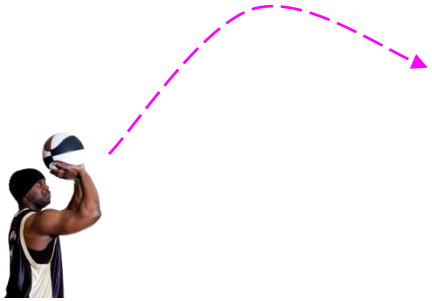
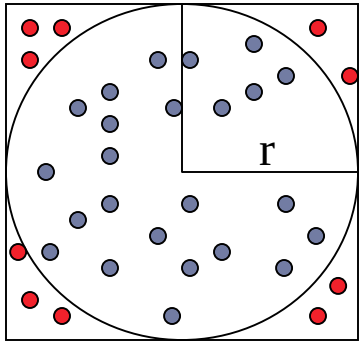
8 随机算法
Random Algorithm

给一枚硬币，试计算圆周率。

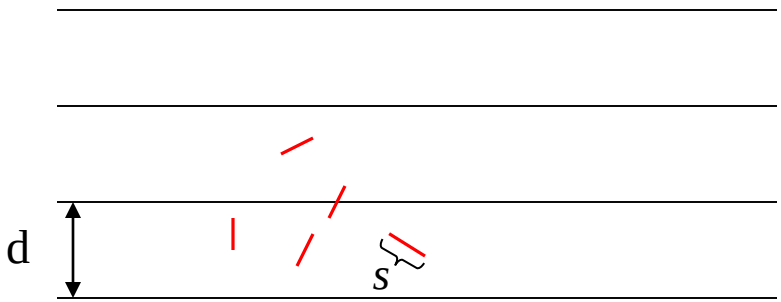
圆周率计算





$$\frac{S_{circle}}{S_{square}} = \frac{\pi \cdot r^2}{4 \cdot r^2} \approx \frac{n_{in\_circle}}{n_{in\_square}} \qquad \pi \approx 4 \cdot \frac{n_{in\_circle}}{n_{in\_square}}$$

```
public static double darts(int n)
{   // 用随机投点法计算pi值
    int k=0;
    for (int i=1;i <=n;i++) {
        double x=dart.fRandom();
        double y=dart.fRandom();
        if ((x*x+y*y)<=1) k++;
    }
    return 4*k/(double)n;
}
```



18世纪，法国数学家布丰和勒可莱尔提出的“投针问题”，记载于布丰1777年出版的著作中：“在平面上画有一组间距为d的平行线，将一根长度为s（s<d）的针任意掷在这个平面上，求此针与平行线中任一条相交的概率。”

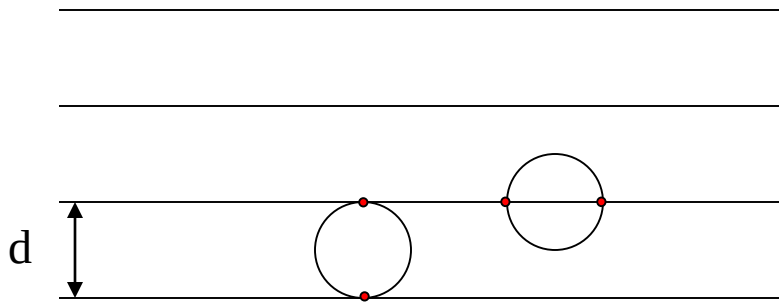
1) 取一张白纸，在上面画上许多条间距为d的平行线。

2) 取一根长度为s（s<d） 的针，随机地向画有平行直线的纸上掷n次，观察针与直线相交的次数，记为m

3) 计算针与直线相交的概率p

布丰本人证明了，这个概率是

$$p = \frac{2s}{\pi d} \approx \frac{m}{n}$$



- 一个直径为d的圆圈，不管如何投掷，必定是有两个交点。那么投掷n，共有2n个交点。
- 把圆圈拉直，变成一条长为πd的线段。显然，这样的线段扔下时与平行线相交的情形要比圆圈复杂些，可能有4个交点，3个交点，2个交点，1个交点，甚至于都不相交。由于圆圈和线段的长度同为πd，根据机会均等的原理，当它们均投掷n次(n是一较大的数)，两者与平行线组交点的总数期望值也是一样的，即均为2n。
- 长度为s的线段，投掷n次后，交点的个数为记为m，那么： $m/2n = s/\pi d$

$$\pi d \text{ ---------- } 2n$$
$$s \text{ ---------- } m$$

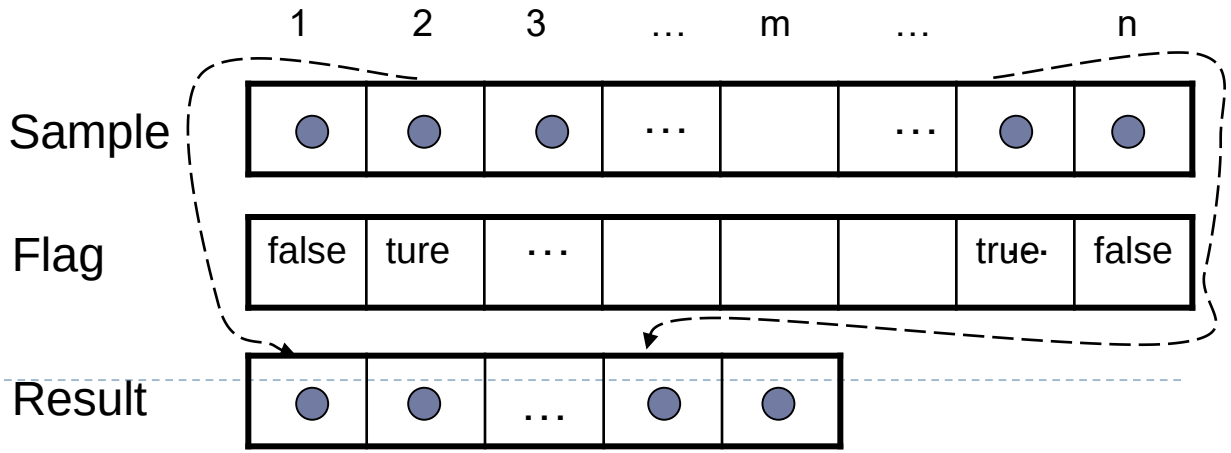
- 布丰投针实验是第一个用几何形式表达概率问题的例子，他首次使用随机实验处理确定性数学问题，为概率论的发展起到一定的推动作用。
- 像投针实验一样，用通过概率实验所求的概率来估计我们感兴趣的一个量，这样的方法称为蒙特卡罗方法（Monte Carlo method）。蒙特卡罗方法是在第二次世界大战期间随着计算机的诞生而兴起和发展起来的。这种方法在应用物理、原子能、固体物理、化学、生态学、社会学以及经济行为等领域中得到广泛利用。
- 蒙特·卡罗方法（Monte Carlo method），也称为统计模拟方法，是二十世纪四十年代中期由于科学技术的发展和电子计算机的发明，而被提出的一种以概率统计理论为指导的一类非常重要的数值计算方法。是指使用随机数（或更常见的伪随机数）来解决很多计算问题的方法。蒙特·卡罗方法的名字来源于摩纳哥的一个城市蒙地卡羅，该城市以赌博业闻名，而蒙特·卡罗方法正是以概率为基础的方法。

