CS 237 Fall 2019 Homework Four Solution

Due date: PDF file due Thursday October 3rd @ 11:59PM in GradeScope with 6-hour grace period

Late deadline: If submitted up to 24 hours late, you will receive a 10% penalty (with same 6 hours grace period)

General Instructions

Please complete this notebook by filling in solutions where indicated. Be sure to "Run All" from the Cell menu before submitting.

There are two sections to the homework: problems 1 - 8 are analytical problems about last week's material, and the remaining problems are coding problems which will be discuss in lab next week.

In [1]:
```
# Here are some imports which will be used in code that we write for CS 237

# Imports potentially used for this lab

import matplotlib.pyplot as plt   # normal plotting

from math import log, pi       # import whatever you want from math
from collections import Counter
from numpy.random import randint, seed

%matplotlib inline

# Useful code

def show_distribution(outcomes, title='Probability Distribution'):
    num_trials = len(outcomes)
    X = range( int(min(outcomes)), int(max(outcomes))+1 )
    freqs = Counter(outcomes)
    Y = [freqs[i]/num_trials for i in X]
    plt.bar(X,Y,width=1.0,edgecolor='black')
    if (X[-1] - X[0] < 30):
        ticks = range(X[0],X[-1]+1)
        plt.xticks(ticks, ticks)
    plt.xlabel("Outcomes")
    plt.ylabel("Probability")
    plt.title(title)
    plt.show()

# This function takes a list of outcomes and a list of probabilities and
# draws a chart of the probability distribution.

def draw_distribution(Rx, fx, title='Probability Distribution for X'):
    plt.bar(Rx,fx,width=1.0,edgecolor='black')
    plt.ylabel("Probability")
    plt.xlabel("Outcomes")
    if (Rx[-1] - Rx[0] < 30):
        ticks = range(Rx[0],Rx[-1]+1)
        plt.xticks(ticks, ticks)
    plt.title(title)
    plt.show()

def round4(x):
    return round(x+0.00000000001,4)

def round4_list(L):
    return [ round4(x) for x in L]


# This is a solution to one of the lab problems, but useful for calculations
# in the analytical problems

#  MUST BE REMOVED BEFORE CREATING DISTRIBUTION VERSION!!!!!

def C(N,K):
    if(K < N/2):
        K = N-K
    X = [1]*(K+1)
    for row in range(1,N-K+1):
        X[row] *= 2
        for col in range(row+1,K+1):
            X[col] = X[col]+X[col-1]
    return X[K]
```

Analytical Problems

You may use ordinary ASCII text to write your solutions, or (preferably) Latex. A nice video introduction to Markdown Cellls and Latex in Jupyter notebooks may be found here (https://www.youtube.com/watch?v=-F4WS8o-G2A). Various complicated examples are shown with Latex here (https://jupyter-notebook.readthedocs.io/en/latest/examples/Notebook/Typesetting%20Equations.html).

Quantitative answers must be given as decimals to 4 significant digits, unless very small, in which case you should give in scientific notation to 4 significant digits (if possible).

You may use fractions throughout your calculations, but for grading I want you to give them in decimal form.

You may present your answers in Markdown or Code cells. You are free to create extra Code cells to do the calculations. Just make sure to indicate clearly where your answer is; one nice way to do this is to put it in a box:

Here is my answer, graders: $\boxed{3.1416}$.

Problem 1

(Permutations and Combinations) Suppose 2 cards are drawn without replacement (the usual situation with cards) from an ordinary deck of 52 randomly shuffled cards. Find the probability that:

(a) The first card is not a ten of clubs or an ace;

(b) The first card is an ace, but the second is not;

(c) The cards have the same rank (i.e., both are Aces, both are 2's, both are 3's, etc.);

(d) At least one card is a Diamond;

(e) Not more than 1 card is a picture card (Jack, Queen, King).

