

VÉLETLEN PERMUTÁCIÓ ELŐÁLLÍTÁSA

Az alábbi algoritmusban $X(1..N)$ tömb elemeinek egy véletlen permutációját állítjuk elő. Természetesen elvárjuk, hogy a $\text{HalmazFelsorolás}(X)$ előfeltétel teljesüljön. Talán első hallásra meglepő lesz az ötlet: induljunk ki a permutációgenerálás algoritmusának elkészítésénél a rendezésből. A következő (P1.) algoritmust pl. a minimumkiválasztásos néven ismert rendezésből eredeztetjük.

P1. Algoritmus:

Permutálás („Keverés”)	Rendezés
Ciklus $i=1$ -től $N-1$ -ig $j := \text{VéletlenSzám}(i..N)$ $\text{Csere}(X(i), X(j))$ Ciklus vége	Ciklus $i=1$ -től $N-1$ -ig $j := \text{MinIndex}(i..N)$ $\text{Csere}(X(i), X(j))$ Ciklus vége

Jelölések:

X – sorozat; X' – az X sorozat algoritmus végrehajtás utáni értéke

$P(X'_k = X_j)$ – annak a valószínűsége, hogy az algoritmusbeli ciklus után az $X'_k = X_j$ lesz.

$i, j \in \mathbb{N}, \alpha \in [0, 1] \subset \mathbb{R}$

$\text{VéletlenSzám}(i..j)$ – (egyenletesen) véletlen egész szám $\in [i, j]$.

VéletlenSzám – (egyenletesen) véletlen valós szám $\in [0, 1] \subset \mathbb{R}$.

Feltevések:

F1. $P(\text{VéletlenSzám}(i..j) = k) = P(\text{VéletlenSzám}(i..j) = m) \quad k, m = i..j$

F2. $P(\text{VéletlenSzám} < \alpha) = \alpha$

Lemma:

$P(\text{VéletlenSzám}(i..j) = k) = 1 / (j - i + 1) \quad k = i..j$

Bizonyítás:

Az i és j között $j - i + 1$ féle érték van, tehát ezek közül valamelyik kijövetelének valószínűsége biztos esemény (**1**); és $P(\mathbf{1}) = 1$; továbbá egymást kizáró események

\Rightarrow

$$\sum_{k=i}^j P(\text{VéletlenSzám}(i..j) = k) = 1.$$

Az 1. feltevés miatt ez a $j - i + 1$ esemény egyenletesen osztozik az 1 valószínűségen.

Ebből már adódik az állítás.

Qed.

Például:

$P(\text{VéletlenSzám}(1..N) = k) = 1 / (N - 1 + 1) = 1 / N$

P1-1. Állítás:

A P1. algoritmus a bemeneti sorozat egy permutációját állítja elő.

Bizonyítás:

1. a kimeneti állapot hossza megegyezik a bemenetiével,
Mivel az X sorozat változása csak elemeinek helycseréjét jelenti, ezért
 2. új elem az eredményssorozatba nem kerül,
 3. minden eleme a sorozatban marad.
- $\Rightarrow \forall i \in [1..N] : X'_i \in X$

Qed.**P1-2. Állítás:**

Az X' minden elemében az X bármely eleme azonos eséllyel lesz a P1. algoritmus végrehajtása után: $P(X'_i = X_k) = 1/N \quad \forall i, k \in [1..N]$. (Előfeltétel: $X_i \neq X_j \quad i \neq j$, vagyis HalmazFelsorolás(X) = Igaz.)

Bizonyítás:

Indukcióval bizonyítunk: belátjuk, hogy

$$(i=1) \quad P(X'_1 = X_k) = 1/N \quad \forall k \in [1..N]$$

(azaz bármelyik elem azonos valószínűséggel kerülhet az első helyre)

$$(i \Rightarrow i+1) \quad P(X'_{i+1} = X_k) = 1/N \quad \forall k \in [1..N]$$

(azaz bármelyik elem azonos valószínűséggel kerülhet az $i+1$. helyre)

feltéve, hogy

$$(a) \quad P(X'_j = X_m) = 1/N \quad \forall j \in [1..i], m \in [1..N]$$

(azaz bármelyik elem azonos valószínűséggel kerülhet az i -re vagy az előtti helyre) és

$$(b) \quad X' \in \text{Perm}(X) \quad \heartsuit \quad (X' \text{ a ciklus egyszeri lefutása után } X\text{-et jelöli})$$

($i=1$)

A csere juttathat elemet valahova, pl. X'_1 -be X_k -t, –az algoritmus szerint– a ciklus $i=1$ esetében (máskor nem!), amikor a VéletlenSzám($1..N$) éppen k -t eredményez. Ennek ($P(X'_1 = X_k)$) a lemma szerint $1/N$ az esélye

($i \Rightarrow i+1$)

Indukciós feltételként feltesszük (a)-t.

