研究性学习结题报告书

2020 年 10 月 14 日

课题名称 用信息方法研究遗传学问题

课题负责人 杨景云

课题成员 blablabla

指导教师 李丽华老师

所在班级 高二 (9) 班

1 约定

真值运算符 若 [] 内表达式为真，则是 1，否则是 0。

2 定义

2.1 基因集合

我们用 $\mathbb{G}$ 来表示基因集合。

对于只有显隐性的情况，基因集合由一系列大写字母和小写字母组成，大写字母表示显性，小写字母表示隐性。对于只有两对等位基因 `A,B` 的情况，$\mathbb{G} = \{\texttt{A,B,a,b}\}$。

对于另一些更复杂的情况，拿喷瓜举例，基因集合可以写作 $\mathbb{G} = \{\texttt{g}^-, \texttt{g}^+, \texttt{G}\}$。

2.2 对于集合元素的标号

创建基因集合到 $\{1, 2, \cdots |\mathbb{G}|\}$ 的映射 $f : \mathbb{G} \to \mathbb{Z}$。

基因的顺序就是标号的顺序。

容易发现其有逆运算 f'。

2.3 集合到向量的转化

一个集合 S 可以转化为一个 $|S|$ 维向量 v，其中 $v_i = [f'(i) \in S]$。

若基因集合为 $\{\texttt{A,B}\}$，`A` 标号为 1，`B` 标号为 2，那么集合 $\{\texttt{A}\}$ 可以转化为 $(1, 0)$。

2.4 基因片段

基因片段是一个向量。记基因片段组成的集合为 $\mathbb{P}$。

2.4.1 配子基因片段

我们用 $\vec{G}$ 来表示配子基因片段。

我们可以将一个具有 k 个基因的配子用一个 k 维有序向量 $\{a_i\}$ 表示，$a_i \in \mathbb{G}$。

2.4.2 个体基因片段

我们用 $\vec{I}$ 来表示个体基因片段。

我们可以将一个具有 k 对等位基因的个体用一个 k 维有序向量 $\{(l_i, r_i)\}$ 表示，$l_i, r_i \in \mathbb{G}$。

2.5 基因片段的运算

2.5.1 加法运算 $+$

对于 $L, R \in \mathbb{P}$，而且 L, R 同为配子基因片段或个体基因片段，定义加法运算为两基因片段的 **有序**拼接。

如 $(\texttt{A}, \texttt{C}) + (\texttt{B}) = (\texttt{A}, \texttt{B}, \texttt{C})$。

2.5.2 结合运算 $\oplus$

对于 $L, R \in \mathbb{P}$，而且 L, R 同为配子基因片段，而且长度相等，定义结合运算为按位有序结合：

$$(L \oplus R)_i = (\max(L_i, R_i), \min(L_i, R_i))$$

$\max, \min$ 为取序号较大/较小者。
如 $(\mathtt{A}, \mathtt{b}) + (\mathtt{a}, \mathtt{B}) = (\mathtt{(A,a)}, \mathtt{(B,b)})$。

2.6 生成函数 (Generating function)

定义：

$$\mathbf{A} = \sum_i a_i x^i$$

是序列 $\{a_i\}$ 的生成函数。
我们不关心 x 的取值和级数是否收敛，把 x 作为形式，只关心系数 a_i。

2.7 基因片段生成函数

定义：

$$\mathbf{A} = \sum_{i \in \mathbb{P}} a_i x^i$$

是序列 $\{a_i\}$ 的基因片段生成函数。

2.8 基因片段生成函数的系列运算

2.8.1 乘法运算 $\times$

$$x^L \times x^R = x^{L+R}$$

2.8.2 结合乘法运算 $\otimes$

$$x^L \otimes x^R = x^{L \oplus R}$$

2.9 基因片段生成函数的应用

- 求基因型为 AaBB 的个体产生的配子数量比。

构造生成函数：

$$\begin{aligned}\mathbf{G} &= (\frac{1}{2}x^{\mathtt{A}} + \frac{1}{2}x^{\mathtt{a}})(\frac{1}{2}x^{\mathtt{B}} + \frac{1}{2}x^{\mathtt{B}}) \\ &= \frac{1}{2}x^{\mathtt{AB}} + \frac{1}{2}x^{\mathtt{aB}}\end{aligned}$$