随机非重复采样问题

- 设有n 个样本，要求从中随机选出m 个样本(m<n/2)。请设计一个随机算法来完成该任务，并分析该算法的时间复杂度。
- 思路：
  - 将n个样本存放于数组Sample[1…n]中。数组Result[1…m]存放选中的m个样本。
  - 定义一个长度为n的标志数组Flag[1…n]。Flag[i]=true表示对应位置的样本已经被选中；否则为未选中。
  - 随机生成一个落在区间[1,n]的随机数r。若Flag[r]为false，Sample[r]追加到数组Result中,并将Flag[r]置为true；若Flag[r]为true,则重新生成随机数r，直到Flag[r]为false。重复此步骤，直到m个样本选择完成。

输入：样本数组Sample[1…n], 选择样本的个数m, 且m<n/2
输出：选中的样本Result[1..m]

```
1.for i←1 to n
2.  Flag[i] ←false //boolean数组，表示对应的样本是否已被选中
3. end for
4. k←0
5. while k<m
6.     r←random(1,n)
7.     if not Flag[r] then
8.         k←k+1
9.         Result[k] ← Sample[r]
10.        Flag[r] ← true
11.    end if
12.end while
13.return Result[1…m]
```



时间复杂度分析

- 很显然，这是一个典型的迭代算法，因而可以使用计算迭代次数的技术来分析。
- 观察：然而每次运行此算法，迭代次数都可能是不同的。

- 引理1：在某个空间中抛掷硬币，设正面朝上的概率是p，反面朝上的概率为q(q=1-p)。用随机变量X表示“出现正面朝上”需要连续抛掷的次数，则随机变量X的分布满足几何分布

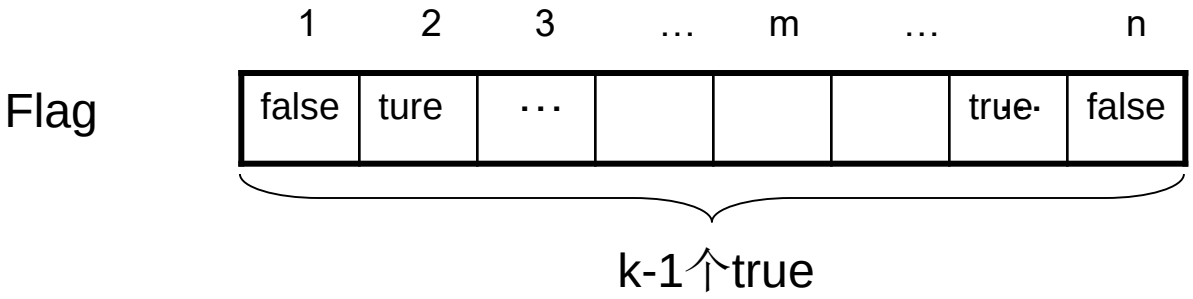
$$\mathbf{Pr}(X=k)=\begin{cases} pq^{k-1} & if\ k\geq 1 \\ 0 & if\ k<1 \end{cases}$$

X的数学期望是

$$\mathbf{E}(X)=\sum_{k=1}^{\infty}k\cdot\mathbf{Pr}(X=k)=\sum_{k=1}^{\infty}kpq^{k-1}=\frac{1}{p}$$

E(X)的含义是：平均连续抛掷1/p次，会出现一次正面。

- 分析：假设已经选中了k-1个样本，当前需要选择第k个样本。



“空位置个数”： n-(k-1)

“全部位置个数”： n

“随机数r落在空位置的概率”：n-(k-1) /n

用随机变量X_k 表示“落在空位置”需要连续生成随机数的次数

(为选中第k个样本算法需要的迭代次数)。

由引理可知：$E(X_k)=n/(n-k+1)$

- 用随机变量Y表示为了从n个样本中选出m个样本而生成的总的随机数个数(即算法总的迭代次数)，根据数学期望的线性性质，我们有

$$
\begin{aligned}
E(Y) &= E(X_1) + E(X_2) + \ldots + E(X_m) \\
&= \sum_{k=1}^{m} \mathbf{E}(X_k) \\
&= \sum_{k=1}^{m} \frac{n}{n-k+1} \\
&= n\sum_{k=1}^{\mathbf{n}} \frac{1}{n-k+1} - n\sum_{k=\mathbf{m+1}}^{\mathbf{n}} \frac{1}{n-k+1} \\
&= n\sum_{k=1}^{\mathbf{n}} \frac{1}{k} - n\sum_{k=1}^{\mathbf{n-m}} \frac{1}{k}
\end{aligned}
$$

由于：$\sum_{k=1}^{n}\frac{1}{k}\le \ln n+1,\quad \sum_{k=1}^{n-m}\frac{1}{k}\ge \ln(n-m+1)$

则有：$E(Y)=n\sum_{k=1}^{\mathbf{n}}\frac{1}{k}-n\sum_{k=1}^{\mathbf{n-m}}\frac{1}{k}$

$$\le n(\ln n+1)-n\ln(n-m+\mathbf{1})$$

$$\le n(\ln n+1)-n\ln(n-m)$$

$$\le n(\ln n+1)-n\ln(n-\mathbf{n/2}) \qquad (\because \mathbf{m\le n/2})$$

$$=n(\ln n+1-\ln(n/2))$$

$$=n(\ln 2+1)$$

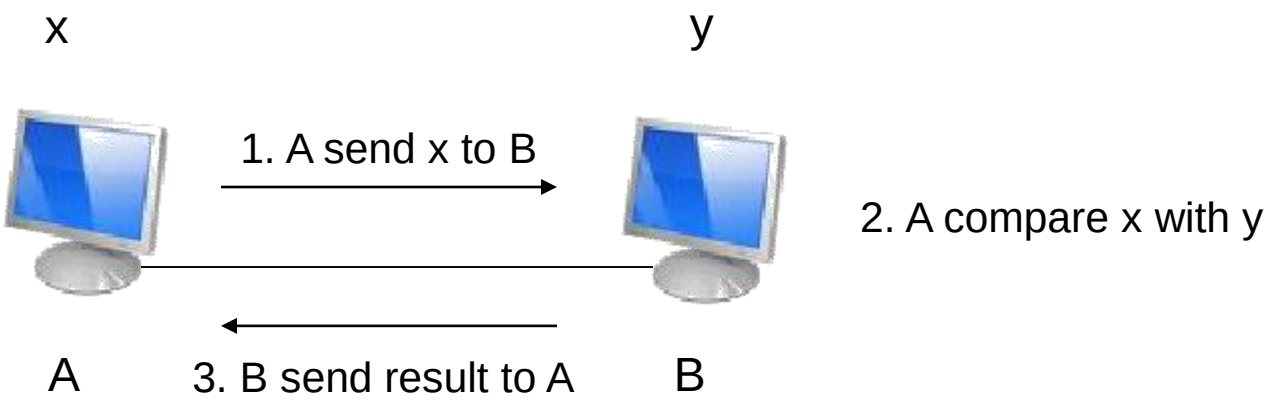
$$\approx 1.69n$$

算法的期望运行时间T(n)主要有两部分组成：1)将boolean数组S[1…n]置为false耗时Θ(n)，2)产生随机数r←random(1,n)的期望运行时间E(Y)≈1.69n；因此，期望运行时间

