

Zadatak 2

Grupa 8

Oktober 2024

1 Uredjeni izbori elemenata

Ponekad je poredak u kojem se elementi nalaze bitan, a ponekad nije. Na primer, reč STOP je različita od reči POTS, bez obzira što su obe reči formirane od slova iz skupa O, P, S, T. S druge strane, zbir brojeva $1+2+3$ je isti kao zbir $2+3+1$, bez obzira što je redosled ovih brojeva promenjen. U ovom odeljku ćemo naučiti kako da prebrojimo izbore elemenata kod kojih je poredak bitan.

1.1 Klasifikacija

Prilikom prebrojavanja elemenata neke kolekcije važno je razmotriti dva pitanja:

1. Da li je važno kako su izabrani elementi uredjeni?
2. Da li su ponavljanja elemenata dozvoljena?

Odgovor na prvo pitanje određuje da li govorimo o uredjenom ili neuredjenom izboru elemenata, obzirom da ovde govorimo samo o uredjenim izborima, odgovor na prvo pitanje je jasan.

Odgovor na drugo pitanje zapravo određuje da li radimo sa skupovima ili sa multiskupovima.

Ipak, kod uredjenih izbora elemenata ova dva pitanja nisu u potpunosti dovoljna za klasifikaciju već se mora postaviti i treće pitanje:

3. Da li biramo sve elemente kolekcije ili ne?

Odgovor na ovo pitanje određuje da li radimo sa permutacijama ili sa m -permutacijama koje se još nazivaju i varijacije.

Na osnovu ovih pitanja možemo dobiti sledeću klasifikaciju:

1. Uredjenje m elemenata iz multiskupa koji ima n elemenata:
 - (a) m -permutacije multiskupa ($m \leq n$, $M = [b_1, \dots, b_l]_{m_1, \dots, m_l}$)
 - (b) Permutacije multiskupa ($m = n$, $M = [b_1, \dots, b_l]_{m_1, \dots, m_l}$)
2. Uredjenje m elemenata iz skupa od n elemenata
 - (a) m -permutacije skupa ($m \leq n$)
 - (b) Permutacije skupa ($m = n$)

1.2 m-permutacije skupa

M-permutacija elemenata skupa A je bilo koja m -torka elementa skupa A u kojoj su svaka dva elementa međusobno različita.

Formula za prebrojavanje m-permutacija

Broj m-permutacija skupa od n elemenata može se izračunati pomoću sledeće formule:

$$P(n, m) = \frac{n!}{(n-m)!}$$

Primer

Ako imamo skup $A = \{1, 2, 3, 4\}$ (gde je $n = 4$) i želimo da izaberemo $m = 2$ elementa, broj m-permutacija bi bio:

$$P(4, 2) = \frac{4!}{(4-2)!} = \frac{4!}{2!} = \frac{24}{2} = 12$$

To znači da postoji 12 različitih načina da se izaberu i rasporede 2 elementa iz skupa od 4 elementa.

Teorema

Broj m-permutacija skupa $B = \{b_1, \dots, b_n\}$ jednak je

$$P(n, m) = n(n-1)(n-2) \dots (n-m+1).$$

Dokaz (indukcija po m):

Baza: Za $m = 1$: Uredjena torka sa jednim elementom je sam taj element. Broj načina da se taj jedan element izabere jednak je n , za svako $n \in \mathbb{N}$.

Induktivna hipoteza: Neka tvrdnja važi za $P(n, m)$ za svako $n \geq m$. Sada treba pokazati da važi i za $P(n, m+1)$ za svako $n \geq m+1$.

Znači, treba pokazati da je broj uredjenih torki dužine $m+1$ skupa B sa osobinom da su svi elementi međusobno različiti jednak:

$$P(n, m+1) = n(n-1)(n-2) \dots (n-(m+1)+1) = n(n-1)(n-2) \dots (n-m).$$

Svaka uredjena torka sa datom osobinom pripada skupu

$$B_1 \cup B_2 \cup \dots \cup B_n,$$

gde su definisani skupovi:

$$B_i = \{(b_i, c_2, \dots, c_{m+1}) : (c_2, \dots, c_{m+1}) \text{ je } m\text{-permutacija skupa } B \setminus \{b_i\}\}.$$

Prema indukcijskoj hipotezi, za svako $i \in \{1, \dots, n\}$, broj elemenata u B_i jednak je broju m -permutacija elemenata skupa $B \setminus \{b_i\}$, to jest,

$$P(n-1, m) = (n-1)(n-2) \dots (n-m).$$

Prema principu sume, broj $(m+1)$ -permutacija je:

$$P(n, m+1) = |B_1 \cup B_2 \cup \dots \cup B_n| = |B_1| + |B_2| + \dots + |B_n| = P(n-1, m) + P(n-1, m) + \dots + P(n-1, m).$$

Tako dobijamo:

$$P(n, m+1) = n \cdot P(n-1, m) = n(n-1)(n-2) \dots (n-m),$$

što je i trebalo dokazati.