Solution:

(a) If we remove the aces and the ten of clubs, we have 47 cards left, so

$$\frac{47}{52} = 0.9038.$$

(b) P(first card is an ace) $*$ P(second card not an ace) =

$$\frac{4}{52} * \frac{48}{51} = \frac{192}{2652} = 0.0724.$$

(c) Two ways to do this. The simplest way is that after the first card is chosen, you have to match the denomination with the second card, so you have 3 chances out of 51 or: $\frac{3}{51} = \frac{1}{17}$.

If you want to use combinations, choose a denomination and then choose two of the four of that denomination:

$$\frac{\binom{13}{1} * \binom{4}{2}}{\binom{52}{2}} = \frac{1}{17} = 0.0588.$$

(d) As usual, "at least" should tip you off to consider the inverse method.

P(no diamonds) = P(first card not a diamond) $*$ P(second card not a diamond) =

$$\frac{39}{52} * \frac{38}{51} = \frac{1482}{2652} = 0.5588,$$

so P(at least one diamond) = $1.0 - 0.5588 = 0.4412$.

(e) There are $4 * 3 = 12$ picture cards (J,Q,K of each suit). The choices are 0, 1, or 2 picture cards; the question asks for 0 or 1, but using the inverse method we can derive the answer by just considering 2.

P(first a picture card) = $\frac{12}{52}$, and P(second a picture card) = $\frac{11}{51}$.

So P(not more than 1 picture card) =

$$1.0 - (\frac{12}{52} * \frac{11}{51}) = 0.9502.$$

Problem 2

Suppose you shuffle a standard deck and draw the card on top of the deck; the probability of this card being an Ace is 4/52 = 1/13. Now suppose, instead of drawing it from the top, you draw the **second** card from the original deck; since the deck was randomly shuffled, it should not matter **where** in the deck you draw one card from, there should still be a 1/13 probability of an Ace.

But this implies that if you draw a card from the top, toss it away without looking at it, and draw the second card, the probability of an Ace is 1/13 (though you are drawing it from a deck of 51 cards). This is a little confusing (at least to me!), so let's make sure we are thinking straight about this.

(a) Draw a tree diagram of what happens when you draw two cards without replacement from a well-shuffled deck; in each of the two steps, consider the cases of an Ace or a non-Ace.

(b) Calculate the probability that the second draw is an Ace, and observe (nothing more to do!) that it is indeed 1/13, no matter what the first card is. (In this problem, use fractions instead of rounding decimals, since we want to have an absolutely precise result.)

Solution

IMG-5723.JPG

Problem 3

(Permutations and Combinations) We draw 5 cards at random from an ordinary deck of 52 cards without replacement. In this case, we will think of the 5 cards as a **sequence**, and we are going to consider the relationship between the first two cards drawn and the last three cards drawn (each of which will in fact be treated as sets, so we have a sequence of sets).

Consider the following events:

```
A = "the first two cards are both spades"
C = "the last 3 cards are all spades"
```

(a) Calculate P(A)

(b) Calculate P(C)

(c) Calculate $P(C \mid A)$.

Hint: For (b), think about your answer to the previous problem.

Solution:

(a) If we use permutations, we have:

$$P(A) = \frac{13}{52} \cdot \frac{12}{51} = 0.0588$$

If we use combinations, we have

$$P(A) = \frac{\binom{13}{2}}{\binom{52}{2}} = 0.0588$$

A = "the first two cards are both spades" B = "there is at least one spade among the first two cards"

(b) It does not matter how cards are drawn from a well-shuffled deck, so event C is calculated in the same way (we can ignore the first two cards).

$$P(C) = \frac{13}{52} \cdot \frac{12}{51} \cdot \frac{11}{50} = 0.0129$$

or:

$$P(C) = \frac{\binom{39}{3}}{\binom{52}{3}} = 0.0129$$

(c) The event $C \cap A$ = "all five cards are spades," so

$$P(C \cap A) = \frac{\binom{39}{5}}{\binom{52}{5}} = 0.0004952$$

therefore:

$$P(C \mid A) = \frac{P(C \cap A)}{P(A)} = \frac{0.0004952}{0.0588} = 0.0084$$

Problem 4

(Permutations and Combinations) This problem is a continuation of Problem 3.

