
    - $f(X)$的位数不超过41位，传输一次指纹$f(X)$可节省约2.5万倍
    - 根据下面所做的分析，错判率小于$\frac{1}{10^6}$
    - 若取5次指纹，错判率小于$\frac{1}{10^{30}}$

Monte Carlo算法——字符串相等判断

- 错判率分析
  - B接到指纹$f(X)$后与$f(Y)$比较，
    - 若$f(X) \neq f(Y)$，当然有$X \neq Y$。
    - 若$f(X) = f(Y)$而$X \neq Y$，此时是一个错误匹配。
  - 错误匹配的概率有多大?
    - 随机取一个小于$2n^2$的素数p，则对于给定的X和Y，
    - 错判率

$$P_{fail} = \frac{X \neq Y, \text{但使得} f(X) = f(Y) \text{的小于} 2n^2 \text{的素数的个数}}{\text{小于} 2n^2 \text{的素数的总个数}}$$

Monte Carlo算法——字符串相等判断

- 错判率分析
  - 错判率：$P_{fail}=\frac{X\neq Y,\text{但使得}f(X)=f(Y)\text{的小于}2n^2\text{的素数的个数}}{\text{小于}2n^2\text{的素数的总个数}}\leq\frac{\pi(n)}{\pi(2n^2)}\leq\frac{1}{n}$

数论定理1：设$\pi(n)$是小于n的素数个数，则$\pi(n)\approx\frac{n}{\ln n}$

则小于$2n^2$的素数的总个数为：$\pi(2n^2)\approx\frac{n^2}{ln\ n}$

数论定理2：设$a\equiv b(mod\ p)\ \ iff\ \ p$能整除$|a-b|$

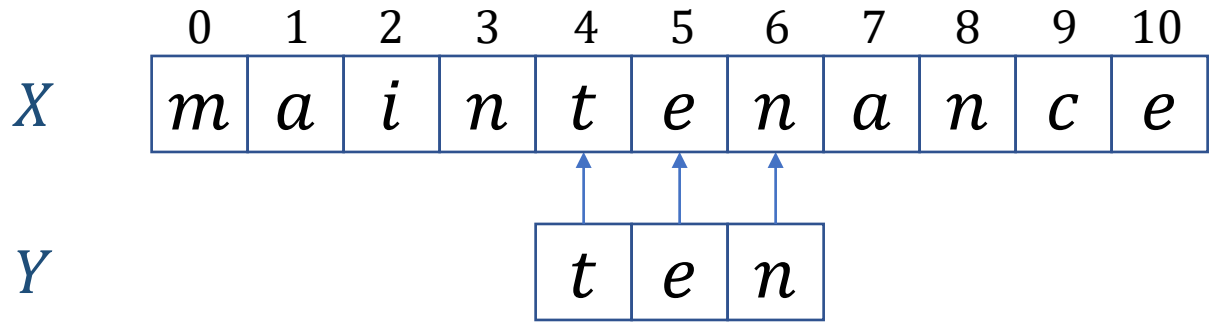
使得$f(X)=f(Y)$的素数的个数=能够整除$|X-Y|$素数的个数

数论定理3：若$a<2^n$，则能整除a的素数个数不超过$\pi(n)$个

$|X-Y|<\max\{X,Y\}\leq 2^n-1$，所以能整除整除$|X-Y|$的素数个数$\leq\pi(n)$

子串匹配问题

- 问题定义：
  - 输入：目标串$X = “x_1 x_2 \cdots x_n“$，模式串$Y = ”y_1 y_2 \cdots y_m“$，$m \leq n$
  - 输出：判断模式串Y是否为目标串X的子串
- 问题实例
  - $X = “maintenance”$，$Y = "ten"$



Y是X的子串，且在X中的位置为4

子串匹配问题

- 随机算法
  - 记$X(j) = x_j x_{j+1} \cdots x_{j+m-1}$
  - 逐一比较$X(j)$的指纹$f\big(X(j)\big) \equiv X(j)(mod\ p)$与$Y$的指纹$f(Y) \equiv Y(mod\ p)$，$1 \le j \le n - m + 1$， p是小于$2mn^2$的素数
  - $f\big(X(j+1)\big)$可根据$f\big(X(j)\big)$计算，故算法可很快完成

子串匹配问题

- 随机算法

数论定理4：$(xy+z)\ (mod\ p) = (x(y\ mod\ p)+z)\ (mod\ p)$

  - $f(X(j+1))$可根据$f(X(j))$计算
    - 不失一般性，令X和Y都是0-1串（二进制编码）
    - $f(X(j+1)) = (x_{j+1}\cdots x_{j+m-1}x_{j+m})\ (mod\ p)$
    - $= (2(x_{j+1}\cdots x_{j+m-1}) + x_{j+m})\ (mod\ p)$
    - $= (2(x_{j+1}\cdots x_{j+m-1}) + 2^m x_j - 2^m x_j + x_{j+m})\ (mod\ p)$
    - $= (2(x_j x_{j+1}\cdots x_{j+m-1}) - 2^m x_j + x_{j+m})\ (mod\ p)$
    - $= \left(2\left((x_j x_{j+1}\cdots x_{j+m-1})(mod\ p)\right) - 2^m x_j + x_{j+m}\right)\ (mod\ p)$
    - $= \boxed{(2f(X(j)) - (2^m\ mod\ p)x_j + x_{j+m})}\ (mod\ p)$

被余数都在p附近，与p相差很小（$[-(p-1), 2p-1]$），只需一两次加减法即可求余

子串匹配问题

- 错判率分析
  - 当$Y \neq X(j)$，但$f(Y) = f\big(X(j)\big)$时产生错误
  - 而$f(Y) = f\big(X(j)\big)$当且仅当p能整除$|Y - X(j)|$
    - 若p能整除$|Y - X(j)|$，p自然也能整除$Z = |Y - X(1)| \times \cdots \times |Y - X(n-m+1)|$
    - 因为$|Y - X(j)| \leq 2^m$，
    - 所以$Z = |Y - X(1)| \times \cdots \times |Y - X(n-m+1)| \leq (2^m)^{n-m+1} \leq 2^{mn}$