1.3 Permutacije skupa

Neka je $B = \{b_1, \dots, b_n\}$. U slučaju kada je $m = n$, za m -permutaciju skupa B se kaže da je ona permutacija. Broj permutacija skupa B označava se sa $P(n)$.

Teorema

Broj permutacija skupa B jednak je

$$P(n) = n(n-1) \cdot \dots \cdot 2 \cdot 1$$

Dokaz: Koristimo teoremu m -permutacija skupa za specijalni slučaj $m = n$.

Baza $n = 1$: Ako imamo jedan element, postoji samo jedan način da ga permutujemo (tj. $1!$). Dakle, broj permutacija jednog elementa je 1, što je tačno $n! = 1!$.

Induktivna pretpostavka: Pretpostavimo da tvrdnja važi za $n' = n$, tj. da broj permutacija skupa sa n elemenata iznosi $P(n) = n!$.

Induktivni korak: Dokazaćemo da tvrdnja važi za $n+1$, tj. da je broj permutacija skupa sa $n+1$ elemenata jednak $(n+1)!$.

Razmotrimo skup sa $n+1$ elemenata. Za svaku permutaciju ovog skupa, možemo odabrati bilo koji od $n+1$ elemenata kao prvi element. Nakon što smo odabrali prvi element, preostalih n elemenata možemo permutovati na $P(n) = n!$ načina (prema induktivnoj pretpostavci). Dakle, ukupan broj permutacija skupa sa $n+1$ elemenata je:

$$P(n+1) = (n+1) \cdot P(n)$$

Prema induktivnoj pretpostavci, $P(n) = n!$ pa imamo:

$$P(n+1) = (n+1) \cdot n! = (n+1)!$$

Zaključak: Broj permutacija skupa sa $n+1$ elemenata iznosi $(n+1)!$, što dovršava indukcionu dokaz. Iz ovog dokaza sledi formula $P(n) = n(n-1) \cdot \dots \cdot 2 \cdot 1$.

1.4 m-permutacije multiskupa

Neka je dat skup $B = \{b_1, \dots, b_n\}$ sa $n \geq 1$ elemenata i neka je

$$M = [b_1, \dots, b_n]_{m, \dots, m}$$

multiskup u kojem se svaki element iz B pojavljuje tačno m puta. Ako izaberemo m -toclani podskup od M i uredimo ga dobijamo jednu m -permutaciju multiskupa.

Definicija

M-permutacija elemenata multiskupa $M = [b_1, \dots, b_n]_{m, \dots, m}$ je bilo koja uredjena m -torka elemenata iz M , tj. bilo koja uredjena m -torka u kojoj je svaka komponenta element iz skupa B .

Teorema

Formula za prebrojavanje m -permutacija multiskupa $M = [b_1, \dots, b_n]_{m, \dots, m}$ jednaka je:

$$P(n, m) = n^m$$

Na osnovu definicije m -permutacija elemenata multiskupa, svaki element skupa $B \times \dots \times B$ predstavlja jednu m -permutaciju multiskupa M . Prema tome, broj takvih m -torki elemenata iz B je na osnovu principa proizvoda:

$$|B \times \dots \times B| = |B|^m = n^m$$

Teorema

Svaki skup X sa n elemenata ima tačno 2^n podskupova ($n \geq 0$).

Posmatrajmo proizvoljan podskup A skupa X i definišimo preslikavanje $f_A : X \rightarrow \{0, 1\}$. Za $x \in X$ odredjujemo:

$$f_A(x) = \begin{cases} 1 & \text{ako } x \in A \\ 0 & \text{ako } x \notin A \end{cases}.$$

Ovo preslikavanje se često sreće u matematici i naziva se karakteristična funkcija skupa A .

Različiti skupovi A imaju različite funkcije f_A , i obrnuto, svako preslikavanje $f : X \rightarrow \{0, 1\}$ odredjuje skup $A = \{x \in X \mid f(x) = 1\}$, tako da je $f = f_A$.

