Q38-关于一种特殊斯坦纳系的构造解法

查翔[1], 查飞[2], 周义超[3]

[1] 复旦大学 21307140052

[2] 中国科学技术大学 PB21010418

[3] 复旦大学 21307140094

摘要

摘要：选题所描述的特殊平衡的不完全区组设计问题实际上等价于一种特殊斯坦纳系，即 $S(n^2, n, 1)$ 的构造问题。本文经过数学建模后，对 n 作为质数与合数分开进行了讨论。对质数来说，可以对分组情况进行简单的数学公式化描述；而对合数，则需满足特殊数列 W(y) 的存在性方可求解。因此，本文对 $S(n^2, n, 1)$ 问题的可解性，猜想只有当 n 满足 $n = p^q$ 时 (p 为质数) 时，原问题有解，称之为质方猜想。

关键字：特殊斯坦纳系、W(y) 数列、质方猜想

1 问题背景

平衡不完全区组设计 (balanced incompleteblock design) 是一类重要的区组设计。若 X 为 v 元点集，居为 X 的一些 k 元子集组成的族 (这些 k 元子集称为区组)，使得 X 中的任意点对少恰好出现在几个区组中，则称此区组设计为一个平衡不完全区组设计。区组设计在试验设计、计算优化、通讯和同步优化有着广泛的应用，因此，对平衡不完全区组设计问题的研究，确有其实际意义。

该问题最早来源于 19 世纪中叶英格兰数学家科克曼提出的 “15 女生问题”，又叫做 “科克曼女生散步问题”。该问题提出，15 个女生每天出去散步一次，每次散步 3 个人一组，如何分组能使得 7 天内任意两个人恰好在一起散步一次。其问题实际上就是 BIBD 问题中参数 $(b, v, r, k, \lambda) = (35,15,7,3,1)$ 的情况。科克曼同时给出了一个答案，但碍于当时的信息并不流通，科克曼提出的这个问题并未受到重视。而到后来几何学家斯坦纳在研究四次曲线的两切线问题时再次提出了这一组合问题并发表在克雷儿杂志上，由于这一原因以及斯坦纳的声望，B(3,1,v) 一直被称为斯坦纳三元系，并将一般的 B(k,1,v) 称作斯坦纳系。显而易见，选题即 $v = k^2$ 的情形。

对于一般的平衡不完全区组设计 BIBD(balence incomplete block design) 的研究中，费舍尔 (R.AFisher) 和叶斯 (F.Yates) 在 1938 年的一本著作 [Fisher R A,Yates F. Satistical Tables for Biological ,Agricultural and Medical Research.Edinburgh:Oliver and Boyd,1938.156] 中给出了一个 $B(k, \lambda, v)$ 的各个参数间满足的基本关系式：rv = bk 与 $r(k-1) = \lambda(v-1)$，其中 r 表示每个区组中都含有 r 个元素，b 表示全部的区组个数。显然，这是一个 $B(k, \lambda, v)$ 的必要而非充分条件。1940 年，费舍尔又得到了 $B(k, \lambda, v)$ 存在的一个限定条件，即费舍尔不等式 $v \leq b$.

给出和证明 $B(k, \lambda, v)$ 存在性的充要条件是很困难的。三元系在被提出后，又经过了 100 年左右的探索明知道 1961 年才由哈纳尼 (H.Hanani) 证明了下面的式子确是 $B(3, \lambda, v)$ 设计的充分条件：

$$\lambda\, v - 1 \ = 0 \quad (mod 2)$$

$$\lambda v(v-1) = 0 \quad (mod 6)$$

同时，他还指出这也是 $B(4, \lambda, v)$ 设计存在的充分条件。$k \geq 5$ 时，问题变得更加复杂，对每个指定过的 $k \geq 5$, 都满足上式但却不存在 $B(k, \lambda, v)$ 的 (v, λ) 数值组。1975

年，威尔逊 (R.M.Wilson) 证明了："对给定的正整数 k 和 λ，除去有限个正整数 v 以外，上面所给的式子时 $B(k,\lambda,v)$ 存在的充要条件。其中 $v \geq k$. 这一结论宣告了 $B(k,\lambda,v)$ 存在性问题的基本解决。

以上便是平衡不完全区组设计问题的背景，通过对背景知识的大概了解，可以清楚解决 BIBD 问题的难度，虽然选题属于 BIBD 问题的一个特例，但其难度并不小，下面给出一个关于此类特殊问题的一个构造性解法。

2 问题分析

对于选题中的斯坦纳系 $(k^2,k,1)$ 问题，可仿照科克曼女生问题将其描述为 k^2 个女生每天出去散步，k 人一组，共 k 组，规定 k+1 天内任意两人恰有一次编在同组，要求给出一个连续散步 k+1 天的分组方案。

为了方便起见，设定记录天数的方式由 $1 \sim k+1$ 天变为 $0 \sim k$ 天，其中任意一天编号为 n $0 \leq n \leq k$. 同时，将女生的编号 $number(number \in [0,k^2-1])$ 变为 $i\{j\}(i,j \in [0,k-1])$ 的形式, 实际上，number = i * k + j，二者等价。易知，对于身份编号 $a\{b\}$ 与 $a'\{b'\}$, 只有当且仅当 a=a' ,b=b' 时，二者才是同一人。

下面参考了一种可行的构造方法，并以此给出具体的分组方案。

2.1 k 为质数

第一步：确定第 k 天 (也就是最后一天) 时的分组情况

由于首先讨论第 k 天，其余天数尚未明确，不妨以组位号作为女生编号，即 i 组 j 位的女生编号为 $i\{j\}$。于是，第 k 天的身份分组的分配方式如下：

$$\begin{aligned}
&\text{第 0 组}: 0\{0\}, 0\{1\}, 0\{2\} \cdots, 0\{k-1\}\\
&\text{第 1 组}: 1\{0\}, 1\{1\}, 1\{2\}, \cdots, 1\{k-1\}\\
&\vdots\\
&\text{第 k-1 组}: k-1\{0\}, k-1\{1\}, k-1\{2\}, \cdots, k-1\{k-1\}
\end{aligned}$$

