

## **Counting Pedigrees up to Isomorphism**

**1. Introduction**

**2. Definition**

**3. Sex-labelled Discrete Generation Model**

**4. Sex Label a given unlabelled pedigree**

**5. Unsex-labelled Discrete Generation Model**

**6. Summary**

**Appendix**

## 1. Introduction

When investigating problems like reconstruction of pedigrees or the probability of data given a particular pedigree, interesting questions related to pedigrees arise. Simulations based on a model of human population history and geography find that an individual that is the genealogical ancestor of all living humans existed just a few thousand years ago, as suggested in a recently published article by Prof. Hein.<sup>1</sup> One related fascinating question, which we are going to investigate in this project, is that how many different pedigrees there actually are up to isomorphism if we trace back a certain number of generations.

To do this, we will introduce some definitions.

## 2. Definition

**Pedigree:**

A pedigree is a directed graph showing the relationship between individuals according to their parentage. Both of (2.1) and (2.2) below are examples of pedigrees.

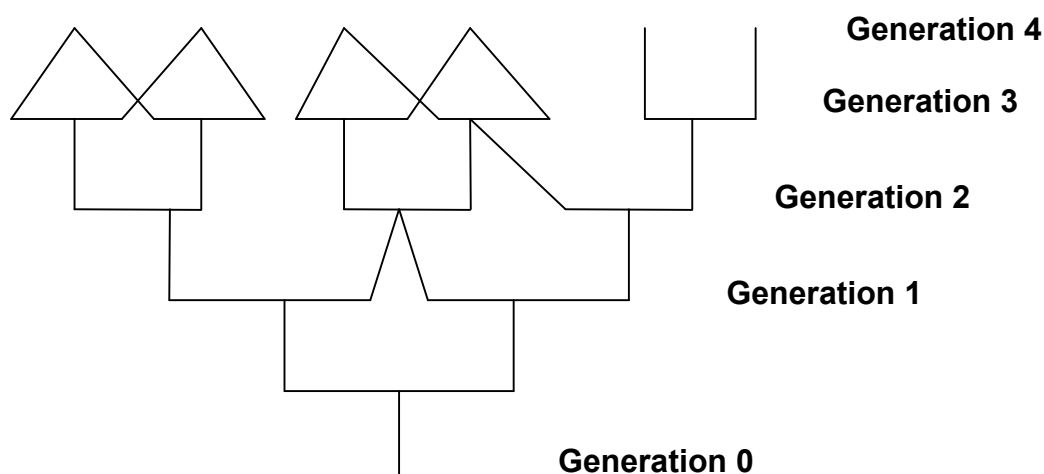
**Generation 0:**

We call the most recent generation ‘generation 0’, in this project, there is only 1 individual in generation 0.

**Discrete:**

As we said above, the most recent generation is ‘generation 0’, then let the parents of the individual in generation  $g$  be in generation  $g+1$ , and allow the case that more than one generation number are assigned to one single individual. If no individual in the pedigree belongs to more than one generation, we say the pedigree is discrete. Pedigree (2.1) as below is an example of a discrete pedigree, while (2.2) being non-discrete.

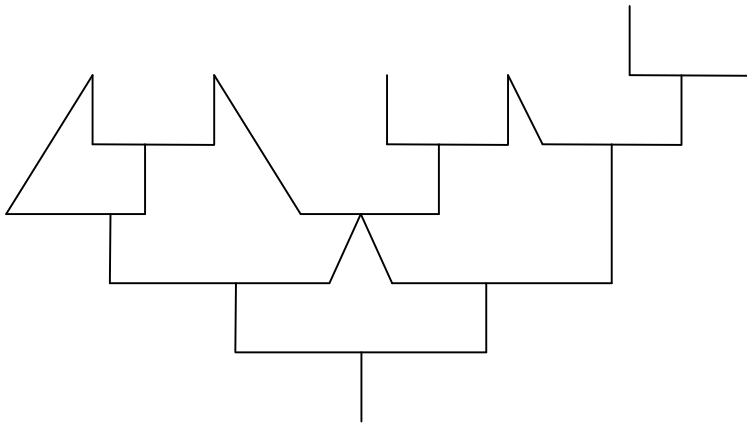
(2.1)



---

<sup>1</sup> Hein, Jotun. ‘Pedigrees for all humanity’ Nature Vol 431 September 2004, Page 518-519

(2.2)



### 3. Sex-labelled Discrete Generation Model

To start off, we are going to concentrate on a simpler version of the original question, in which we are only interested in the pedigrees which are sex-labelled, and are of discrete time and generation.

