

# Count of Matches for a Highly Ambiguous Regular Expression

Mike French

2022-11-19

## Abstract

We evaluate the number of matches of the regular expression  $(a?)^n (a^*)^n$  for the string  $a^n$ . We show the total match count is the dot product of two vectors taken from Pascal's Triangle. A formula is given for the total count and values are calculated for  $n=1..10$ .

## Problem Statement

Let the exponential meta-operator '^' mean repetition. So 'a^4' for a string means repeating the 'a' character 4 times 'aaaa', and '(a?)^4' for a regex means '(a?a?a?a?)'. We will consider a regex of the form  $(a?)^n (a^*)^n$  matching a string of 'a^n', which is a highly ambiguous exaggeration of the example given in [Cox].

## Definitions

Consider the match counts for each operator in the regular expression:

- Optional quantifier *zero or one* '?' matches 0 or 1 characters.  
The counts for the first half of the expression ' $?^n$ ' are a sequence of  $n$  binary digits
- Star quantifier *zero or more* '\*' matches  $0..n$  characters.  
The counts for the second half ' $?^n$ ' are a sequence of  $n$  numbers in the range  $0..n$ .

Count the total number of ways to get a specific partial sum of matches  $k=0..n$  for each half of the expression. Assemble these counts into two vectors of  $n+1$  values over index  $k=0..n$ :

- $S_{?n}[k]$  ways for ' $?^n$ ' to match  $k$  characters.
- $S_{*n}[k]$  ways for ' $*^n$ ' to match  $k$  characters

For a successful match, the two counts for each half of the expression must add up to  $n$ : if the second half matches  $k$ , the first half must have matched  $n-k$ .

So the total count is the pairwise multiplication of the S vectors:

$$\text{Total count : } S_n = \sum_{k=0..n} S_{?n}[n-k] \times S_{*n}[k]$$

## ? Quantifiers

The count value  $S_{?n}[k]$  is:

- The number of ways to get  $n$  *zero-or-one* matches accepting a total of  $k$  characters.
- The count of  $n$ -digit binary numbers that have  $k$  bits set (1s).
- The number of ways of choosing  $k$  from  $n$ , which is the binomial coefficient  $nCk$ :

$$S_{?n}[k] = nCk$$

The table of binomial coefficients is just Pascal's Triangle with the recurrence relation:

$$nCk = (n-1)C(k-1) + (n-1)Ck$$

The vector of counts  $S_{?n}[k]$  is  $nCk$  for  $k=0..n$ , which is just the  $n^{th}$  diagonal in Pascal's Triangle.

The diagonal vector is symmetric because:  $nCk = nC(n-k)$  and so  $S_{?n}[k] = S_{?n}[n-k]$

which means we can invert the  $S_{?n}$  vector index and justify the dot product formulation:

$$\text{Total count: } S_n = \sum_{k=0..n} S_{?n}[k] \times S_{*n}[k] = S_{?n} \bullet S_{*n}$$

## \* Quantifiers

Change variable to  $r$ , to avoid confusion with the binomial index  $n$ . The count value  $S_{*r}[k]$  is:

- The number of ways to get  $r$  zero-or-more matches accepting a total of  $k$  characters.
- *Sum of digits* problem: the ways  $r$  numbers in the range  $0..r$  can have sum of  $k$ .

Construct a recurrence relation for the sum of digits problem. Each set of  $r-1$  numbers with a sum in the range  $0..k$  is uniquely made up to sum  $k$  by adding the  $r^{\text{th}}$  number  $r-k$ :

$$S_{*r}[k] = \sum_{k=0..r} S_{*r-1}[k] = S_{*r}[k-1] + S_{*r-1}[k]$$

Each entry is the sum of values in the row above, up to and including the same column  $k$ , and hence also the sum of the two terms to the left ( $r, k-1$ ) and above ( $r, k$ ).

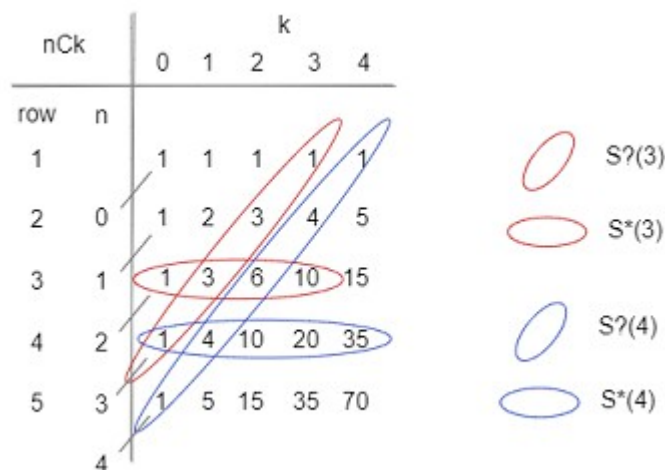
The recurrence is grounded at 1 on the left and at the top:

$$\forall_{k=0..r} S_{*1}[k] = 1, \quad \forall_{r>0} S_{*r}[0] = 1$$

This is just Pascal's Triangle accessed by  $r$ , the 1-based row number, rather than  $n$ , the 0-based diagonal. The indices are linked by  $n=r+k-1$ , so the count is the binomial coefficient:

$$S_{*r}[k] = (r+k-1)Ck$$

## Vectors in Pascal's Triangle



## Final Formula

$$\text{Total count: } S_n = S_{?n} \cdot S_{*n} = \sum_{k=0..n} nCk \times (n+k-1)Ck$$

## Specific Examples

$$\begin{aligned}
 S(1) &= [1,1] \cdot [1,1] = 1+1 = 2 \\
 S(2) &= [1,2,1] \cdot [1,2,3] = 1+4+3 = 8 \\
 S(3) &= [1,3,3,1] \cdot [1,3,6,10] = 1+9+18+10 = 38 \\
 S(4) &= [1,4,6,4,1] \cdot [1,4,10,20,35] = 1+16+60+80+35 = 192 \\
 S(5) &= [1,5,10,10,5,1] \cdot [1,5,15,35,70,126] = 1+25+150+350+350+126 = 1,002 \\
 S(6) &= [1,6,15,20,15,6,1] \cdot [1,6,21,56,126,252,462] = 1+36+315+1120+1890+1512+462 = 5,336
 \end{aligned}$$

| $n$   | 1 | 2 | 3  | 4   | 5     | 6     | 7      | 8       | 9       | 10        |
|-------|---|---|----|-----|-------|-------|--------|---------|---------|-----------|
| $S_n$ | 2 | 8 | 38 | 192 | 1,002 | 5,336 | 28,814 | 157,184 | 864,146 | 4,780,008 |

## References

[Cox] "Regular Expression Matching Can Be Simple And Fast", Russ Cox, January 2007 [\[web\]](#).