Counting card hands (revisited)

How many 5 card hands are there (unordered selection from a standard 52 card deck) that consist of a single pair of the same value, and three other cards of different values? Two possible examples are:
2♥, 2♦, 7♣, 9♦3♥ and A♥, A♣, 4♦, 6♦3♥

We can encode a pair with a string as follows:
{P, O1, O2, O3} {PS1, PS2} {S1, S2, S3}
Where P represents the pair values and O1, O2, and O3, represent the other values. PS1, and PS2 are the suits for the pair, and S1, S2, S3, are suits for the 3 other cards. This leads to the following (incorrect) expression:

\[
\frac{13!}{9!} \binom{4}{2} \binom{4}{1}^3
\]
We now verify that we have a valid bijection between the string representation of a hand and the hand itself. The hand 2♥, 2♦, 7♣, 9♦3♥ is encoded as 2, 7, 9, 3, ♥, ♦, ♣, ♠, ♦, ♥. However the hand 2♥,2♦,7♣,3♥,9♦ is considered to be equivalent to 2♥,2♦,7♣,9♦3♥ because we are not concerned with the ordering of the cards. This is **bad news** because 2♥,2♦,7♣,3♥,9♦ is encoded as 2,7,3,9,♥,♦,♣,♥,♦, so the expression above over-counts the number of hands by 3!, the number of permutations of the 3 other cards. Therefore, to remedy this we divide by 3! yielding the expression:

\[
\frac{\binom{13}{1}\binom{4}{2}\binom{12}{3}\binom{4}{1}}{3!}
\]

A very good resource that shows how to count various (poker) card hands can be found at:

https://en.wikipedia.org/wiki/Poker_probability
Recursive Functions

A function is *recursively defined* if the function definition refers to itself.

**Example:** factorial function
0! = 1
n! = n (n-1)! For all natural numbers greater than or equal to 1.

Example: exponentiation
Let a be a non-zero real number then:
a^0 = 1
a^n= a (a^{n-1}) For all natural numbers greater than or equal to 1.
Recursively defined functions come in two parts, the base case, and the recursive step.

**Example:**
Base: $T(0) = 1$
Recursive step: $T(n) = 2T(n-1)$ For all natural numbers greater than or equal to 1.

Let’s work out a few terms of $T(n)$ for small values of $n$
$T(1) = 2$
$T(2) = 4$
$T(3) = 8$
$T(4) = 16$

Can you see a pattern developing?
Guess $T(n) = 2^n$.

Theorem: Let $T(0) = 1$, and $T(n) = 2T(n-1)$ for all natural numbers greater than or equal to 1, then $T(n) = 2^n$.

**Proof:** (Proof by mathematical induction)

**Base:** $T(0) = 1 = 2^0$

**Induction Hypothesis:** Assume that $T(k) = 2^k$ for $k \geq 1$.

**Induction Step:**

$T(n+1) = 2T(n)$

$= 2 \cdot 2^k$ (Use induction assumption)

$= 2^{k+1}$ □
**Compound Interest**

Alice deposits $1000 in a bank account that returns a fixed interest rate of 1.5% annually. After 1 year Alice would then have $1000 \times 1.015 = $1015. After 2 years Alice would have $1015 \times 1.015 = $1030.225.

Let $A(n)$ denote the amount that has accumulated in the bank after $n$ years. Then we have the recursively defined function

\[
A(0) = 1000.
\]
\[
A(n) = A(n-1) \times 1.015.
\]

A good guess for the closed form equation is:

\[
A(n) = 1000(1.015)^n, \text{ and we can prove this using induction.}
\]
Given the recursively defined function
A(0) = 1000
A(n) = 1.015 A(n-1)
P(n) is the proposition that the closed form equation for
A(n) is: A(n) = 1000(1.015)^n

Proof:
Base: A(0) = 1000(1.015)^0 = 1000.
Induction Hypothesis: P(k) is true.
Induction Step: Prove P(k+1) using P(k).
A(k+1) = 1.015A(k)
= 1.015[1000 (1.015)^k] (induction hypothesis)
= 1000[1.015 (1.015)^k] (commutative law)
= 1000(1.015)^{k+1}
Spending options
Suppose we have $n$ dollars to spend on treats. We can buy one treat per day. We have two favourite treats, T1 at a cost of $1$, and T2 at a cost of $2$. In how many ways can we spend the $n$ dollars buying our favourite treats?