Prema tome, broj podskupova X je isti kao broj svih preslikavanja $X \rightarrow \{0, 1\}$, a to je 2^n .

1.5 Permutacije multiskupa

Neka je dat multiskup

$$M = [b_1, \dots, b_l]_{m_1, \dots, m_l} = \{\underbrace{b_1, \dots, b_1}_{m_1}, \dots, \underbrace{b_l, \dots, b_l}_{m_l}\}$$

i neka je $n = m_1 + \dots + m_l$ broj elemenata datog multiskupa. Svaka uređjena n -torka elemenata multiskupa M je permutacija tog skupa.

Definicija: Permutacija multiskupa M je proizvoljna n -torka u kojoj se b_1 pojavljuje m_1 puta, b_2 se pojavljuje m_2 puta, \dots , b_l se pojavljuje m_l puta.

Teorema: Broj permutacija multiskupa M je

$$P(m_1, m_2, \dots, m_l) = \frac{(m_1 + \dots + m_l)!}{m_1! \cdot \dots \cdot m_l!}$$

Dokaz: Ako bi svi elementi skupa M bili različiti, broj permutacija tog skupa bio bi jednak $m_1 + \dots + m_l$!. Međutim, zbog ponavljanja određenih elemenata, imamo $m_1! \cdot \dots \cdot m_l!$ istih permutacija.

1.6 Veze izmedju osobina funkcija i prethodnih objekata

Moguće je definisati vezu izmedju kombinatornih objekata i osnovnih osobina preslikavanja (injekcije i surjekcije).

1.6.1 Veza izmedju preslikavanja i m -permutacija multiskupa

Funkcija skupa A u skup B je skup svih uređenih parova (binarnih relacija) takvih da su prvi članovi elementi skupa A (domena funkcije), a drugi članovi elementi skupa B (kodomena). Neophodno je da se svaki element skupa A pojavi tačno jednom kao prva komponenta relacije, dok može postojati više relacija sa istom drugom komponentom (više elemenata skupa A se može preslikavati u isti element skupa B).

Obzirom da se elementi skupa B (slike) mogu ponavljati, znamo da preslikavanje možemo povezati sa permutacijama multiskupa. Recimo da je m broj elemenata skupa A , a l broj elemenata skupa B . Pri preslikavanju biramo m od mogućih l slika tako da znamo da skupu svih preslikavanja odgovara m -permutacija multiskupa.

Teorema 23 *Neka je $A = \{a_1, \dots, a_m\}$ i $B = \{b_1, \dots, b_l\}$. Broj svih preslikavanja skupa A u skup B jednak je*

$$|\{f : A \rightarrow B\}| = l^m.$$

Figure 1: Veza izmedju preslikavanja i m -permutacija multiskupa.

1.6.2 Veza izmedju injekcije i m -permutacija skupa

Ako imamo skupove $A = \{a_1, \dots, a_m\}$ i $B = \{b_1, \dots, b_n\}$, za preslikavanje $f : A \rightarrow B$ se kaže da je injektivno ako važi da kada je $f(a) = f(b)$, sledi $a = b$. Svaki element skupa A slika se u različit element skupa B .

$$\forall i, j \in \{1, \dots, m\} \quad i \neq j \Rightarrow f(a_i) \neq f(a_j).$$

Figure 2: Veza izmedju injekcije i m -permutacija skupa.

Pošto je u svakom preslikavanju drugi element različit, znamo da se radi o permutacijama skupa. Potrebno je m originala preslikati u skup B , koji ima n članova. Na osnovu ovoga vidimo vezu izmedju injekcija i m -permutacija skupa.

Teorema 24 *Neka je $A = \{a_1, \dots, a_m\}$ i $B = \{b_1, \dots, b_n\}$, gde je $1 \leq m \leq n$. Broj svih injektivnih preslikavanja skupa A u skup B jednak je*

$$|\{f : A \xrightarrow{1-1} B\}| = n \cdot (n-1) \cdot \dots \cdot (n-m+1).$$

Figure 3: Injektivna preslikavanja kao m -permutacije skupa.

1.6.3 Veza izmedju surjekcije i kombinatornih objekata

Neka su $A = \{a_1, \dots, a_m\}$ i $B = \{b_1, \dots, b_n\}$. Preslikavanje $f : A \rightarrow B$ je surjektivno ako za svaki element skupa B važi da postoji element skupa A koji se u njega preslikava.

$$\forall j \in \{1, \dots, n\} \exists i \in \{1, \dots, m\} f(a_i) = b_j.$$

Figure 4: Veza izmedju surjekcije i kombinatornih objekata.

