Doc/Articles/Play203

COLLABORATORS			
	TITLE : Doc/Articles/Play203		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		March 26, 2009	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME
10	2009-03-25 07:33:34	follow guidelines, finish code testing.	RicSherlock
9	2009-03-25 07:01:33	More work on text. Need to test code from q2 on	RicSherlock
8	2009-03-24 20:56:01		RicSherlock
7	2009-03-24 20:52:59	to q2	RicSherlock
6	2009-03-23 08:29:50	update code-block syntax	RicSherlock
5	2008-12-08 18:45:51	converted to 1.6 markup	localhost
4	2005-12-16 10:10:38	Openning {{{ likes to sit on a line of its own	OlegKobchenko
3	2005-12-15 13:08:25	format fixes	ChrisBurke
2	2005-12-15 13:03:15	format fixes	ChrisBurke

Contents

1 At Play With J Giddyap

• ***Eugene McDonnell***

The *OED* doesn't have a *giddyap* entry; the *Concise Oxford Dictionary* has a *giddap* entry; Webster 3 has an entry for *giddap, giddyap, giddyup*. I think it must be a children's word; I don't think I've ever heard it used by an adult. When I was much younger I know that when I pretended I was riding a horse, I swung my imaginary whip on my imaginary horse as I pranced about, shouting *giddyap* with every stroke of the whip. I find it to be a suitable title for this article because it concerns horseraces, and also treats of the speeding up of programs that solve a horseracing problem.

I don't recall now where it was that I found the problem, but when I ran across it, it sounded as if it might a suitable challenge for the J Forum. In any event, on September 25, 2003 I sent this message to the J Forum:

N horses enter a race. Given the possibility of ties, how many different finishes to the horse race exist? Write a program that shows all the possibilities.

By way of example: here is the solution by brute force for N=3. There are 13 solutions. Horses are named `a`, `b` and `c`. The expression `{{b,c},a}` denotes a finish in which `b` and `c` tie for first and `a` comes in next.

```
{a, b, c}, {a, c, b}, {b, a, c}, {b, c, a}, {c, b, a}, {c, a, b},
{a,{b,c}}, {{b,c},a}, {b,{a,c}}, {{a,c},b}, {c,{a,b}}, {{a,b},c}, {{a,b,c}}
```

1.1 Methods for finding how many different finishes

There were over two dozen responses over the next two weeks. The first response misunderstood the problem, and assumed that a race with three horses was the only one to consider. Since I had already given the solution of this one, it was clear that more had to be done than to submit the number 13. The answers to the first question, the number of solutions, were various. The brute force way is good only for the first few number of horses - the answer for eight horses is already 545,835. The nicest early entry used a table of Stirling subset numbers and factorials to give the number of different finishes effectively:

```
   ] fc=:!i.9
1 1 2 6 24 120 720 5040 40320
   require 'math/misc/numbers'
   ] s8=:subsets 8
1 0   0   0    0    0   0  0 0
0 1   0   0    0    0   0  0 0
0 1   1   0    0    0   0  0 0
0 1   3   1    0    0   0  0 0
0 1   7   6    1    0   0  0 0
0 1  15  25   10    1   0  0 0
0 1  31  90   65   15   1  0 0
0 1  63 301  350  140  21  1 0
0 1 127 966 1701 1050 266 28 1
   s8 +/ . * fc
1 1 3 13 75 541 4683 47293 545835
```

Here's another way that uses curtailed binomial lists and the list of terms so far found:

```
   1 +/ . * 1
1
   1 2 +/ . * 1 1
3
   1 3 3 +/ . * 1 1 3
13
   1 4 6 4 +/ . * 1 1 3 13
75
   1 5 10 10 5 +/ . * 1 1 3 13 75
541
```

Which can be done either iteratively or recursively.