Consider the following events:

```
B = "there is at least one spade among the first two cards"
C = "the last 3 cards are all spades"
```

(a) Calculate P(B)

(b) Calculate $P(C \mid B)$.

Hint: For (b), consider the two cases of 1 and 2 Spades (in B) and what happens to C in each case.

Solution:

(a) You could calculate these with separate cases: let D = "no spades in first two cards" and E = "there is exactly one spade among the first two cards," so A, E, and D exhaust all possibilities and are disjoint, and $B = A \cup E$ (a disjoint union), thus:

$$P(A) + P(E) + P(D) = 1.0$$

and

$$P(E) = \frac{\binom{13}{1} \cdot \binom{39}{1}}{\binom{52}{2}} = 0.3823529$$

so

$$P(B) = 0.05882352 + 0.3823529 = 0.4412.$$

But easier to use combinations and the inverse method:

$$P(B) = 1.0 - \frac{\binom{39}{2}}{\binom{52}{2}} = 0.4412$$

(b) Clearly, using results from (a), we have

$$C \cap B = C \cap (A \cup E) = (C \cap A) \cup (C \cap E)$$

(a disjoint union), and we have calculated $P(C \cap A)$ in Problem 3. Now

$$P(C \cap E) = P(E) * \frac{\binom{12}{3}}{\binom{50}{3}} = 0.3823529 * 0.01122448 = 0.004291712478992$$

$$P(C \cap B) = 0.004291712478992 + 0.0004952 = 0.004786912478992$$

therefore:

$$P(C \mid B) = \frac{P(C \cap B)}{P(B)} = \frac{0.004786912478992}{0.4412} = 0.01085$$

Problem 5

Suppose you draw two cards (a sequence) from a standard 52-card deck without replacement.

Let A = "the first card is a spade" and B = "the second card is an Ace." These two events "feel" (at least to me) as if they should be independent, but we will see, surprisingly, that they are not. A tree diagram will help with the analysis.

(a) Calculate $P(A)$

(b) Calculate $P(B)$

(c) Calculate $P(B \mid A)$

(d) Show that A and B are not independent (trivial, just observe that the results of (a) and (c) are different!)

Solution

(a)

$$P(A) = \frac{13}{52} = 0.25$$

(b) You could prove this using a tree diagram, but from the previous problem, it should be obvious that this is the same as asking "what is the probability that the first card is an Ace?"

$$P(B) = \frac{4}{52} = 0.0784$$

(c) For this, we first calculate $P(A \cap B)$ and for this we DO need to draw a diagram, branching first on the possibility of an Ace or not in the first draw (we do know it is a spade) and an Ace in the second draw.

$$P(A \cap B) = \frac{1}{52} \cdot \frac{3}{51} + \frac{12}{52} \cdot \frac{4}{51} = 0.0192307692307$$

Therefore,

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)} = \frac{0.01923}{0.25} = 0.0769$$

(d) Since $0.0784 \neq 0.0769$ we see the two events are NOT independent.

In [2]:
```
p0 = 3/(52*51)
p1 = 48/(52*51)
p2 = 12*4/(52*51)
p3 = 12*47/(52*51)


p0+p2
```

Out[2]: `0.019230769230769232`

Problem 6

(Combinations -- Poker Probabilities) Suppose you deal a poker hand of 5 cards from a standard deck as discussed in lecture.

(a) What is the probability of a flush (all the same suit) of all red cards?

(b) What is the probability of a full house where the 3-of-a-kind include two black cards, and the 2-of-a-kind are not clubs?

(c) What is the probability of a pair, where among the 3 non-paired cards, we have 3 distinct suits?