第二步: 确定第 $0 \sim (k-1)$ 天时的 i 组 j 位上的女生编号

定义运算法则：

$$\{a\} = \{b\}, \quad iff \quad a \equiv b \pmod{k}$$

于是，第 n 天 i 组 j 位的女生编号规定为 $j\{nj+i\}$, 实际上为 $j\{nj+i \mod k\}$

例如第 n 天第 i 组的分组情况：

$$0\{i\}, 1\{j+i\}, 2\{2j+i\}, \cdots, k-1\{(k-1)j+i\}$$

第三步：证明以上分组符合要求

引理 1 当 k 为质数，且 $i,j,n \in [0,k-1]$，任意 $i \neq nj+i \pmod{k}$

证明 1 若存在 $i = nj+i \pmod{k}$

则

$$nj+i = sk+i$$

于是 $nj = sk$

因为 sk 里面含有质因子 k，而 $n,j < k$，所以 n 和 j 都不包含质因子 k

故 nj 不包含质因子 k

所以 $nj \neq sk$, 矛盾

故假设不成立，原命题得证。

求证一：任意第 n 天内的组位号都可以找到唯一一个与之对应的女生身份编号

引理 2 任意第 n 天内都不会出现相同的身份编号

证明 2 若第 n 天里的 a 组 b 位与 a’ 组 b’ 位的身份重合，即

$$b\{nb+a\} = b'\{nb'+a'\}$$

则

$$b = b'\ nb+a \equiv nb'+a' \pmod{k}$$

所以 a = a’ , 即同一天内只有组位号一致，才能身份一致

引理 3 任意第 n 天内的组位号都可以找到一个与之对应的身份编号

证明 3 由第 n 天的分组规则，显然

于是，由上面两个引理，任意第 n 天内的每个区位号对应着一个身份编号，且各不相同，所以任意第 n 天内的区位号都可以找到唯一一个与之对应的女生身份编号。

求证二：任意两个女生曾被且仅被分到同一组

不妨假设两个女生在第 n 天的 i 组一起散步过，其身份编号为 $a\{b\}$ $a'\{b'\}$

引理 4 在第 k 天一组的两个女生不会在 $0 \sim k-1$ 天还是同一组，在 $0 \sim k-1$ 的某一天中分到同一组的两个女生不会在第 k 天被分到同一组

证明 4 依据第 k 天时的分组，可知 $a = a'$，而根据 $0 \sim k-1$ 天内的分组情况，同一组中 $a \neq a'$, 所以这两个女生不会在 $0 \sim k-1$ 天再被分到同一组。

同理，若她们被分到第 k 天，也不会在 $0 \sim k-1$ 天中被分到同一组

引理 5 在 $0 \sim k-1$ 天中的某一天被分到同一组的两个女生，不会在 $0 \sim k-1$ 天中其它某一天依然被分到同一组

证明 5 不妨假设两位女生在第 n_1 天被同时分到第 i 组的 j_1 和 j_2 位

则

$$a = j_1, a' = j_2, b \equiv n_1 j_1 + i, b' \equiv n_1 j_2 + i(均为 modk)$$

若她们还在第 n_2 天被同时分到第 i' 组的 j_1' 和 j_2' 位

则

$$a = j_1', a' = j_2', b \equiv n_1 j_1' + i', b' \equiv n_1 j_2' + i'(均为 modk)$$

经过推导后可得

$$(n_1 - n_2)(a - a') \equiv 0(modk)$$

所以 $(n_1 - n_2)(a - a') = sk(s为整数)$,

因为 $1-k \leq n_1 - n_2 \leq k-1$, 且 $n_1 - n_2 \neq 0, 1-k \leq a-a' \leq k-1$, 且 $a-a' \neq 0$,

又因为 $n_1 - n_2$ 与 a-a' 均不包含质因子 k, 所以

$$(n_1 - n_2)(a - a') \neq sk$$

矛盾, 所以她们不会被分到其它 *0 k-1* 的某一天的同一组中。

综上，任意两个女生会且仅会被分到同一组中。

至此，k 为质数时的斯坦纳系 $(k^2, k, 1)$-设计构造完成，其任意 n 天 $(n \in [0, k])$i 组 j 位对应的女生身份编码表达式为：

$$a\{b\} = j\{nj + i\}, n \in [0, k-1]$$
$$a\{b\} = i\{j\}, n = k$$

本设计运用质数性质，构造了 k 为质数时斯坦纳系 $(k^2, k, 1)$-设计，但当 k 为合数时，$i \neq nj + i(modk)$ 不恒成立，因此，此公式仅仅适合 k 为质数时的设计。

下面以 k=3 为例进行设计：

第 0 天：

00，10，20；01，11，21；02，12，22

第 1 天：

00，11，22；01，12，20；02，10，21

第 2 天：

00，12，21；01，10，22；02，11，20

第 3 天：

00，01，02；10，11，12；20，21，22

等价于：

第 0 天：

0，3，6；1，4，7；2，5，8

第 1 天：

0，4，8；1，5，6；2，3，7

第 2 天:
0，5，7；1，3，8；2，4，6
第 3 天:
0，1，2；3，4，5；6，7，8
经过验证后符合要求

2.2 k 为偶合数

当 k 为合数时，由于 $i \neq i + nj(modk)$，因此无法完成 $(k^2, k, 1)$-设计。为了完成 k 为合数时的 $(k^2, k, 1)$-设计，可进行如下构造:

2.2.1 第一步：构造 W(y) 数列

构造规则如下:

- 数列 W(y) 由 2 y 的连续自然数，共 y-1 项组合而成;
- 定义其加法运算法则为 mod(y+1);
- W(y) 数列满足：数列任意起点连续 z 项相加，其和不相等，不等于 z，且不等于 0

规则解释如下:
如当 y=6 时，可构造如下数列 $W(6) = \{5\ 6\ 4\ 2\ 3\}$.
按规则一：由 6-1=5 项构成
按规则二：5+6=11 mod7=4 , 6+4+2+3=15 mod7=1
按规则三：例如当 z=2 时，

$$[5 + 6 = 4] \neq [6 + 4 = 3] \neq [4 + 2 = 6] \neq [2 + 3 = 5] \neq 2 \text{ 且 } \neq 0$$

当 z=3 时，

$$[5 + 6 + 4 = 1] \neq [6 + 4 + 2 = 5] \neq [4 + 2 + 3 = 2] \neq 3 \text{ 且 } \neq 0$$

根据构造规则,W(y) 数列的排列具有可逆性,因此 $W(6) = \{5\ 6\ 4\ 2\ 3\} = \{3\ 2\ 4\ 6\ 5\}$, 性质相同，我们称之为鸳鸯数列。在以后的计算中，任取其一即可。

2.2.2 第二步: 构造第 k 天 (最后一天) 时的分配方案

分配方案同 1 情形

2.2.3 第三步：当 y=2m 时，利用 W(y) 数列构造 $(k^2,k,1)$-设计，k=y+2

以 W(6) 数列为例，利用 W(6) 构造 $(8^2,8,1)$-设计 (8=6+2)。

1. 将 0 k-1 依次排成 0 列，本设计中，将 0 7 依次排列构成 0 列。如下：

   0 列：0，1，2，3，4，5，6，7

2. 将 1，0，x，W(6) 依次排序构成 X 列，如下：

   X 列：1，0，x，5，6，4，2，3

3. 分别将 $3\cdots7$ 代入 x，与 W(6) 各项依次进行递加运算，得到 1 列，要求:

   - 定义加法运算法则为 mod(y+1) 运算。且当 $(x+a_1+a_2+\cdots+a_s)=0(mody+1)$ 时，取 0 = y+1，此设计中，y+1=7;
   - 0 列 j 位数字不能和 1 列 j 位数字重合，否则继续代入其它 $3\cdots7$;
   - 1 列中 $(x+a_1+\cdots+a_n)mod7$ 值不可出现 1，否则继续代入其它 $3\cdots7$

     如当 x=3 时，
     0 列：0，1，2，3，4，5，6，7
     X 列：1，0，3，5，6，4，2，3
     1 列：1，0，3，$1\cdots$
     不符合要求 3(注:3+5 = 1(mod7))

     当 x=6 时，符合构造要求
     0 列：0，1，2，3，4，5，6，7
     X 列：1，0，6，5，6，4，2，3
     1 列：1，0，6，4，3，7，2，5

     当 x=7 时，
     0 列：0，1，2，3，4，5，6，7
     X 列：1，0，7，5，6，4，2，3
     1 列：1，0，7，5，$4\cdots$
     不符合要求 2(注：7+5+6=4 mod7)
     将符合要求的 1 列除首项 “1” 外的 0，6，4，3，7，2，5 成为本源数列 A(n)，$n\in[0,k-1]$，对应代码为 $A(0),A(1\sim6)$. 由 W(y) 构造规则可知，除 A(0)(A(0)=0) 外，任意 $A(n+z)-A(n)\neq A(n+L+z)-A(n+L)\neq z$

4. 按照如下方式构造 $2\sim7$ 列：

   (1). 每列首项数值递增，并与列名吻合;

(2). 每增加 1 列，0 后退 1 位；

(3). 除 0 外，每增加 1 列，A(n) 后退 1 位并且数值加 1，变为 A(n+1);

(4). 定义加法运算法则为 mod(y+1) 运算，如：y+1 = y+1，6+1=0=7，7+1=1(mod 7)；

(5). 最右端数字向下列移动至次列第 1 位

可知：A(1) A(6) 在移动过程中不会转换成 0

设计结果如下：

0 列：0,1,2,3,4,5,6,7

1 列：1,0,6,4,3,7,2,5

2 列：2,6,0,7,5,4,1,3

3 列：3,4,7,0,1,6,5,2

4 列：4,3,5,1,0,2,7,6

5 列：5,7,4,6,2,0,3,1

6 列：6,2,1,5,7,3,0,4

7 列：7,5,3,2,6,1,4,0

以上 0 7 列成为等价数列

5. 将 0 ∼ 7 列横向铺开，并对应代入组值 i，可得身份编码 $a\{b\}$，构成第一天散步分配方案，如下所示：

   第 0 组：00，11，22，33，44，55，66，77

   第 1 组：01，10，26，34，43，57，62，75

   第 2 组：02，16，20，37，45，54，61，73

   第 3 组：03，14，27，30，41，56，65，72

   第 4 组：04，13，25，31，40，52，67，76

   第 5 组：05，17，24，36，42，50，63，71

   第 6 组：06，12，21，35，47，53，60，74

   第 7 组：07，15，23，32，46，51，64，70

6. 1 ∼ 7 天时，每增加 1 天，身份编号 $a\{b\}$ 中的 a 位值不变，b 位值向左移动 1 位，1 位值移动到末位，可构造出 2 ∼ 7 天的散步方式，如下为第 2 天的分组方案：

   第 0 组：00，12，23，34，45，56，67，71

   第 1 组：01，16，24，33，47，52，65，70

   第 2 组：02，10，27，35，44，51，63，76

   第 3 组：03，17，20，31，46，55，62，74

   第 4 组：04，15，21，30，42，57，66，73

   第 5 组：05，14，26，32，40，53，61，77

第 6 组：06，11，25，37，43，50，64，72
第 7 组：07，13，22，36，41，54，60，75
其它 $2 \sim 7$ 天的分组方案在此不再赘述

7. 将 $(0 \sim 7)\{0\}, \cdots (0 \sim 7)i, \cdots, (0 \sim 7)\{7\}$ 依次放入第 0 天，构成第 0 天的分组方案

   至此 $(8^2, 8, 1)$-设计构造完毕，详细分组方案见附表

2.2.4 第四步: 证明

1. 任意第 n 天时，不同组位号 ij 和 $i'j'$ 对应的身份编号 $a\{b\}, a'\{b'\}$ 不同：

   证明 6 当 $n=0$ 或 k 时，显然成立

   当 $n \in [1, k-1]$ 时，由于第 2 到第 $k-1$ 天均由第一天变换而来，只需考虑第一天即可。

   对于第 1 天，首先若在同一组中，则 $a \neq a'$, 故不会相同

   若在不同组中，要想身份编号相同，则需位号相同，即 $j=j'$，而在第一天的分组构造 *(参考第三步)* 中，相同位号的身份编号中的 b 值不同，即 $b\,b'$, 故二者也不相同

   由上述讨论可知：同一天中不同组位号对应的身份编号不相同，原命题得证

2. 任意第 n 天任意同组内的身份编号，不会再次出现在任意第 m 天的同组内

   证明 7 当 $n=0$ 时，由于第 0 天中同一组内身份编号 $a\{b\}$ 中的 b 值均相等，在第 $1 \sim K$ 天中同一组内 b 值各不相同，所以，第 0 天出现在同一组的身份编号不会出现在第 $1 \sim k$ 天中的同一组内。

   当 $n=k$ 时，由于第 k 天中同一组内身份编号 $a\{b\}$ 中的 a 值均相等，在第 $0 \sim K-1$ 天中同一组内 a 值各不相同，所以，第 k 天出现在同一组的身份编号不会出现在第 $0 \sim k-1$ 天中的同一组内。

   当 $n \in [1, k-1]$ 时，首先肯定的是出现第 $1 \sim k-1$ 天同一组的两个身份编号不会出现在第 0 天和第 k 天的同一组中。

   其次当其中一个身份编号为 $0\{0 \sim k-1\}$ 时，则其所在组的组号不会随着天数的变化而变化，而另一个的组号则不会保持原状，所以二者不会再在同一组内。

   当两个身份编号中的 a 都不为 0 时，由第三步可推出第 n 天 i 组 j 位的身份编号为 $j\{A[n+j-i-1]+i-1\}$*(注：*$n+j-i-1$ 为 $mod\, k-1$ 操作，$A[n+j-i-1]+i-1$ 为 $mod\, k$ 操作*)*。