In this particular case, since every individual in a pedigree is labelled by the sex, there is a unique link between generation 0 (our starting individual) and any other individuals in the pedigree. This is a very useful characteristic of the sex-labelled pedigrees, in the sense that it guarantees every individual is uniquely determined, which saves us a lot of trouble when counting. Also, the discrete time and generation constraints make it possible for us to adopt a recursion moving back generation by generation.

The idea of the counting is as follows:

Sampling one individual, the number of grandparents is a natural number between 2 and 4 inclusive. Similarly, the number of different ancestors  $g$  generations back in time (the ancestors in generation  $g$  only) is a natural number between 2 to  $2^g$  inclusive. The number of different pedigrees back to  $g$  generations can be easily calculated if we know the number of different pedigrees of 1 generation with  $c$  number of children and  $p$  number of parents ( $c$  is any natural number from 1 to  $2^{g-1}$  inclusive, and  $p$  is any natural number from 2 to  $2^g$  inclusive). This number, however, can be calculated easily.

Now, let's write the idea above out in symbolic terms. Let  $N_g(p)$  denote the number of different pedigrees going back  $g$  generations with 1 individual in generation 0 and  $p$  individuals in generation  $g$ , and  $A(c, p)$  denote the number of different pedigrees of 1 generation with  $c$  children and  $p$  parents. Adopting these notations, our recursions can be shown as the following expressions:

$$A(c, p) = \sum_{\substack{1 \leq i, j \leq c \\ i+j=p}} S_{c,i} \times S_{c,j} \quad (3.1)$$

$$N_{g+1}(p) = \sum_c N_g(c) \times A(c, p) \quad (3.2)$$

In Equation (3.1),  $Sc,j$ , the Stirling number of second kind, is the number of different ways to partition  $c$  elements into  $j$  sets. Here, we assume among those  $p$  parents,  $i$  of them are female and  $j$  of them male. Since the partitions of females and males are independent,  $A(c, p)$ , the number of different pedigrees of 1 generation with  $c$  children and  $p$  parents is just the product of  $Sc,i$  and  $Sc, j$  summing over all possible  $i,j$ -s.

Notice that among these  $p$  parents, at least one of them is female and at most  $p-1$  of them are female, we can rewrite Equation (1.1):

$$A(c, p) = \sum_{i=1}^{\text{Min}[c, p-1]} Sc, i \times Sc, p-i \quad (3.3)$$

Following this recursion algorithm, we can write a programme called ‘countp’ in Mathematica.

- Step 1. Delete any existing function with the name ‘countp’;
- Step 2. Declare function ‘countp’ to have input ‘generation’, a integer number variable;
- Step 3. Declare matrix variables ‘matrixn’, ‘matrixa’, number variable ‘g’, ‘c’, ‘p’;  
‘matrixn’—the  $g,p^{\text{th}}$  entry in matrixn is  $Ng(p)$  above.  
‘matrixa’—the  $c,p^{\text{th}}$  entry in matrixa is  $A(c,p)$  above.
- Step 4. Define matrixn to have ‘generation’ number of rows and  $2^{\text{‘generation’}}$  number of columns with  $1,2^{\text{th}}$  entry being 1 and all the other entries being 0;
- Step 5. Define matrixa to have  $2^{(\text{‘generation’} - 1)}$  number of rows and  $2^{\text{‘generation’}}$  number of columns with all entries being 0;
- Step 6. Assign ‘g’ value 2;
- Step 7. If  $g \leq \text{‘generation’}$ , let  $c = 1$ ;  
Otherwise, go to Step 12;
- Step 8. If  $c \leq 2^{(g-1)}$ , let  $p = 2$ ;  
Otherwise, go to Step 11;
- Step 9. If  $p \leq 2c$ , let  $\text{matrixa}[[c,p]] = \text{Summation of } Sc,j * Sc,p-j \text{ over integer values for } j$   
from 1 to the minimum of  $c$  and  $p-1$ ,  
let  $\text{matrixn}[[g,p]] = \text{matrixn}[[g,p]] + \text{matrixn}[[g-1,c]] * \text{matrixa}[[c,p]]$ ,  
 $p=p+1$ ;  
repeat Step 9;  
Otherwise, go to Step 10;
- Step 10. Let  $c=c+1$ , then go to Step 8;
- Step 11. Let  $g=g+1$ , then go to Step 7;
- Step 12. Sum the last row of matrixn and return the value.  
(Matrix[[m,n]] denote the  $m,n^{\text{th}}$  entry in matrix called ‘Matrix’)