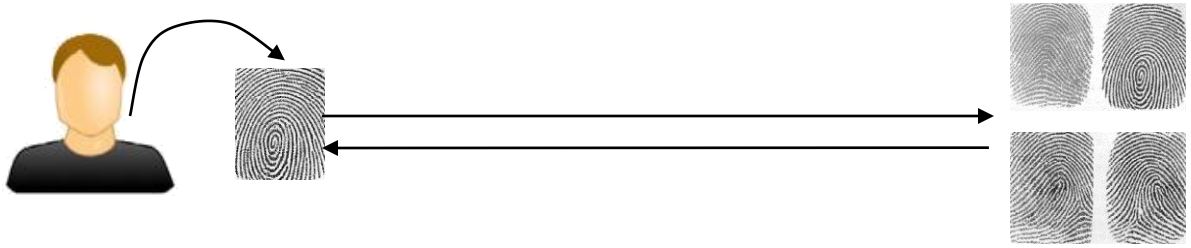
$$T(n)\approx \Theta(n)+1.69n=\Theta(n)$$

测试串的相等性

- 通信问题：发射端A发送信号x（x一般为很长的二进制串），接收端B收到信号y，要确定是否x = y。



目标：降低通信量，以减少对通信资源的占用。

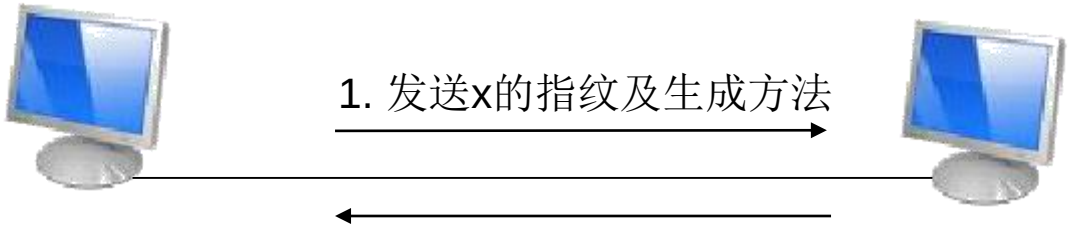


嫌疑人
(A地)

指纹库
(B地)

x ⟶ 生成x的指纹

y



1. 发送x的指纹及生成方法

3. 回传比对结果

A

B

2. 利用同样的方法生成y的指纹，并和x的指纹比对：**如果指纹不相同，**认为$x \neq y$；**否则，认为** $x = y$。

生成指纹

对于一个n位(bit)的二进制串x，I(x)为该二进制串所表示的一个整数。一种生成指纹的方法是：选择一个素数p，令

$$I_p(x) = I(x) \pmod p$$

$I_p(x)$即作为x的指纹。

因为$I_p(x)<p$，则$I_p(x)$可用一个不超过$\lfloor \log p \rfloor+1$位的二进制串来表示。因此，如果p 不是很大，那么，指纹$I_p(x)$可以作为一个较短的串来发送，串长度为O(log p)。

例如：$x=(101011)_2=(43)_{10}$, 取$p=(101)_2=(5)_{10}$,

则$I(x) \pmod p = 43 \pmod 5 = 3 = (11)_2$, 仅需2比特.

分析

- 上述算法如果给出$x \neq y$，肯定正确; 如果给出$x = y$,则可能出错。
- 出现错误匹配情形是：

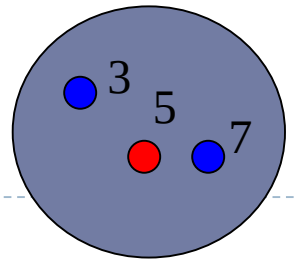
$$x \neq y，但\ I_p(x) = I_p(y)$$

$$\Updownarrow$$

$$x \neq y，\ p\ 整除 I(x)-I(y)$$

例：$x=(101011)_2=(43)_{10}$, $y=(110101)_2=(53)_{10}$。若取 $p=(101)_2=(5)_{10}$, 则会出现错误匹配，即：

$$x \neq y, 但\ I_p(x) = 3 = I_p(y).$$

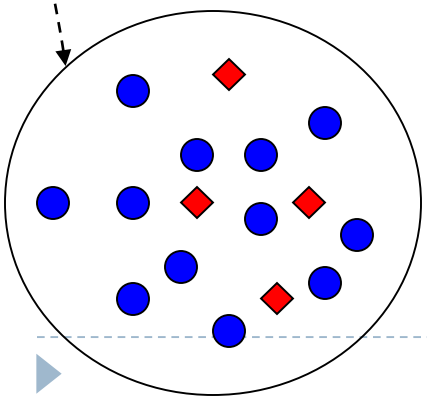


下面我们先给出算法的框架，再分析出错的概率。

算法：

1. 从**小于M 的素数集**中随机选择一个p
2. A 将p 和 $I_p(x)$发送给B
3. B 检查是否$I_p(x) = I_p(y)$，从而确定x 是否和y 相等。

当算法报告x=y时，可能会出错。是否会出错取决于所选择的素数p。因此，该算法的出错概率为：



$$P_{error} = \frac{\text{会导致出错的素数个数(红色)}}{\text{候选素数个数(红色+蓝色)}}$$

1. 用π(n)表示小于整数n 的不同素数的个数。那么π(n)渐趋近于n/ln(n)。
2. 如果整数$k < 2^n$，那么能够整除k 的素数的个数小于$\pi(n)$。

$$P_{error} = \frac{\text{会导致出错的素数个数}}{\text{候选素数个数}} = \frac{\left|\{p \mid x \neq y, p\text{整除}I(x) - I(y)\}\right|}{\pi(M)}$$

由于，x, y均为n位二进制串，所以：

$$0<I(x),I(y)<2^n，\ -2^n < \overbrace{I(x)-I(y)}^{k} < 2^n.$$

则由结论2可知，整除I(x)-I(y)的素数个数小于$\pi(n)$ 。

$$P_{error} \leq \frac{\pi(n)}{\pi(M)}$$

若取$M=2n^2$，则：

$$P_{error} \leq \frac{\pi(n)}{\pi(M)} = (n\ln n)/(2n^2/\ln 2n^2) \approx \frac{1}{n}$$

假如算法连续执行k次的运行结果为“相等”，则出错概率可以降低为：

$$P_{error} \le \left(\frac{1}{n}\right)^k$$

例子：传输一个百万位的串即n=1,000,000，取$M=2n^2=2\times10^{12}$，传输素数p至多需$\lfloor\log(M)\rfloor+1=41$位，传输指纹也至多41位，共82位。验证1次发生假匹配的概率为1/1,000,000，重复验证5次，假匹配概率可减小到$(10^{-6})^5=10^{-30}$.