假设两个女生在第 n 天第 i 组的 j_1 位与 j_2 位，则

$$A[n+j1-i-1]+i-1=b$$
$$A[n+j2-i-1]+i-1=b\text{'}$$

如果她们两还出现在第 n' 天第 i' 组的 j_1 位和 j_2 位，则

$$A[n\text{'}+j_1-i\text{'}-1]+i\text{'}-1=b$$
$$A[n\text{'}+j_2-i\text{'}-1]+i\text{'}-1=b\text{'}$$

易知 $i\text{'}\neq i, n\text{'}\neq n$ 则

$$A[n+j_1-i-1]-A[n+j_2-i-1]=A[n\text{'}+j_1-i\text{'}-1]-A[n\text{'}+j_2-i\text{'}-1]$$

由第三步中本源数列是由 *W(y)* 数列得到的，而根据 *W(y)* 数列的要求 *3*，可知

$$A[n+j_1-i-1]-A[n+j_2-i-1]\ A[n\text{'}+j_1-i\text{'}-1]-A[n\text{'}+j_2-i\text{'}-1]$$

矛盾，故她们不会再被分到一组。综上，原命题得证。

2.3 k 为奇合数

2.3.1 第一步：构造 W(y) 数列

方法同偶合数，如 y=7 时，$W(7)=\{7,5,6,3,4,2\}=\{2,4,3,6,5,7\}$

2.3.2 第二步：构造第 k 天时的分配方案

方法同偶合数

2.3.3 第三步：当 y=2m+1 时，利用 W(y) 数列构造 $(k^2,k,1)$-设计 (k = y+2)

以 W(7) 数列为例，利用数列 W(7) 构造 $9^2,9,1$-设计 (9=7+2)

1. 同偶合数，将 $0\sim k-1$ 依次排序构成 0 列，本设计中，将 $0\sim 8$ 依次排列构成 0 列 0 列：0，1，2，3，4，5，6，7，8
2. 与偶合数情形不同，将 1，0，x，W(7) 按照如下要求排列构成 X 列，规则如下：
   - 首项为 1；
   - 0 位于 j=(k+1)/2 处；
   - x 位于 0 后；
   - x 后排列 W(y) 数列，到了 j=y+1 处，就转移到 j=1 处

以 $W(7)=\{7\ 5\ 6\ 3\ 4\ 2\}$ 为例，结果如下：X 列：1，6，3，4，2，0，x，7，5

3. 分别将 $3\cdots7$ 代入 x，与 W(7) 各项依次进行递加运算，要求同 2，将符合要求的 X 列称为 1 列。所得 1 列如下：

   1 列：1，5，8，4，6，0，3，2，7

   将 $0,3,2,7,5,8,4,6$ 称为本源数列 A(n)，$n\in[0,k-1]$，对应代码为 A(0),$A(1)\sim A(7)$，同样，由 W(y) 构造规则可知，除 A(0)[A(0) = 0] 外任意 $A(n+z)-A(n)\neq A(n+L+z)-A(n+L)\neq z$.

4. 同 2 构造等价数列，如下所示：

   0 列：0，1，2，3，4，5，6，7，8
   1 列：1，5，8，4，6，0，3，2，7
   2 列：2，8，6，1，5，7，0，4，3
   3 列：3，4，1，7，2，6，8，0，5
   4 列：4，6，5，2，8，3，7，1，0
   5 列：5，0，7，6，3，1，4，8，2
   6 列：6，3，0，8，7，4，2，5，1
   7 列：7，2，4，0，1，8，3，5，6
   8 列：8，7，3，5，0，2，1，6，4

   同样可知，$A(1)\sim A(7)$ 在移动过程中不会转换成 0.

   5、6、7 同偶合数情形，则 $(9^2,9,1)$-设计构造完毕。

2.3.4 第四步：证明

证明方法同偶合数，只不过当两个女生的身份编号的 a 值均不为 0 时，其 n 天 i 组 j 位的身份编号推导表达式为

$$a\{b\}=j\{A[n+j+(k+1/2)-1-i]+i-1\}$$

，其余不变，推导方法类似，在此不再赘述。

3 理论及算法实现

3.1 输入

选题要求的输入模式有两种，一是输入两位女生的身份编号以求其所在同一组的编号，二是输入一个组号以求其所有女生的身份编号。

针对这中情况，首先需要根据输入的不同确定对应的模式，这里用 type 表示。易知输入中“,”的数量决定着输入的模式。可以从这个方面来决定 type，再根据 type 的

值决定实现哪一种输出。规定输出女生身份编号 type=0，输出组号 type=1。具体可由 *get_value* 函数实现。以输入：总人数，总组数，任意两人 ID 为例：(注：input 是 string 类型表示输入，*input_type* 是一个局部变量表示都逗号的数量，group, ID1, ID2 分别表示输入的组数，两位女生的身份编号)

```
for(int i = 0 ; i < input.length() ; i++){
    if(input[i] - ',' == 0) {
        input_type--;
        continue;
    }
    if(input_type == 3){
        n_square = n_square * 10 + input[i] - '0';
    }
    else if(input_type == 2){
        group =  group * 10 + input[i] - '0';
    }
    else if(input_type == 1){
        ID1 = ID1 * 10 + input[i] - '0';
    }
    else ID2 = ID2 * 10 + input[i] - '0';
}
```

这样便成功收集到了总人数、总组数及两人 ID 的数据，第二种输入模式数据的采集与其类似，详细内容参考源码

3.2 *BIBD_group* 数组的构建

BIBD_group$[n+1][n][n]$ 是一个三维数组，存储着 n+1 天内 n^2 个女生的分组信息，按照 2 中给出的方法，有两种情况，分为质数和合数。**step1: 判断 n 为质数还是合数**

采用遍历算法即可

```
int prime_number(int n){
    if(n == 1) return 0;
    if(n == 2) return 1;
    else{
        for(int i = 2 ; i < sqrt(n) + 1 ; i++){
            if(n % i == 0) return 0;
        }
```

```
    }
    return 1;
}
```

step2: 在 n 为质数或合数的基础上实现 2 中算法

case1：n 为质数由 3 中的推导可知 $BIBD\_group[i][j][k] = k*n+(j+i*k)modn(i < n)$，利用循环结构赋值即可。如下：

```
for(int i = 0 ; i < n+1 ; i++){
    for(int j = 0 ; j < n ; j++){
        for(int k = 0 ; k < n ; k++){
            if(i < n)BIBD_group[i][j][k] = k * n + (j + i * k
                ) % n;
            else BIBD_group[i][j][k] = j * n + k;
        }
    }
}
```

case2:n 为合数

(1)W(y) 数列的求解,(n = y + 2) W(y) 数列的求解是解决当 n 为合数时 $S(n^2, n, 1)$-设计的关键，若 W(y) 数列不存在，则无法通过本文中所给的方法来求解 $S(n^2, n, 1)$-设计。下面给出一个求解 W(y) 数列的算法, 2 到 11 的可构造 W(y) 数列：