Broj surjektivnih preslikavanja može se odrediti primenom principa uključenja-isključenja.

Teorema 25 *Neka su A i B skupovi sa osobinom $|A| = m$, $|B| = n$ i $1 \leq n \leq m$. Broj surjektivnih preslikavanja skupa A u skup B jednak je*

$$|\{f : A \xrightarrow{na} B\}| = \sum_{i=0}^{n-1} \binom{n}{i} (-1)^i (n-i)^m.$$

Figure 5: Princip uključenja-isključenja za brojanje surjektivnih preslikavanja.

Dokaz

Definisaćemo skupove koji nisu surjektivni, tj. bar za jedan njihov element ne postoji original koji se u njega slika.

$$\begin{aligned} B_1 &= \{f : A \rightarrow B \mid b_1 \notin im(f)\} \\ B_2 &= \{f : A \rightarrow B \mid b_2 \notin im(f)\} \\ \dots &\dots \dots \\ B_n &= \{f : A \rightarrow B \mid b_n \notin im(f)\}. \end{aligned}$$

Figure 6: Skupovi koji nisu surjektivni.

Može se zaključiti da je skup svih preslikavanja jednak skupu svih preslikavanja koja jesu surjektivna i skupu svih preslikavanja koja nisu surjektivna.

$$\begin{aligned}\{f : A \rightarrow B\} &= \{f : A \rightarrow B : f \text{ je "na"}\} \cup \{f : A \rightarrow B : f \text{ nije "na"}\}, \text{ tj.} \\ \{f : A \rightarrow B\} &= \{f : A \rightarrow B : f \text{ je "na"}\} \cup B_1 \cup \dots \cup B_n.\end{aligned}$$

Figure 7: Veza izmedju surjektivnih i nesurjektivnih preslikavanja.

Na osnovu principa sume sledi:

$$|\{f : A \rightarrow B\}| = |\{f : A \xrightarrow{na} B\}| + |B_1 \cup \dots \cup B_n|.$$

koristimo princip uključenja-isključenja.

$$|B_1 \cup \dots \cup B_n| = \sum_{\emptyset \neq I \subseteq \{1, \dots, n\}} (-1)^{|I|-1} \left| \bigcap_{i \in I} B_i \right|.$$

Figure 8: Princip uključenja-isključenja za računanje broja surjekcija.

Ranije smo videli da je ukupan broj preslikavanja jednak n^m , i kada uvrstimo princip uključenja-isključenja dobijamo:

$$\begin{aligned}|\{f : A \xrightarrow{na} B\}| &= |\{f : A \rightarrow B\}| - |B_1 \cup \dots \cup B_n| \\ &= n^m - n(n-1)^m + \binom{n}{2}(n-2)^m - \dots + (-1)^{n-1}.\end{aligned}$$

Figure 9: Konačan izraz za broj surjektivnih preslikavanja.

1.7 Algoritmi za generisanje prethodnih kombinatornih objekata

Za svaki od prethodnih kombinatornih objekata možemo da napravimo algoritam za njihovo generisanje na osnovu ulaza. Naredni programi su napisani u programskom jeziku python.

1.7.1 Algoritam za generisanje m-permutacija multiskupa


```

while True:
    try:
        l=int(input("Unesite velicinu skupa:"))
        if l<=0:
            print("Velicina skupa mora biti pozitivan  
ceo broj!")
            continue
        break
    except ValueError:
        print("Niste uneli ceo broj!")

while True:
    try:
        m=int(input("Unesite broj elemenata koji se  
uredjuje(ujedno i broj ponavljanja svakog  
elementa):"))
        if m<=0:
            print("Broj elemenata uredjenja mora biti  
pozitivan ceo broj!")
            continue
        break
    except ValueError:
        print("Niste uneli ceo broj!")

skup=[]
for i in range(1):
    while True:
        try:
            x=int(input("Unesite element skupa:"))
            if(x in skup):
                print("Element vec postoji u skupu!")
                continue
            break
        except ValueError:
            print("Niste uneli ceo broj!")
    skup.append(x)

multiskup=[]

for i in range(1):
    for j in range(m):
        multiskup.append(skup[i])

print("Uneli ste multiskup: "+str(multiskup))
print("Resenje:")
#formula: l^m
broj=l**m
print("Broj m-permutacija multiskupa: "+str(broj))

resenja=[]