例：$x=(101011)_2=(43)_{10}$, $y=(110101)_2=(53)_{10}$,

取$p=(101)_2=(5)_{10}$, $I_p(x)=3=I_p(y)$，出现假匹配；

再取p=3，$I_p(x)=1\neq I_p(y)=2$，验证不通过；

模式匹配问题

- 问题描述：给一个字模式串$X=x_1x_2\ldots x_n$，模式串$Y=y_1y_2\ldots y_m$，其中$m<n$，问：模式串Y是否在模式串X中出现？不失一般性，假设模式串定义在字符集{0,1}上。
- 最直观的方法：沿模式串X滑动Y，逐个比较子模式串$X(j)=x_j\ldots x_{j+m-1}$和Y。时间复杂度：最坏情况下为O(mn)，例如：

  X="00000000000000000000000000000000000001"

  Y="000001"

随机算法

- 思想：同样沿着模式串X滑动Y，但不是直接将模式串Y与每个子模式串$X(j)=x_j \ldots x_{j+m-1}$进行比较；而是借鉴指纹匹配的思想，将Y的指纹与子模式串X(j)的指纹比较，从而判断Y是否与X(j)相同。
- 直接使用上述方法时间复杂性不能有效降低，因为指纹计算同样要耗费时间。
- 然而，分析可以发现：新串X(j+1)的指纹可以很方便地从X(j)的指纹计算出来，即：

$$I_p(X(j+1))=(2I_p(X(j)) - 2^m x_j + x_{j+m}) \pmod p$$

- 为何有上述结论？

$$X(j) = x_j x_{j+1} \cdots x_{j+m-1}$$

$$I(X(j)) = x_j 2^{m-1} + x_{j+1} 2^{m-2} + \cdots + x_{j+m-1} 2^0 \quad (1)$$

$$I(X(j+1)) = x_{j+1} 2^{m-1} + x_{j+2} 2^{m-2} + \cdots + x_{j+m-1} 2^1 + x_{j+m} 2^0 \quad (2)$$

2×(1): $$2I(X(j)) = x_j 2^m + x_{j+1} 2^{m-1} + \cdots + x_{j+m-1} 2^1 \quad (3)$$

(3) - (2): $$2I(X(j)) - I(X(j+1)) = x_j 2^m - x_{j+m} 2^0$$

$$\Downarrow$$

$$I(X(j+1)) = I(X(j)) - x_j 2^m + x_{j+m} 2^0$$

$$\Downarrow$$

$$I_p(X(j+1)) = (2I_p(X(j)) - 2^m x_j + x_{j+m}) (\mathrm{mod}\ p)$$

输入：模式串X，长度为n，模式串Y，长度为m
输出：若Y在X中出现，则返回Y在X中的第1个位置，否则返回-1
1. 从小于M的素数集中随机选择一个素数p
2. j←1
3. 计算$W_p=2^m(\mathrm{mod}\ p)$, $I_p(Y)$ 和 $I_p(X_1)$ //O(m)+O(m)+O(m)
4. while $j\le n-m+1$ //$X_j = x_j\ldots x_{j+m-1}$, 须有$j+m-1\le n$，即$j\le n-m+1$
5. if $I_p(X_j) = I_p(Y)$ then return j //(可能)找到了匹配子串, O(log*p*)
6. $I_p(X_{j+1}) \leftarrow (2I_p(X_j) - x_jW_p + x_{j+m})$ //每步O(1)时间，共O(n)
7. j←j+1
8. end while
9. return -1 //Y肯定不在X中(若在，已由第5步返回j值)

时间复杂度分析：O(m) + O(n) + O(log*p*) = O(m+n)

遗传算法Genetic Algorithm (GAs)

基于进化论的随机优化算法

遗传算法(Genetic Algorithm, GA)是近年来发展起来的一种崭新的全局优化算法，它借用了生物遗传学的观点，通过自然选择、遗传、变异等作用机制，实现各个个体的适应性的提高。

遗传算法是由美国的J. Holland教授于1975年在他的专著《自然界和人工系统的适应性》中首先提出。

基 本概 念

1. 适应度与适应度函数

适应度(fitness)就是借鉴生物个体对环境的适应程度，而对所求解问题中的对象设计的一种表征优劣的测度。适应度函数(fitness function)就是问题中的全体对象与其适应度之间的一个对应关系，即对象集合到适应度集合的一个映射。 它一般是定义在论域空间上的一个实数值函数。

2. 染色体及其编码

遗传算法以生物细胞中的染色体(chromosome)代表问题中的个体对象。而一个染色体可以看作是由若干基因组成的位串,所以需要将问题中的个体对象编码为某种位串的形式。这样,原个体对象也就相当于生命科学中所称的生物体的表现型(phenotype),而其编码即“染色体”也就相当于生物体的基因型(genotype)。遗传算法中染色体一般用字符串表示,而基因也就是字符串中的一个个字符。例如,假设数字9是某问题中的个体对象,则我们就可以用它的二进制数串1001作为它的染色体编码。

3. 种群

种群(population)就是模拟生物种群而由若干个染色体组成的群体, 它一般是整个论域空间的一个很小的子集。 遗传算法就是通过在种群上实施所称的遗传操作,使其不断更新换代而实现对整个论域空间的搜索。

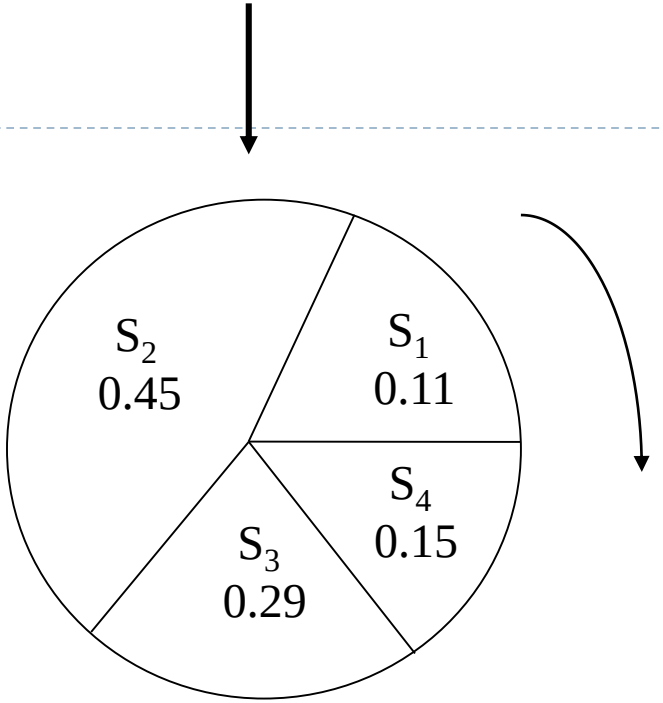
4. 遗传操作

遗传算法中有三种关于染色体的运算: 选择-复制、交叉和变异,这三种运算被称为遗传操作或遗传算子(genetic operator)。

选择-复制　选择-复制(selection　reproduction)操作是模拟生物界优胜劣汰的自然选择法则的一种染色体运算，就是从种群中选择适应度较高的染色体进行复制,以生成下一代种群。选择-复制的通常做法是，对于一个规模为N的种群S,按每个染色体$x_i \in S$的选择概率$P(x_i)$所决定的选中机会，分N次从S中随机选定N个染色体，并进行复制。 这里的选择概率$P(x_i)$的计算公式为

$$P(x_i)=\frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)}$$

其中，f为适应度函数，$f(x_i)$为x_i的适应度。可以看出，染色体x_i被选中的概率就是其适应度$f(x_i)$所占种群中全体染色体适应度之和的比例。 显然，按照这种选择概率定义，适应度越高的染色体被随机选定的概率就越大，被选中的次数也就越多，从而被复制的次数也就越多。相反,适应度越低的染色体被选中的次数也就越少,从而被复制的次数也就越少。如果把复制看做染色体的一次换代的话,则这就意味着适应度越高的染色体其后代也就越多,适应度越低的染色体其后代也就越少，甚至被淘汰。 这正吻合了优胜劣汰的自然选择法则。



