# Še o generiranju permutacij

↓↓↓
Aleksander Vesel

→

## Uvod

*Permutacija* je bijektivna preslikava končne množice $A$ nase. Predstavimo jo lahko kot razporeditev elementov množice v neko zaporedje. Brez izgube splošnosti lahko pri tem predpostavimo, da množico $A$ sestavlja prvih $n$ naravnih števil oziroma $A = \{1, 2, \ldots, n\}$. Zanimajo nas torej urejene izbire vseh elementov iz $A$, pri čemer ponavljanje elementov ni dovoljeno. Permutacijo množice $A$ z $n$ elementi bomo označili kot zaporedje $a = (a_1, a_2, \ldots, a_n)$, kjer $a_i \in \{1, 2, \ldots, n\}$ in $a_i \neq a_j$, če $i \neq j$, za vse $i, j \in \{1, 2, \ldots, n\}$.

Znano je, da je število permutacij v množici z $n$ elementi enako $n! = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1$, pri čemer z zapisom $n!$ označimo *fakulteto* naravnega števila $n$. Opazimo lahko, da število permutacij glede na število elementov v množici zelo hitro narašča. Če je $n = 20$, je število permutacij množice $\{1, 2, \ldots, n\}$ enako $20! = 2432902008176640000$.

V Preseku [2] so nas že zanimali algoritmi za konstruiranje vseh permutacij množice z $n$ elementi. Spomnimo, da je včasih zaželeno, da algoritem vrne urejeno zaporedje permutacij. Pri tem običajno mislimo na *leksikografsko urejenost*. Permutacija $a = (a_1, a_2, \ldots, a_n)$ je manjša od permutacije $b = (b_1, b_2, \ldots, b_n)$ glede na leksikografsko urejenost, če in samo če je $a_j < b_j$ za najmanjši $j$, v katerem se $a$ in $b$ razlikujeta. Permutacija $a = (4, 2, 3, 1, 5)$ je tako manjša od permutacije $b = (4, 2, 3, 5, 1)$, saj se, gledano od leve proti desni, prvič razlikujeta v četrtem elementu in je $a_4 < b_4$. Najmanjša permutacija množice z $n$ elementi v leksikografski ureditvi je permutacija $(1, 2, \ldots, n)$, največja pa $(n, n-1, \ldots, 1)$. Vse permutacije lahko leksikografsko razporedimo od najmanjše do največje in oštevilčimo od 0 do $n! - 1$.

Zaradi hitrega naraščanja števila permutacij generiranja vseh permutacij večje množice v praksi ne moremo izvesti, saj je permutacij hitro zelo veliko in njihovo generiranje traja preveč časa. Pri reševanju nekaterih problemov se zato zadovoljimo s tem, da konstruiramo samo izbrane permutacije, pogosto pa nas zanima tudi konstrukcija zaporedja naključno izbranih permutacij.

Na spletni strani `projecteuler.net/` so zbrani nekateri zanimivi matematično računalniški problemi v okviru Projekta Euler. Problem 24 je zastavljen takole: Poišči milijonto leksikografsko permutacijo množice $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Povedano drugače, poišči permutacijo množice $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ z zaporednim številom 999999 glede na leksikografsko ureditev.

V tem prispevku bomo opisali način, kako konstruiramo permutacijo z zaporednim številom $i$ glede na leksikografsko ureditev ter algoritem, ki vrne naključno izbrano permutacijo. Za majhne množice bi lahko za ta namen uporabili tudi algoritem, ki generira leksikografsko zaporedje vseh permutacij ter potem vrne $i$-to permutacijo, kjer je $i$ vhodni podatek ali naključno generirana vrednost med 0 in $n! - 1$. Opisana rešitev pa je prostorsko zelo potratna in za večje množice neuporabna, saj število permutacij, ki jih moramo straniti, hitro preseže velikost delovnega pomnilnika v računalniku.