```

```

def generisi_permutacije(m, trenutna_permutacija):
    if len(trenutna_permutacija) == m:
        resenja.append(tuple(trenutna_permutacija)) #
        Cuvamo trenutnu permutaciju
    return

    for element in skup:
        trenutna_permutacija.append(element) #
        Dodajemo element
        generisi_permutacije(m, trenutna_permutacija)
        trenutna_permutacija.pop() # Uklanjam
        poslednji element za sledeci izbor

generisi_permutacije(m, [])

print("Generisane m-permutacije multiskupa:")
for permutacija in resenja:
    print(permutacija)

```

```

Unesite velicinu skupa:3
Unesite broj elemenata koji se uredjuje(ujedno i broj ponavljanja svakog elementa):2
Unesite element skupa:1
Unesite element skupa:2
Unesite element skupa:3
Uneli ste multiskup: [1, 1, 2, 2, 3, 3]
Resenje:
Broj m-permutacija multiskupa: 9
Generisane m-permutacije multiskupa:
(1, 1)
(1, 2)
(1, 3)
(2, 1)
(2, 2)
(2, 3)
(3, 1)
(3, 2)
(3, 3)

```

Figure 10: Izgled ulaza i izlaza za algoritam generisanja m-permutacija multiskupa

1.7.2 Algoritam za generisanje m-permutacija skupa

```

while True:
    try:
        n=int(input("Unesite velicinu skupa:"))
        if n<=0:

```

```

        print("Velicina skupa mora biti pozitivan ceo
              broj!")
        continue
    break
except ValueError:
    print("Niste uneli ceo broj!")

while True:
    try:
        m=int(input("Unesite broj elemenata koji se
                     uredjuje:"))
        if m<=0:
            print("Broj elemenata uredjenja mora biti
                  pozitivan ceo broj!")
            continue
        break
    except ValueError:
        print("Niste uneli ceo broj!")

skup=[]
for i in range(n):
    while True:
        try:
            x=int(input("Unesite element skupa:"))
            if(x in skup):
                print("Element vec postoji u skupu!")
                continue
            break
        except ValueError:
            print("Niste uneli ceo broj!")
    skup.append(x)

print("Uneli ste skup: "+str(skup))
print("Resenje:")
#formula: n(n-1)(n-2)...(n-m+1)
broj=1
for i in range(n, n-m, -1):
    broj*=i
print("Broj m-permutacija skupa: "+str(broj))

resenja=[]

def generisi_permutacije(m, trenutna_permutacija, dostupni):
    if len(trenutna_permutacija) == m:
        resenja.append(tuple(trenutna_permutacija)) #
        Cuvamo trenutnu permutaciju
    return

    for i in range(len(dostupni)):
        # Izbor i uklanjanje elementa

```

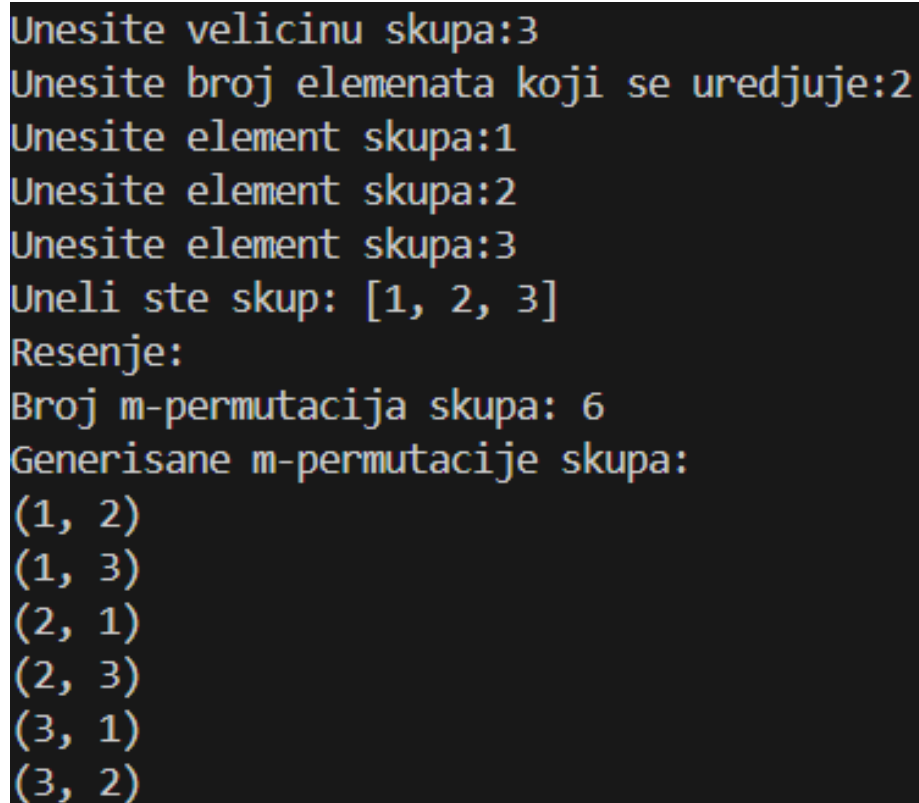
```

        element = dostupni[i]
        trenutna_permutacija.append(element) # Dodajemo
            element
        # Kreiramo novi skup dostupnih elemenata bez
            trenutnog
        novi_dostupni = dostupni[:i] + dostupni[i + 1:]
        generisi_permutacije(m, trenutna_permutacija,
            novi_dostupni)
        trenutna_permutacija.pop() # Uklanjamo poslednji
            element za sledeci izbor

generisi_permutacije(m, [], skup)

print("Generisane m-permutacije skupa:")
for permutacija in resenja:
    print(permutacija)