A továbblépés csak akkor értelmes, ha teljesül (b). Ez viszont az algoritmusból azért következik, mert az elemek legfeljebb helyet tudnak cserélni, de egyéb módon megváltozni nem, azaz az elemek permutálódhatnak csupán.

Meghatározzuk annak valószínűségét, hogy egy elem az $i+1$. helyre kerüljön. Az alábbi két esemény fennállásának valószínűségét kell meghatározni:

\heartsuit Perm = Permutációhalmaz

- nem került az 1..i helyek egyikére sem ($=1-i/N$) és
 - a „maradékok” ($N-i$) közül épp őt sorsolta ki cserére ($=1/(N-i)$)
- Vagyis $(1-i/N) * 1/(N-i) = (N-i)/N * 1/(N-i) = 1/N$

Qed.

P1-3. Állítás:

A P1. algoritmus azonos eséllyel állítja elő az $X_1..N$ elemek tetszőleges permutációját.

(Előfeltétel: $X_i \neq X_j \quad i \neq j$, vagyis HalmazFelsorolás(X)=Igaz.)

Bizonyítás:

Belátjuk, hogy $P(X' = (X_{i_1}, \dots, X_{i_N})) = 1/N!$.

$$P(X' = (X_{i_1}, \dots, X_{i_N})) = P(X'_1 = X_{i_1}) * \\ P(X'_2 = X_{i_2} \mid X'_1 = X_{i_1}) * \dots * \\ P(X'_N = X_{i_N} \mid X'_1 = X_{i_1} \wedge \dots \wedge X'_{N-1} = X_{i_{N-1}})$$

P1-1. \Rightarrow

$$P(X'_1 = X_{i_1}) = 1/N$$

[P1-1.](#) & [P1-2.](#) & [F1.](#) \Rightarrow

$$P(X'_2 = X_{i_2} \mid X'_1 = X_{i_1}) = 1/(N-1)$$

(hiszen csak $N-1$ közül, azonos valószínűséggel választható a 2.)

...

[P1-1.](#) & [P1-2.](#) & [F1.](#) \Rightarrow

$$P(X'_N = X_{i_N} \mid X'_1 = X_{i_1} \wedge \dots \wedge X'_{N-1} = X_{i_{N-1}}) = 1/(N - (N-1)) = 1/1$$

(hiszen csak 1 közül választható az N .)

Végülis:

$$P(X' = (X_{i_1}, \dots, X_{i_N})) = P(X'_1 = X_{i_1}) = 1/N * 1/(N-1) * \dots * 1/1 = 1/N!$$

Qed.

Az elmélet után jöjjön egy csipet gyakorlat! Győződjünk meg arról, hogy a fentiekben belátott „egyenletesség” a permutációk terében, mennyire és milyen számú permutációgenerálás után mutatható ki.

Írjunk programot, amely a [P1.](#) eljárással előállít számos permutációt, s közben számolja, hogy az éppen generált permutáció hányadszorra jött ki. Azt várjuk, hogy nagyjából minden permutációra ez kb. azonos érték lesz.

Mivel a permutációk száma $N!$, ezért

1. ahhoz, hogy elvileg minden permutáció létrejöhessen: legalább $N!$ -szor kell az eljárást meghívni;
2. a számláló tömbnek is $N!$ eleműnek kell lennie (a legszorosabban számolva[♦]). Pl. $8! = 40320$, ami már a Turbo Pascal memóriában tartott tömbjének a méretét meghaladja.

[♦] Pirossal emeltük ki a cserében résztvevő elemeket.

Segíthetünk ez utóbbi problémás helyzetben, ha építünk a [P1-2. állításra](#). Ugyanis e szerint „elegendő” azt számlálni, hogy az egyes X' -beli elemek gyanánt hány ízben fordultak elő az eredeti X -beli elemek. Vagyis egy $N \times N$ -es mátrix is elég a számláláshoz, ha N az X hossza.