We can solve this problem using a recursive function.

Let Sp(n) denote the number of ways to spend $n$ dollars on treats.
We use [ ] to separate the different strategies
Sp(1) = [1]: 1 way.
Sp(2) = [1,1];[2]: 2 ways.
Sp(3) = [1,1,1];[1,2];[2,1]: 3 ways.
Sp(4) = [1,1,1,1];[1,1,2];[1,2,1];[2,1,1];[2,2]: 5 ways.
Sp(5) = [1,1,1,1,1];[1,1,1,2];[1,1,2,1];[1,2,1,1];[1,2,2]
[2,1,1,1];[2,1,2];[2,2,1]: 8 ways.
**Observation:** On the first day we can either spend $1, or $2. If we spend $1 then we have Sp(n-1) different ways to spend the rest, and if we spend $2 we have Sp(n-2) different ways to spend the rest.

A recursive formula for Sp(n) can be given as:

\[ \text{Sp}(n) = \text{Sp}(n-1) + \text{Sp}(n-2) \]

With the base cases:

\[ \text{Sp}(1) = 1, \text{Sp}(2) = 2 \]

Staring at the sequence of values of the Sp function:

\[ \text{Sp}(1) = 1. \]
\[ \text{Sp}(2) = 2. \]
\[ \text{Sp}(3) = 3. \]
\[ \text{Sp}(4) = 5. \]
\[ \text{Sp}(5) = 8. \]
\[ \text{Sp}(6) = 13. \]
\[ \text{Sp}(7) = 21. \]
\[ \text{Sp}(8) = 34. \]

won’t likely help in guessing a closed form expression for this recursively defined function.
The Fibonacci function
Fib(0) = 0
Fib(1) = 1
Fib(n) = F(n-1) + F(n-2) for all Natural numbers n ≥ 2.

NOTE: (Some variants start the function with Fib(1) = 1, and Fib(2) =1.)

The first few Fibonacci numbers are
Fib(0) = 0, Fib(1) = 1, Fib(2) = 1,
Fib(3) = 2, Fib(4) = 3, Fib(5) = 5,
Fib(6) = 8, Fib(7) = 13, Fib(8) = 21,
Fib(9) = 34.

The Fibonacci sequence is named after Fibonacci. His real name was Leonardo Pisano Bogollo, and he lived between 1170 and 1250 in Italy."Fibonacci" was his nickname, which roughly means "Son of Bonacci". As well as being famous for the Fibonacci Sequence, he helped spread through Europe the use of Hindu-Arabic Numerals (like our present number system 0,1,2,3,4,5,6,7,8,9) to replace Roman Numerals (I, II, III, IV, V, ...).
The Fibonacci sequence is often motivated by a story about bunnies.

Once a bunny pair reaches two months they produce another pair once a month, forever. (These bunnies never die!) The population growth is illustrated below.
Fibonacci sequences occur in nature, for more information see https://plus.maths.org/content/life-and-numbers-fibonacci.
The first few Fibonacci numbers are
Fib(0) = 0, Fib(1) = 1, Fib(2) = 1,
Fib(3) = 2, Fib(4) = 3, Fib(5) = 5,
Fib(6) = 8, Fib(7) = 13, Fib(8) = 21,
Fib(9) = 34.