```



```

Unesite velicinu skupa:3
Unesite broj elemenata koji se uredjuje:2
Unesite element skupa:1
Unesite element skupa:2
Unesite element skupa:3
Uneli ste skup: [1, 2, 3]
Resenje:
Broj m-permutacija skupa: 6
Generisane m-permutacije skupa:
(1, 2)
(1, 3)
(2, 1)
(2, 3)
(3, 1)
(3, 2)

```

Figure 11: Izgled ulaza i izlaza za algoritam generisanja m-permutacija skupa

1.7.3 Algoritam za generisanje permutacija multiskupa

```
import math
from typing import Counter

while True:
    try:
        l=int(input("Unesite velicinu skupa:"))
        if l<=0:
            print("Velicina skupa mora biti pozitivan ceo broj!")
            continue
        break
    except ValueError:
        print("Niste uneli ceo broj!")

skup=[]
brojPojava=[]
for i in range(1):
    while True:
        try:
            x=int(input("Unesite element skupa:"))
            if(x in skup):
                print("Element vec postoji u skupu!")
                continue
            m=int(input("Unesite broj ponavljanja elementa:"))
            if m<=0:
                print("Broj ponavljanja elementa mora biti pozitivan ceo broj!")
                continue
            break
        except ValueError:
            print("Niste uneli ceo broj!")
    skup.append(x)
    brojPojava.append(m)

multiskup=[]

for i in range(1):
    for j in range(brojPojava[i]):
        multiskup.append(skup[i])

print("Uneli ste multiskup: "+str(multiskup))
print("Resenje:")
#formula: (m1+m2+...+ml)!/(m1!*m2!*...*ml!)
broj=1
zbir=0
faktorijel=1
proizvod=1
```

```

for i in range(1):
    zbir+=brojPojava[i]
    faktorijel=1
    for j in range(1,brojPojava[i]+1):
        faktorijel*=j
    proizvod*=faktorijel
broj=math.factorial(zbir)//proizvod

print("Broj permutacija multiskupa: "+str(broj))

resenja=[]

def generisi_permutacije_multiskupa(multiskup,
    trenutna_permutacija, dostupni):
    if sum(dostupni.values()) == 0: # Kada nema dostupnih
        elemenata
        resenja.append(tuple(trenutna_permutacija.copy()))
        # Cuvamo trenutnu permutaciju
        return

    for element in dostupni:
        if dostupni[element] > 0: # Ako je element dostupan
            trenutna_permutacija.append(element) #
            Dodajemo element
            dostupni[element] -= 1 # Smanjujemo broj
            dostupnih
            generisi_permutacije_multiskupa(multiskup,
                trenutna_permutacija, dostupni)
            trenutna_permutacija.pop()
            dostupni[element] += 1

# Broj pojavljivanja za generisanje
dostupni = Counter(multiskup)

# Pozivamo funkciju za generisanje permutacija
generisi_permutacije_multiskupa(multiskup, [], dostupni)

print("Generisane permutacije multiskupa:")
for permutacija in resenja:
    print(permutacija)