W(2) = {2}

W(3) = {2, 3}

W(5) = {3, 2, 5, 4}

W(6) = {3, 2, 4, 6, 5}

W(7) = {2, 4, 3, 6, 5, 7}

W(9) = {2, 7, 8, 6, 5, 3, 4, 9}

W(11)= {6, 10, 5, 4, 2, 11, 9, 8, 3, 7}

由上述数列可猜想：

$$W(n) + W(y - n) = y + 2$$

为了简化 W(y) 数列的构造，默认为 W(y) 数列具有这样一种和对称的性质。在这样一种特性的基础上，可以使用回溯法来进行对 W(y) 数列的构造。

使用回溯法，则需要一种判断标准 (即生成的 W(y) 数列是否合格) 来判断是否回溯，为此，可以采取生成一种三角阵列的方式来直观显示所生成的 W(y) 数列是否合格。以 W(6) = 3,2,4,5,6 为例：

第五行 6

第四行：1 3

第三行：2 5 1

第二行：5 6 3 4
第一行：3 2 4 6 5
其中第一行为 W(y) 数列，往上依次为 z 个数相加并 mod k+1 的值不难发现该阵列具有如下性质：

- 以任意一行的某个值为等边三角形的顶点，则其等于位于底边的一行数的和 mod k+1 的余数，例如第三行 $5 = 2 + 4 + 6 = 12 \pmod 7$
- 任意在该阵列中取一个菱形，则相对的顶点和 mod k+1 相等。

依靠性质二，可以快速地根据第一二行算出整个阵列，即 a[i][j] = a[i-1][j] + a[i-1][j+1] − a[i-2][j+1]

由于 W(y) 任意项连续和的要求，则该阵列要求第 i 行不能出现 0 或 i，下面给出一个错误的例子：
第五行 6
第四行：0 4
第三行：2 5 1
第二行：5 0 2 4
第一行：2 3 4 5 6

如上所示，该阵列第二行及第四行均出现错误，所以 $W(6) = \{2, 3, 4, 5, 6\}$ 是错误的，需要回溯。

接下来详细介绍回溯法：

由于 W(y) 和对称的性质，故只需要讨论到 W(y) 的 y/2 或 (y+1)/2 项即可，且讨论的每一项，需与前面讨论的所有项均不同，且不等于它们的对称项。若讨论到第 y/2 或 (y+1)/2 项，则展开判断。若判断成功，则说明该数列就是需要的 W(y) 数列，若不成功，则需要回溯。注意回溯时，上一项可以直接从回溯前的值加上 1 开始讨论，不必从 2 开始。此外若某一项达到了 y+1, 则也需要往前回溯。

若直到第一项超过了 y/2 或 (y+1)/2(因为 W(y) 和对称性的缘故，我们需要的 W(y) 第一项的值总是不超过这两个值的，若超过了，则其鸳鸯数列首项满足不超过这两个值) 就说明不存在这样的 W(y) 数列，就无法构造出我们想要的 $S(n^2, n, 1)$-设计，可以直接结束算法。

代码实现如下：

```
1  //生成W(y)数列
2  int make_array(int n , int W[]){
3      int equal[n] , symetric[n] , mode_value[n][n];
4      for(int i = 1 ; i < n ; i++){
5          equal[i] = 0;
```

```
6           symetric[i] = 0;
7           for(int j = 0 ; j < n ; j++) mode_value[i][j] = 0;
8       }
9       W[1] = 2;
10      equal[1] = 2;
11      symetric[1] = n;
12      mode_value[1][1] = 2;
13      if(n % 2 == 0) W[n/2] = n/2 + 1;
14      int j = 2 , i = 1;
15      while(1){
16          for(int k = 1 ; k < i ; k++){
17              if(j == equal[k] || j == symetric[k] || (j +
                    mode_value[k][i-1]) % (n+1) == 0 || (j +
                    mode_value[k][i-1]) % (n+1) == i - k + 1){
18                  j++;
19                  break;
20              }
21              else if(j > n){
22                  i--;
23                  j = W[i] + 1;
24                  W[i] = 0;
25                  equal[i] = 0;
26                  symetric[i] = 0;
27                  for(int s = 1 ; s <= i ; s++) mode_value[s
                        ][i] = 0;
28                  break;
29              }
30              else{
31                  W[i] = j;
32                  j = 2;
33                  equal[i] = W[i];
34                  symetric[i] = n + 2 - W[i];
35                  for(int s = 1 ; s <= i ; s++) mode_value[s
                        ][i] = (mode_value[s][i-1] + W[i] ) % (n
                        + 1);
36                  i++;
37                  break;
```

```
38              }
39          }
40          if(i > (n - 1) / 2){
41              if(judge(W , n) == 1) return 1;      //判断生成的
                    W(y)是否符合要求
42              else{                                //往前回溯
43                  i--;
44                  j = W[i] + 1;
45                  W[i] = 0;
46                  equal[i] = 0;
47                  symetric[i] = 0;
48                  for(int s = 1 ; s <= i ; s++) mode_value[s
                        ][i] = 0;
49              }
50          }
51          if(i == 1){
52              if(j > (n + 1) / 2) return 0;
53              else{
54                  W[i] = j;
55                  j = 2;
56                  equal[i] = W[i];
57                  symetric[i] = n + 2 - W[i];
58                  mode_value[1][1] = W[i];
59                  i++;
60              }
61          }
62      }
63
64  }
```

(2) 本源数列的求解

由 3，本源数列根据 W(y) 数列进行求解，分为 y 是偶数和 y 是奇数两种情况。两种情况可以定义一个 trial[n] 数列来做中间数列。trial 数列定义如下：

