Suggested Approach for Teaching a Lesson of Combination

Yip Ming Ham
Hong Kong Baptist University
Mak Kin Wai
Cheung Chuk Shan College

Introduction

When we teach the topic of binomial expansion[1] for the HKDSE Mathematics Extended Module 1 curriculum, we need to teach students on how to find the binomial coefficients using C_r^n. Also, the following nice relationship will be introduced:

$C_r^{n+1} = C_r^n + C_{r-1}^n$ ---------- (*)

Instead of giving the algebraic proof based on the formula $C_r^n = \frac{n!}{r!(n-r)!}$, I attempted to deliver a mental proof to explain why (*) is true. In this way, students could be given a chance to appreciate more about combinatorics. The ideas and the flows of the lessons will be described next.

Details of the lesson

Question: Show that $C_r^{n+1} = C_r^n + C_{r-1}^n$ ------(*).

Solution:

Consider (n + 1) objects, where one of the objects is represented by the symbol Δ and the others are represented by the symbol X. (See Figure 1)

1 Curriculum Development Council, The Hong Kong Examinations and Assessment Authority. *Mathematics Curriculum and Assessment Guide*. http://334.edb.hkedcity.net/doc/eng/curriculum/Math%20C&A%20Guide_updated_e.pdf

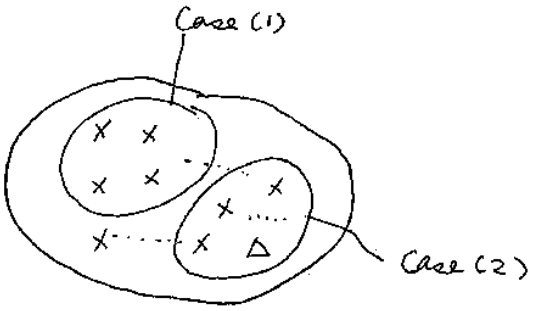


Figure 1

Then, C_r^{n+1} simply means the number of ways to select *r* objects from (*n* + 1) objects without reordering.

Now, there are two cases here (refer to Figure 1 again).

Case (1) The group does NOT include the object Δ.
Since 1 object is excluded from the selection, we simply need to select r objects (some "X" in the diagram) from these remaining n objects. That is given by C_r^n.

Case (2) The group includes the object Δ.
Since 1 object is already included in the selection, we simply need to select $(r-1)$ objects from the *n* objects (i.e. All the "X" in the diagram). That is given by C_{r-1}^n.

Combining Cases (1) and (2), the grand total should be $C_r^n + C_{r-1}^n$. The result follows.

After delivering the above explanation to our students, a few number of students found it difficult to understand. I then considered a small number of objects and illustrated the relationship in the following manner.

➔ To show that $C_2^3 = C_2^2 + C_1^2$, we consider the set *S*={*A*, *B*, *X*} having only 3 elements.

To select any two letters from *S*, the outcomes are {*A, B*}, {*B, X*} and {*A, X*} in which the grand total is 3. That is, $C_2^3 = 3$.

We may consider the selection based on the following 2 cases:

Case (1) Without "X"
The only outcome is $\{A, B\}$ and this is given by $C_2^2 = 1$.

Case (2) With "X"
The outcomes are $\{A, X\}$ and $\{B, X\}$ and this is given by $C_1^2 = 2$.
(Remarks: From the letters $\{A, B\}$, we just pick 1 to pair up with X.)

Combining the two cases, we have $C_2^2 + C_1^2 = 1 + 2 = 3 = C_2^3$. Then we thus verified the validity of the formula when $n = 3$ and $r = 2$.

Students understood better afterwards. Based on my approach, a student quickly asked, "What if we want a formula for C_r^{n+2}? Do we have the same formula?" My response was, "No, the original formula doesn't work as the situation becomes more complicated." In fact, by adopting the same technique, one can prove that $C_r^{n+2} = 2C_{r-1}^n + C_{r-2}^n + C_r^n$ without tedious algebraic working steps. However, it should be too demanding for students to prove this new formula. Later on, I thought of an approach to inspire my students in this aspect. I asked: "Can you think of a story explaining the phenomenon $C_3^7 = 2C_2^5 + C_1^5 + C_3^5$ with a simple story?"

I stayed for a while and drew the following diagram (note the special guests Δ and ✯ in Figure 2).