```

```
Unesite velicinu skupa:3
Unesite element skupa:1
Unesite broj ponavljanja elementa:1
Unesite element skupa:2
Unesite broj ponavljanja elementa:2
Unesite element skupa:3
Unesite broj ponavljanja elementa:1
Uneli ste multiskup: [1, 2, 2, 3]
Resenje:
Broj permutacija multiskupa: 12
Generisane permutacije multiskupa:
(1, 2, 2, 3)
(1, 2, 3, 2)
(1, 3, 2, 2)
(2, 1, 2, 3)
(2, 1, 3, 2)
(2, 2, 1, 3)
(2, 2, 3, 1)
(2, 3, 1, 2)
(2, 3, 2, 1)
(3, 1, 2, 2)
(3, 2, 1, 2)
(3, 2, 2, 1)
```

Figure 12: Izgled ulaza i izlaza za algoritam generisanja permutacija multiskupa

1.7.4 Algoritam za generisanje permutacija skupa

```
while True:
    try:
        n=int(input("Unesite velicinu skupa:"))
        if n<=0:
            print("Velicina skupa mora biti pozitivan ceo broj!")
            continue
        break
    except ValueError:
        print("Niste uneli ceo broj!")

skup=[]
for i in range(n):
    while True:
        try:
            x=int(input("Unesite element skupa:"))
            if(x in skup):
                print("Element vec postoji u skupu!")
                continue
            break
        except ValueError:
            print("Niste uneli ceo broj!")
    skup.append(x)

print("Uneli ste skup: "+str(skup))
print("Resenje:")
#formula: n!
broj=1
for i in range(n, 0, -1):
    broj*=i
print("Broj permutacija skupa: "+str(broj))

resenja=[]

def generisi_permutacije(m, trenutna_permutacija, dostupni):
    if len(trenutna_permutacija) == m:
        resenja.append(tuple(trenutna_permutacija)) #
        Cuvamo trenutnu permutaciju
        return

    for i in range(len(dostupni)):
        # Izbor i uklanjanje elementa
        element = dostupni[i]
        trenutna_permutacija.append(element) # Dodajemo
        element
        # Kreiramo novi skup dostupnih elemenata bez
        trenutnog
        novi_dostupni = dostupni[:i] + dostupni[i + 1:]
```



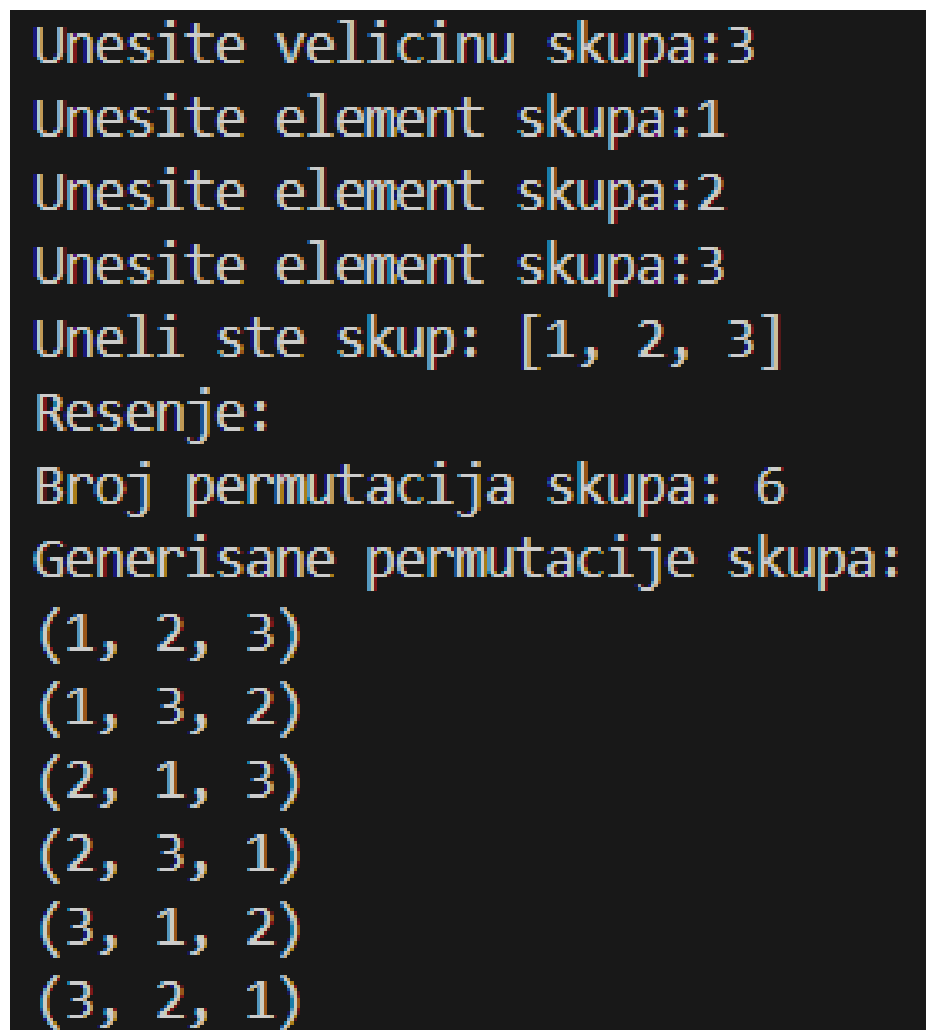
```

generisi_permutacije(m, trenutna_permutacija,
    novi_dostupni)
    trenutna_permutacija.pop() # Uklanjamo poslednji
        element za sledeci izbor

generisi_permutacije(n, [], skup)

print("Generisane permutacije skupa:")
for permutacija in resenja:
    print(permutacija)

