Isidore Harris - zdn5uf
APMA 3100
Fall 2024

Project 1: An Exploration of Counting Methods

On my honor as a student, I have neither given nor received unauthorized aid on this assignment/examination.

Isidore Harris

Isidore Harris - zdn5uf
APMA 3100
Fall 2024

Project 1: An Exploration of Counting Methods

Due Friday, Sept 6 at 11:59pm

1. *A student is registering for classes for next semester. The courses she can enroll in are calculus, physics, organic chemistry, history, literature, French, and astronomy. She has room for four classes…*

    a. *Make the beginnings of a carefully ordered list of all possible book orders. (Your list should include a minimum of 20 outcomes and should be ordered systematically so that someone else can see and continue the pattern and not miss any of the outcomes if they were to finish the list.)*

Astronomy(A), Calculus (C), French (F), History (H), Literature (L), Organic Chemistry (O), Physics (P)

System Rules:

- Maintain alphabetic order.
- Cycle book subjects starting from last position in order, ending with first position in order.
- Do not repeat combinations (i.e. in non-alphabetic order).

****START FIRST 20 COMBINATIONS****
ACFH ← *Start all possible combinations beginning with Astronomy*
ACFL
ACFO
ACFP
ACHL
ACHO
ACHP
ACLO
ACLP
ACOP ← *End all possible combinations beginning with Astronomy, Calculus*
AFHL
AFHO
AFHP
AFLO
AFLP
AFOP ← *End all possible combinations with Astronomy, French*
AHLO
AHLP
AHOP ← *End all possible combinations beginning with Astronomy, History*
ALOP ← *End all possible combinations beginning with Astronomy*
****END FIRST 20 COMBINATIONS****
Combination list would continue with:
CFHL ← Start all possible combinations beginning with Calculus

1b. Now make the beginnings of a carefully ordered list of all the possible versions of the student's schedule, assuming she can't register one class for multiple slots. (Your list should include at minimum of 20 outcomes and should be ordered systematically so that someone else can see and continue the pattern and not miss any of the outcomes if they were to finish the list.)

Astronomy(A), Calculus (C), French (F), History (H), Literature (L), Organic Chemistry (O), Physics (P)

System Rules:

- Cycle class subjects from last timeslot to first time slot.
- Follow alphabetic order of subjects when cycling classes.
- Each subject can only occur once per schedule.

****START FIRST 20 COMBINATIONS****
ACFH ← *Start all possible scheduling beginning with Astronomy, Calculus.*
ACFL
ACFO
ACFP ← *End all possible scheduling beginning with Astronomy, Calculus, French.*
ACHF
ACHL
ACHO
ACHP ← *End all possible scheduling beginning with Astronomy, Calculus, History.*
ACLF
ACLH
ACLO
ACLP ← *End all possible scheduling beginning with Astronomy, Calculus, History.*
ACOF
ACOH
ACOL
ACOP ← *End all possible scheduling beginning with Astronomy, Calculus, Organic Chemistry.*
ACPF
ACPH
ACPL
ACPO ← *End all possible scheduling beginning with Astronomy, Calculus.*
****END FIRST 20 COMBINATIONS****
Combination list would continue starting with:
AFCH ← Start all possible scheduling beginning with Astronomy, French.

1c. Here is a program written in Python…
Here is the same program written in a slightly better way …
Does this count the possible book orders, schedules, or neither? Briefly explain.

These programs count the possible schedules. This calculation is a permutation of the 7 subjects.

- without replacement - Each class is only used once per schedule.
- order doesn't matter - We want all possible orders of the classes in each time slot.

How the program works:

- This program iterates through the list ("books") using 'for' loops. Four 'for' loops are nested to accommodate 4 positions of the schedule - 9am, 10am, 11am, 12pm.
- The program prevents repetition by verifying strings are not repeated using 'if' loops and inequalities (!=) to compare positions (a, b, c, d) to each other.
- The program outcomes are incremented in the order in which the list ("books") was made. In this case the increments are alphabetical (history -> literature -> organic_chemistery -> etc.).

Listed below are the first 8 results of the program. We can see from this list that multiple orders of the same four subjects are allowed (indicated in bold).

astronomy calculus french history
astronomy calculus french literature
astronomy calculus french organic_chemistry
astronomy calculus french physics
astronomy calculus history french
astronomy calculus history literature
astronomy calculus history organic_chemistry
astronomy calculus history physics

1d. Write a program using for loops that lists AND counts all the possible book orders. Paste your code below. What are the first 20 outcomes that your program lists? What is the total number of outcomes?