若 y 为偶数：

1). 首项为 1，第一项为 0，第二项未定；

2). 后 y-1 项为 W(y) 数列

若 y 为奇数：

1). 首项为 1，第 (n+1)/2 项为 0，第 1+(n+1)/2 项未定

2). 后 y-1 项为 W(y) 数列，若超出数组上限，则从第一项开始

然后未定项从 3 到 n 开始迭代，遍历整个 trial 数组进行加法运算，同时设置 flag 标签，若 flag 经历加法运算后仍为 1 则说明该未定项的值符合要求，跳出迭代。之后从 0 开始，依次往后作加法运算即可得到本源数列的各项值。代码实现如下：

```
1  //生成本源数列
2  int make_original_array(int W[] , int original[] , int n){
3      int trial[n] , add_value = 0 , flag = 1 ;
4      if(n % 2 == 0){
5          trial[0] = 1 ;
6          trial[1] = 0;
7          for(int i = 3 ; i < n ; i++){
8              trial[i] = W[i - 2];
9          }
10         for(int i = 3 ; i < n ; i++){
11             flag = 1;
12             add_value = i;
13             for(int j = 3 ; j < n ; j++){
14                 add_value += trial[j];
15                 add_value = add_value % (n-1);
16                 if(add_value == 1 || add_value == j){ //满
                       足要求2和3且所得数不重复
17                     flag = 0 ;
18                     break;
19                 }
20
21             }
22             if(flag == 1){
23                 trial[2] = i;
24                 //for(int k = 0 ; k < n ; k++) cout <<
                       trial[k] << " ";
25                 //cout << endl;
26                 original[0] = 0;
27                 original[1] = i;
28                 for(int k = 3 ; k < n ; k++){
29                     add_value = 0;
30                     for(int s = 2 ; s <= k ; s++){
31                         add_value += trial[s];
```

```
32                      }
33                      original[k-1] = add_value % (n-1);
34                      if(original[k-1] == 0) original[k-1] =
                            n - 1;
35                  }
36
37                  return 1;
38              }
39          }
40      }
41      else{
42          int pos = 0;
43          trial[0] = 1;
44          trial[(n+1)/2] = 0;
45          for(int i = 1 ; i < n - 2 ; i++){
46              pos = i + (n+1)/2 + 1;
47              if(pos > n - 1) pos = pos % (n-1);
48              trial[pos] =W[i];
49          }
50          for(int i = 3 ; i < n ; i++){
51              pos = 0;
52              flag = 1;
53              add_value = i;
54              for(int j = (n+1)/2 + 2 ; j < n + (n+1)/2 ; j
                    ++){
55                  pos = j;
56                  if(pos > n - 1) pos = pos % (n-1);
57                  add_value += trial[pos];
58                  add_value = add_value % (n-1);
59                  if(add_value == 1 || add_value == pos){ //
                        满足要求2和3且所得数不重复
60                      flag = 0 ;
61                      break;
62                  }
63
64              }
65              if(flag == 1){
```

```
                    trial[(n+1)/2 + 1] = i;
                    //for(int k = 0 ; k < n ; k++) cout <<
                        trial[k] << " ";
                    //cout << endl;
                    original[0] = 0;
                    original[1] = i;
                    for(int k = 3 ; k < n ; k++){
                        pos = 0;
                        add_value = 0;
                        for(int s = 2 ; s <= k ; s++){
                            pos = (n+1)/2 + s -1;
                            if(pos > n - 1) pos = pos % (n-1);
                            add_value += trial[pos];
                        }
                        original[k-1] = add_value % (n-1);
                        if(original[k-1] == 0) original[k-1] =
                            n - 1;
                    }

                    return 1;
                }
            }
        }
        return 0;
}
```

该代码兼具判断是否能生成本源数列的功能，若可以则返回 1，否则返回 0.

(3) 等价数列的求解

等价数列由本源数列生成由于本源数列受到奇偶性的影响，等价数列同样要对 n 为奇数还是偶数分开进行讨论，等价数列用 equivalence[n][n] 表示。

第 0 列无关奇偶性，都是从 0 到 n-1 等价数列最重要的是对第 1 列进行赋值，奇偶的区别就主要体现在这里, 而与余下列则是通过对第 1 列的变换得到。代码编写如下:

```
int equi_pos;
for(int i = 0 ; i < n ; i++){
    for(int j = 1 ; j < n ; j++){
        if(i == 0) equivalence[i][j] = j;   //第 0列
        else if(i == 1){                    //第 1列
```

```
6           if(n % 2 == 0) equivalence[i][j] =
7                 original[j-1]; //奇数
8           else{      //偶数
9               int pos;
10              pos = j - (n+1)/2;
11              if(pos < 0 ) pos += n-1;
12              equivalence[i][j] = original[pos];
13          }
14        }
15        else{                              //其余列
16              equi_pos = j-1;
17              if(equi_pos == 0 ) equi_pos = n-1;
18              if(equivalence[i-1][equi_pos] == 0)
                    equivalence[i][j] = 0 ;
19              else if(equivalence[i-1][equi_pos]==n-1)
                    equivalence[i][j] = 1 ;
20              else equivalence[i][j] = equivalence[i-1][
                    equi_pos] + 1;
21
22        }
23    }
24}
```

(4)*BIBD_group* 数组的赋值由 2 中分析，得到等价数列之后，第一天的分组情况可照等价数列进行赋值。第 2 到第 n-1 天由第一天分组变换而来，第 0 天与第 n 天则与第一天无关。

```
1 int BIBD_pos;
2 for(int i = 0 ; i < n+1 ; i++){
3     for(int j = 0 ; j < n ; j++){
4         for(int k = 0 ; k < n ; k++){
5             if(i == 0)BIBD_group[i][j][k] = k * n + j;
                      //第0天
6             else if(i == n) BIBD_group[i][j][k] = j * n + k
                  ;      //第n天
7             else{
8                 if(k != 0) {
9                     BIBD_pos = i + k - 1;
```

```
                    if(BIBD_pos >= n) BIBD_pos = BIBD_pos -
                        n + 1;
                    BIBD_group[i][j][k] = k * n +
                        equivalence[j][BIBD_pos];
                }
                else BIBD_group[i][j][k] = j; //第1~n-1天的
                    每组第1个人固定(k=0)
            }
        }
    }
}
```

3.3 输出

得到 *BIBD_group* 之后，则根据在输入中得到的 type 值决定输出模式。type=0，则输出对应组号 *group_ID* 的全员女生身份编号；type=1，则输出两个女生身份编号对应的组号。两者均可通过遍历 *BIBD_group* 数组来得到。前者较为简单，后者只需编写一个 belong 函数来判断一个女生是否在一组中。代码编写如下：

```
//判断数列是否包含某个数
int belong(int a , int b[] , int n){
    if(n < 1) return 0;
    for(int i = 0 ; i < n ; i++){
        if(a == b[i]) return 1;
    }
    return 0;
}

//输出
if(type == 0){
    int number = 0;
    for(int i = 0 ; i < n+1 ; i ++){
        for(int j = 0 ; j < n ; j++){
            if(number == group_ID){
                for(int k = 0 ; k < n ; k++){
                    cout << BIBD_group[i][j][k] << " ";  //
                        输出全组女生编号
                }
```

```
                cout << endl;
            }
            number++;
        }
    }
}
else{
    int number = 0;
    for(int i = 0 ; i < n+1 ; i ++){
        for(int j = 0 ; j < n ; j++){
            if(belong(ID1 , BIBD_group[i][j] , n) == 1 &&
               belong(ID2 , BIBD_group[i][j] , n) == 1){
                cout << "G" << number << endl;    //输出对应
                   组号
            }
            number++;
        }
    }
}
```