Still another way to get the number of different finishes uses the Weighted Taylor coefficient adverb `t:` defined in the J Dictionary as:

- The result of `u t: k` is `(!k)*u t. k`. In other words, the coefficients produced by `t:` are the Taylor coefficients *weighted* by the factorial. As a consequence, the coefficients produced by it when applied to functions of the exponential family show simple patterns. For this reason it is sometimes called the *exponential generating function*.

The exponential generating function for the our numbers is (1/(2-e ^{n})) so we can write a function `fn` using it, modified by the Weighted Taylor adverb:

```
   fn =: (%@(2 - ^)) t:
   fn 8
545835
   fn i. 9
1 1 3 13 75 541 4683 47293 545835
```

All of these methods are discussed in Sloane's *On-Line Encyclopedia of Integer Sequences*, sequence 670.

1.2 Methods for representing all the possible finishes

The method I used for representing all 13 of the finishes for a three-horse race was informal. Various methods were used in the J solutions. This is one of the J solutions for a three-horse race:

```
+-----+-----+-----+-----+-----+-----+----+----+----+----+----+----+---+
|a b c|a c b|b a c|b c a|c a b|c b a|a bc|bc a|ab c|c ab|b ac|ac b|abc|
+-----+-----+-----+-----+-----+-----+----+----+----+----+----+----+---+
```

Results using boxed arrays, like this, were relatively slow. Faster results were obtained using a list of post positions to show the finish, with the finish order of the horse in post position `k` given as the value of item `k` of the result. Here is how the finishes of a three-horse race are displayed using this method:

```
0 0 0       abc
0 1 1       a bc
1 0 0       bc a
0 0 1       ab c
0 1 0       ac b
1 0 1       b ca
1 1 0       c ab
0 1 2       a b c
0 2 1       a c b
1 0 2       b a c
1 2 0       c a b
2 0 1       b c a
2 1 0       c b a
```

I've appended to the right of each finish the order of finish of three horses `a`, `b` and `c`, having post positions `0`, `1` and `2`, respectively. Horses tied in a finish are shown by abutting letters. For example, `a bc` shows horse `a` in first place and horses `b` and `c` tied for second place.

The function to produce the finishes in this order is `fin3`, by Roger Hui:

```
   rankings=: ,"1 0~@i. , /:"1@=@i.@>:
   ext     =: [: ,/ _1&,. {"2 1 rankings@#@~.@{.
   fin3    =: ([: ; >./"1 <@ext/. ])@$:@<: ` (i.@(1&,)) @. (1&>:)
```

I found that it was difficult to understand how these functions obtained the proper result, so if you are in the same boat, I'll try to explain them. The first part of `fin3`, `([: ; >./"1 <@ext/. ])`, is where most of the action occurs. Its argument is predecessor of the table to be produced. For example, to produce the table of order 3, one needs the table of order 2, which I'll call `q2`.

```
   ]q2=: 0 0 , 0 1 ,: 1 0
0 0
0 1
1 0
```

The left argument to `<@ext/.` is a list of the row-maxima of q2, or 0 1 1. The key adverb `/.` modifying `<@exp` uses this list to partition `q2` into as many parts as there are keys, in this case two. Thus `<@ext` is applied separately to each part of `q2`, which I'll call `q2a` and `q2b`:

```
   ]q2a=: ,: 0 0
0 0
   ]q2b=: 0 1 ,: 1 0
0 1
1 0
```

The application to `q2a`

```
   q2aa=:(_1&,.) q2a
   q2aa
_1 0 0
   q2aa{"2 1 rankings 1
0 0 0

0 1 1

1 0 0
   ,/q2aa{"2 1 rankings 1
0 0 0
0 1 1
1 0 0
   rankings # ~. {. q2a
0 0
1 0
0 1
   q2aa{"2 1 rankings # ~. {. q2a
0 0 0

0 1 1

1 0 0
   rankings 1
0 0
1 0
0 1
   q2ab=: rankings 1
   q2aa {"2 1 q2ab
0 0 0

0 1 1

1 0 0
   $q2aa
1 3
   q2ab
0 0
1 0
0 1
   q2aa
_1 0 0
   _1 0 0 { 0 0
0 0 0
```

```
   _1 0 0 { 0 1
1 0 0
   _1 0 0 { 1 0
0 1 1
```