赌轮选择示例

上述按概率选择的方法可用一种称为赌轮的原理来实现。即做一个单位圆，然后按各个染色体的选择概率将圆面划分为相应的扇形区域(如图所示)。这样，每次选择时先转动轮盘，当轮盘静止时,上方的指针所正对着的扇区即为选中的扇区,从而相应的染色体即为所选定的染色体。 例如, 假设种群S中有4个染色体：s_1,s_2，s_3，s_4,其选择概率依次为：0.11，0.45，0.29，0.15，则它们在轮盘上所占的份额如图中的各扇形区域所示。

在算法中赌轮选择法可用下面的子过程在计算机上进行模拟:

① 在［0, 1］区间内产生一个均匀分布的伪随机数r。

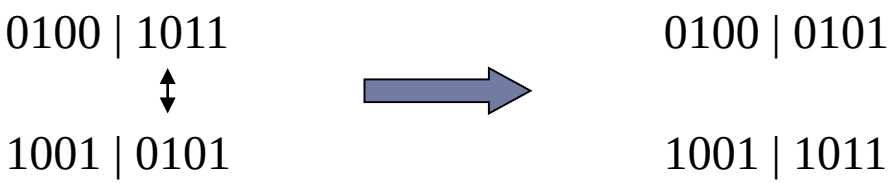
② 若$r \le q_1$,则染色体x_1被选中。

③ 若$q_{k-1} < r \le q_k (2 \le k \le N)$, 则染色体$x_k$被选中。

其中的q_i称为染色体$x_i (i=1, 2, \ldots, n)$的积累概率, 其计算公式为

$$q_i = \sum_{j=1}^{i} P(x_j)$$

交叉（crossover）亦称交换、交配或杂交，就是互换两个染色体某些位上的基因。例如，设染色体s_1=01001011，s_2=10010101，交换其后4位基因，即



则得新串s_1'=01000101，s_2'=10011011。s_1'和s_2'可以看做是原染色体s_1和s_2的子代染色体。

变异（mutation）亦称突变，就是改变染色体某个(些)位上的基因。例如，把染色体s=11001101的第三位上的0变为1，则得到新染色体s'=11101101。

简单遗传算法(Simple Genetic Algorithm)

简单来讲，遗传算法就是对种群中的染色体反复做三种遗传操作，使其朝着适应度增高的方向不断更新换代，直至出现了适应度满足目标条件的染色体为止。

在算法的具体步骤中，还需给出若干控制参数，如种群规模、最大换代数、交叉率和变异率等等。

——种群规模就是种群的大小，用染色体的个数表示。

——最大换代数就是算法中种群更新换代的上限，它也是算法终止的一个条件。

——交叉率（crossover rate)就是参加交叉运算的染色体个数占全体染色体总数的比例，记为P_c,取值范围一般为0.4～0.99。由于生物繁殖时染色体的交叉是按一定的概率发生的,因此参加交叉操作的染色体也有一定的比例，而交叉率也就是交叉概率。

——变异率(mutation rate)是指发生变异的基因位数所占全体染色体的基因总位数的比例，记为P_m，取值范围一般为0.0001～0.1。由于在生物的繁衍进化过程中，变异也是按一定的概率发生的，而且发生概率一般很小，因此变异率也就是变异概率。

基本遗传算法：

步1　在论域空间U上定义一个适应度函数$f(x)$，给定种群规模N，交叉率P_c和变异率P_m，代数T；

步2　随机产生U中的N个染色体$s_1, s_2, \ldots, s_N$，组成初始种群$S=\{s_1, s_2, \ldots, s_N\}$，置代数计数器$t=1$；

步3　计算S中每个染色体的适应度f()；

步4　若终止条件满足，则取S中适应度最大的染色体作为所求结果，算法结束。

步5　按选择概率$P(x_i)$所决定的选中机会，每次从S中随机选定1个染色体并将其复制，共做N次，然后将复制所得的N个染色体组成群体S_1；

步6 按交叉率P_c所决定的参加交叉的染色体数c，从S_1中随机确定c个染色体，配对进行交叉操作，并用产生的新染色体代替原染色体，得群体S_2；

步7 按变异率P_m所决定的变异次数m，从S_2中随机确定m个染色体，分别进行变异操作，并用产生的新染色体代替原染色体，得群体S_3；

步8 将群体S_3作为新一代种群，即用S_3代替S，t=t+1，转步3；

简单遗传算法是D. J. Goldberg总结出的一种统一的最基本的遗传算法。在简单遗传算法的基础上，现在已派生出遗传算法的许多变形，可以说已形成了一个遗传算法家族。

在应用遗传算法解决实际问题时，还需给出结构模式的表示方案、适应度的计算方法、终止条件等。表示方案通常是把问题的搜索空间的每一个可能的点,编码为一个看做染色体的字符串,字符通常采用二进制数0、1。适应度的计算方法一般根据实际问题而定。

简单遗传算法应用举例

例: 利用遗传算法求解区间［0,31］上的二次函数$y=x^2$的最大值。(尽量简化，纯粹用作示例)

分析 可以看出,只要能在区间［0,31］中找到函数值最大的点a,则函数$y=x^2$的最大值也就可以求得。于是, 原问题转化为在区间［0, 31］中寻找能使y取最大值的点a的问题。显然, 对于这个问题, 任一点$x\in$［0, 31］都是可能解, 而函数值$f(x)=x^2$也就是衡量x能否为最佳解的一种测度。那么,用遗传算法的眼光来看, 区间［0, 31］就是一个(解)空间,x就是其中的个体对象, 函数值$f(x)$恰好就可以作为x的适应度。这样, 只要能给出个体x的适当染色体编码, 该问题就可以用遗传算法来解决。

解

(1) 定义适应度函数,编码染色体。由上面的分析,函数$f(x)=x^2$就可作为空间U上的适应度函数。显然$y=x^2$是一个单调增函数,其取最大值的点x=31是个整数。另一方面, 5位二进制数也刚好能表示区间［0, 31］中的全部整数。所以, 我们就仅取［0,31］中的整数来作为参加进化的个体,并且用5位二进制数作为个体x的基因型编码, 即染色体。

(2) 设定种群规模, 产生初始种群。我们将种群规模设定为4, 取染色体s_1=01101(13), s_2=11000(24), s_3=01000(8), s_4=10011(19)组成初始种群S_1。

(3) 计算各代种群中的各染色体的适应度，并进行遗传操作，直到适应度最高的染色体(该问题中显然为“11111”=31)出现为止。

计算S_1中各染色体的适应度、选择概率、积累概率等并列于表1中。

表1. 第一代种群S_1中各染色体的情况

染色体	适应度	选择概率	积累概率	估计被选中次数
s_1=01101	169	0.14	0.14	1
s_2=11000	576	0.49	0.63	2
s_3=01000	64	0.06	0.69	0
s_4=10011	361	0.31	1.00	1