Using the programme we wrote in Mathematica, we have got the following result:

**Number of Parents**

<b>Generation</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>Total</b>
<b>1</b>	1	0	0	0	0	0	0	1
<b>2</b>	1	2	1	0	0	0	0	4
<b>3</b>	4	28	84	98	52	12	1	279

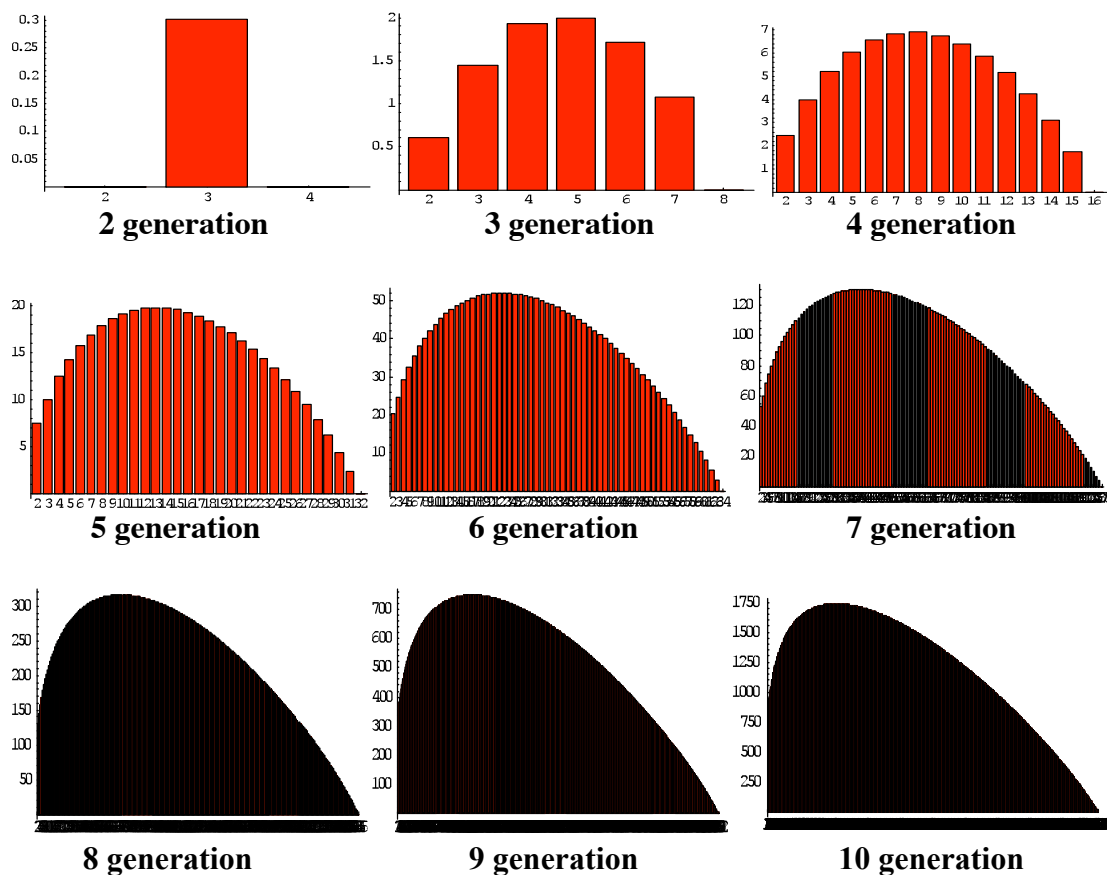
### Number of different sex labelled pedigrees (#)<sup>2</sup>