The first 20 outcomes from my program are listed below the code.

The total number of outcomes is **35**.

My code is compatible with Python 3.10.

```
orders = 0
books = ['astronomy', 'calculus', 'french', 'history', 'literature',
'organic_chemistry', 'physics']
book_lists = []

for a in books:
  for b in books:
    if a > b:
      for c in books:
        if c > a and c > b:
          for d in books:
            if d > a and d > b and d > c:
              book_lists.append([a, b, c, d])
              orders += 1
print(orders)
print(*book_lists, sep='\n')
```

Listed below are the first 20 results of the program:

calculus astronomy french history
calculus astronomy french literature
calculus astronomy french organic_chemistry
calculus astronomy french physics
calculus astronomy history literature
calculus astronomy history organic_chemistry
calculus astronomy history physics
calculus astronomy literature organic_chemistry
calculus astronomy literature physics
calculus astronomy organic_chemistry physics
french astronomy history literature
french astronomy history organic_chemistry
french astronomy history physics
french astronomy literature organic_chemistry
french astronomy literature physics
french astronomy organic_chemistry physics
french calculus history literature
french calculus history organic_chemistry
french calculus history physics
french calculus literature organic_chemistry

2a. An 8-digit password is required to have three 0's and five 1's. You will determine how many unique passwords are possible.

Write a program that lists and counts the unique passwords. ***Provide your code, the list of all outcomes, and the count. Briefly communicate which notation you are using for the outcomes.*** *(The list of outcomes you provide should match with the notation you're using in your code.)* ***Also briefly explain how your program works*** *(3-5 sentences) so that a programmer not familiar with this language could easily reproduce your results using a different language.*
Write a program that lists and counts the unique passwords. Provide your code, the list of all outcomes, and the count. Briefly communicate which notation you are using for the outcomes.

My outcomes are listed below my other responses to this Problem 2A in 8 digit notation.
The total unique number of unique passwords is **56.**
My code Python 3.10 is listed below, it is compatible with Python 3.10.

```
count = 0
code_list = []

for i in range(0, 7):
  for j in range(i+1, 8):
    for k in range(j+1, 8):
      code = ['1', '1', '1' ,'1' ,'1' ,'1' ,'1' ,'1']
      code[i] = '0'
      code[j] = '0'
      code[k] = '0'
      code_list.append(code) #Save entire 8 digit code
      count += 1

print("The total unique number of unique passwords is: ", count)
print("Listed below are the possible codes in 8 digit format:")
print(*code_list, sep='\n')
```

PROGRAM EXPLANATION

This program uses nested 'for' loops to iterate through possible outcomes of codes.

There are three 'for' loops, one for each possible position of the zero (0) digits.

- The inner loop ranges add one (1) to the previous loop integer to prevent counting a possible position of zero (0) multiple times.
- The innermost loops initializes a list with eight (8) positions, uses the three loop integers to update the list with zero (0) positions, and also updates a code counter variable.

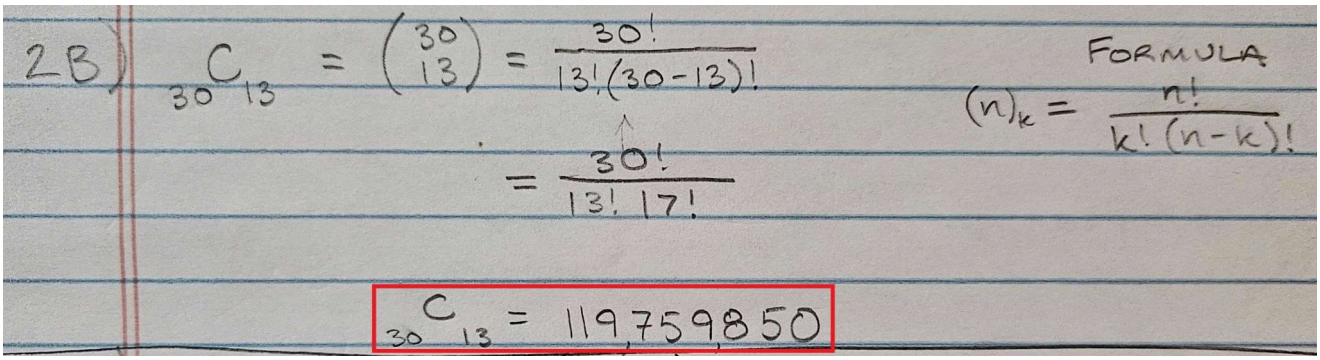
(continue 2a next page)

2a (cont.).