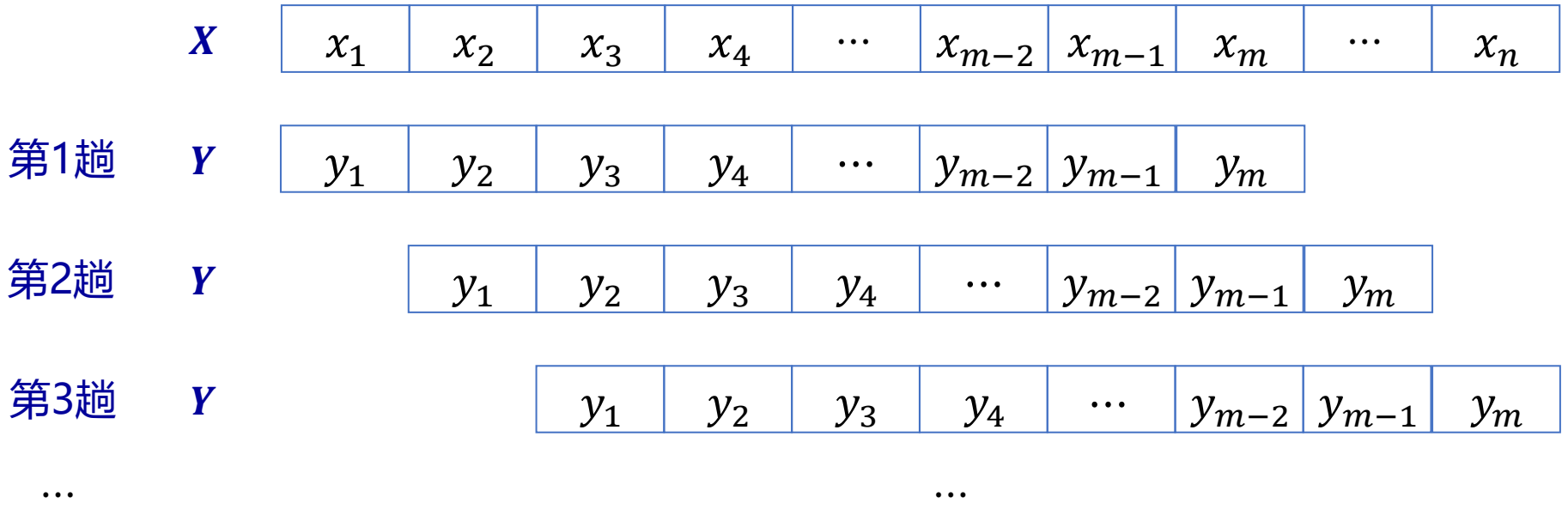
数论定理3：若$a < 2^n$，则能整除a的素数个数不超过$\pi(n)$个

  - 所以能整除Z的素数个数不超过$\pi(mn)$个

$$P_{fail} = \frac{X\text{不包含}Y\text{，但能整除}Z\text{的素数的个数}}{\text{小于}2mn^2\text{的素数的总个数}} \leq \frac{\pi(mn)}{\pi(2mn^2)} = \frac{1}{n}$$

子串匹配问题

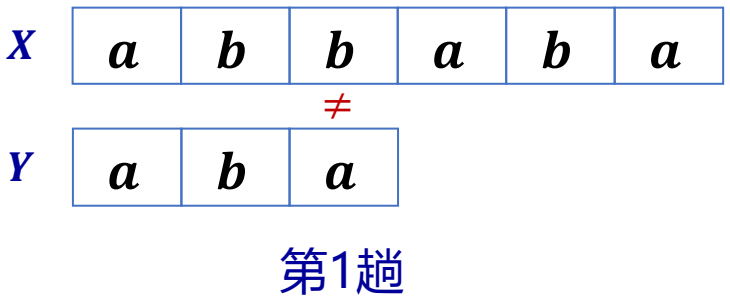
- 确定算法1——穷举模式匹配算法
  - 匹配失败，目标串X回溯，模式串Y从头开始