Here is his explanation:

- To generate the finishes for `n`, `fin3` first partitions the finishes for `n-1` by the maximum ranks. Then for each partition with maximum rank `m` and each finish `v` therein, (`_1,v` is indexed into the matrix `,"1 0~i.1+m` (tieing the new competitor with each possible rank) and into the matrix `/:"1=i.2+m` (slotting the new competitor into each possible position ("no ties")).

  Therefore, if `c` is a vector of the number of finishes with maximum ranks `i.#c`, the corresponding counts for `1+#c` are: `(c*1+i.#c)` are the number of finishes for ranks `i.#c` coming from ties, and `c*2+i.#c` are the number of finishes for ranks `1+i.#c` coming from non-ties. For example, for `n`=2:

```
                          each finish indexed into
   max rank    finishes      ties      nonties
    0            0 0         0 0        1 0
                                        0 1
    1            0 1         0 1 0      1 2 0
                 1 0         0 1 1      0 2 1
                                        0 1 2
```

  There is 1 finish for max rank 0 and 2 finishes for max rank 1. The new counts are for `n`=3 are:

```
   max ranks    0     1     2
   ties         1*1   2*2
   nonties            2*1   3*2
   total        1     6     6
```

  And for `n`=4:

```
   max ranks    0     1     2     3
   ties         1*1   2*6   3*6
   nonties            2*1   3*6   4*6
   total        1     14    36    24
```

  The following functions encode the algorithm:

```
   ntie1=: 0: ,~ ] * 1&+@i.@#
   ntie0=: 0: ,  ] * 2&+@i.@#
   nfin2=: (ntie1 + ntie0)@$:@<: ` ((,1x)"0) @. (1&>:)

   nfin2 1
1
   nfin2 2
1 2
   nfin2 3
1 6 6
   nfin2 4
1 14 36 24
```

[end of Hui's explanation].

In a race where there are no ties the order of finish is a permutation. Notice that the bottom six results give the permutation table of order 3. In the rankings function you see `/:1@=@i.` This is the 'magical matrix' described in my last column (*Vector 20.2*, October 2003), and it is used in precisely the same way: to produce a table of permutations.

The results of the function `nfin2` above give the number of finishes of `n` horses having `k` as the maximum rank. If we form a triangle from these results:

```
1
1  2
1  6  6
1 14 36 24
```

and ravel it, `1 1 2 1 6 6 1 14 36 24` we get a list that is sequence 19538 in (http://www.research.att.com/~njas/sequences/). It is described as "the number of ways `n` labeled objects can be distributed into `k` nonempty parcels". I wanted to obtain a different triangle, one showing the number of finishes having each leading digit. An easy way to do this is to look at the first column of the table of order `n`, as found by `fin3`. Thus if one transposes table `fin3 n`, takes its head item, applies tally modified by the key adverb and reflexive to tally one would get the number of instances of each leading digit, in order.

```
   # /. ~ {. |: fin3 3
6 5 2
```

This says that the digits 0, 1 and 2 occur 6, 5 and 2 times as leading digits, respectively, in the table of order 3. You can verify this by inspecting the table above. Here are the results for tables of order 1 through 7:

```
   1
   2    1
   6    5    2
  26   25   18    6
 150  149  134   84   24
1082 1081 1050  870  480  120
9366 9365 9302 8700 6600 3240 720
```

I've submitted this triangle to Sloane's Online Encyclopedia. I've come across the sequence given by the row sums of the triangle above, namely `1 3 13 75 541 4683 47294` in several different contexts. In 1977 I found that it enumerates the number of different left arguments for APL's dyad transpose. In 2000 it enumerated the number of distinct basic lists, I called Blists, which mathematicians call preferential arrangements. And now, here they come galloping again to enumerate horserace finishes!