Hint: These are straight-forward modifications of the formulae given in lecture. You may quote formulae already given in lecture or lab without attribution and without explaining how to get the formula.

Solution: These are all modifications of the formulae presented in lecture.

(a) Choose the suit (only 2 are red) then the hand from that suit:

$$\frac{\binom{2}{1} * \binom{13}{5}}{\binom{52}{5}} = 9.903 * 10^{-4} = 0.001$$

(b) Adapting the formula from lecture, we just have to change the selection of suits for each, since there are only 2 ways of choosing one of the two non-black suits for 3 cards, and 3 ways of choosing 2 from 3 non-clubs for the pair:

$$\frac{\binom{13}{1} * \binom{2}{1} * \binom{12}{1} * \binom{3}{2}}{\binom{52}{5}} = 3.602 * 10^{-4} = 0.0004.$$

(c) Again we adapt the standard formula, but for the 3 non-paired cards, we have to choose 3 different denominations (so there are no more paired cards) and 3 suits for each of the three::

$$\frac{\binom{13}{1} * \binom{4}{2} * \binom{12}{3} * P(4,3)}{\binom{52}{5}} = 0.1585.$$

Problem 7

(Combinations) Consider the following problem: "From an ordinary deck of 52 cards, seven cards are drawn at random and without replacement. What is the probability that at least one of the cards is a King?" A student in CS 237 solves this problem as follows: To make sure there is a least one King among the seven cards drawn, first choose a King; there are $\binom{4}{1}$ possibilities; then choose the other six cards from the 51 cards remaining in the deck, for which there are $\binom{51}{6}$ possibilities. Thus, the solution is

$$\frac{\binom{4}{1}\binom{51}{6}}{\binom{52}{7}} = 0.5385.$$

However, upon testing the problem experimentally, the student finds that the correct answer is somewhat less, around 0.45.

(a) Calculate the correct answer using the techniques presented in class;

(b) Explain carefully why the student's solution is incorrect.

Solution:

(a) We use the inverse method: there are $\binom{48}{7}$ seven-card hands with no Kings, so the probability of at least one King is

$$1.0 - \frac{\binom{48}{7}}{\binom{52}{7}} = 0.4496$$

(b) The problem is that hands with more than one King are counted multiple times. For example,

```
{ KH, KD, 2H, 5S, JH, 10C, AD }
```

is counted twice: once when the KH is counted as part of the $\binom{4}{1}$ and the KD is counted as part of $\binom{51}{6}$, and again when KD is part of $\binom{4}{1}$ and KH is part of $\binom{51}{6}$. So a hand with n Kings, $n > 1$, will be counted n times.

Problem 8

(Combinations, Subsets, and Partitions) Suppose you have a committee of 10 people.

(a) How many ways are there to choose a group of 4 people from these 10 if two particular people (say, John and Dave) can not be in the group together?

(b) How many ways are there to choose a team of 5 people from these 10 with one particular person being designated Captain and another particular person being designated Co-Captain?

(c) How many ways are there to separate these 10 people into two groups, if no group can have less than 2 people?

Solution:

For (a), we have to remove from the total those committees on which John and Dave both sit; but since the only choice is the other two people, this is

$$\binom{10}{4} - \binom{8}{2} = 210 - 28 = 182.$$

(b) The captain and co-captain are a sequence, and the remaining 3 are a set, so

$$P(10, 2) * \binom{8}{3} = 5040.$$

For (c) and (d) , note that if we choose one group, then the other group is determined as well; and so we should consider all possible subsets (one group), which determines the other set. However, this number overcounts by a factor of 2, since we have counted each group and its complement (each of which produces the same pair of groups, just exchanged); e.g., if one group is {1,2}, then other group is {3,4,5,6,7,8,9,10}; but if we choose {3,4,5,6,7,8,9,10}, then the other group is {1,2}, and these represent the same "way" of separating into two groups. So count the number for one group and divide by two.