## Konstruiranje $i$-te permutacije v leksikografski ureditvi

Kot smo pokazali v uvodnem poglavju, lahko permutacije glede na leksikografsko ureditev razvrstimo od najmanjše do največje oziroma jih oštevilčimo z zaporednimi števili od 0 do $n! - 1$. Konstrukcija $i$-te permutacije je osnovana na posebnem številskem sistemu, kjer posamezne števke zmnožimo s fakultetami števil od 1 do $n - 1$ in ga zato imenujemo *faktorialni številski sistem*. V faktorialnem številskem sistemu je vsako celo število $i$ iz množice $\{0, 1, \ldots, n! - 1\}$ na enoličen način predstavljeno kot

- $i = x_1(n-1)! + x_2(n-2)! + \ldots + x_{n-1}1! + x_n 0!$,

kjer za $j \in \{1, \ldots, n\}$ velja, da $x_j \in \{0, 1, \ldots, n-j\}$.

→

Ker je $x_n = 0$, lahko zadnji člen tudi izpustimo in dobimo

- $i = x_1(n-1)! + x_2(n-2)! + \ldots + x_{n-1}1!$.

Kot primer predstavimo število 81 v faktorialnem številskem sistemu. Pretvorbo lahko opravimo na podoben način kot pretvorbo iz desetiškega v dvojiški številski sistem. Namesto zaporednih delitev s številom 2 v tem primeru po vrsti delimo z zaporednimi naravnimi števili $1, 2, 3, \ldots$, tako da po deljenju celoštevilski količnik postane deljenec na naslednjem koraku, ostanek pa predstavlja števko faktorialnega številskega sistema $x_j$, za vsak $j$ med $n$ in 1:

- $81 = 81 \cdot 1 + 0, \quad x_5 = 0$
  $81 = 40 \cdot 2 + 1, \quad x_4 = 1$
  $40 = 13 \cdot 3 + 1, \quad x_3 = 1$
  $13 = 3 \cdot 4 + 1, \quad x_2 = 1$
  $3 = 0 \cdot 5 + 3, \quad x_1 = 3$

Preverimo, da velja

- $81 = 3 \cdot 4! + 1 \cdot 3! + 1 \cdot 2! + 1 \cdot 1! + 0 \cdot 0! =$
  $3 \cdot 24 + 1 \cdot 6 + 1 \cdot 2 + 1 \cdot 1.$

Algoritem, ki pretvori naravno število v števke faktorialnega številskega sistema $x_1, x_2, \ldots, x_n$, prepuščamo za vajo bralcu.

Najmanjši permutaciji $(1, 2, \ldots, n)$ v leksikografski ureditvi ustreza število 0, ki je v opisanem številskem sistemu predstavljeno s števkami $x_1 = x_2 = \ldots = x_n = 0$. Značilnost najmanjše permutacije je, da je na $j$-tem mestu permutacije vedno število $j$ oziroma $a_j = j$. To tudi pomeni, da desno od števila $a_j$ ni nobenega števila, ki bi bilo manjše od $a_j$. V splošnem za poljubno permutacijo $(a_1, \ldots, a_n)$ velja, da je pripadajoča števka faktorialnega številskega sistema $x_j$ enaka številu elementov manjših od $a_j$, ki so v permutaciji desno od števila $a_j$.

Ker je v permutaciji $(4, 2, 3, 1, 5)$ element 4 na prvem mestu, so desno od njega trije manjši elementi in dobimo $x_1 = 3$. Na podoben način dobimo še druge vrednosti $x_j$, ki so prikazane v levem delu tabele 1. Na podlagi števk $x_j$ lahko izračunamo zaporedno število permutacije $(4, 2, 3, 1, 5)$:

- $3 \cdot 4! + 1 \cdot 3! + 1 \cdot 2! + 0 \cdot 1! + 0 \cdot 0! = 80.$

Poskusimo še ugotoviti, katera je permutacija množice $\{1, 2, 3, 4, 5\}$ z zaporednim številom 81. Kot smo že izračunali, velja $x_5 = 0$, $x_4 = x_3 = x_2 = 1$ in $x_1 = 3$. Ker je $x_1 = 3$, morajo biti desno od $a_1$ tri števila manjša od $a_1$, iz česar sledi, da je $a_1$ četrti najmanjši element množice $\{1, 2, 3, 4, 5\}$ oziroma $a_1 = 4$. Podobno, ker je $x_2 = 1$, je $a_2$ drugi najmanjši element množice $\{1, 2, 3, 5\}$, torej $a_2 = 2$. Zaradi $x_3 = x_4 = 1$ dobimo $a_3 = 3$ in $a_4 = 5$. Ko vstavimo v permutacijo prve štiri vrednosti, ostane množica z elementom 1, zato je $a_5 = 1$.

Vrednosti $x_1, x_2, x_3, x_4, x_5$ za permutaciji $(4, 2, 3, 1, 5)$ in $(4, 2, 3, 5, 1)$ so prikazane v tabeli 1.

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 1 | 1 | 0 | 0 | 3 | 1 | 1 | 1 | 0 |

**TABELA 1.**
Vrednosti $x_j$ za permutacijo $(4, 2, 3, 1, 5)$ (levo) in $(4, 2, 3, 5, 1)$ (desno)

Na podlagi zapisanega bi lahko zapisali algoritem, ki najprej pretvori vhodni podatek $i$ v števke faktorialnega številskega sistema $x_1, x_2, \ldots, x_n$ in nato poišče pripadajočo permutacijo. Algoritem lahko poenostavimo, saj se izkaže, da ni potrebno shranjevati števk faktorialnega številskega sistema. V ta namen je potrebno pretvorbo v faktorialni številski sistem spremeniti tako, da izračun poteka v obratni smeri: od prve števke $x_1$ do zadnje $x_n$.

Postopek spet pojasnimo za primer $i = 81$. Najprej 81 delimo s 4!. Dobimo količnik 3, ki je enak vrednosti $x_1$. Na naslednjem koraku ostanek pri prejšnjem deljenju delimo s 3!, da dobimo $x_2$. Nato postopek nadaljujemo še za $x_3$ in $x_4$. Kot vemo, vedno velja $x_5 = 0$, zato lahko zadnji izračun izpustimo:

- $81 = 3 \cdot 4! + 9, \quad x_1 = 3$
  $9 = 1 \cdot 3! + 3, \quad x_2 = 1$
  $3 = 1 \cdot 2! + 1, \quad x_3 = 1$
  $1 = 1 \cdot 1! + 0, \quad x_4 = 1$

Algoritem Vrni permutacijo (Algoritem 1) vrne $i$-to najmanjšo permutacijo množice $\{1, 2, \ldots, n\}$ glede na leksikografsko ureditev. V zanki algoritem za vrednost spremenljivke $j$ med 1 in $n - 1$ izračuna element permutacije $a_j$. Pri tem si pomaga z množico

$E$, v kateri imamo elemente množice $\{1, 2, \ldots, n\}$, ki še niso bili uporabljeni za konstrukcijo permutacije. Na začetku seveda velja $E = \{1, 2, \ldots, n\}$. Algoritem spremenljivki $x$ v $j$-ti ponovitvi zanke priredi vrednost $x_j$. Kot smo že pojasnili, je $a_j$ enak $(x_j + 1)$. neporabljenemu elementu množice $\{1, 2, \ldots, n\}$, kar ustreza $(x_j+1)$. elementu množice $E$, saj po vstavitvi v permutacijo element $a_j$ odstranimo iz $E$.

**Algoritem 1:** Vrni permutacijo

```
1 Vhod: Naravno število n in nenegativno celo število i
2 Izhod: (a_1, a_2, ..., a_n), i-ta permutacija množice {1, 2, ..., n} glede na leksikografsko ureditev
3 E := {1, 2, ..., n};
4 for j := 1 to n − 1 do
5     x := ⌊ i / (n−j)! ⌋;
6     i := i mod (n − j)!;
7     a_j := (x + 1). najmanjši element iz E;
8     E := E \ {a_j};
9 a_n := element iz E;
```

Če želimo izračunati permutacijo množice, ki ni enaka $\{1, 2, \ldots, n\}$, je potrebno ustrezno spremeniti začetno vrednost spremenljivke $E$. Za rešitev problema 24 bi tako določili $E := \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ ter poklicali algoritem Vrni permutacijo za vhodna podatka $i = 999999$ in $n = 10$.

