

# Re: Help with Permutations

---

*Source:* <http://sci.tech-archive.net/Archive/sci.math/2006-01/msg00893.html>

---

- *From:* "Dana DeLouis" <delouis@xxxxxxxxxxxxxx>
  - *Date:* Sun, 8 Jan 2006 02:03:05 -0500
- 

>> gives 1265

Hi. Just for feedback... using brute force with Mathematica also gives 1265 as an answer.

Here we take all Subsets, remove the null subset (with Rest) and use Union to remove duplicates.

Then do a Permutation on all subsets...

```
t = Union[Rest[Subsets[{a,c,c,c,d,d,e}]]];
```

```
Length[Flatten[Permutations/@t,1]]
```

1265

If you had different letters (ie a,b,c) then maybe this equation...

<http://www.research.att.com/cgi-bin/access.cgi/as/njas/sequences/eisA.cgi?Anum=A007526>  
(E\*n!\*Gamma[n, 1])/Gamma[n]

(15 when n equals 3, ect.)

--

Dana

"Jonas" <asdf@xxxxxxxxxxxxxx> wrote in message  
[news:43c06b71\\$0\\$163\\$edfadb0f@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](news:43c06b71$0$163$edfadb0f@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

>

> <mareg@xxxxxxxxxxxxxxxxxxxxxx> wrote in message

> [news:dpolh3\\$im4\\$1@xxxxxxxxxxxxxxxxxxxxxxxxxxxx](news:dpolh3$im4$1@xxxxxxxxxxxxxxxxxxxxxxxxxxxx)

>> In article <1136584344.300707.31850@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>,

>> "barliow" <jack.c.barlow@xxxxxxxxxx> writes:

>>>Hi all,

>>>

>>>I'm not mathematically fantastic so I hope this makes sense.

>>>

>>>I've written an algorithm to generate all permutations of words from a

>>>given alphabet.

>>>

>>>Alphabet:

>>>

>>>"abc"

## Re: Help with Permutations

```
>>>
>>>Gives permutations:
>>>
>>>a, b, c, ab, ac, ba, bc, ca, cb, abc, acb, bac, bca, cab, cba
>>>
>>>In order to keep track of the progress through the algorithm, I need to
>>>be able to calculate the total number of permutations in advance. I can
>>>do this using the following formula:
>>>
>>>Number of permutations of size k taken from n objects is:
>>>
>>> n!
>>>n_P_k = -----
>>> (n - k)!
>>>
>>>So I can calculate the total number of permutations off all lengths as:
>>>
>>>P = (3! / (3! - 1!)) + (3! / (3! - 2!)) + (3! / (3! - 3!)) = 15
>>>
>>>However, the algorithm is designed to eliminate repeated permutations
>>>that arise as a result of having repeated letters:
>>>
>>>Alphabet:
>>>
>>>"acc"
>>>
>>>Gives permutations:
>>>
>>>a, c, ac, ca, cc, acc, cac, cca
>>>
>>>This means that the formula above no longer works. I'm wondering if
>>>someone can provide a formula to calculate the total of number of
>>>permutations of *all* lengths that will be outputted in the following
>>>scenarios:
>>>
>>>"abc" (should = 15)
>>>
>>>"accd" (should = 34)
>>>
>>>"accdde" (uncertain as to what the result should be)
>>>
>>
>> I don't know a closed formula for this total number (which is not to say
>> that there isn't one!) but it is not difficult to calculate it.
>>
>> Suppose k1, k2, ..., kr are the numbers of each of the distinct letters
>> in your string. (For example, for "accdde", k1=1, k2=3, k3=k4=2.)
>> Then the total number of what you describe above as "permutations" is
>> (view this in fixed width font):
>>
>> k1 k2 kr
```

## Re: Help with Permutations

```
>> ---
>> \\ (i1 + i2 + ... + ik)!
>>> ...> ----- - 1
>> /// i1! i2! ... ik!
>> ---
>> i1=0 i2=0 ik=0
>>
>> (The -1 at the end is to avoid counting the empty string, which
>> corresponds
>> to i1=i2=...ir=0 in the sum.)
>>
>> You should be able to write a program to evaluate this expression for
>> given
>> k1, k2, ..., kr. Doing it for the example above, k1=1, k2=3, k3=k4=2
>> gives 1265, as quasi reported.
>>
>> Derek Holt.
>
> I have calculated your sum and I get 4952 or 5023 depending on if the one
> is inside the sum or outside but not 1265.
>
>
>
```

---

### • *References:*

- ◆ [\*Help with Permutations\*](#)  
◇ *From:* barliow
- ◆ [\*Re: Help with Permutations\*](#)  
◇ *From:*
- ◆ [\*Re: Help with Permutations\*](#)  
◇ *From:* Jonas

- Prev by Date: [\*internal language of a 2 category\*](#)
- Next by Date: [\*Re: Proof of the Riemann hypothesis\*](#)
- Previous by thread: [\*Re: Help with Permutations\*](#)
- Next by thread: [\*Re: Help with Permutations\*](#)
- Index(es):
  - ◆ [\*Date\*](#)
  - ◆ [\*Thread\*](#)