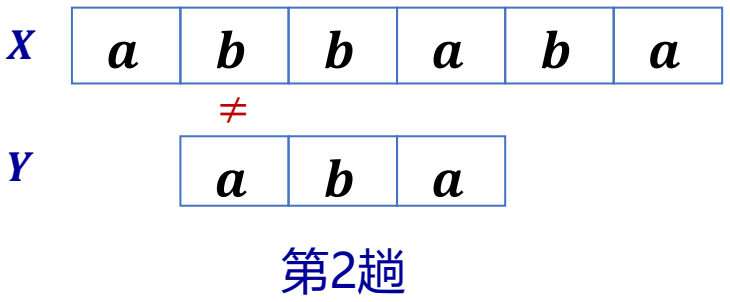
时间复杂度$O(mn)$

子串匹配问题

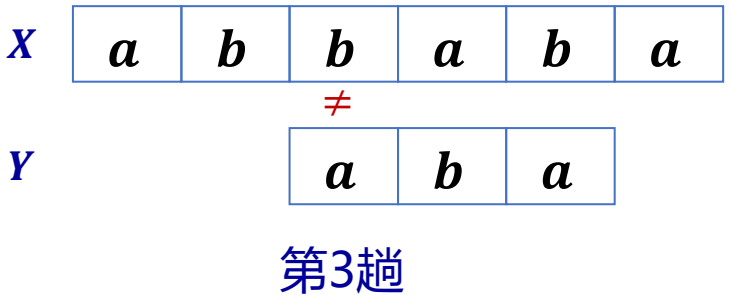
- 穷举模式匹配算法
  - 匹配失败，目标串X回溯，模式串Y从头开始



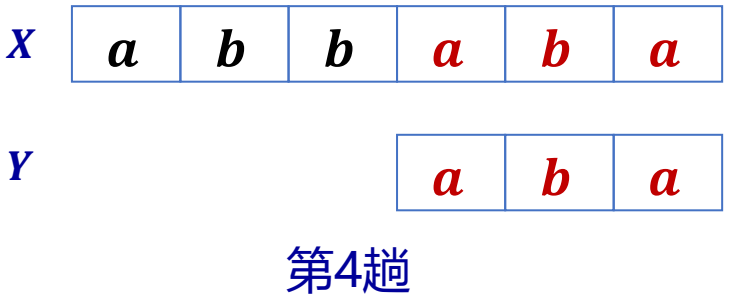
第1趟



第2趟



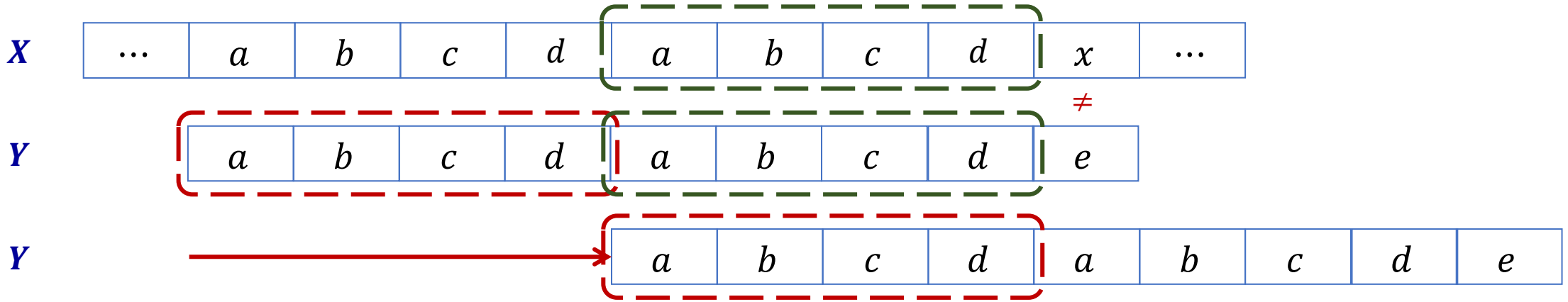
第3趟



第4趟

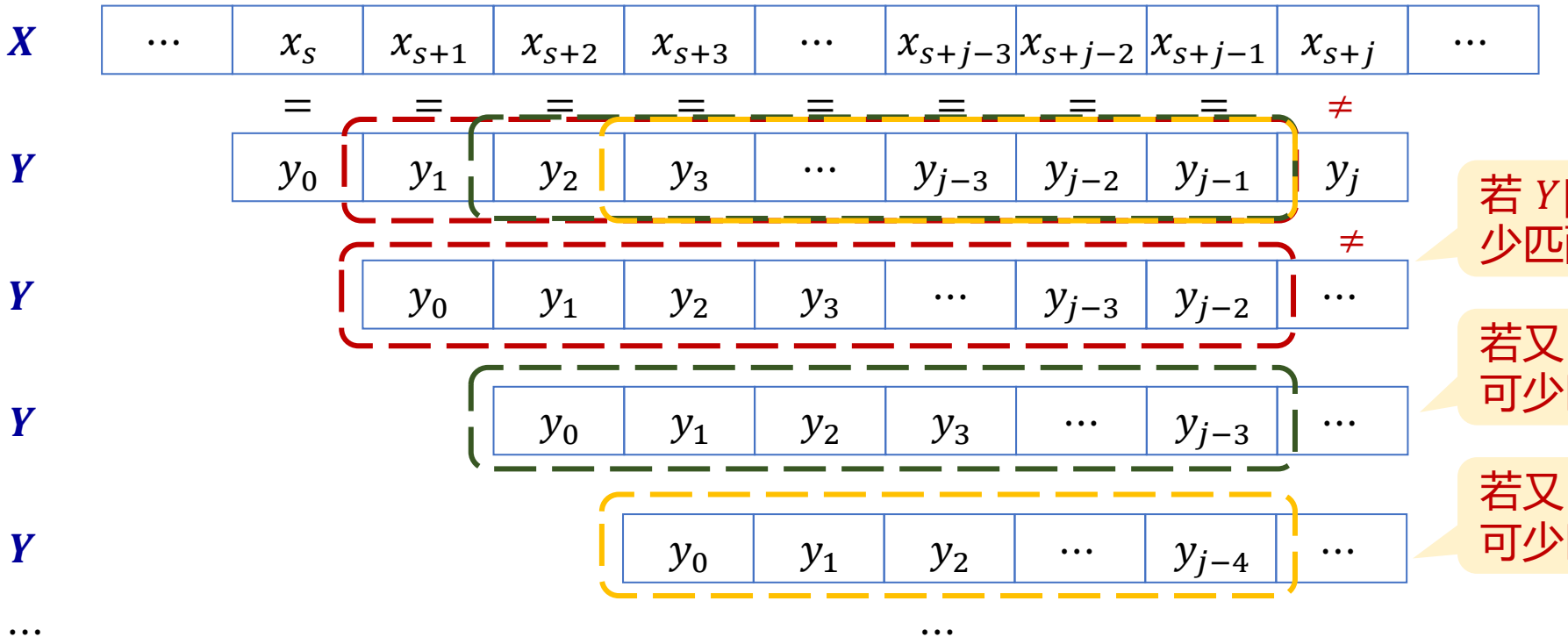
子串匹配问题

- 确定算法2——Knuth-Morris-Pratt（KMP）算法
  - 【例】模式串Y中重复出现$abcd$，但是e和x不匹配时，可直接向右滑动4个字符开始匹配，可少匹配3趟



子串匹配问题

- Knuth-Morris-Pratt（KMP）算法



设 $X[s, s+j-1] = Y[0, j-1]$，但 $X[s, s+j] \neq Y[0, j]$

若 $Y[0, j-2] \neq Y[1, j-1]$，可少匹配1趟

若又 $Y[0, j-3] \neq Y[2, j-1]$，可少匹配2趟

若又 $Y[0, j-4] \neq Y[3, j-1]$，可少匹配3趟

类推直到前缀$Y[0, k+1]$ ≠ 后缀$Y[j-k-2, j-1]$，
但是前缀$Y[0, k]$ = 后缀 $Y[j-k-1, j-1]$ 时，
可少匹配 $j-k-2$ 趟，相当于Y直接向右滑动$j-k-1$个字符

子串匹配问题

- Knuth-Morris-Pratt（KMP）算法
  - 对模式串Y进行预处理，计算可以滑过多少个字符

$$next(j)=\begin{cases}-1, & \text{当}j=0\\ k+1, & \text{当}0\le k<j-1\text{，且使}y_0y_1\cdots y_k=y_{j-k-1}y_{j-k}\cdots y_{j-1}\text{的最大数}\\ 0, & \text{其他情况}\end{cases}$$