## Konstruiranje naključno izbrane permutacije

V tem razdelku bomo pojasnili, kako zapišemo algoritem, ki vrne naključno permutacijo množice $\{1, 2, \ldots, n\}$. Glede na prejšnji razdelek se enostavna rešitev ponuja kar sama: na naključen način izberemo število $i$ med 0 in $n! - 1$ ter nato uporabimo algoritem Vrni permutacijo (Algoritem 1), ki poišče $i$-to permutacijo v leksikografski ureditvi.

Opisana rešitev je sicer res enostavna, a je primerna le za manjše množice. Na začetku prispevka smo zapisali vrednost 20!, ki je število s kar 20 števkami, kar že presega največje celo število, ki ga imamo običajno na voljo v programskem jeziku. Potrebno je torej poiskati algoritem, ki bo deloval tudi za množice z večjim številom elementov.

Poglejmo najprej, kako bi lahko skonstruirali permutacijo množice $\{1, 2, \ldots, n\}$.

Če začnemo od desne proti levi, velja:

- $a_n$ lahko izberemo na $n$ načinov (izbiramo iz $\{1, 2, \ldots, n\}$);
- $a_{n-1}$ lahko izberemo na $n - 1$ načinov (izbiramo iz $\{1, 2, \ldots, n\} \setminus \{a_n\}$);
- $a_{n-2}$ lahko izberemo na $n - 2$ načinov (izbiramo iz $\{1, 2, \ldots, n\} \setminus \{a_n, a_{n-1}\}$);
- : ;
- $a_2$ lahko izberemo na 2 načina.

Zgornjo konstrukcijo uporabimo v algoritmu naključna permutacija (Algoritem 2). Algoritem vrne naključno izbrano permutacijo, ki jo zgradi iz najmanjše permutacije leksikografske ureditve $a = (1, 2, \ldots, n)$, določene v prvi zanki algoritma. V drugi zanki spremenljivka $j$ teče od $n$ do 2. Na $j$-tem koraku zanke algoritem v spremenljivko $i$ shrani naključno število iz množice $\{1, 2, \ldots, j\}$ in nato zamenja $i$-ti in $j$-ti element permutacije $a$.

**Algoritem 2:** naključna permutacija

```
1 Vhod: Naravno število n
2 Izhod: naključno izbrana permutacija (a_1, ..., a_n) množice {1, 2, ..., n}
3 for j := 1 to n do
4     a_j := j
5 for j := n downto 2 do
6     i := naključno celo število med 1 in j;
7     zamenjaj(a_i, a_j);
```

| $j$ | $i$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ |
|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 |
| 5 | 1 | 5 | 2 | 3 | 4 | 1 |
| 4 | 1 | 4 | 2 | 3 | 5 | 1 |
| 3 | 3 | 4 | 2 | 3 | 5 | 1 |
| 2 | 2 | 4 | 2 | 3 | 5 | 1 |