选择-复制　设从区间［0, 1］中产生4个随机数如下：

r_1=0.450126, r_2=0.110347, r_3=0.572496, r_4=0.98503

按赌轮选择法，染色体s_1, s_2, s_3, s_4的被选中次数依次为：1, 2, 0, 1。于是，经复制得群体：

s_1’=11000（24）, s_2’=01101（13）, s_3’=11000（24）, s_4’=10011（19）

可以看出，在第一轮选择中适应度最高的染色体s_2被选中两次，因而被复制两次；而适应度最低的染色体s_3一次也没有选中而遭淘汰。

交叉 设交叉率p_c=100%，即S_1中的全体染色体都参加交叉运算。设s_1'与s_2'配对，s_2'与s_4'配对。分别交换后两位基因，得新染色体：

s_1''=11001（25），s_2''=01100（12），

s_3''=11011（27），s_4''=10000（16）

变异 设变异率p_m=0.001。这样，群体S_1中共有5*4*0.001=0.02位基因可以变异。0.02位显然不足1位，所以本轮遗传操作不做变异。

现在，我们得到了第二代种群S_2：

s_1=11001（25），s_2=01100（12），

s_3=11011（27），s_4=10000（16）

表2 第二代种群S_2中各染色体的情况

染色体	适应度	选择概率	积累概率	估计被选中次数
s_1=11001	625	0.36	0.36	1
s_2=01100	144	0.08	0.44	0
s_3=11011	729	0.41	0.85	2
s_4=10000	256	0.15	1.00	1

假设这一轮选择-复制操作中，种群S_2中的4个染色体都被选中（因为选择概率毕竟只是一种几率，所以4个染色体恰好都被选中的情况是存在的），我们得到群体：

s_1'=11001（25），s_2'=01100（12），s_3'=11011（27），s_4'=10000（16）

然后，做交叉运算，让s_1'与s_2'，s_3'与s_4'分别交换后三位基因，得

s_1''=11100（28），s_2''=01001（9），s_3''=11000（24），s_4''=10011（19）

这一轮仍然不会发生变异。于是，得第三代种群S_3：

s_1=11100（28），s_2=01001（9），s_3=11000（24），s_4=10011（19）

表3 第三代种群S_4中各染色体的情况

染色体	适应度	选择概率	积累概率	估计被选中次数
s_1=11100	784	0.44	0.44	2
s_2=01001	81	0.04	0.48	0
s_3=11000	576	0.32	0.80	1
s_4=10011	361	0.20	1.00	1

设这一轮的选择-复制结果为：

s_1'=11100（28）, s_2'=11100（28）, s_3'=11000（24）, s_4'=10011（19）

然后，做交叉运算，让s_1'与s_4'，s_2'与s_3'分别交换后两位基因，得

s_1''=11111（31）, s_2''=11100（28）, s_3''=11000（24）, s_4''=10000（16）

这一轮仍然不会发生变异。于是，得第四代种群S_4：

s_1=11111（31）, s_2=11100（28）, s_3=11000（24）, s_4=10000（16）

显然，在这一代种群中已经出现了适应度最高的染色体s_1=11111。于是，遗传操作终止，将染色体“11111”作为最终结果输出。

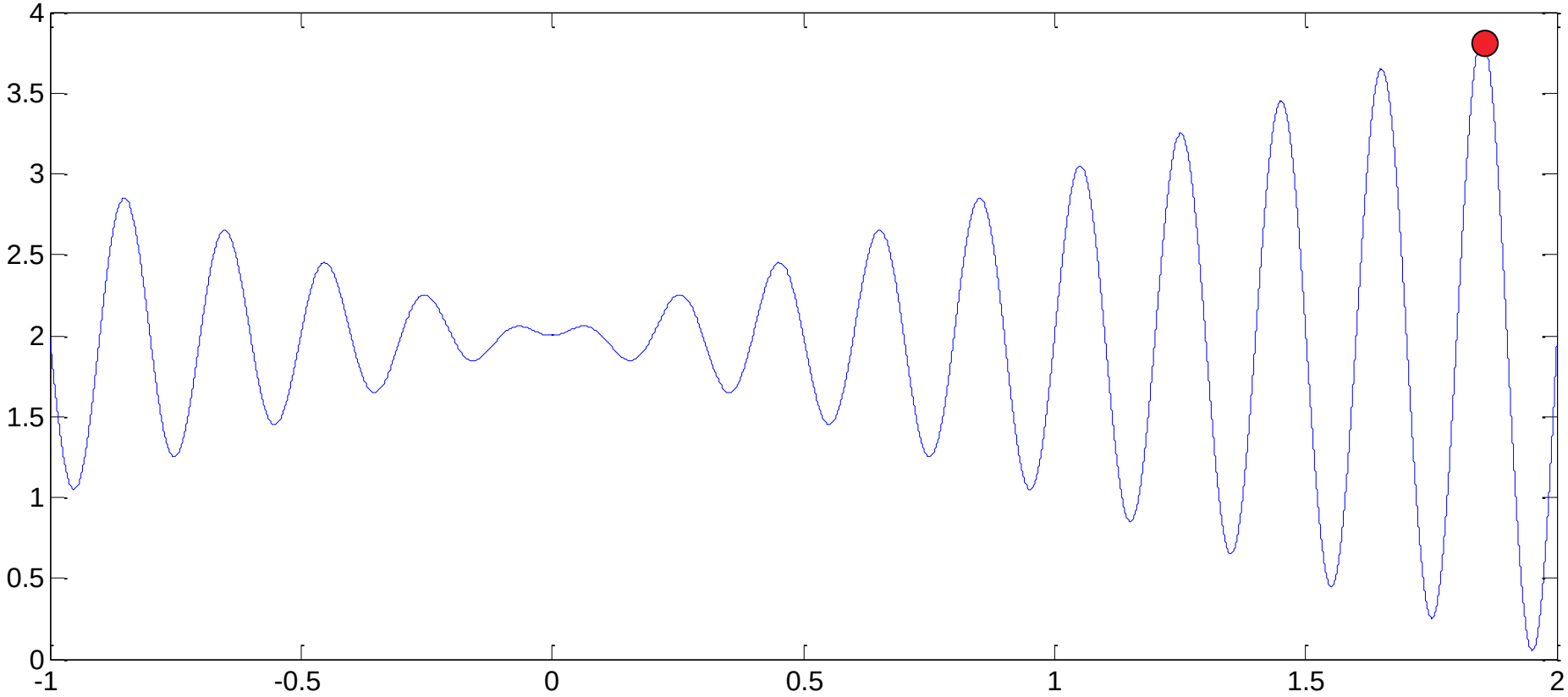
然后，将染色体“11111”解码为表现型，即得所求的最优解：31。将31代入函数$y=x^2$中，即得原问题的解，即函数$y=x^2$的最大值为961。

使用简单遗传算法(SGA)解决下面问题(编程实现)