Generation	1	2	3	4	5	6
#	1	4	279	$2.8766 \cdot 10^7$	$2.81597 \cdot 10^{20}$	$7.40974 \cdot 10^{52}$

Generation	7	8	9	10
#	$2.76739 \cdot 10^{131}$	$2.88199 \cdot 10^{317}$	$3.52352 \cdot 10^{749}$	$3.89498 \cdot 10^{1737}$

From the numbers about, we can see that the number of different sex labelled pedigrees grows very fast with generations. If we trace for 3 generations, then there are 279 sex-labelled pedigrees up to isomorphism, and if we trace back one generation more, the number of pedigrees up to isomorphism rises up to  $2.8766 \cdot 10^7$ ! The recursion programme in Mathematica we used above can calculate up to 10 generations. Beyond that, calculations get very slow.

Tracing back each number of generations, I plotted the log of number of different pedigrees which have  $p$  distinct parents in the earliest generation against  $p$ , getting a log distribution of the number of different pedigrees having a certain number of parents in the last generation for each generation.



According the bar chart above, we can tell that all of them are of an 'n' shape, and the position of maximum value shifts to the left as the number of generations increases. The following table gives the position (the number of different parents in the earliest generation)

<sup>2</sup> See Appendix 1

where there are most different pedigrees for tracing back different generations.

Generation	1	2	3	4	5
Position for most pedigrees	2	3	5	8	13
Most number of Parents	2	4	8	16	32

Generation	6	7	8	9	10
Position for most pedigrees	22	39	68	120	214
Most number of Parents	64	128	256	512	1024

#### 4. Sex Label a given unlabelled pedigree

Now since we have an idea how many sex labelled pedigrees up to isomorphism tracing back to any generation from no more than 10. A natural question to ask would be how many ways there are to sex label a given pedigree if it is not sex labelled.

Consider the number of ways to sex-label a pedigree with  $c$  children and  $p$  parents with 1 generation back only, we notice that there are only two ways to sex-label such a pedigree if it is not possible to partition all the parents and children into two or more none empty set such that any two children in different sets share no parent and any two parents in two different sets have no common child. We call the children and parents group with the above characteristics 'closed'.

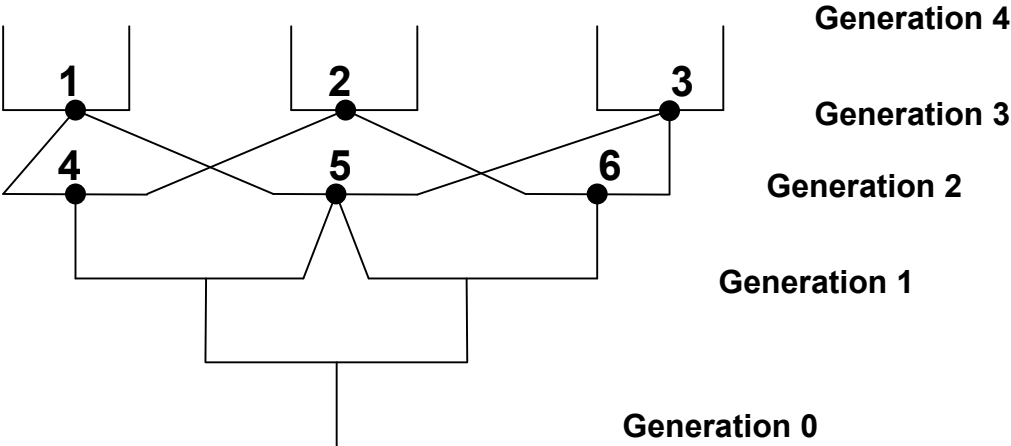
For any given pedigree with only 2 generations (tracing back only 1 generation) of finite number of parents and children, it is possible to check the number of closed groups. Since sex-labelling all these different closed groups are independent, and there are precisely 2 ways to sex-label each of the closed group, the number of ways to sex-label a pedigree with 2 generation is 2 to the power of the number of different proper closed groups. To say a closed group is proper, we mean it is able to sex-label this closed group. Naturally, we call a pedigree that is not able to be sex-labelled improper. An example of an improper closed group is 3 parents in which any two of them share a child.