****STARTING NEXT LINE IS CODE OUTPUT****

```
The total unique number of unique passwords is:  56
Listed below are the 56 possible codes in 8 digit format:
['0', '0', '0', '1', '1', '1', '1', '1']
['0', '0', '1', '0', '1', '1', '1', '1']
['0', '0', '1', '1', '0', '1', '1', '1']
['0', '0', '1', '1', '1', '0', '1', '1']
['0', '0', '1', '1', '1', '1', '0', '1']
['0', '0', '1', '1', '1', '1', '1', '0']
['0', '1', '0', '0', '1', '1', '1', '1']
['0', '1', '0', '1', '0', '1', '1', '1']
['0', '1', '0', '1', '1', '0', '1', '1']
['0', '1', '0', '1', '1', '1', '0', '1']
['0', '1', '0', '1', '1', '1', '1', '0']
['0', '1', '1', '0', '0', '1', '1', '1']
['0', '1', '1', '0', '1', '0', '1', '1']
['0', '1', '1', '0', '1', '1', '0', '1']
['0', '1', '1', '0', '1', '1', '1', '0']
['0', '1', '1', '1', '0', '0', '1', '1']
['0', '1', '1', '1', '0', '1', '0', '1']
['0', '1', '1', '1', '0', '1', '1', '0']
['0', '1', '1', '1', '1', '0', '0', '1']
['0', '1', '1', '1', '1', '0', '1', '0']
['0', '1', '1', '1', '1', '1', '0', '0']
['1', '0', '0', '0', '1', '1', '1', '1']
['1', '0', '0', '1', '0', '1', '1', '1']
['1', '0', '0', '1', '1', '0', '1', '1']
['1', '0', '0', '1', '1', '1', '0', '1']
['1', '0', '0', '1', '1', '1', '1', '0']
['1', '0', '1', '0', '0', '1', '1', '1']
['1', '0', '1', '0', '1', '0', '1', '1']
['1', '0', '1', '0', '1', '1', '0', '1']
['1', '0', '1', '0', '1', '1', '1', '0']
['1', '0', '1', '1', '0', '0', '1', '1']
['1', '0', '1', '1', '0', '1', '0', '1']
['1', '0', '1', '1', '0', '1', '1', '0']
['1', '0', '1', '1', '1', '0', '0', '1']
['1', '0', '1', '1', '1', '0', '1', '0']
['1', '0', '1', '1', '1', '1', '0', '0']
['1', '1', '0', '0', '0', '1', '1', '1']
['1', '1', '0', '0', '1', '0', '1', '1']
['1', '1', '0', '0', '1', '1', '0', '1']
['1', '1', '0', '0', '1', '1', '1', '0']
['1', '1', '0', '1', '0', '0', '1', '1']
['1', '1', '0', '1', '0', '1', '0', '1']
['1', '1', '0', '1', '0', '1', '1', '0']
['1', '1', '0', '1', '1', '0', '0', '1']
['1', '1', '0', '1', '1', '0', '1', '0']
['1', '1', '0', '1', '1', '1', '0', '0']
['1', '1', '1', '0', '0', '0', '1', '1']
['1', '1', '1', '0', '0', '1', '0', '1']
['1', '1', '1', '0', '0', '1', '1', '0']
['1', '1', '1', '0', '1', '0', '0', '1']
['1', '1', '1', '0', '1', '0', '1', '0']
['1', '1', '1', '0', '1', '1', '0', '0']
['1', '1', '1', '1', '0', '0', '0', '1']
['1', '1', '1', '1', '0', '0', '1', '0']
['1', '1', '1', '1', '0', '1', '0', '0']
['1', '1', '1', '1', '1', '0', '0', '0']
```

2b. Now suppose a 30-digit password is required to have thirteen 0's and seventeen 1's. You don't need to list all the possible passwords, but you need to determine how many exist. You may solve this however you wish (you don't have to write a program), but briefly compare your method to what you used in part (a) and justify your chosen method.