$next(j)$直观含义：$[0,j-1]$中前缀和后缀相等的最大长度

$next(j)$直观作用：可滑过$j-next(j)$位不用匹配

下标 j	0	1	2	3	4	5	6	7
Y	a	b	c	d	a	b	c	e
$next(j)$	-1	0	0	0	0	1	2	3

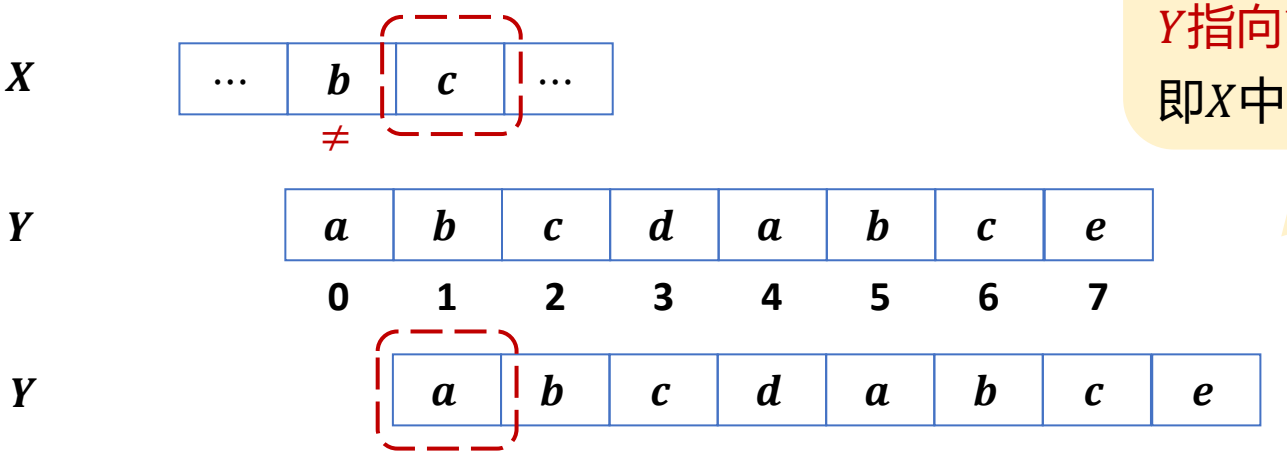
子串匹配问题

下标 j	0	1	2	3	4	5	6	7
Y	a	b	c	d	a	b	c	e
$next(j)$	-1	0	0	0	0	1	2	3

- Knuth-Morris-Pratt（KMP）算法
  - 匹配过程：遇到$Y[j] \neq X[k]$时，Y向右滑动$j-next(j)$位

$$next(j)=\begin{cases}-1, & \text{当} j=0\\ k+1, & \text{当} 0\le k<j-1\text{，且使} y_0y_1\cdots y_k=y_{j-k-1}y_{j-k}\cdots y_{j-1}\text{的最大数}\\ 0, & \text{其他情况}\end{cases}$$

$j=0$时匹配失败，查找模式串Y的$next(0)=-1$，
模式串Y向右滑动$j-next(j)=1$位，X指针加1，Y指向$Y[0]$，
即X中c与Y中$Y[0]=a$进行比较

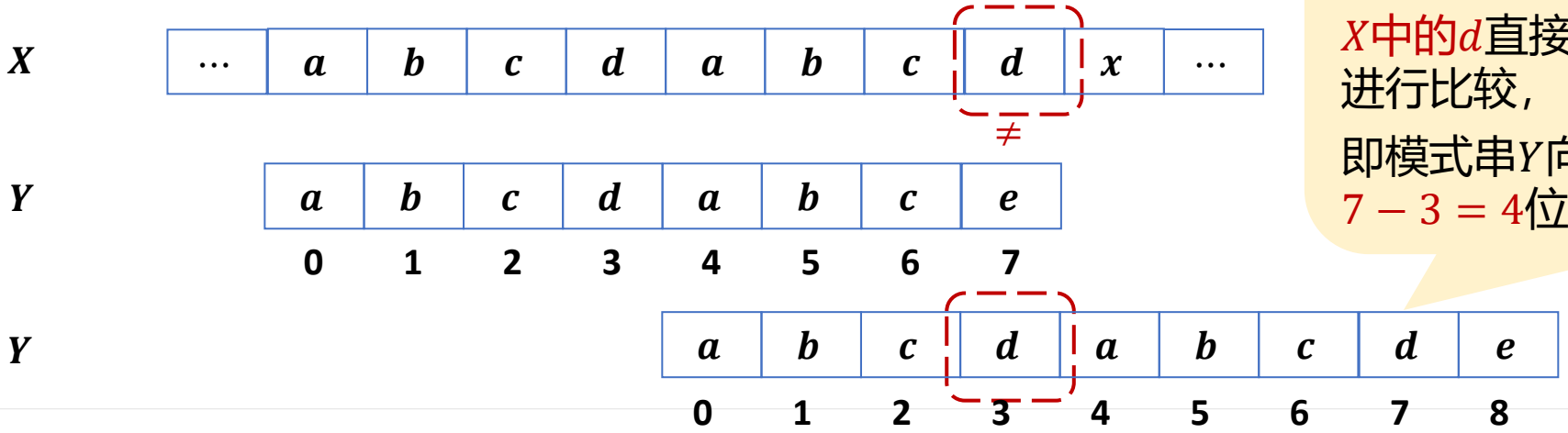


子串匹配问题

下标 j	0	1	2	3	4	5	6	7
Y	a	b	c	d	a	b	c	e
$next(j)$	-1	0	0	0	0	1	2	3

- Knuth-Morris-Pratt（KMP）算法
  - 匹配过程：遇到$Y[j] \neq X[k]$时，Y向右滑动$j-next(j)$位

$$next(j)=\begin{cases}-1, & \text{当} j=0\\ \boldsymbol{k+1}, & \boldsymbol{\text{当} 0 \le k < j-1,\ \text{且使} y_0 y_1 \cdots y_k = y_{j-k-1} y_{j-k} \cdots y_{j-1} \text{的最大数}}\\ 0, & \text{其他情况}\end{cases}$$