求下列一元函数的最大值：

$$f(x) = x \cdot \sin(10\pi \cdot x) + 2.0$$

其中, x∈[-1,2]，求解结果精确到6位小数。

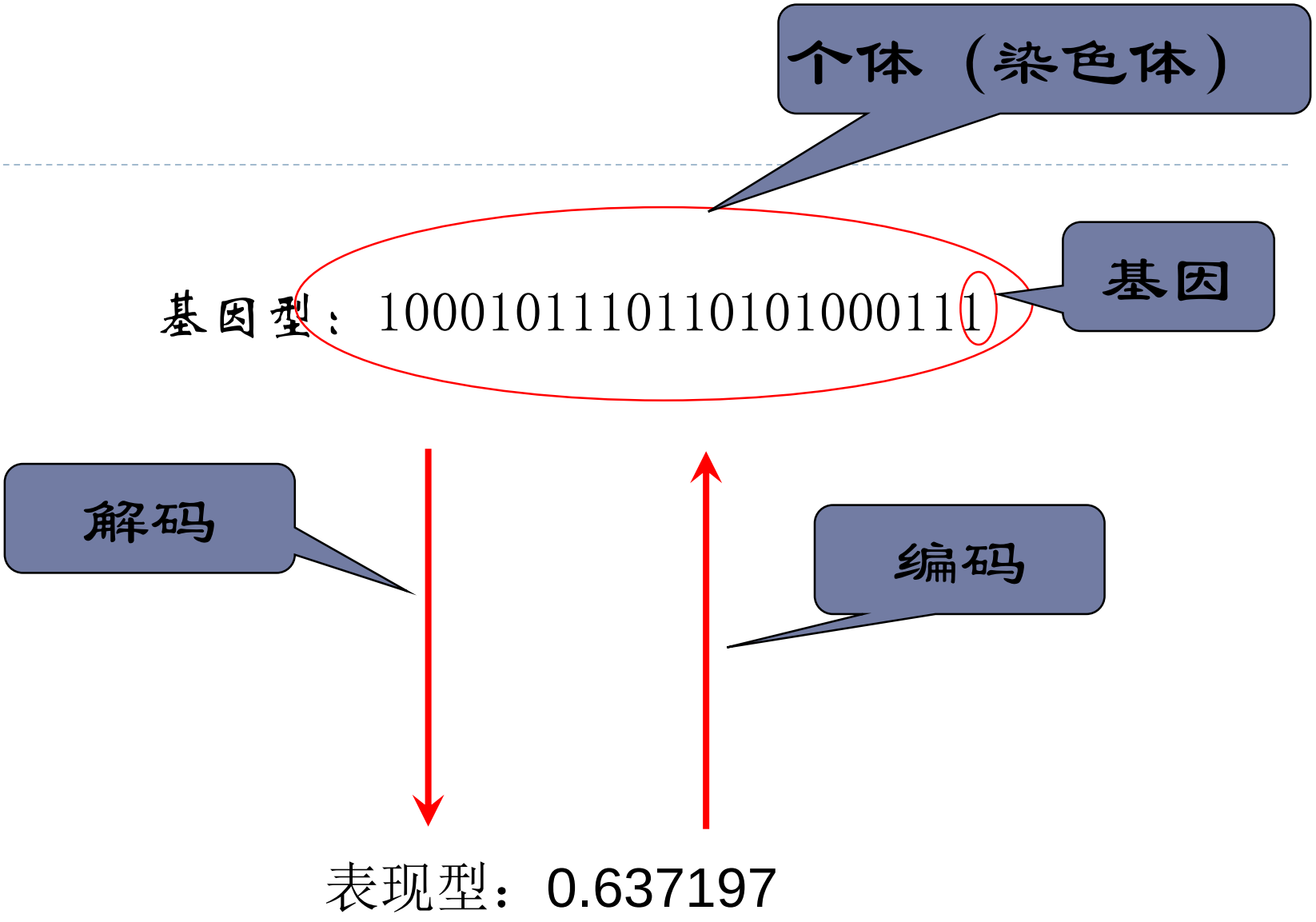

4
3.5
3
2.5
2
1.5
1
0.5
0
-1
-0.5
0
0.5
1
1.5
2

编码：

由于区间长度为3，求解结果精确到6位小数，因此可将自变量定义区间划分为3×10^6等份。又因为$2^{21}<3\times10^6<2^{22}$，所以本例的二进制编码长度至少需要22位，本例的编码过程实质上是将区间[-1，2]内对应的实数值转化为一个二进制串（$b_{21}b_{20}\cdots b_0$）。

0000000000000000000000 ⟶ -1

0000000000000000000001 ⟶ -0.999999


个体（染色体）
基因型：10001011101101010001111
基因
解码
编码
表现型：0.637197

初始种群

采用随机方法生成若干个个体的集合，该集合称为初始种群。初始种群中个体的数量称为种群规模。

适应度函数

遗传算法对一个个体（解）的好坏用适应度函数值来评价，适应度函数值越大，解的质量越好。适应度函数是遗传算法进化过程的驱动力，也是进行自然选择的唯一标准，它的设计应结合求解问题本身的要求而定(问题依赖的)。

选择算子

遗传算法使用选择运算来实现对群体中的个体进行优胜劣汰操作：适应度高的个体被遗传到下一代群体中的概率大；适应度低的个体，被遗传到下一代群体中的概率小。选择操作的任务就是按某种方法从父代群体中选取一些个体，遗传到下一代群体。遗传算法中选择算子采用轮盘赌选择方法。

轮盘赌选择方法

轮盘赌选择又称比例选择算子，它的基本思想是：各个个体被选中的概率与其适应度函数值大小成正比。设群体大小为n，个体i 的适应度为 F_i，则个体i 被选中遗传到下一代群体的概率为：

$$P_i = F_i / \sum_{i=1}^{n} F_i$$

轮盘赌选择方法的实现步骤

（1） 计算群体中所有个体的适应度函数值（需要解码）；

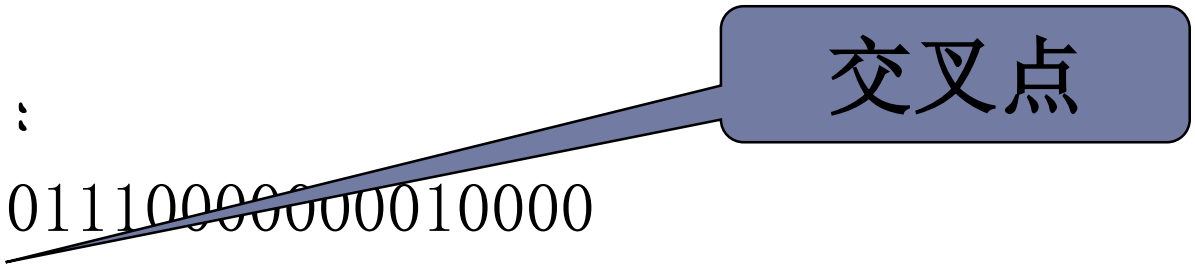
（2） 利用比例选择算子的公式，计算每个个体被选中遗传到下一代群体的概率；

（3） 采用模拟赌盘操作（即生成0到1之间的随机数与每个个体遗传到下一代群体的概率进行匹配）来确定各个个体是否遗传到下一代群体中。

交叉算子

所谓交叉运算，是指对两个相互配对的染色体依据交叉概率 P_c 按某种方式相互交换其部分基因，从而形成两个新的个体。交叉运算是遗传算法区别于其他进化算法的重要特征，它在遗传算法中起关键作用，是产生新个体的主要方法。SGA中交叉算子采用单点交叉算子。