即配子数量比为 $\texttt{AB}:\texttt{aB}=1:1$。

- 求其自交后个体的基因型比例。

构造生成函数：

$$
\begin{aligned}
\mathbf{I} &= \mathbf{G}\otimes\mathbf{G} \\
&= \frac{1}{4}x^{\texttt{AABB}}+\frac{1}{2}x^{\texttt{AaBB}}+\frac{1}{4}x^{\texttt{aaBB}}
\end{aligned}
$$

即基因型数量比为 $\texttt{AABB}:\texttt{AaBB}:\texttt{aaBB}=1:2:1$。

2.10 表现型集合

定义 $\mathbb{E}$ 为表现型集合，一般地，$\mathbb{E}=\mathbb{G}$。

2.11 表现型映射

我们创建映射：$exp:\mathbb{G}\times\mathbb{G}\to\mathbb{E}$，对于一对等位基因 $l,r\in G$ 使得 $exp(l,r)$ 为这个个体的表现型。
比如 $exp(\texttt{A},\texttt{a})=\texttt{A}$，$exp(\texttt{a},\texttt{a})=\texttt{a}$。

2.12 表现型映射的性质

- $exp(i,j)=exp(j,i)$。
- $exp(i,i)=i$。

2.13 计算个体的表现型

个体的表现型可以用一个 k 维向量 $\vec{E}$ 表示，其中

$$\vec{E}_i=exp(\vec{I}_i)$$

2.14 卷积

给定环 R 上的 n 维向量 $\vec{A}=\{a_i\},\vec{B}=\{b_i\}$ 和下标运算 $\circ$，设 $C=\{c_i\}=A*B$，则满足：

$$c_i=\sum_{j,k}[j\circ k=i]a_jb_k$$

称 C 为 A 和 B 关于 $\circ$ 的离散卷积，以下简称卷积。
记 $C=A*_{\circ}B$，如果不引起混淆，简记为 $C=A*B$，其中 $*$ 为卷积算子。
若 $\circ=+$，就是我们熟悉的多项式乘法运算。

2.15 卷积与生成函数运算的联系

若满足运算 $x^L\times x^R=x^{L\circ R}$，那么生成函数 $\mathbf{F}=\sum f_ix^i$ 的乘法：

$$\mathbf{H}=\mathbf{F}\times\mathbf{G}$$

和卷积 $\vec{F} = \{f_i\}, \vec{G} = \{g_i\}, \vec{H} = \vec{F} *_\circ \vec{G} = \{h_i\}$ 等价。

2.16 快速沃尔什变换的实现

算法 1 快速莫比乌斯变换

输入: 集合幂级数 f
输出: f 的莫比乌斯变换

function FastMobiusTransform(f)
  for $i \leftarrow 1$ **to** n **do**
    for all $S \subseteq U \setminus \{i\}$ **do**
      $f_{S\cup\{i\}} \leftarrow f_{S\cup\{i\}} + f_S$
    end for
  end for
  return f
end function

算法 2 快速莫比乌斯反演

输入: 集合幂级数 f
输出: f 的莫比乌斯反演

function FastMobiusInversion(f)
  for $i \leftarrow 1$ **to** n **do**
    for all $S \subseteq U \setminus \{i\}$ **do**
      $f_{S\cup\{i\}} \leftarrow f_{S\cup\{i\}} - f_S$
    end for
  end for
  return f
end function

算法 3 多维广义离散傅里叶变换

输入: 幂级数 f，单位根 w_k，操作符 opr 代表正变换还是逆变换。
输出: f 的傅里叶变换

```
function FourierTransform(f, w_k, opr)
    if then opr = 1
        matrix_{i,j} = w_k^{(i-1)(j-1)}
    else
        matrix_{i,j} = (1/k) w_k^{-(i-1)(j-1)}
    end if
    for i ← 1 to n do
        for The k vectors satisfying 1···k on the i-th bit and the other bits are same. do
            v ← the k vectors
            for j ← 1 to k do
                g_j ← f_{v_j}
            end for
            g ← g × matrix
            for j ← 1 to k do
                f_{v_j} ← g_j
            end for
        end for
    end for
    return f
end function
```