$j=7$时匹配失败，查找模式串Y的$next(7)=3$，

X中的d直接与$Y[next(7)]=Y[3]=d$进行比较，

即模式串Y向右滑动$j-next(7)=7-3=4$位

子串匹配问题

下标 j	0	1	2	3	4	5	6	7
Y	a	b	c	d	a	b	c	e
$next(j)$	-1	0	0	0	0	1	2	3

- Knuth-Morris-Pratt（KMP）算法
  - 匹配过程：遇到$Y[j] \neq X[k]$时，Y向右滑动$j-next(j)$位

$$next(j)=\begin{cases}-1, & \text{当} j=0\\ k+1, & \text{当} 0\le k<j-1\text{，且使} y_0y_1\cdots y_k=y_{j-k-1}y_{j-k}\cdots y_{j-1}\text{的最大数}\\ \mathbf{0}, & \textbf{其他情况}\end{cases}$$

$j=4$时匹配失败，查找模式串Y的$next(4)=0$，

X中的x直接与$Y[next(4)]=Y[0]=a$进行比较，

即模式串Y向右滑动$j-next(4)=4-0=4$位

X	…	a	b	c	d	a	b	c	d	x	…
										$\neq$	
Y		a	b	c	d	a	b	c	e		
		0	1	2	3	4	5	6	7		

Y	a	b	c	d	a	b	c	d	e
	0	1	2	3	4	5	6	7	8

子串匹配问题

- Knuth-Morris-Pratt（KMP）算法

```
KMP (X, Y, next)
j = 0;
for i = 0 to n − 1 do
    while j > 0 and Y[j] ≠ X[i] do
        j = next(j);      /*向右滑动j − next(j)位*/
    if Y[j] == X[i] then
        j = j + 1;
    if j == m then        /*找到一个匹配子串*/
        Print "Partten occurs with shift" i − m;
        j = next(j);      /*继续找下一个匹配字符*/
```

- 如果已经预处理得到所有$next(j)$

时间复杂度$\theta(n)$

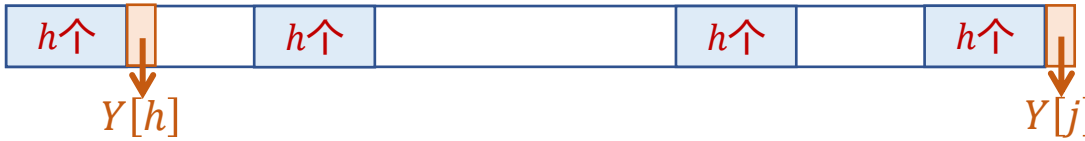
如何求所有的$next(j)$？

子串匹配问题

下标 j	0	1	2	3	4	5	6	7
Y	a	b	c	d	a	b	c	d
$next(j)$	-1	0	0	0	0	1	2	3

- **快速求$next(j)$的方法**
  - **假设已知$next(j)=k$，现在计算$next(j+1)$**
    - 若$Y[k]=Y[j]$，则$next(j+1)=k+1=next(j)+1$

    - 否则，设$next(k)=h$，此时有$Y[0,h-1]=Y[k-h,k-1]=Y[j-h,j-1]$



      - 若$Y[h]=Y[j]$，则$next(j+1)=h+1$
      - 否则，令$next(h)=t$，此时有$Y[0,t-1]=Y[h-t,h-1]=Y[j-t,j-1]$
        - 继续判断是否$Y[t]=Y[j]$，直到找到或者到$next(0)=-1$

子串匹配问题

- Knuth-Morris-Pratt（KMP）算法

Next-Function (Y)

