研究性学习结题报告书

2020 年 10 月 24 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：

$$c_i=\sum_{j,k}[j\circ k=i]a_jb_k \tag{1}$$

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。

3 只有显隐性情况群体自由交配的计算

参考 2.9 中做法，我们需要分成两部分计算，第一部分是求配子生成函数 $\mathbf{G}$，第二部分是求基因片段生成函数 $\mathbf{I}$。

3.1 配子生成函数的求法

将基因片段对应到一个二进制数，如 $\texttt{AB}=(11)_2=3,\texttt{aB}=(01)_2=1$。

3.1.1 朴素求法

模拟生成配子的过程，每次生成一个长度为 k 的二进制数，若第 i 位为 0，则选择第 i 对等位基因的其中一个，否则选择另一个。

拿 AaBB 举例：

选择的二进制数	得到的配子
00	AB
01	AB
10	aB
11	aB

生成二进制数的时间复杂度（time complexity）为 $\mathcal{O}(2^k)$，而计算配子的时间复杂度为 $\mathcal{O}(k)$。

所以总时间复杂度是 $\mathcal{O}(k2^k)$，对于 n 个个体都计算一次，时间复杂度为 $\mathcal{O}(nk2^k)$，是不能接受的。

3.1.2 快速做法

考虑维护配子出现次数函数 f，一开始为 $x^{\texttt{None}}$，考虑每次加入一对基因，f 的变化。假设它变为 f'。

若加入的基因是一对显性基因，如 AA，那么 $f'(x\times 2+1)=2f(x)$。

若加入的基因是一个显性和一个隐形基因，如 Aa，那么 $f'(x\times 2+1)=f(x),f'(x\times 2)=f(x)$。

若加入的基因是一对隐性基因，如 aa，那么 $f'(x\times 2)=2f(x)$。

加入 k 等位基因，每次都 $\mathcal{O}(2^k)$ 计算，时间复杂度和上面没有区别，看似没有优化。

但是程序处理时,加入到第 i 个等位基因时,可以只用考虑 $0\sim 2^i$ 的函数值,总时间复杂度是 $\mathcal{O}(\sum_{i=1}^{k}2^i)=\mathcal{O}(2^k)$，可以将一个 k 优化掉。

对于 n 个个体都计算一次，时间复杂度为 $\mathcal{O}(n2^k)$，比较快速。

3.2 基因片段生成函数的求法

我们想求出一个基因片段生成函数乘法的快速实现。

朴素做法

考虑朴素地实现 (1) 中的卷积，要枚举 j,k，通过下标运算 $\circ$ 计算出 c_i，时间复杂度是 $\mathcal{O}(4^k)$，是不能接受的。

优化的第一步

我们发现 **对于只有显隐性情况的基因片段生成函数，可以转化为集合生成函数**。而且集合生成函数已经存在快速算法。

集合生成函数

可以使用符号：

$$f = \sum_{S \subseteq U} f_S x^S$$

来表示一个集合生成函数。

这里我们定义算子 $\circ = \cup$，即：$x^L \times x^R = x^{L \cup R}$。

容易发现集合生成函数的乘法运算恰好为 **集合并卷积**。

基因片段生成函数到集合生成函数的转换

定义全集 U 是：$\{\mathtt{A}, \mathtt{B}, \cdots\}$。

我们将基因片段中的显性基因抽取出来，形成一个集合，如 `ABc` $\Rightarrow \{\mathtt{A}, \mathtt{B}\}$。

这样发现集合并卷积刚好符合“显性基因克制隐形基因”的条件，因为只要某一位有对应的显性基因，那么个体就表现为显性，可以结合集合运算表来理解：

$\cup$	$\{\mathtt{A}\}$	$\varnothing$
$\{\mathtt{A}\}$	$\{\mathtt{A}\}$	$\{\mathtt{A}\}$
$\varnothing$	$\{\mathtt{A}\}$	$\varnothing$

集合生成函数的快速卷积算法：FWT

仿照 FFT 的思路，**我们求出 f 的一种变换 $\hat{f}$，使得 $f * g = h \Rightarrow \hat{f}_i \times \hat{g}_i = \hat{h}_i$**，即将系数表示法转化为点值表示法。

我们给出关于集合并卷积的 FWT 运算，即快速莫比乌斯变换。

$$\hat{f}_S = \sum_{T \subseteq S} f_T$$

证明：

$$\begin{aligned}
\hat{h}_S &= \sum_L \sum_R [(L \cup R) \subseteq S] f_L g_R \\
&= \sum_L \sum_R [L \subseteq S][R \subseteq S] f_L g_R \\
&= \sum_L [L \subseteq S] f_L \sum_R [R \subseteq S] g_R \\
&= \hat{f}_S \hat{g}_S
\end{aligned}$$