Examining the first few Fibonacci numbers you could try to guess a closed form expression. However, in this case the closed form expression is somewhat hard to guess.
The closed form for the nth Fibonacci number is:

\[
F_n = \frac{\varphi^n - (-\varphi)^{-n}}{\sqrt{5}}
\]

where \( \varphi \) is a quantity known as the golden ratio with the value:

\[
\varphi = \frac{1 + \sqrt{5}}{2} \approx 1.61803 39887 \ldots
\]

\( \varphi \) is the positive solution to the equation \( x^2 - x - 1 = 0 \).
The first few Fibonacci numbers are
$\text{Fib}(0) = 0$, $\text{Fib}(1) = 1$, $\text{Fib}(2) = 1$,
$\text{Fib}(3) = 2$, $\text{Fib}(4) = 3$, $\text{Fib}(5) = 5$,
$\text{Fib}(6) = 8$, $\text{Fib}(7) = 13$, $\text{Fib}(8) = 21$,
$\text{Fib}(9) = 34$

You can find Fibonacci numbers in Pascal’s triangle. that is:

$$F_n = \sum_{k=0}^{\left\lfloor \frac{n-1}{2} \right\rfloor} \binom{n-k-1}{k}$$

where:

$$\left\lfloor \frac{n-1}{2} \right\rfloor$$
denotes $(n-1)/2$ rounded down.
Floor and Ceiling Functions

Integer values are very often used instead of more accurate fractional values, or real values. Rounding should be familiar to all of you. Here are two useful variants to rounding. (These variants can be informally thought of as rounding up or rounding down.)

Ceiling
Let $x$ be any real number. The \textit{ceiling} of $x$, denoted by

$$\lceil x \rceil$$

is the smallest integer greater than or equal to $x$.

Floor
Let $x$ be any real number. The \textit{floor} of $x$, denoted by

$$\lfloor x \rfloor$$

is the greatest integer less than or equal to $x$. 
Let’s see what happens when we sum Fibonacci numbers.

\[
\begin{align*}
0 &= 0 \\
0 + 1 &= 1 \\
0 + 1 + 1 &= 2 \\
0 + 1 + 1 + 2 &= 4 \\
0 + 1 + 1 + 2 + 3 &= 7 \\
0 + 1 + 1 + 2 + 3 + 5 &= 12
\end{align*}
\]
\[ 0 = 0 = 1 - 1 = \text{Fib}(2) - 1 \]
\[ 0 + 1 = 1 = 2 - 1 = \text{Fib}(3) - 1 \]
\[ 0 + 1 + 1 = 2 = 3 - 1 = \text{Fib}(4) - 1 \]
\[ 0 + 1 + 1 + 2 = 4 = 5 - 1 = \text{Fib}(5) - 1 \]
\[ 0 + 1 + 1 + 2 + 3 = 7 = 8 - 1 = \text{Fib}(6) - 1 \]
\[ 0 + 1 + 1 + 2 + 3 + 5 = 12 = 13 - 1 = \text{Fib}(7) - 1 \]

This is just a guess and may or may not be true. How would we prove this?
Fib(0) + Fib(1) + ... + Fib(n) = Fib(n+2) - 1, for all integers n, n ≥ 1.

**Proof:** (Mathematical Induction)

**Base:** Fib(0) = Fib(2) - 1 = 0

**Induction Hypothesis:** P(k) is true.

**Induction Step:** Prove P(k+1) using P(k).

Fib(0) + Fib(1) + ...
   + Fib(k) + Fib(k+1) = Fib(k+2) - 1 + Fib(k+1)
   = Fib(k+1) + Fib(k+2) - 1
   = Fib(k+3) - 1 □
Recall: Sp(n)  
A recursive formula for Sp(n) can be given as:  
Sp(n) = Sp(n-1) + Sp(n-2)  
With the base cases:  
Sp(1) = 1, Sp(2) = 2

We can compare the Sp and Fib functions

Sp(1) = 1; Sp(2) = 2; Sp(n) = Sp(n-1) + Sp(n-2)

Fib(2) = 1; Fib(3) = 2 Fib(n) = Fib(n-1) + Fib(n-2)

We can conclude that Sp(n) = Fib(n+1).