In order to distinguish the improper cases, we can adopt the following algorithm. For any pedigree with two generations,  $c$  children and  $p$  parents, we label the parents from 1 to  $p$ . Start from parent 1 and assign it value 0, and then assign all the parents who share at least one children with parent 1 the value 1. Suppose parent  $m$  is the one with the smallest ordinal number sharing kid(s) with parent 1 ( $1 < m \leq p$ ), we move on the consider parent  $m$ . Take all parents who share at least one child with parent  $m$ , and in which, we assign parents who have no sex value assigned yet the opposite sex value of parent  $m$  (0 in this case). For those who share child(ren) with parent  $m$  but have got a sex value already, we check whether their assigned sex is indeed the opposite sex from parent  $m$ , if it fails for any of them, this group is improper.

Let's have a look at an example. For pedigree (4.1), consider the generation with 1,2,3 as parents and 4,5,6 as children. Assign parent 1 with value 0, and then note that both 4,5 are children of parent 1. For child 4, 2 is the other parent and for child 5, 3 is the other parent, so

we assign value 1 to both parent 2 and parent 3. Then we move on the parent 2, note that both 4 and 6 are children of parent 2. Since we have done with child 4, we only need to consider child 6. Parent 3 is the other parent of child 6. The value assigned to parent 3 was 1, but the value of parent 2 is 1 as well, contradiction! We cannot sex-label pedigree (4.1).

(4.1)

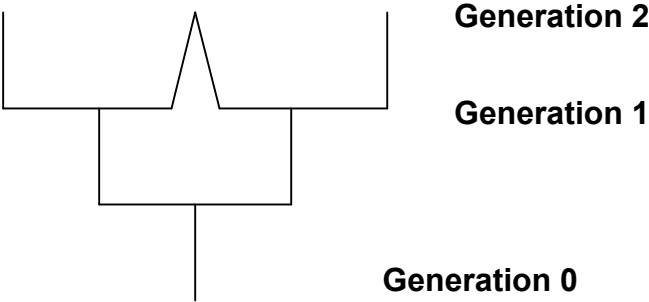


For similar pedigrees as in Section 3, where there is only one individual in generation 0, we can use the method above to figure out the number of ways to sex-label them. But we cannot simply adopt it if there are two or more individuals in generation 0. This is because the individuals in generation 0 are identical and will cause further counting repetitions if we don't adjust the method as above. Also, note that this only works for pedigrees of discrete generation, since for non-discrete pedigrees, there may only be one way to sex-label the parents of some individuals.

**5. Unsex-labelled Discrete Generation Model**

For the case that all individuals are not sex-labelled, but generation discrete, is it possible to make a similar recursion to calculate the number of different pedigrees up to isomorphism? The idea of the original attempt is that each time we trace back a generation, we check on every individual and keep a track on the isomorphic groups all these individuals form using a list of numbers. For instance, if we go back 2 generations and the pedigree looks like the following:

(5.1)



We can write it as:

generation 0: 1,0,0,0,...  
 generation 1: 0,1,0,0,...  
 generation 2: 1,1,0,0,...

In generation 0, there is only one individual, so there is one isomorphic group with only 1 element, and no other groups at all, denote it 1,0,0,0,...

In generation 1, there are 2 individuals in total, and they are isomorphic, so that we write it as 0,1,0,0,.... to denote that there are 1 group of 2 isomorphic elements and no other groups in this generation.

In generation 2, there are 3 individuals in total, two and only two of which are isomorphic. So, we denote it as 1,1,0,0,.... meaning that there is 1 group of 1 element, and 1 group of 2 elements in this generation.