(c) We can count any group that does not have 0, 1, 9, or 10 people (the latter two would produce a complement with 1 or 0 people). There is only one group with 0, one group with 10, ten groups with 1, and ten groups with 9. The number of subsets of a 10-element set is 210. So we have 210 - 22 = 1024 - 22 = 1002. Dividing by 2 we get 501 ways.

Problem 9

Among 25 Senate candidates, the 11 (all Republicans) think global warming is a myth, the 8 (all Democrats) believe that global warming is real, and the rest (from the "Spineless Party") have no opinion ("I'm not a scientist"). A newspaper interviews a random sample of 5 of the candidates. What is the probability that

(a) all 5 think global warming is a myth;

(b) all 5 share the same position (i.e., all think it is a myth, all believe it is real, or all have no opinion);

(c) 3 share the same position and 2 share a different position (for example, 3 believe it is a myth and 2 have no opinion).

(d) 2 share the same position, 2 share the same position, but different from the first two, and the remaining candidate has a position different from the other four.

Hint: Worry about overcounting in (d).

Note that this is really the same as poker probabilities, but you have a deck of 25 cards, with 11 of one denomination, 8 of another, and 6 of a third.

Solution: There are $\binom{25}{5}$ ways of choosing the five for the interview.

(a) There are $\binom{11}{5}$ ways of choosing 5 who think it is a myth, so

$$\frac{\binom{11}{5}}{\binom{25}{5}} = 0.0087.$$

(b) There are $\binom{8}{5}$ ways of choosing 5 who think it is real, and $\binom{6}{5}$ ways of choosing 5 who have no opinion. All of these are disjoint, so we can simply add the probabilities:

$$\frac{\binom{11}{5} + \binom{8}{5} + \binom{6}{5}}{\binom{25}{5}} = 0.0097.$$

(c) There are six different permutations of positions among the groups:

```
majority of 3        minority of 2         Number of possible groups
-------------        -------------         -------------------------
   myth                 real                C(11,3) * C(8,2)   = 4620
   myth                 no opinion          C(11,3) * C(6,2)   = 2475
   real                 myth                C(8,3)  * C(11,2)  = 3080
   real                 no opinion          C(8,3)  * C(6,2)   = 840
   no opinion           myth                C(6,3)  * C(11,2)  = 1100
   no opinion           real                C(6,3)  * C(8,2)   = 560
                                                              Total: 12705
```

So the probability is 12705/C(25,5) = 0.2391. Note that you can NOT solve (c) as in the full-house card example (choose a position, then the number of choices, then the second position, and the number of choices for that) because the number of choices is different among the three positions; hence we must explicitly list all possibilities.

(d) This is similar to (c), except that we have to worry about double-counting, since the "ways" of dividing between 2 and 2 are redundant, as explained in lecture on Tuesday 9/25. Also note that once the positions for the 2 pairs are chosen, there is no choice for the remaining candidate so we can forget about him/her. So

```
 Group of 2           Group of 2           Number of possible groups
-------------        -------------         -------------------------
   myth                 real                C(11,2) * C(8,2) * C(6,1)  = 9240
   myth                 no opinion          C(11,2) * C(6,2) * C(8,1)  = 6600
   real                 no opinion          C(8,2)  * C(6,2) * C(11,1)  = 4620

                                                              Total: 20460
```

So the probability is 20460/C(25,5) = 0.3851.

Lab Problems

We will continue our investigation of efficient implementations of the standard algorithms in probability theory, first considering how to implement various kinds of sampling (for which we used numpy last time), and then how to calculate combinations and permutations.

Problem 10: Choosing, Shuffling, and Sampling from a List

Now we will we will create our own versions of the `sample(...)`, `shuffle(...)`, and `choice(...)` functions, which we used in the last lab (in their numpy versions, but now we write our own!).

All you need to do is to demonstrate these as shown. We could test them using probability distributions, but it will sufficient to check the results by eye...