3.4 算法复杂度评估

因为 n 为质数与 n 为合数的时候所需的操作步骤完全不一样，两种情况下的算法复杂读也截然不同。

n 为质数时：最复杂的步骤：$BIBD\_group$ 数组的赋值，复杂度在 n^3 级别

n 为合数时：最复杂的步骤：W(y) 数列的生成，复杂度在 (n/2)!! 级别，也是整个算法里最复杂的一步。

3.5 内存占用

由于算法中涉及到的数组固定，而最大的数组为 $BIBD\_group[n+1][n][n]$，故而所占用的内存大概在 n^3 级别。

4 实验数据及软硬件介绍

本算法相关代码的构建仅使用了较为简单的 c++ 语言，同时因为该选题所涉及到的算法输入输出较为简洁，需要一个验证实验来判断该算法的最终结果是否正确。

介于该选题最重要的部分是 $S(n^2, n, 1)$-设计，即 n^2 位女生在 n+1 天内的分组情况。故而可针对算法里的 $BIBD\_group$ 数组进行验证实验。由选题要求，$BIBD\_group$ 数组 (为说明方便，以区组位进行划分) 具有如下性质：

- 每一区里为 0 到 $n^2 - 1$ 的互不相同的数字;
- 每一组内出现的两个数字不会在其它组出现

由上述算法给出的分组方案 (无论是质数还是合数)，可知同一区内不会有重复的数字，所以性质 1 已然满足。

下面主要验证第二个性质：对于任意两个数，遍历 $BIBD\_group$ 数组进行验证，判断其是否同时出现在一个组里，若均符合性质 2，则说明符合要求。

具体代码如下：

```
cout << "是否启动验证程序:(yes or no?)" << endl;
string verify;
cin >> verify ;
int flag = 0 ; //设置标签
if(verify.compare("yes") == 0){
    for(int i = 0 ; i < n+1 ; i++){
        for(int j = 0 ; j < n ; j++){
            for(int k = 0 ; k < n ; k++){
                cout << BIBD_group[i][j][k] << " ";
            }
            cout << " | ";
            }
        cout << endl;
    }
    for(int i = 0 ; i < n*n-1 ; i++){
        for(int j = i+1 ; j < n ; j++){
            flag = 0;
            for(int k = 0 ; k < n+1 ; k++){
                for(int s = 0 ; s < n ; s++){
                    if(belong(i , BIBD_group[k][s] , n) ==
                       1 && belong(j , BIBD_group[k][s] , n)
                        == 1) flag++; //若两数出现在同一组一
                       次, flag就加1
                }
            }
```

```
23                 if(flag > 1){
24                     cout << "i 和 j 同时出现在至少两组内 , 结
                           果错误" << endl;
25                     return 0;
26                 }
27
28             }
29         }
30         cout << "经验证 , BIBD_group数组符合所要求的性质 , 所
               以结果正确" << endl;
31 }
```

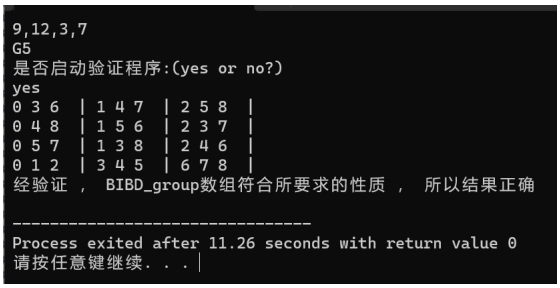
验证实验同主程序一样，均由 c++ 代码编写，在 visual studio , vscode , dev c++ 等平台上均可运行。

5 实验结果及主要结论

5.1 实验结果

根据代码所显示的结果，以下选择部分进行展示：

- S(9,3,1):(n = 3 ,k 为质数)



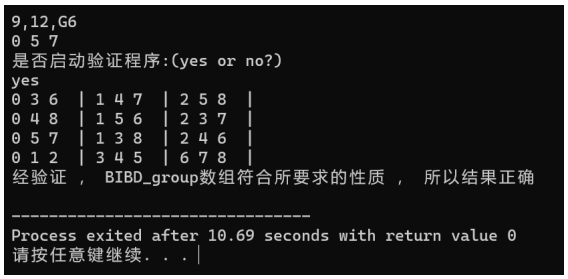


图 1: k 为质数

- S(16,4,1)(n = 4 , 为偶合数)

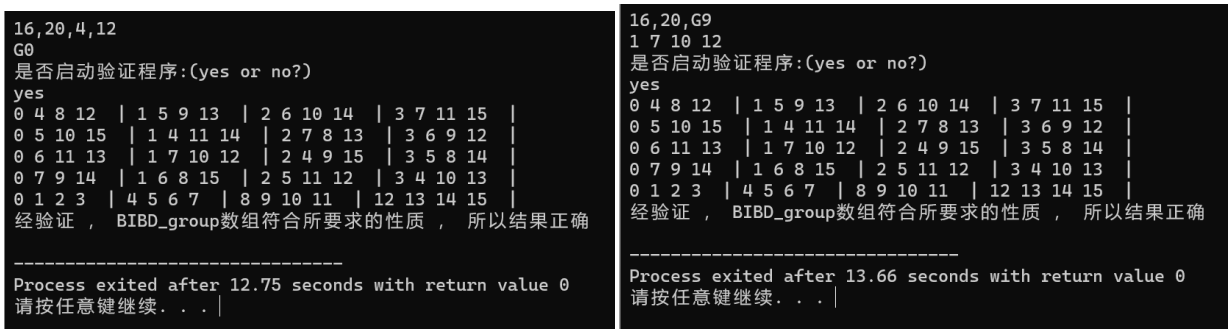


图 2: k 为偶合数

- S(81,9,1)(n=9 , 为奇合数)