单点交叉运算



- 交叉前：
- 00000|0111000000010000
- 11100|0000011111000101
- 交叉后：
- 00000|0000011111000101
- 11100|0111000000010000

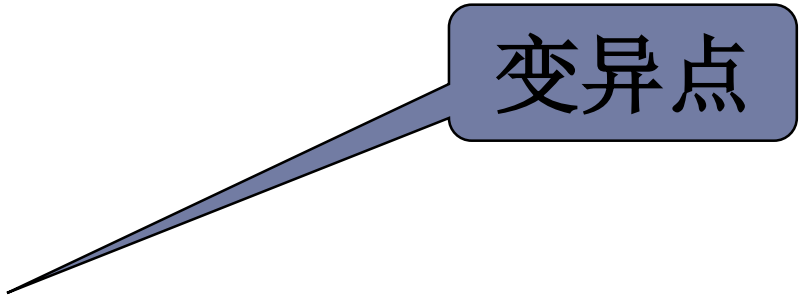
变异算子

所谓变异运算，是指依据变异概率 P_m 将个体编码串中的某些基因值用其它基因值来替换，从而形成一个新的个体。遗传算法中的变异运算是产生新个体的辅助方法，它决定了遗传算法的局部搜索能力，同时保持种群的多样性。交叉运算和变异运算的相互配合，共同完成对搜索空间的全局搜索和局部搜索。SGA中变异算子采用基本位变异算子。

基本位变异算子

基本位变异算子是指对个体编码串随机指定的某一位或某几位基因作变异运算。对于基本遗传算法中用二进制编码符号串所表示的个体，若需要进行变异操作的某一基因座上的原有基因值为0，则变异操作将其变为1；反之，若原有基因值为1，则变异操作将其变为0 。

基本位变异算子的执行过程



- 变异前：
- 00000111000**0**000010000
- 变异后：
- 00000111000**1**000010000

运行参数

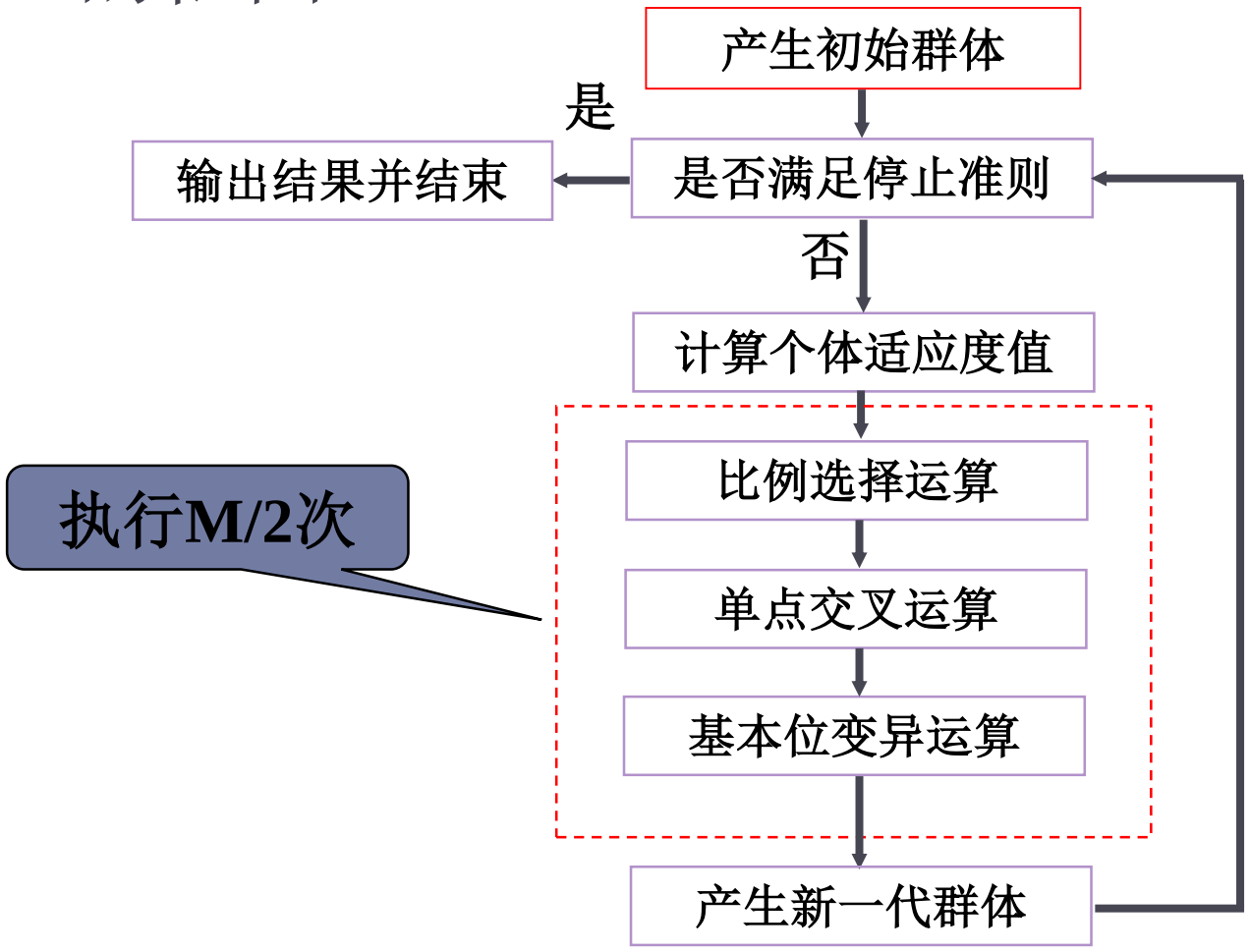
（1）M ： 种群规模

（2）T ： 遗传运算的终止进化代数

（3）P_c ： 交叉概率

（4）P_m ： 变异概率

SGA的框图


产生初始群体
是
输出结果并结束
是否满足停止准则
否
计算个体适应度值
比例选择运算
单点交叉运算
基本位变异运算
产生新一代群体
执行M/2次

控制参数的选择

Schaffer建议的最优参数范围是：

$M = 20\text{-}100$,

$T = 100\text{-}500$,

$P_c = 0.4\text{-}0.9$,

$P_m = 0.001\text{-}0.01$。

遗传算法的特点与优势

由上所述，我们看到,遗传算法模拟自然选择和有性繁殖、遗传变异的自然原理，实现了优化搜索和问题求解。遗传算法的主要特点和优势体现在以下几个方面:

(1)群体搜索，易于并行化处理；
(2)不是盲目穷举，而是启发式搜索；
(3)遗传算法寻找的是优秀解(最优解或次优解)，一般是设法尽快找到解(当然包括优解)，所以遗传算法又是一种优化搜索算法；
(4)遗传算法的适应性强，除需知适应度函数外，几乎不需要其他的先验知识；
(5)遗传算法长于全局搜索，它不受搜索空间的限制性假设的约束,不要求连续性，能以很大的概率从离散的、多极值的、含有噪声的高维问题中找到全局最优解。