NOTE: You may use `randint(...)` from the numpy random library (imported above) but no other library functions which generate random results.

Part (a): Choosing from a list with replacement

This part is easy: to choose a member of a list randomly and with replacement, simply generate a random integer as an index and return the member at the index.

In [3]:

```
[5, 6, 1, 4, 4, 4, 2, 4]
[6, 3, 5, 1, 1, 5, 3, 2]
[1, 2, 6, 2, 6, 1, 2, 5]
[4, 1, 4, 6, 1, 3, 4, 1]
```

Part (b): Shuffling a list

This is exactly the same as shuffling a deck and then dealing out a number of cards from the top. In order to do this, we shall use the following method, known as the Fisher-Yates Shuffle (https://en.wikipedia.org/wiki/Fisher%E2%80%93Yates_shuffle), which works in $O(n)$. Basically, you maintain a shuffled part of the list and an unshuffled part of the list, and at each step randomly select a number from the unshuffled part and move it to the shuffled part.

```
-- To shuffle an array A of n elements (indices 0..n-1):
for i from n−1 downto 1 do
     j ← random integer such that 0 ≤ j ≤ i
     exchange A[j] and A[i]
```

Since this shuffle works in-place, you should create a copy of the list before shuffling it.

In [4]:

```
[3, 9, 5, 10, 2, 7, 8, 4, 1, 6]
[4, 6, 2, 3, 10, 9, 1, 7, 8, 5]
[3, 4, 9, 5, 6, 2, 1, 7, 10, 8]
[7, 2, 10, 3, 8, 6, 9, 1, 4, 5]
```

Part (c) Sampling without replacement from a list

This is easy: just slice the list produced by shuffling.

In [5]:

```
[3, 9, 5, 10, 2]
[4, 6, 2, 3, 10]
[3, 4, 9, 5, 6]
[7, 2, 10, 3, 8]
```

Problem 11: Calculating P(N,K) and C(N,K)

(a) The first problem is very simple: you must write a function P(N,K) and then print out the value for P(300,200). The only thing you have to remember is to make it as efficient as possible (which means don't use two calls to factorial and a division). [Hint: to check your solution, use Wolfram Alpha.]

(b) This problem is about creating an efficient implementation for C(N,K). Before starting this problem, read about Pascal's Triangle and its relationship to C(N,K) here (https://en.wikipedia.org/wiki/Pascal%27s_triangle). Then, you must write a function C(N,K) and print out the value C(1000,250) and C(1000,750) (which should be the same). However, as remarked in lecture, the standard mathematical definition is very inefficient; instead you *could* use the recursive definition based on Pascal's Triangle:

Lab02.P2.png

but this, although much more efficient than the definition using factorials, does a lot of repeated computation. So you must write a bottom-up version of this definition, which replaces the top-down recursion with bottom-up iteration. You *could* create a two-dimensional array and fill in by row and column, and then return the result (which will be in the lower right corner); this is essentially Pascal's Triangle tilted counter-clockwise 45 degrees, e.g., for C(6, 3) you would get

```
                 1                                     C(0,0)
              1      1                           C(1,0)       C(1,1)
           1     2      1                  C(2,0)      C(2,1)       C(2,2)
        1     3      3     1         C(3,0)      C(3,1)       C(3,2)      C(3,3)
           4     6      4
             10     10                                  etc.
                20


    1  1  1  1           C(0,0)  C(1,1)  C(2,2)  C(3,3)          Rule: top row and left column a
re all 1;
    1  2  3  4           C(1,0)  C(2,1)  C(3,2)  C(4,3)          every other value is sum of the
two values
    1  3  6 10           C(2,0)  C(3,1)  C(4,2)  C(5,3)          immediately above and to the im
mediate left.
    1  4 10 20           C(3,0)  C(4,1)  C(5,2)  C(6,3)  ....
```

Note: If we think of this as a 2D array, with indices starting at 0, then the value for C(N,K) is to be found at row N-K and column K.