$j = 0;\ k = -1;\ next(0) = -1;$

while $j < m$ ***do***

  if $k == -1$ or $Y[j] == Y[k]$ ***then***

    $j = j + 1;$

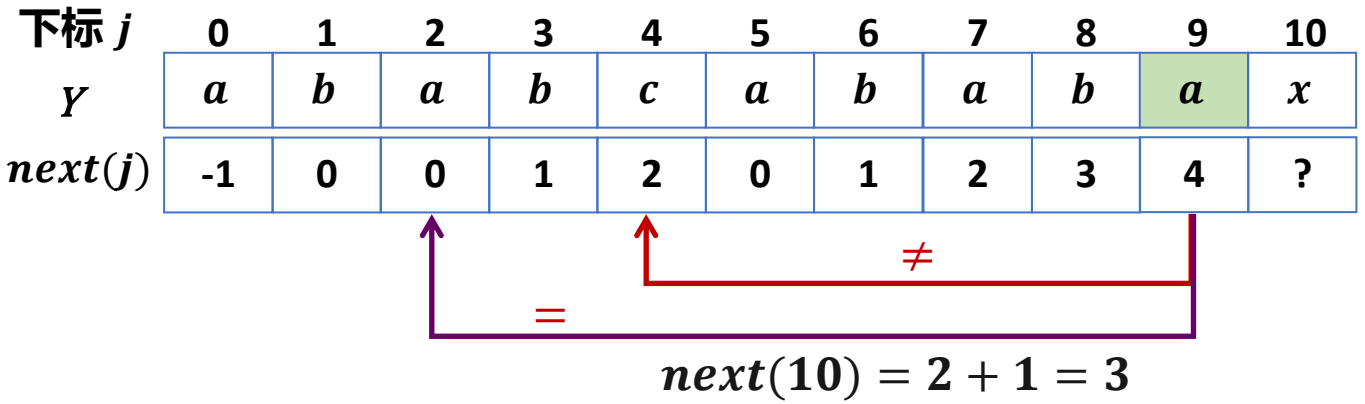
    $k = k + 1;$

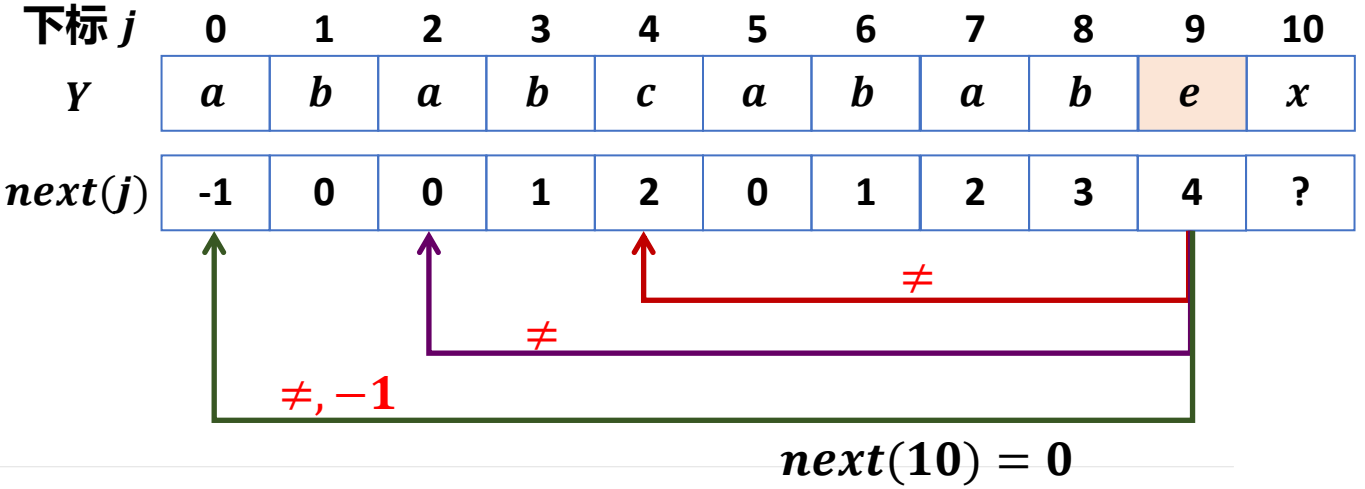
    $next(j) = k;$

  else

    $k = next(k);$

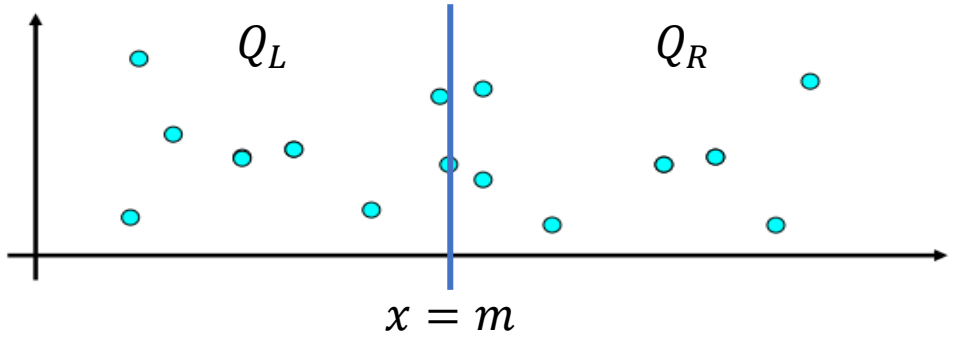
return $next;$





最近点对随机算法

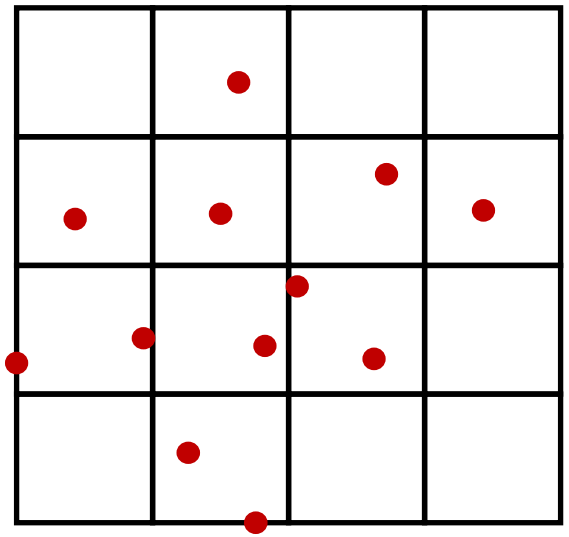
- 问题定义
  - 输入：欧几里得空间上n个点的集合Q
  - 输出：$A, B \in Q$，$dis(A,B) = \min\{dis(P_i, P_j) | P_i, P_j \in Q\}$
- 分治算法求解需$O(n \log n)$时间



- 用随机算法求解可达到$O(n)$期望时间

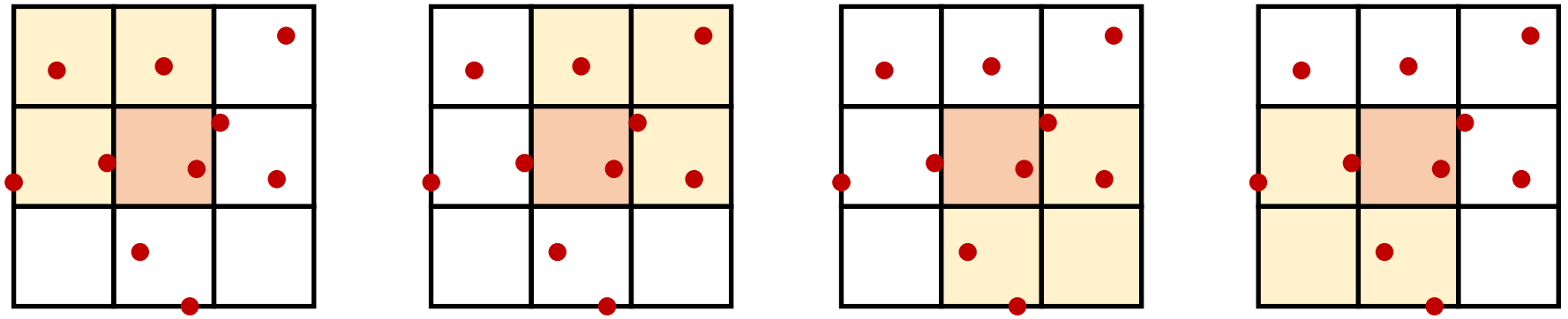
最近点对随机算法

- 随机算法
  - 平面上的n个点(x_1, x_2)，$(i = 1,2, \dots, n)$，以Δ为尺寸构造网格，Δ≥最近距离Δ^*
    - 网格以$(\min\{x_i\}, \min\{y_j\})$为最左下点
    - 以步长Δ不断向右向上伸展
    - $(\max\{x_i\}, \max\{y_j\})$含在某一格