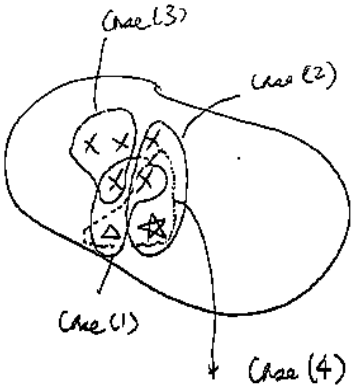


Figure 2

I asked, "Using the same trick like what I did before, if we want to take any 3 objects from a group of 7 objects, how many different cases can we consider this time? " One student voiced out, "There should be 4." I then probe

other students with more guidance and eventually they worked out the following story.

Case (1) The selection includes the object Δ:
For this case, we need to select 2 "X" from a set of 5 "X", in which the total number of ways is given by C_2^5 .

Case (2) The selection includes the object ✯:
By similar argument as stated in case 1, the total number of such groupings is given by C_2^5.

Case (3) The selection includes only the object "X", but excluding the objects Δ and ✯:
For this case, we need to select 3 "X" from a set of 5 "X", in which the total number of ways is given by C_3^5 .

Case (4) The selection includes both Δ and ✯:
For this case, we simply need to select 1 "X" to come together with Δ and ✯ in which the total number of such groupings is given by C_1^5.

Thus, combining all the 4 cases, we can prove that:
$C_3^7 = 2C_2^5 + C_1^5 + C_3^5$ --- (#)

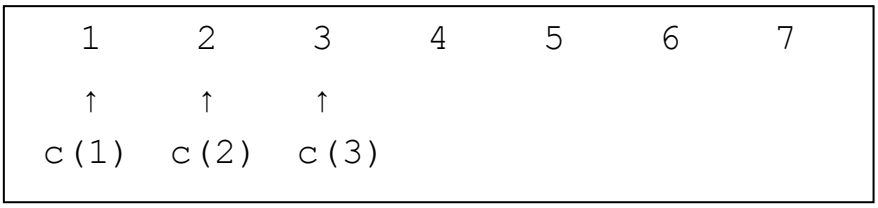
Also, by setting $n + 2 = 7$ and $r = 3$ for the left hand side of (#), students may infer the following formula as a result:
$C_r^{n+2} = 2C_{r-1}^n + C_{r-2}^n + C_r^n$

As a final remark, students still wondered if the grand total is really $C_3^7 = 35$. So I think it is really beneficial to show a full list of output for the required groupings. I finally used a computer program written by Dr. Yip for this purpose. (See the algorithm, output and source code written in Scilab[2] shown below.)

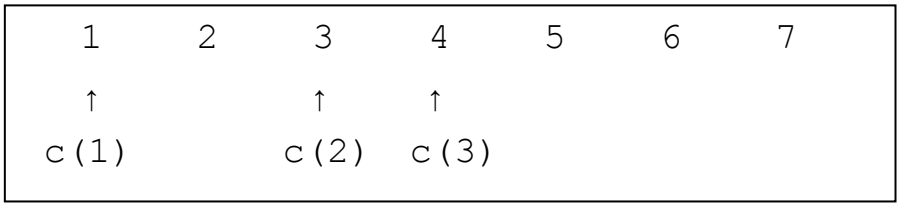
2 Scilab. *Scilab for very beginners tutorial*. http://www.scilab.org/content/view/full/849

- The Algorithm

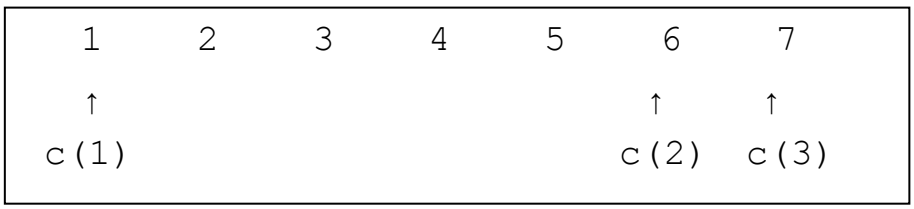
The method used to generate the combinations can be described as follows. To obtain all C_3^7 combinations, first, we initialize the combination to (1 , 2 , 3).