我们求出 $\hat{h}_S$ 后，当然需要将 $\hat{h}$ 转化为 h，于是需要反演运算：

$$f_S = \sum_{T \subseteq S} (-1)^{|S|-|T|} \hat{f}_T$$

可以用容斥简单证明。

朴素的变换和反演的实现

枚举 T 和 S，并且判断是否 $T \subseteq S$，时间复杂度 $\mathcal{O}(4^n)$，没有太大的变化。

经过优化的变换和反演的实现

通过程序精细实现，能够以 $\mathcal{O}(2^{|S|})$ 的时间复杂度枚举 S 的子集。

如果对于所有的 $S \subseteq U$，都这样枚举子集，时间复杂度为：

$$\mathcal{O}(\sum_{i=1}^{k} \binom{k}{i} 2^i) = \mathcal{O}(3^k)$$

比上述做法稍有进步。

进一步优化的变换和反演的实现

我们使用递推的思路，推导出 $\hat{f}_S$。

设 $\hat{f}_S^{(i)} = \sum_{T \subseteq S} [(S \setminus T) \subseteq \{1, \cdots, i\}] f_T$，$\hat{f}_S^{(n)}$ 即是目标序列。

首先有 $\hat{f}_S^{(0)} = f_S$，因为只有当 $S \setminus T$ 为空集时，才能属于空集。

对于所有 $i \notin S$ 的 S，满足 $\hat{f}_S^{(i)} = \hat{f}_S^{(i-1)}, \hat{f}_{S \cup \{i\}}^{(i)} = \hat{f}_S^{(i-1)} + \hat{f}_{S \cup \{i\}}^{(i-1)}$。

我们解释一下两个式子。

$$\begin{aligned}
\hat{f}_S^{(i)} &= \sum_{T \subseteq S} [(S \setminus T) \subseteq \{1, \cdots, i\}] f_T \\
&= \sum_{T \subseteq S} [(S \setminus T) \subseteq \{1, \cdots, i-1\}] f_T \\
&= \hat{f}_S^{(i-1)}
\end{aligned}$$

这里我们发现 $i \notin (S \setminus T)$，所以可以直接把 $\{i\}$ 去掉，也是等价的。

$$\begin{aligned}
\hat{f}_{S \cup \{i\}}^{(i)} &= \sum_{T \subseteq (S \cup \{i\})} [(S \cup \{i\}) \setminus T) \subseteq \{1, \cdots, i\}] f_T \\
&= \sum_{T \subseteq (S \cup \{i\}) \text{ and } i \notin T} [((S \cup \{i\}) \setminus T) \subseteq \{1, \cdots, i\}] f_T + \sum_{T \subseteq (S \cup \{i\}) \text{ and } i \in T} [((S \cup \{i\}) \setminus T) \subseteq \{1, \cdots, i-1\}] f_T \\
&= \sum_{T \subseteq S \text{ and } i \notin T} [(S \setminus T) \subseteq \{1, \cdots, i-1\}] f_T + \sum_{T \subseteq (S \cup \{i\}) \text{ and } i \in T} [((S \cup \{i\}) \setminus T) \subseteq \{1, \cdots, i-1\}] f_T \\
&= \hat{f}_S^{(i-1)} + \hat{f}_{S \cup \{i\}}^{(i-1)}
\end{aligned}$$

这样，我们 $\mathcal{O}(n2^n)$ 求出 $\hat{f}_S, \hat{g}_S$，按位乘，然后再反演回去即可。

快速莫比乌斯变换和反演的伪代码实现

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

4 只有显隐性情况群体自由交配的计算的推广

4.1 共显性问题

有一种花卉，基因型为 `AA` 时表现为红色，基因型为 `Aa` 时表现为粉色，基因型为 `aa` 时表现为白色。

我们将基因片段中的显性和隐性基因抽取出来，形成一个集合，如 $\texttt{ABc} \Rightarrow \{\texttt{A},\texttt{B},\texttt{c}\}$。

也可以理解为把一对等位基因拆成两位：

- $\texttt{A} \Rightarrow 10$
- $\texttt{a} \Rightarrow 01$

容易发现这样做的时间复杂度为 $\mathcal{O}(2k \times 2^{2k}) = \mathcal{O}(2k \times 4^k)$，和朴素做法差不多，是不可接受的。

4.2 喷瓜问题

喷瓜的性别由等位基因 $\texttt{g}^-,\texttt{g}^+,\texttt{G}$ 决定，其中：

$\oplus$	$\mathtt{g}^-$	$\mathtt{g}^+$	$\mathtt{G}$
$\mathtt{g}^-$	$\mathtt{g}^-$	$\mathtt{g}^+$	$\mathtt{G}$
$\mathtt{g}^+$	$\mathtt{g}^+$	$\mathtt{g}^+$	$\mathtt{G}$
$\mathtt{G}$	$\mathtt{G}$	$\mathtt{G}$	$\mathtt{G}$

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
        $matrix_{i,j} = \frac{1}{k} w_k^{-(i-1)(j-1)}$
    end if
    for $i \leftarrow 1$ **to** n **do**
        for `The k vectors satisfying` $1 \cdots k$ `on the i-th bit and the other bits are same.` **do**
            $v \leftarrow$ `the k vectors`
            for $j \leftarrow 1$ **to** k **do**
                $g_j \leftarrow f_{v_j}$
            end for
            $g \leftarrow g \times matrix$
            for $j \leftarrow 1$ **to** k **do**
                $f_{v_j} \leftarrow g_j$
            end for
        end for
    end for
    return f
end function