Now we can use a similar formula as before to do the recursion:

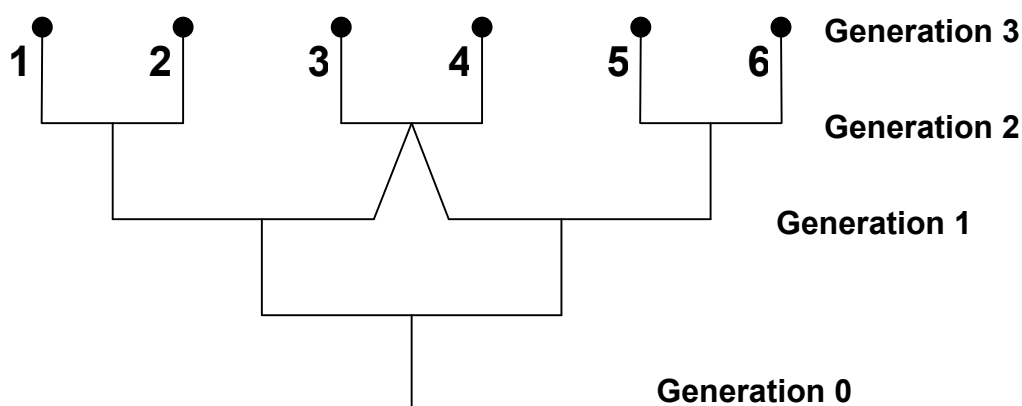
$$Ng+1(p) = \sum_c Ng(c) \times A(c, p) \tag{5.2}$$

Now, c and p are not the number of children and parents, but the lists of isomorphic groups for the child generation and the parent generation, as our example above. Ng(c) is the number of different unsex-labelled pedigrees there are up to isomorphism up to generation g. A(c,p) is the number of different cases that individuals as listed in c can have parents as listed in p.

Everything seems to work out so far, but then, we discovered a vital problem in this model. The method to form the list, our way to keep tracks of isomorphic groups, is not informative enough. It may not be sufficient to identify the isomorphic grouping relation in generation n if we only look at how the pedigree rises from generation n-1 to generation n.

For example, if using the method as above for the six individuals in Generation 3 in pedigree (5.3), we will get a list of 0,1,0,1,...., 3 and 4 in a group and the rest in the other.

(5.3)

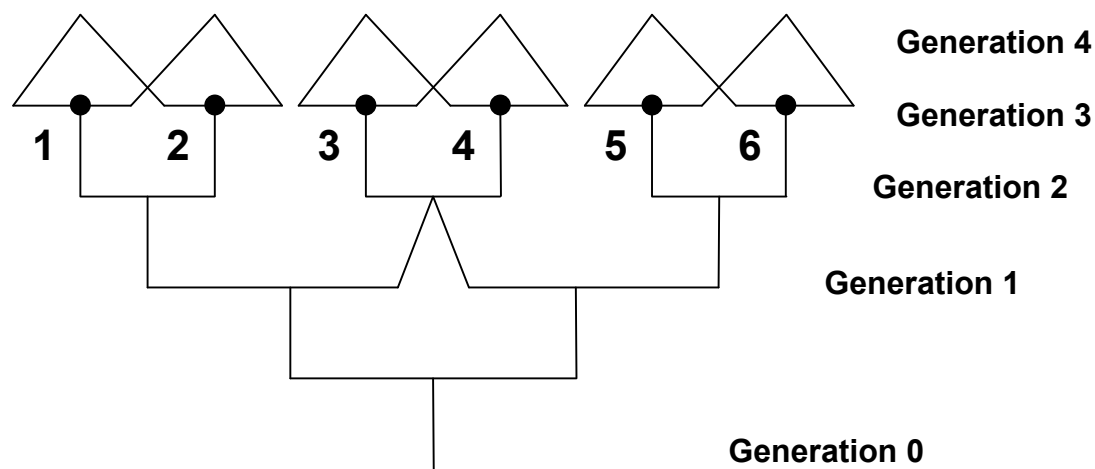


It seems that individual 1,2,5,6 are all in one isomorphic group. However, at this stage it is too early to tell. If the pedigree up to generation 4 is as pedigree (5.4), then individual 1, 2, 5, 6 would be in the same isomorphic group. But if the pedigree up to generation 4 has the form of