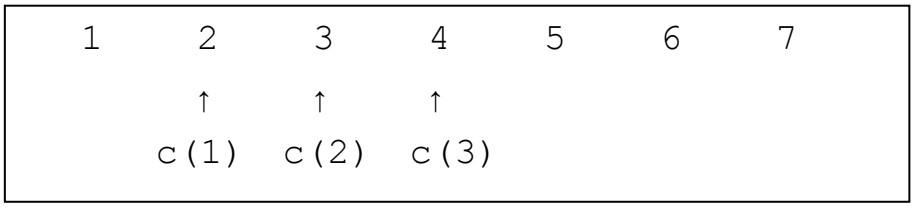


The next combination (1 , 2 , 4) is then obtained by moving c(3) one step forward (realized in the code by increasing the value of c(3) from 3 to 4). This procedure is repeated until c(3) = 7 which cannot be moved further. In this case, we move the previous entry c(2) one step forward and at the same time reset all entries in front of c(2) to the positions that immediately follow c(2). This generates the combination (1 , 3 , 4) in the next figure.



In general, we start from the last entry c(3) and try to move it forward. If c(3) is already at the final position, then we try moving the previous entry c(2) one step forward and reset c(3) to the position next to c(2). If c(2) is already in the second last place, then we try moving c(1) one step forward and reset c(2) and c(3) to the positions immediately after c(1), see the following figures.





When all entries cannot be moved forward, all combinations have been generated and the procedure finishes.

```
  1     2     3     4     5     6     7
                        ↑     ↑     ↑
                      c(1)  c(2)  c(3)
```

- The related output

The outputs of the program are depicted in the following diagrams. The first fifteen (C_2^6) combinations contain the element "1". The next twenty (C_3^6) combinations do not contain the element "1". The total number of combinations is $C_3^7 = C_2^6 + C_3^6$.

```
Combination   1:   1   2   3
Combination   2:   1   2   4
Combination   3:   1   2   5
Combination   4:   1   2   6
Combination   5:   1   2   7
Combination   6:   1   3   4
Combination   7:   1   3   5
Combination   8:   1   3   6
Combination   9:   1   3   7
Combination  10:   1   4   5
Combination  11:   1   4   6
Combination  12:   1   4   7
Combination  13:   1   5   6
Combination  14:   1   5   7
Combination  15:   1   6   7
```

```
Combination  16:   2   3   4
Combination  17:   2   3   5
Combination  18:   2   3   6
Combination  19:   2   3   7
Combination  20:   2   4   5
Combination  21:   2   4   6
Combination  22:   2   4   7
Combination  23:   2   5   6
Combination  24:   2   5   7
Combination  25:   2   6   7
Combination  26:   3   4   5
Combination  27:   3   4   6
Combination  28:   3   4   7
Combination  29:   3   5   6
Combination  30:   3   5   7
Combination  31:   3   6   7
Combination  32:   4   5   6
Combination  33:   4   5   7
Combination  34:   4   6   7
Combination  35:   5   6   7
```

- The source code

```
// A Scilab program that shows all nCr combinations

n = 7;
r = 3;

c = 1:r;   // initial combination
count = 0;   // counter of number of combinations generated
while 1,
    count = count + 1;

    // display the combination
    mfprintf(6,'Combination %3d:',count)
    for i = 1:r,
        mfprintf(6,' %3d',c(i));
    end
    mfprintf(6,'\n');

    // Search backward the first number in the current combination that
    // can be incremented without conflict
    // (i.e. largest k s.t. c(k) < c(k+1)-1 = n-r+k)
    k = r;
    while c(k)==n-r+k,
        k = k - 1;

        if k==0 then
            // no value can be incremented
            // all combinations have been formed
            return
        end
    end

    // increment the kth entry by 1
    // reset the remaining entries to immediately after
    c(k:r) = c(k) + (1:(r-k+1));
end
```

Reflection

Combinatorics is a funny topic in mathematics. Students were so lucky to be given a chance to learn this concept from both the mathematics compulsory part and extended module 1. According to the observation of my lessons, students understand better after we include some concrete examples with small values of n and r.

Acknowledgement

Special thanks go to my S6A students who gave me a lot of insights during the lessons of mathematics extended module 1. Also, thanks wholeheartedly for

Dr. Yip's generous support in sharing with me his Scilab code so as to make the lessons to be more attractive.

First author's e-mail: mhyipa@hkbu.edu.hk