However, you don't need to create an actual 2D array, you can use a single list (for the row) which you initialize to [1, 1, 1, ..., 1] and then keep keep scanning down, creating each successive row. That is, your lists would be:

```
    [1, 1, 1, 1]
    [1, 2, 3, 4]
    [1, 3, 6, 10]
    [1, 4, 10, 20]
```

But there is one more refinement to make it as efficient as possible! Since C(N,K) is symmetric, i.e., C(N,K) = C(N,N-K), the array is symmetric around the diagonal, and thus you are doing twice as much work as necesary! Really, you only need to compute the values at, above, and to the right of the diagonal. So, you can calculate starting at the diagonal, i.e., in the nth row you can just start at the nth column; note that the numbers on the diagonal are simply twice the number above it, and then the rest are calculated as usual by summing the number to the left and above:

```
    1  1  1  1     C(0,0)  C(1,1)  C(2,2)  C(3,3)          New rule: Top row is all 1s; values on
diagonal are
    1  2  3  4     C(1,0)  C(2,1)  C(3,2)  C(4,3)              twice the value immediately above;
 all others
    1  3  6 10     C(2,0)  C(3,1)  C(4,2)  C(5,3)              calculated as previously from value
s immediately
    1  4 10 20     C(3,0)  C(4,1)  C(5,2)  C(6,3)  ....        above and immediately to the left.
```

When writing your code, you will need to make sure that K >= N/2, so you must exploit the symmetry of C(N,K), i.e.,

```
def C(N,K):
    if(K < N/2):
        K = N-K        # because C(N,K) = C(N, N-K) this has no effect on result returned
```

Whew, that's it! You have to write your function C(N,K) so that it only calculates the numbers in the upper diagonal. Think about it...... my solution uses a double for loop and is only 6 lines long.....

Compared with the recursive definition based on the mathematical formula, this is WAY more efficient; [here (CombinationTest.html)](CombinationTest.html) is a trace of a very large computation which demonstrates the effectiveness of this approach.

In [6]:

```
Out[6]: 327943773098001918628170562380683064484501303900608517398125137849389970121216851842356184685
712825632014264627117755632402345611746842669512470914001200904928134714415837392622354400568
340396297551130259875864371286738821821734751391938425380405054844783183834887733584127846679
520819537945049903902962831618186851330195682974549249883451789905767522387567180014514916250
01485807722958727751263051646500864000000000000000000000000000000000000000000000000000
```

In [4]:

```
1 [1, 1, 1, 1, 1]
2 [1, 1, 1, 1, 1]
3 [1, 2, 1, 1, 1]
4 [1, 2, 1, 1, 1]
5 [1, 2, 3, 1, 1]
4 [1, 2, 3, 1, 1]
5 [1, 2, 3, 4, 1]
4 [1, 2, 3, 4, 1]
5 [1, 2, 3, 4, 5]
6 [1, 2, 3, 4, 5]
2 [1, 2, 3, 4, 5]
3 [1, 2, 6, 4, 5]
4 [1, 2, 6, 4, 5]
5 [1, 2, 6, 10, 5]
4 [1, 2, 6, 10, 5]
5 [1, 2, 6, 10, 15]
6 [1, 2, 6, 10, 15]
2 [1, 2, 6, 10, 15]
3 [1, 2, 6, 20, 15]
4 [1, 2, 6, 20, 15]
5 [1, 2, 6, 20, 35]
6 [1, 2, 6, 20, 35]
35
```

Problem 12: Creating Enumerations, Permutations, and Combinations

Here is a short recursive function `enumerate(...)` which will enumerate all sequences of L letters chosen *with replacement* from an initial collection of N letters from ['A', 'B', 'C', ..., 'Z'] (i.e, if N = 3, then you are using only the letters 'A', 'B', and 'C'; and we are assuming the N <= 26).

