Generation of chained permutations

Pavle Guduric [pavle@guduric.info]
Fulda University of Applied Sciences

Problem

Problem is to create array of 2-dimensional permutation elements, where each successive par won't have more than one same element and which doesn't have more than two successively repeated elements with one same element.

Problem example

For given elements *A*, *B*, *C* and *D* we have next 2-dimensional permutations: *AB*, *AC*, *AD*, *BC*, *BD*, *CD*, *BA*, *CA*, *DA*, *CB*, *DB* and *DC*. Exactly 12 pairs according to the next equation:

$$p = \frac{n!}{(n-k)!} = \frac{4!}{(4-2)!} = \frac{4!}{2!} = 4 \cdot 3 = 12$$

$$P = \{ AB, AC, AD, BC, BD, CD, BA, CA, DA, CB, DB, DC \}$$

Array P can be sorted in many ways depending of wanted usage. In this case I want to create array that has next 2 characteristics: 1) the maximal possible distance between the pars that have two same elements and 2) the lowest possible successive repetition of pars that have one same element. Examples of arrays that do not satisfy 1) and 2) are given respectively: 1) Array $P_1=\{AB, BA\ldots\}$ and 2) $P_2=\{AB, AC, AD\ldots\}$

Problem solution

Let *P* be the array of permutation elements x_{ij} where *i*, *j* are indexes of elements and $i \neq j$. Now *P* can be expressed as:

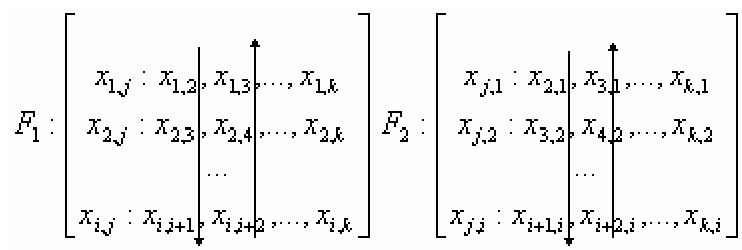
$$P = \{x_{1,2}, x_{1,3}, ..., x_{1,k,} x_{2,3}, x_{2,4}, ..., x_{2,k}, ..., x_{k-1,k}, x_{2,1}, x_{3,1}, ..., x_{k,1}, x_{3,2}, x_{4,2}, ..., x_{k,2}, ..., x_{k,k-1}\}$$

where *k* is maximal number of elements in *P*.

Array *P* can be observed through the next two groups:

$$F_1 : \begin{bmatrix} x_{1,j} : x_{1,2}, x_{1,3}, ..., x_{1,k} \\ x_{2,j} : x_{2,3}, x_{2,4}, ..., x_{2,k} \\ ... \\ x_{i,j} : x_{i,i+1}, x_{i,i+2}, ..., x_{i,k} \end{bmatrix} \quad F_2 : \begin{bmatrix} x_{j,1} : x_{2,1}, x_{3,1}, ..., x_{k,1} \\ x_{j,2} : x_{3,2}, x_{4,2}, ..., x_{k,2} \\ ... \\ x_{j,i} : x_{i+1,i}, x_{i+2,i}, ..., x_{k,i} \end{bmatrix}$$

where each next $x_{i,j} : i = 1, k$ will have one element less and $i<k$. In order to create the array *P'* that will satisfy characteristics 1) and 2), we can chain every two successive elements from different rows in the formation F. For each $x_{i,j}$ where *i* is odd number we move in the direction $x_{i,j} \rightarrow x_{i+1,j}$. For each $x_{i,j}$ where *i* is even number we move in the direction $x_{i,j} \rightarrow x_{i-1,j}$. Graphically this can be depicted as[1]:



If we follow the arrows array *P'* would have next order:

$$P' = \{x_{1,2}, x_{2,3}, ..., x_{k-1,k}, x_{k-2,k}, x_{k-3,k-1}, ..., x_{1,3}, ..., x_{1,k}, x_{2,1}, x_{3,2}, ..., x_{k,k-1}, x_{k,k-2}, x_{k-1,k-3}, ..., x_{3,1}, ..., x_{k,1}\}$$

The power of this algorithm is that only *k* iterations is needed because in each iteration we can order all elements at the same time, that belongs to the same row in forms F_1 and F_2.

Problem solution example

For our example with *A*, *B*, *C* and *D*, array *P* would be formed as follow:

P={ *AB*, *BC*, *CD*, *BD*, *AC*, *AD*, *BA*, *CB*, *DC*, *DB*, *CA*, *DA* }

[1] F_1 and F_2 are not matrices.

Implementation of algorithm

```
p=P.length*(P.length-1);  t=p/2; k=P.length-1; g=0; z=0;

for i=1 to P.length
begin
  current=P[i];
  for j=1 to P.length
  begin
    z=z+g;
    element=[current,P[j]];
    reversed element=[P[j],current];
    P'[z]=element;
    P'[z+t]=reversed element;
    g=P.length-j+i;
  end
g=0;
z=i+1;
--k;
end
```