Permutációk

	1	2	3	4	5	6	7
1:	1452	1467	1433	1472	1463	1388	1405
2:	1480	1424	1442	1400	1450	1439	1445
3:	1446	1424	1397	1422	1490	1412	1489
4:	1375	1408	1399	1445	1441	1521	1491
5:	1454	1454	1447	1448	1406	1465	1406
6:	1496	1415	1475	1447	1399	1444	1404
7:	1377	1488	1487	1446	1431	1411	1440
Szumm:	10080	10080	10080	10080	10080	10080	10080

1. ábra. A program eredményének magyarázata.

A 7 (N) elemű 7! számú permutációt a P1. algoritmussal generálva azt tapasztaltuk, hogy a generált permutációkban 1. helyen az 1 (X_1) 1452-szor, ... a 7. helyen ugyanő 1377-szer fordult elő, ... a 7 (X_7) az 1. helyen 1405-szor, ... a 7. helyen ugyanő 1440-szer. Ellenőrizhető, hogy mind a sor-, mind az oszlopösszegek 10080 ($=2*N!$), aminek az az oka, hogy –biztos, ami biztos– ennyi ($2*N!$) darab véletlen permutációt generáltunk a programmal.

A kiértékelést vezérlő algoritmus nagyjából az alábbi lehet:

```

Procedure PermutacioVezeres;
Var
    i:LongInt;
Begin
    PermInic;
    For i:=1 to Fakt(N) {esetleg 2*Fakt(N)} do
        Begin
            VeletlenPermutacio;
            PermErtekeles;
        End;
    GyakorisagMegjelenites('Permutációk');
End;

```

A VéletlenPermutáció eljárása állítja elő a globálisan elérhető tömbben (legyen ez $p:TPermutacio$) az új permutációt:

```

Procedure VeletlenPermutacio;
Var
    i,j:integer;
    Procedure Csere(Var x,y:byte);
        Var
            s:integer;
        Begin
            s:=x; x:=y; y:=s
        End;
    Begin
        For i:=1 to N-1 do
            Begin
                j:=Random(N-i+1)+i; Csere(p[i],p[j])
            End;
    End;

```

A PermErtekeles a gyakoriságokat számláló mátrix ($gy:TGyakorisag$) megfelelő elemét növeli:

^{*} Ugyanis e mögött az a feltevés húzódik, hogy minden egyes permutációhoz kölcsönösen egyértelműen tudunk a (számláló tömb-beli) indexértéket rendelni. Ha belegondolunk, ez nem is könnyű feladat!

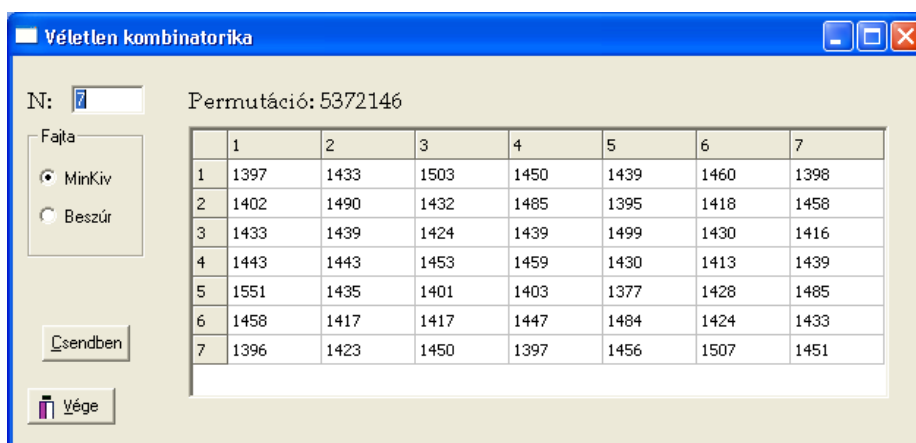
```

Procedure PermErtekeles;
  Var
    i:Byte;
  Begin
    For i:=1 to N do Inc(gy[p[i],i]);
  End;

```

Az egészet lásd egyben: [VELKOMB.PAS](#), illetve működés közben: [VELKOMB.EXE](#).

Ugyanennek egy barátságosabb, vizuális (Lazarus) környezetbeli változatának eredményét mutatja a 2. ábra. (Kipróbálhatja: [veletlenkombinatorika.exe](#).)



2. ábra. A Lazarus alkalmazás eredménye.