In [5]:

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R',
'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
['A', 'A']
['A', 'B']
['A', 'C']
['B', 'A']
['B', 'B']
['B', 'C']
['C', 'A']
['C', 'B']
['C', 'C']
```

Here is an example of the tree traversed during enumerate(3,2) (two-letter permutations of the letters 'A', 'B', and 'C'):</h3>

Screen%20Shot%202019-09-28%20at%201.43.07%20AM.png

This recursive paradigm is called "recursive backtracking" because it explores a tree of all possible sequences, where the sequences exist at the leaves of the tree, and the recursion explores all the ways that such sequences can be created. Specifically, it uses a `for` loop to go through all the letters, and for each letter, places it in the location `I` in the array A, and then recursively fills in the rest of the array. In the base case, the array is printed out. Please look carefully at this function and understand what it does before doing the rest of this problem. You will be modifying this function in two ways, to account for permutations (doing the same thing, but *without replacement*), and for combinations (doing the same thing, but without replacement and creating sets).

(a) You must write functions `permute(...)` and `permuteAux(...)` by analogy with the the functions `enumerate(....)` and `enumAux(....)`. Basically, the changes are to implement the "without replacement" condition: you must keep track of which letters have been used using a Boolean list in previous stages of the recursion (e.g., B[0] will be true if at an earlier stage of the recursion, 'A' has already been inserted into X; simply don't do the recursive call if that letter has been used).

Print out the sequences for `permute(4,3)`, as shown in the sample printout at the end of this document.

(b) Second, you must write `combinate(...)` by analogy with `permute(....)`. The main trick here is that we will encode combinations as permutations using the following important fact: *if all sets are represented by an ordered list, then the (ordered) permutations are exactly the combinations.*Suppose we say that a letter is *larger* than another if it occurs later in the alphabetic sequence.The trick in the code is to keep track of the last letter to be inserted into your sequence, and then only consider larger letters to be used when you go down into the recursion. Then all the letters will be in order.

Print out the values for `combinate(4,3)`, as shown below.

In [9]:

```
['A', 'B', 'C']
['A', 'B', 'D']
['A', 'C', 'B']
['A', 'C', 'D']
['A', 'D', 'B']
['A', 'D', 'C']
['B', 'A', 'C']
['B', 'A', 'D']
['B', 'C', 'A']
['B', 'C', 'D']
['B', 'D', 'A']
['B', 'D', 'C']
['C', 'A', 'B']
['C', 'A', 'D']
['C', 'B', 'A']
['C', 'B', 'D']
['C', 'D', 'A']
['C', 'D', 'B']
['D', 'A', 'B']
['D', 'A', 'C']
['D', 'B', 'A']
['D', 'B', 'C']
['D', 'C', 'A']
['D', 'C', 'B']
```

In [7]:

```

```

```
0
A
0
A
0
B
0
A
0
B
0
C
['A', 'B', 'C']
C
D
['A', 'B', 'D']
B
C
D
A
D
B
D
C
D
D
['A', 'C', 'D']
C
D
D
A
D
B
D
C
D
D
A
B
D
A
D
B
D
C
D
A
D
B
D
C
D
D
['B', 'C', 'D']
C
D
D
A
D
B
D
C
D
D
B
C
D
A
D
B
D
```

```
C
D
D
D
A
D
B
D
C
D
D
C
D
D
A
D
B
D
C
D
D
```

Hint: Here is what your code should print out:

For clarity, we show a list ['A','A','A'] as a string "AAA"; your printout will of course show lists.

```
(a)   permute(4, 3) returns 24 sequences:

ABC         // really ['A','B','C']
ABD
ACB
ACD
ADB
ADC
BAC
BAD
BCA
BCD
BDA
BDC
CAB
CAD
CBA
CBD
CDA
CDB
DAB
DAC
DBA
DBC
DCA
DCB

(b)   combinate(4, 3) returns 4 sequences:

ABC
ABD
ACD
BCD
```