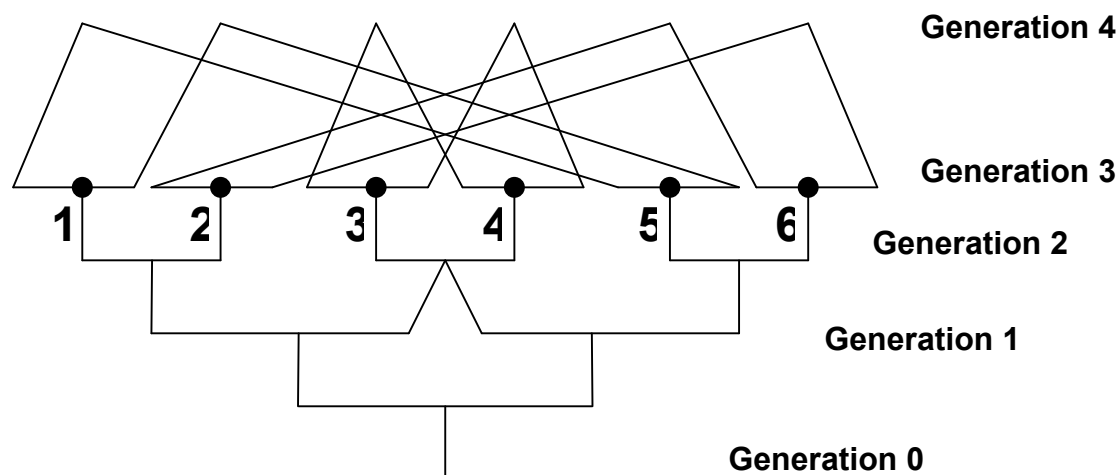
pedigree (5.5), 1, 2, 5, 6 are not all isomorphic any more. (Assume they are isomorphic, then we can use 0,1,0,1 to represent the list of generation 3. Consider individuals 1, 2, 5, 6 only for now, how many ways there are to rise from 0, 0, 0, 1 to 0, 0, 0, 1? One only, which is the pedigree showed as (5.4). But pedigree (5.5) is also a way to rise from the individual 1, 2, 5, 6 in generation 3 to a parent group with 4 isomorphic parents (0,0,0,1). This contradicts our assumption that we can use 0,0,0,1 to present the isomorphic grouping relationship of individuals 1, 2, 5, 6 in generation 3 for (5.3).

It seems that we cannot use a list to describe all the isomorphic relations we need to know about in each generation of the pedigree, suggesting that it might not be possible to find a recursion algorithm to count the number of unsex-labelled pedigrees up to isomorphism.

(5.4)



(5.5)



## 6. Summary

In this project, we have investigated several questions relating to pedigree counting. For discrete pedigrees with only 1 individual in generation 0, we succeeded in finding a recursion to calculate the number of unsex-labelled pedigrees up to isomorphism and in finding an

algorithm for calculating the ways of sex-labelling a given pedigree. The number of different sex-labelled pedigrees rises significantly along the number of generations. It exceeds 1700 digits if we want to go back to 10 generations.

For the unsex-labelled pedigrees, counting is rather hard. Although the number of unsex-labelled pedigrees should be smaller than the one of sex-labelled pedigrees, it is a lot harder to identify if two pedigrees are isomorphic and to find a computable algorithm. We could not adopt similar recursion as in the sex-labelled case and write a computer programme up. Moreover, it might be impossible to write a recursion programme to count for this case.

## Appendix

### 1. Programme in Mathematica

**N[countp[4],6]**

$2.87966 \cdot 10^7$

**N[countp[5],6]**

$2.81597 \cdot 10^{21}$

**N[countp[6],6]**

$7.40974 \cdot 10^{52}$

**N[countp[7],6]**

$2.76739 \cdot 10^{131}$

**N[countp[8],6]**

$2.88199 \cdot 10^{317}$

**N[countp[9],6]**

$3.52352 \cdot 10^{749}$

**N[countp[10],6]**

$3.89498 \cdot 10^{1737}$