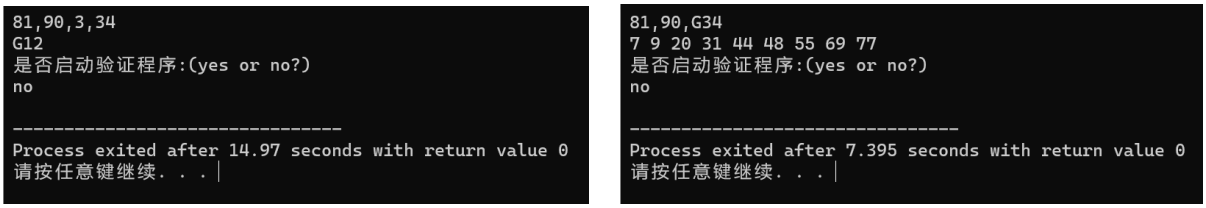


图 3: k 为奇合数

- S(36,6,1)

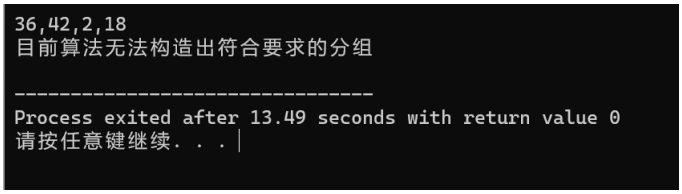


图 4: 分组不存在的情形

5.2 结论

上图展示了 n 为质数、偶合数、奇合数时候的输出及相应的 $S(n^2, n, 1)$-设计。本文所给出的算法针对质数、偶合数、奇合数具均有输出，但并不是所有数运用该算法都可以进行输出，如 n=6 时。说明不是所有的 n 都存在 $S(n^2, n, 1)$-设计。

在背景介绍中已知,目前还并不存在针对斯坦纳系 S(v,k,1) 或者 BIBD 问题 $B(b, v, r, k, \lambda)$ 的通用分组方法。而对更加特殊化的该选题所要求的 $S(n^2, n, 1)$-设计，本算法给出了一个较为全面的设计方案，虽然有对于有些数并不能给出一个分组，但仍旧为该问题的解决提供了一种思路。该算法的运行时间随着 n 的增大变化规律不明显，这主要是因为当

女生个数n/个	4	9	16	25	49	64	81	121	169	256	289
代码返回时间/s	16.26	8.073	8.208	10.66	9.214	11.57	11.4	24.89	17.93	8.97	23.98

图 5: 算法的统计描述

n 是质数时合当 n 是合数时所采取的操作不同导致的。并且当 n 是合数时，W(y) 数列的计算速度也并不一致，这也是不规律性的重要原因。

就准确率而言，由于建立在较为严格的数学模型上，该算法的输出较为准确，基本上能够输出就意味着输出的准确性。

就效率而言，通俗来说在女生人数少于 625 的时候效率较高。

但该算法也存在着一个很大的问题，那就是在计算 W(y) 时的计算复杂度会会随着 n 的增大而急剧扩大。分界点在 $n^2 = 625$ 处，在此之前，算法能够较为快速地计算出 $S(n^2, n, 1)$ 的分组结果，但在 625 之后，计算就显得较为缓慢。

6 评价及展望

综合而言，本文所给出的算法对于质数构造出了一个简单易行的分组方案，而对于合数，当其满足 W(y) 的存在性时，也能够给出一个可行的分组方案。

虽然不能对 W(y) 的数列的存在性给出一个确定的数学判断方法，但结合 11 以内的能够使 W(y) 存在的 y 值，即 3，5，6，7，9，11，因为：

3+2 = 5 是质数，

5+2 = 7 是质数，

6+2 = 8 = 2^3,2 是质数，

7+2 = 9 = 3^2 ,3 是质数，

9+2 = 11，11 是质数，

可以猜想：当 n = y+2 为 p^q(p 为质数时), 相应的 W(y) 数列就存在，此猜想尚待证明。

不难看出 n 为合数时，W(y) 的存在性是解决问题的核心，事实上，对 W(y) 的计算是整个算法中最为复杂的部分，所以要优化此算法，需从 W(y) 数列的生成方式处着手。结合 11 以内的 W(y) 数列：

$W(2) = \{2\}$

$W(3) = \{2, 3\}$

$W(5) = \{3, 2, 5, 4\}$

$W(6) = \{3, 2, 4, 6, 5\}$

$W(7) = \{2, 4, 3, 6, 5, 7\}$

$W(9) = \{2, 7, 8, 6, 5, 3, 4, 9\}$

$W(11) = \{6, 10, 5, 4, 2, 11, 9, 8, 3, 7\}$

对比 2 到 y 的顺序数列，可以发现其改变只有两种方式。

(1). 对称项交换，如 a[i] 与 a[y-i] 进行交换
(2). 相邻项交换，如 a[i] 与 a[i+1] 进行交换
(注意在进行交换的时候，其对称项是绑定的，即 a[i] 的对称项一定是 n-a[i])

以 $W(6) = \{3\ 2\ 4\ 6\ 5\}$ 为例，相较于顺序数列 $\{2\ 3\ 4\ 5\ 6\}$，改变方式如下：
2 与 3 交换，对应的，5 与 6 也要进行交换（改变方式 2），则 W(y) 数列的生成完毕。

以 $W(9) = \{2, 7, 8, 6, 5, 3, 4, 9\}$ 为例，相较于顺序数列 $\{2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\}$，改变方式如下：

3 与 8 交换，4 与 7 交换，5 与 6 交换（改变方式 1）

3 与 4 交换，7 与 8 交换（改变方式 2）

从以上改变方式可以看出，从顺序数列到 W(y) 数列之间的改变并不复杂，相反，仅需很少的步骤就可以达到，所以可以猜想，是否存在一个固定的判断方法来判断该如何移动顺序数列中的某些项，使其越来越接近 W(y) 数列，称之为调整算法或者是趋近算法。这算的上是未来可能的一种解法之一。

本文所给出的算法是通过遍历来求解 W(y) 数列的，虽然较为笨拙，但相较于上面未知的调整算法，优点是可以较为容易实现的。

BIBD 问题是一类重要的组合学问题,而身为 BIBD 问题中的一类特殊问题,$S(n^2, n, 1)$-设计也有着十分广大的应用前景，经过人们有意的设计和规划，完全可以应用在通讯领域或是试验设计的方方面面，因此，本文所给出的关于 $S(n^2, n, 1)$-设计的构造解法，有着积极意义。

参考文献

[1] Fisher,RonaldA.Statistical Tables for Biological, Argicultural and Medical Research /-15th ed[M].Oliver and Boyd,1957.

[2] 刘建军. 科克曼和他的” 女生问题”[J]. 科学, 2004, 56(5):4.DOI:10.3969/j.issn.0368-6396.2004.05.013.

[3] 特殊斯坦纳系设计的一个构造方法》，知乎，2019，https://zhuanlan.zhihu.com/p/89770311

7 附录

工程文件及 tex 源码见网页:https://github.com/Charle170933/SpecialBIBDProblem.git