**TABELA 2.**
Delovanje algoritma naključna permutacija za $n = 5$

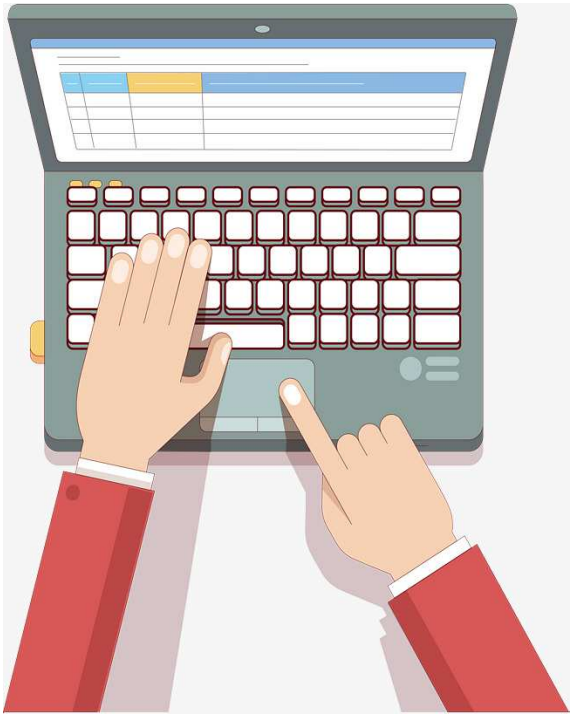
V tabeli 2 je prikazano delovanje algoritma naključna permutacija za $n = 5$, če so spremenljivki $i$ prirejene zaporedne naključne vrednosti 1, 1, 3 in 2.

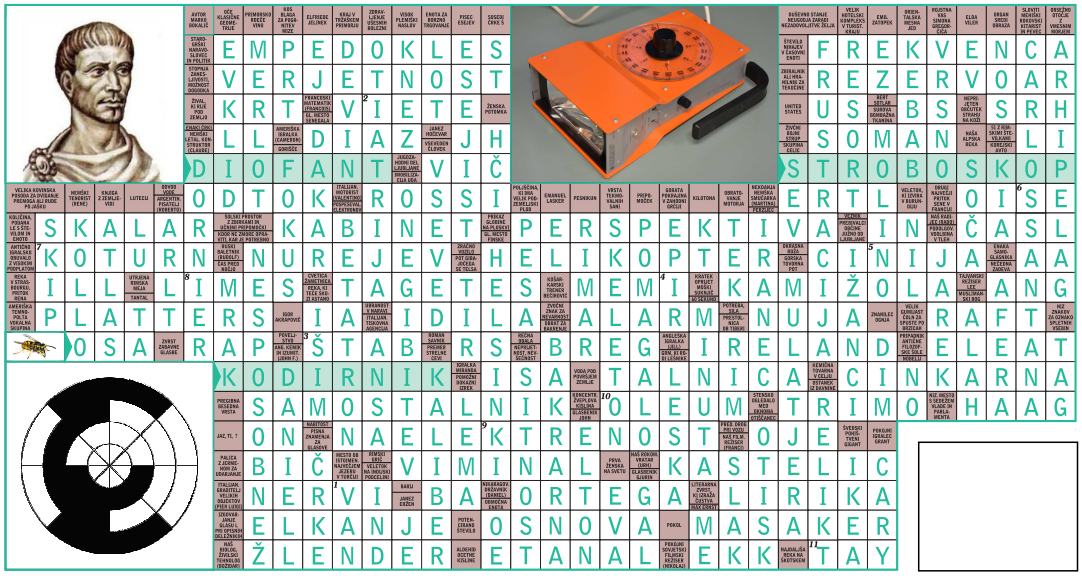
## Literatura


[1] A. Bacher, O. Bodini, H. Hwang in T. Tsai, *Generating Random Permutations by Coin Tossing: Classical Algorithms, New Analysis and Modern Implementation*, ACM Transactions on Algorithms, **13**, 1–43, 2017.

[2] A. Vesel, *Nekaj algoritmov za generiranje permutacij*, Presek, **44** (2016/2017) 26–29.

[3] M. A. Weiss, *Data Structures and Algorithm Analysis*, Benjamin-Cummings Pub Co, 1991.



## REŠITEV NAGRADNE KRIŽANKE

PRESEK 45/6

→ Pravilna rešitev nagradne križanke iz šeste številke Preseka je **Višinski kot**. Izmed pravilnih rešitev so bili izžrebani Rok Strah iz Ljubljane, Marijana Marinšek iz Celja in Marko Kubale iz Rogaške Slatine, ki so razpisane nagrade prejeli po pošti.