In part 2A, I determined the amount of 8 digit codes by letting the program count each iteration.

In part 2B, I determined the amount of 13 digits codes by using the equation for k-permutations with n = 30 and k = 13. I used this method because after simplification it requires a one step calculation (with a calculator) to give us an accurate answer.

3a. An 8-digit password is required to have exactly three 0's. The other 5 digits can be any number 1-7, but numbers 1-7 may not be repeated. Again, consider different options for notating possible outcomes. Then make a list of at least 30 of the possible passwords by hand. Make sure you explain the notation you're using for the passwords.

Notation Rules:

- Begin with three 0's on left side and lowest possible integers ascending left to right.
- Counting proceeds with rightmost digit increasing.
- When rightmost digit can not increase due to limit (7) or duplication (repeated integers), increase next digit to the left.
- Non-zero digits will "reset" to lowest possible integer when next digit to the left increases.
- When digits reach limit (7) and would not be able to increase further, third position 0 moves to the right until it reaches the rightmost position.
- The first move of the third position will appear as follows:
  000765431
  000765432
  001023456 <- First move of third position 0

(continue 3a next page)

3a (cont).

****START FIRST 20 COMBINATIONS****
00012345 <- Start all combinations beginning with 000
00012346
00012347
00012354
00012356
00012357
00012364
00012365
00012367
00012374
00012375
00012376 <- End all combinations beginning with 000123
00012453
00012456
00012457
00012463
00012465
00012467
00012473
00012475
00012476 <- End all combinations beginning with 000124
00012534
00012536
00012537
00012543
00012546
00012547
00012563
00012564
00012567 <- End all combinations beginning with 000125
00012634
****END FIRST 20 COMBINATIONS****

3b. Determine how many unique passwords exist. You do not have to write a program to solve but you can if you choose. However you compute your answer, be sure to justify it fully.

$$({}_8C_3)({}_7P_5) = 141,120\ unique\ passwords$$

8 C 3 → This combination is choosing 3 positions of the 8 possible positions (1, 2, 3, 4, 5, 6, 7, 8) to place 0's.

- **w/o replacement** - We are only interested in placing each zero a single time.
- **order does not matter** - We are only interested in using each possible combination of positions once.

7 P 5 →This permutation is choosing which integers 1-7 to place in the 5 remaining possible positions.

- **w/o replacement** - We are only interested in placing each integer a single time
- **order matters** - We are interested in all possible positions or "orders" of the integers 1-7.

c. Suppose you have a password with m digits required to have exactly n 0's. The other (m-n) digits can be any number 1-9 but numbers 1-9 may not be repeated. How many unique passwords exist? (Your answer should be a mathematical expression written in terms of m and n.)

$$(m^n)({}_9P_{m-n}) = \frac{m^n 9!}{(9-m-n)!}\ unique\ passwords$$

4a. A 5-letter password is to be constructed from the following letters: A, B, C, D, and E and the password must contain exactly 3 of the letters. (Obviously at least one of them will need to be used more than once to create a 5-letter password.)

> *You may solve problem 4 however you wish (with or without a program). If you write a program, you should clearly explain what each section of your code accomplishes. If you don't use a program, you should show all your work and justify each computation.*
>
> *a. Suppose A,B, and C are the letters selected and we use A twice, B twice, and C once. How many different passwords exist in this scenario?*