最近点对随机算法

- 随机算法
  - 平面上的n个点(x_1, x_2)，$(i = 1,2, \dots, n)$，以Δ为尺寸构造网格，Δ≥最近距离Δ^*
    - 若最近点对中的一个点落在某个$\Delta \times \Delta$方格C_Δ中，则包含C_Δ的4个$2\Delta \times 2\Delta$方格C_Δ^i $(i = 1,2,3,4)$中，至少有一个同时含有最近点对的两个点。



尽量给定更小的Δ

最近点对随机算法

- 随机算法
  - 平面上的n个点(x_1, x_2)，$(i = 1,2, \dots, n)$，以Δ为尺寸构造网格，Δ≥最近距离Δ^*。**如何选择Δ？**

$O(n)$ 1. 点集$S(|S| = n)$中随机取子集T，使得$|T| = \lfloor \sqrt{n} \rfloor$

$O(n)$ 2. T中点两两计算距离，求出T中最近点对距离$\Delta(T)$

3. 以$\Delta(T)$为尺寸构造网格，设网格横向m个纵向r个

4. 分情况讨论：

$O(n)$ A. 如果$m \times r \leq cn$，其中c为2~10之间的常数：找到具有最多节点的方格，坐标记为(i, j)，

$O(n)$ a. 如果方格(i, j)包含的节点数≤ $\sqrt{n}$，逐一计算方格(i, j)内任意点对的距离，找到最小值记为Δ′

期望$O(n)$ b. 如果方格(i, j)包含的节点数> $\sqrt{n}$，将方格(i, j)一分为四，点数最多的子方格点数若还> $\sqrt{n}$，继续拆分直至点数≤ $\sqrt{n}$，计算节点数最多方格的最近点距Δ′

最近点对随机算法

- 随机算法
  - 平面上的n个点(x_1, x_2)，$(i = 1,2, \dots, n)$，以Δ为尺寸构造网格，Δ≥最近距离Δ^*。**如何选择Δ？**

4. 分情况讨论：

   A. 如果$m \times r \leq cn$，……

$O(n)$ B. 如果$m \times r > cn$，找一个略小于cn的素数p，用散列函数$H(i,j,p)$把S中的点存至长为p的散列表中，其对应一个方格(i,j)，散列表的每个槽只对应一个方格（若发生冲突，用冲突处理机制），在散列表中找到点数最多的方格(i,j)，然后按照和4.A中同样的方法找到新的Δ′

期望$O(n)$ 5. 以Δ′为新的尺寸构造新网格，最近点对必落在某个$2\Delta' \times 2\Delta'$的方格中，找到所有$2\Delta' \times 2\Delta'$的方格中包含节点数大于2的，并计算其中点对距离，找到最小值

素数测试

- 非对称密码RSA加密算法用到大素数

公钥: (n, e)	n是两个大素数p和q的乘积 (p和q必须保密), e与$(p-1)(q-1)$互质
私钥: (n, d)	$e \times d \equiv 1\ (mod\ (p-1)(q-1))$
加密	$C = M^e\ (mod\ n)$
解密	$M = C^d\ (mod\ n)$

【例】 令$p = 3$，$q = 11$，则$n = p \times q = 3 \times 11 = 33$；$(p-1)(q-1) = 2 \times 10 = 20$；

取$e = 3$，（$e = 3$与$(p-1)(q-1) = 20$互质）

$3 \times d \equiv 1\ mod\ 20$，可取$d = 7$

明文25，加密得密文$16 = 25^3\ mod\ 33$

密文16，解密得明文$25 = 16^7\ mod\ 33$

其中关键过程是得到两个大素数，但是判断一个非常大的数是否是素数并非简单的事

素数测试

- 大整数的素因子分解
  - 如$n = 10^{60}$（比已知最大素数小很多）
  - 若算法时间复杂度为$O(n)$
  - 设高速计算机，每秒1亿亿次(10^{16})基本运算
  - 则需10^{44}秒，大于10^{42}分钟，大于10^{40}小时，大于10^{38}天，大于10^{35}年!
  - $O(n)$时间的算法绝对不能接受!

素数测试

- 准备工作——求$a^m \ (mod\ n)$的算法（$m \le n$）
  - m的二进制表示为$b_k b_{k-1} \cdots b_1 b_0$，　$(b_k = 1, k \approx \log_2 m)$
  - $a^m = a^{b_k b_{k-1} \cdots b_1 b_0}$
  - 【例】$m = 41 = b_5 b_4 b_3 b_2 b_1 b_0 = 101001_{(2)}$，　$k = 5$
  - $a^{41} = a^{101001_{(2)}}$

素数测试

m的二进制表示为$b_k b_{k-1} \cdots b_1 b_0$，（$b_k = 1, k \approx \log_2 m$）

- 准备工作——求$a^m \ (mod\ n)$的算法（$m \le n$）
  - 计算a^m可以用下述方法
    - 从m的二进制的高位到低位，平方，遇1还要乘a
  - 【例】计算$a^{41}, 41 = b_5 b_4 b_3 b_2 b_1 b_0 = 101001_{(2)}$
    - 初始$C = 1$
    - $b_5 = 1$：$C = C^2 = 1$，因为$b_5 = 1$，$C = a * C = a$
    - $b_5 b_4 = 10$：$C = C^2 = a^2$
    - $b_5 b_4 b_3 = 101$：$C = C^2 = a^4$，因为$b_3 = 1$，$C = a * C = a^5$
    - $b_5 b_4 b_3\ b_2 = 1010$：$C = C^2 = a^{10}$
    - $b_5 b_4 b_3 b_2 b_1 = 10100$：$C = C^2 = a^{20}$
    - $b_5 b_4 b_3 b_2 b_1 b_0 = 101001$：$C = C^2 = a^{40}$，因为$b_0 = 1$，$C = a * C = a^{41}$

素数测试

m的二进制表示为$b_k b_{k-1} \cdots b_1 b_0$，$(b_k = 1, k \approx \log_2 m)$

- 准备工作——求$a^m \ (mod\ n)$的算法（$m \le n$）
  - 计算a^m可以用下述方法
    - 从m的二进制的高位到低位，平方，遇1还要乘a
  - 模运算有规则：$(x * y)\ \%\ n = \big((x\ \%\ n) * (y\ \%\ n)\big)\ \%\ n$
  - 求$a^m \ (mod\ n)$可在求a^m的每一步求模

好处：C的最大值不超过$n-1$，中间值不超过$\max\{(n-1)^2, a(n-1)\}$
求$a^m (mod\ n)$时不会占用很多空间

EXPMOD (a, m, n)