A következő (P2.) algoritmus beszűrőrendezés alapgondolata alapján készült.

P2. Algoritmus:

Permutálás („Keverés”)	Rendezés
Ciklus i=2-től N-ig j:=VéletlenSzám(1..i) y:=X(i) JobbraLéptet(X(j..i-1)) X(j):=y Ciklus vége	Ciklus i=2-től N-ig j:=RendezettHelye(1..i) y:=X(i) JobbraLéptet(X(j..i-1)) X(j):=y Ciklus vége

P2-1. Állítás:

A P2. algoritmus a bemeneti sorozat egy permutációját állítja elő.

Bizonyítás:

A P1-1. bizonyításához hasonlóan.

Qed.

P2-2. Állítás:

Az X' minden elemében az X bármely eleme azonos eséllyel lesz a P2. algoritmus végrehajtása után: $P(X'_i = X_k) = 1/N \quad \forall i, k \in [1..N]$. (Előfeltétel: $X_i \neq X_j \quad i \neq j$, vagyis HalmazFelsorolás(X) = Igaz.)

Bizonyítás:

Belátandó: $P(X'_i = X_k) = 1/N \quad \forall i, k \in [1..N] !$

... hf. ...

Qed.

P2-3. Állítás:

A P2. algoritmus azonos eséllyel állítja elő az $X_1..N$ elemek tetszőleges permutációját.

(Előfeltétel: $X_i \neq X_j$ $i \neq j$, vagyis HalmazFelsorolás(X)=Igaz.)

Bizonyítás:

A P2-1. és a P2-2. állítások nyilvánvaló következménye.

Qed.

Ismét a gyakorlat következik: írjunk programot, amely a P2. eljárással előállít számos permutációt, s eközben az [előző programhoz hasonlóan adminisztrálja](#) az elemek előfordulásszámát.

	1	2	3	4	5	6	7
1	1407	1473	1463	1441	1400	1446	1450
2	1468	1407	1437	1414	1496	1440	1418
3	1436	1443	1417	1453	1423	1433	1475
4	1436	1437	1417	1445	1417	1434	1494
5	1461	1473	1456	1403	1425	1458	1404
6	1455	1411	1433	1553	1449	1430	1349
7	1417	1436	1457	1371	1470	1439	1490

3. ábra. A P2. algoritmust megvalósító alkalmazás eredménye.

VÉLETLEN KOMBINÁCIÓ ELŐÁLLÍTÁSA

Az alábbi algoritmus segítségével az $1, 2, \dots, N$ számok K -ad rendű véletlen kombinációit tudjuk előállítani. (Ez nyilvánvalóan nem jelent megszorítást, hiszen a feltehető halmaztulajdonság miatt egy-egy kapcsolatban áll az elem és indexe.)

K1. Algoritmus:

```

Ciklus  $i=1$ -től  $K$ -ig
     $Y(i) := i$ 
Ciklus vége
Ciklus  $i=K+1$ -től  $N$ -ig
    Ha VéletlenSzám  $< K/i$  akkor  $j := \text{VéletlenSzám}(1..K)$ ;  $Y(j) := i$ 
Ciklus vége

```

K1-1. Állítás:

A K1. algoritmus az $\{1..N\}$ számok egy K -ad rendű kombinációját eredményezi.

Bizonyítás:

... hf. ...

(a P1-1. mintája alapján könnyű)

Qed.

K1-2. Állítás:

A K1. algoritmus végrehajtása utáni X' -re igaz, hogy X' bármely eleme helyén x bármely eleme azonos eséllyel (K/N) előfordulhat.

Bizonyítás:

Mivel N elem van, és ebből K -nak kell előfordulni X' -ben, azonos eséllyel, ezért tetszőleges elemre az előfordulás valószínűségének K/N -nek kell lennie.

Két esetet kell vizsgálni:

1. $i \leq K$,
2. $i > K$.