A twice → $({}_5P_2)$

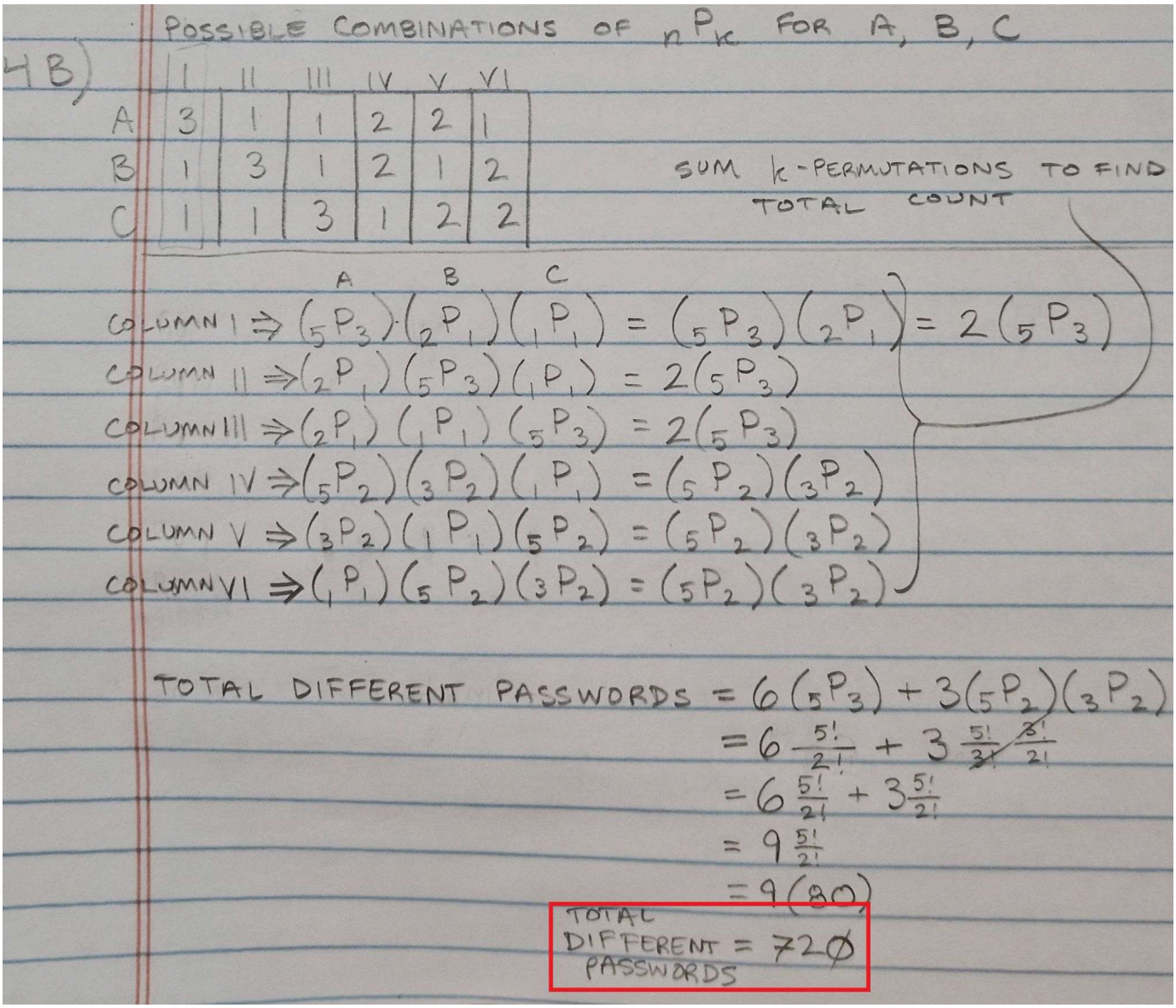
B twice → $({}_3P_2)$ … (3 positions remaining)

C once → $({}_1P_1)$ … (1 position remaining)

$$({}_5P_2)({}_3P_2)({}_1P_1) = 120\ different\ passwords$$

- **order matters** - We want all possible combinations of the letters (A, B, C) selected.
- **w/o replacement** - We don't want to recount letters selected (AAC, ABB, etc.).

4b. Suppose A,B, and C are the letters selected and there are no other restrictions. How many different passwords exist in this scenario?



4c. Now suppose we don't put any restrictions on the letters chosen. (It could be A,B, and C, or it could be A, C, and D, or it could be B, D, and E, etc.) How many different passwords exist in this scenario?

$$\left({}_5C_3 \right) = \frac{5!}{2!3!} = 10\ different\ triad\ combinations\ from\ A,\ B,\ C,\ D,\ E$$

- **without replacing** - We do not want to use a given letter more than once when making triads.
- **order does not matter** - We only need to pick each triad once (i.e. not ABC and BAC).

I took the sum of the 10 sets of possible three letter combinations to find the total possible different passwords (sample space). Each set has the same possible combinations as part 4B (used ABC), 720 different passwords per set.

$$\begin{aligned} Different\ passwords &= (720)ABC + (720)ABD + \ldots + (720)CDE \\ &= 10 * (720) \\ &= 7,200 \end{aligned}$$

There are **7,200 different possible passwords** when using 5 letters to make a 3 letter password.