$C = 1;$

for $j = k$ ***to*** 0

  $C = C^2 \ (mod\ n);$

  if $b_j = 1$ ***then***

    $C = a * C \ (mod\ n);$

return $C;$

例如，计算$a^5 \ (mod\ n)$，$m = 5 = 101_{(2)}$

- 初始$C = 1$
- $b_2 = 1$：$C = C^2 \ (mod\ n)$，因为$b_2 = 1$，$C = a * C \ (mod\ n)$
- $b_2 b_1 = 10$：$C = C^2 \ (mod\ n)$
- $b_2 b_1 b_0 = 101$：$C = C^2 \ (mod\ n)$，因为$b_0 = 1$，$C = a * C \ (mod\ n)$

素数测试

- Fermat小定理
  - 若n为素数，$0 < a < n$，则有$a^{n-1} \equiv 1\ (mod\ n)$
    - 可以从$a = 2,3,\cdots,n-1$对$a^{n-1} \equiv 1\ (mod\ n)$进行测试，如果一旦出现不满足，则$n$不是素数
    - 但是，如果全部满足$a^{n-1} \equiv 1\ (mod\ n)$也不能说明$n$是素数
    - 某些n是合数，对任意a，也满足$a^{n-1} \equiv 1\ (mod\ n)$，称之为Carmichael数，例如前几个561，1105，1729，2465，…，小于1亿只有255个Carmichael数
    - Carmichael数虽然分布很稀，但仍有无穷多个

n为素数，$0 < a < n$ ⇏ **$a^{n-1} \equiv 1\ (mod\ n)$**

素数测试

- Miller-Rabin算法——剔除Carmichael数
  - $1 < a < n$，有$\boldsymbol{a^{n-1} \equiv 1\ (mod\ n)}$
    - 前提：n为奇数（如果为偶数就不用判断它是否为素数了）
    - 那么，$n-1$为偶数，即a^{n-1}可以转换为a^{2k}，即$\boldsymbol{a^{2k} \equiv 1\ (mod\ n)}$
    - 如果$\boldsymbol{a^{2k} \equiv 1\ (mod\ n)}$，那么，**$n$整除$a^{2k}-1$**
    - $\because a^{2k}-1=\left(a^k\right)^2-1=\left(a^k-1\right)\left(a^k+1\right)$
    - $\therefore$ 如果n为素数，那么n整除$\left(a^k-1\right)$或整除$\left(a^k+1\right)$
    - **那么，$a^k\ mod\ n=\begin{cases}1\\n-1\end{cases}$**
  - **二次探测：测试$a^{n-1} \equiv 1\ (mod\ n)$之后，继续测试$a^k\ mod\ n$，其中，$2k=n-1$**

素数测试

- Miller-Rabin算法
  - $1 < a < n$，有$\boldsymbol{a^{n-1} \equiv 1\ (mod\ n)}$
  - 二次探测：测试$a^{n-1} \equiv 1\ (mod\ n)$之后，继续测试$a^k\ mod\ n = 1$或$n-1$，其中，$2k = n-1$
  - **继续探测：**
    - **找到$n-1 = m \times 2^q$，其中，m为奇数。**
    - **用$a^m, a^{m\times 2}, a^{m\times 2^2}, a^{m\times 2^3}, \cdots, a^{m\times 2^q} = a^{n-1}$分别对$n$取模，一旦结果不为1或$n-1$，则判断$n$为合数，此时的$a$称为证据数（witness）**

在$1 \sim n$之间的证据数极多，但目前在理论上只能证明它们超过$(n-1)/2$，即至少有一半以上的证据数。于是答错的概率最多为$1/2$，（实际上低得多）。随机选k个证据数a，答错概率小于$1/2^k$

素数测试

- Miller-Rabin算法
  - 定理：设n为素数，且$0 < x < n$，则当$x^2 \equiv 1(mod\ n)$时，必有 $x = 1$ 或 $x = n - 1$
  - 推论：当$0 < x < n$ 且 $x^2 \equiv 1(mod\ n)$成立时，若$x \neq 1$且$x \neq n - 1$，则n是合数

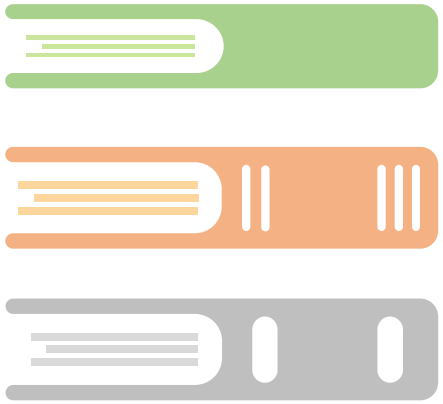
```
Miller-Rabin (m, n)
a = random(1, n − 1);
x = a^m (mod n);
for i = 1 to q do
    y = x^2 (mod n);
    if y = 1 and x ≠ 1 and x ≠ n − 1 then   /*此时满足推论*/
        return "n为合数";
    x = y;
if x ≠ 1 then return "n为合数";
else
    return "n为素数";   /*此时满足Fermat小定理*/
```

求$a^{n-1}\ (mod\ n)$ 相当于求$a^{m*2^q}(mod\ n)$

素数测试

- 【例】测试Carmichael数561是否为素数
  - $n-1=560=2^4*35$
  - 即有$q=4$，$m=35$
  - 假定选$a=7$，则$a^m=7^{35}\equiv 241(mod\ 561)$
  - 7^{35}，7^{70}，7^{140}，7^{280}，7^{560},$mod\ n$后分别为：241，298，166，67，1
  - $x=7^{280}\equiv 67(mod\ 561)$,
  - 而$x^2=7^{560}\equiv 1(mod\ 561)$，$x\neq 1$且$x\neq n-1$
  - 则561是合数

Miller-Rabin (m,n)
$a=random(1,n-1)$;
$x=a^m\ (mod\ n)$;
for $i=1$ ***to*** q ***do***
    $y=x^2\ (mod\ n)$;
    if $y=1$ and $x\neq 1$ and $x\neq n-1$ ***then***
        return "n为合数";
    $x=y$;
if $x\neq 1$ ***then*** ***return*** "n为合数";
else
    return "n为素数";



Q&A