Azért érdekes a szétválasztás, mert az első esetbeli i -k az inicializálás során, *1 valószínűséggel* kerülnek az eredmény sorozatba, így azt kell belátnunk, hogy K/N valószínűséggel benn is tud maradni egy kiválasztott ilyen i . Ezzel szemben a 2. esetben nemcsak a bennmaradással, hanem a bekerüléssel is foglalkozni kell. Tehát:

1. A $K+1$. lépésben (bármely) j . (azaz a j értékű) elem bennmaradásának valószínűsége:

$$\begin{aligned}
 & (1 - K / (K + 1)) \quad [\text{azaz a következő adat nem „akarta” kilökní}] \\
 & + K / (K+1) * (K-1) / K \quad [\text{azaz a következő adat „akarta” ugyan, de}
 \end{aligned}$$

nem sikerült]

$$= (K+1-K) / (K+1) + (K-1) / (K+1) = \\ = K / (K+1)$$

Indukciós feltétel: az L . ($>K$) lépésben a bennmaradás esélye $= K / L$.

Indukciós lépés: az $L+1$. lépésben ...:

$$K / L \text{ [az } L. \text{ lépésig bennmaradt]} \\ * ((1 - K / (L+1)) + K / (L+1) * (K-1) / K) = \\ = K / L * ((L+1-K) / (L+1) + K * (K-1) / ((L+1) * K)) = \\ = K / L * ((K * L + K - K * K + K * K - K) / (K * (L+1))) = \\ = K / L * (K * L) / (K * (L+1)) = K / L * L / (L+1) = K / (L+1)$$

Innen már következik, hogy az N . lépésben a bennmaradás valószínűsége: K / N .

Azaz $\forall i \leq K$: a keresett valószínűség K/N .

2. Az i . lépésben az i bekerülése és az $i+1$. lépésben bennmaradás valószínűsége:

$$K / i \text{ [bekerülés valószínűsége]} \\ * (1 - 1 / (i+1)) \text{ [az adott } j \text{ elem nem módosulásának valószínűsége]}^* \\ = K / i * (i+1-1) / (i+1) = K / (i+1)$$

Indukciós feltétel: az „ i . lépésben bekerül i , s az $i+L$. lépésig a benn is marad” esélye $= K / (i+L)$.

Indukciós lépés: az $i+L+1$. lépésben ...:

$$K / (i+L) \text{ [L.-ig bennvan]} \\ * (1 - i / (i+L+1)) \text{ [„békén hagyták”]} \\ + K / (i+L+1) * (K-1) / K \text{ [megpróbálták kilökní, de nem sikerült]} \\ = K / (i+L) * ((i+L+1) * K - K * K - K * K - K) / ((i+L+1) * K) = \\ \dots \\ = K / (i+L+1)$$

Innen már látszik, hogy az N -nel jelölt utolsó lépésben a bennmaradás valószínűsége: K / N .

Azaz $\forall i > K$: a keresett valószínűség K/N .

Qed.

* $1 - K / (i+1)$ [semmi sem kerül be] + $K / (i+1) * (K-1) / K$ [i+1 bekerül, de nem oda] = $= 1 + (-K + K-1) / (i+1) = 1 - 1 / (i+1)$

K1-3. Állítás:

A K1. algoritmus azonos eséllyel állítja elő az $\{1 \dots N\}$ számok tetszőleges K -ad rendű kombinációját.

Bizonyítás:

A K1-1. és K1-2. állítások nyilvánvaló következménye.

Qed.

Az elmélet után jöjjön ismét a gyakorlat! Győződjünk meg arról, hogy a fentiekben belátott „egyenletesség” a kombinációk terében, mennyire és milyen számú kombinációgenerálás után mutatható ki.

Írjunk programot (vagy folytassuk az előbbit), amely a K1. eljárással előállít számos kombinációt, s közben számolja, hogy az éppen generált hányadszorra jött ki. Azt várjuk, hogy minden kombinációra ez kb. azonos érték lesz.

Bár a kombinációk száma jóval alatta marad a faktoriálisnak, ennek ellenére alkalmazzuk a P1. megvalósításánál ecsetelt számláló módszert! Itt a K1-2. állításra építhetünk. Ugyanis e szerint elegendő azt számlálni, hogy az egyes X' -beli elemek gyanánt hány ízben fordultak elő az eredeti X -beli elemek. Vagyis egy $N \times K$ -as mátrix is elég a számláláshoz, ha N az X hossza, K a kombináció rendje.

...

Kombinációk				
	1	2	3	4
1:	2978	2921	2836	2854
2:	2858	2857	2887	2867
3:	2786	2872	2860	2877
4:	2895	2848	2850	2868
5:	2835	2965	2868	2942
6:	2951	2815	2974	2847
7:	2857	2882	2885	2905
Szumm:	20160	20160	20160	20160

4. ábra. A program eredménye ($N=7$, $K=4$).