```



```

Unesite velicinu skupa:3
Unesite element skupa:1
Unesite element skupa:2
Unesite element skupa:3
Uneli ste skup: [1, 2, 3]
Resenje:
Broj permutacija skupa: 6
Generisane permutacije skupa:
(1, 2, 3)
(1, 3, 2)
(2, 1, 3)
(2, 3, 1)
(3, 1, 2)
(3, 2, 1)

```

Figure 13: Izgled ulaza i izlaza za algoritam generisanja permutacija skupa

1.8 Zadaci

1. Neka je $A = \{1, 2, 3, 4\}$.

- (a) Napisati sve 3-permutacije skupa A .
- (b) Odrediti broj 3-permutacija skupa A .

Rešenje.

- (a) 3-permutacije skupa A su:

123	124	132	134	142	143
213	214	231	234	241	243
312	314	321	324	341	342
412	413	421	423	431	432

- (b) Broj 3-permutacija multiskupa M jednak je

$$P(4; 3) = 4 \cdot 3 \cdot 2 = 24.$$

2. Neka je dat multiskup $M = \{a, a, b, c\}$.

- (a) Napisati sve 3-permutacije multiskupa M .
- (b) Odrediti broj 3-permutacija multiskupa M .

Rešenje.

- (a) 3-permutacije multiskupa M su:

aaa, aab, aac, aba, abc, acb, baa, bab, bac, bca, cab, cac
abc, bac, bca, cab, cac, cba, bba, bbc, bca, cbc, cba.

- (b) Broj 3-permutacija multiskupa M možemo izračunati pomoću sledeće formule:

$$P(n; m) = \frac{n!}{(n - m)!}$$

Za dati multiskup M , gde je broj različitih elemenata 3, a želimo 3-permutacije, imamo:

$$P(4; 3) = \frac{4!}{(4 - 3)!} = 4 \cdot 3 \cdot 2 = 24.$$

3. Napisati Java program koji generiše sve permutacije skupa $A = \{1, 2, 3\}$.

(a) Napisati funkciju koja generiše permutacije skupa.

(b) Ispisati sve permutacije skupa $A = \{1, 2, 3\}$.

Rešenje.

(a) Funkcija u Javi koja generiše permutacije skupa može se napisati na sledeći način:

```
// Java program to generate all permutations of a set
import java.util.ArrayList;
import java.util.List;

public class Permutacije {

    // Function to swap elements
    private static void swap(int[] array, int i, int j) {
        int temp = array[i];
        array[i] = array[j];
        array[j] = temp;
    }

    // Recursive function to generate all permutations
    private static void permute(int[] array, int l, int r,
        List<int[]> results) {
        if (l == r) {
            results.add(array.clone());
        } else {
            for (int i = l; i <= r; i++) {
                swap(array, l, i);
                permute(array, l + 1, r, results);
                swap(array, l, i); // backtrack
            }
        }
    }

    // Main function
    public static void main(String[] args) {
        int[] A = {1, 2, 3};
        List<int[]> results = new ArrayList<>();
        permute(A, 0, A.length - 1, results);

        // Print the results
        for (int[] perm : results) {
            for (int num : perm) {
                System.out.print(num + " ");
            }
            System.out.println();
        }
    }
}
```

(b) Izlaz za skup $A = \{1, 2, 3\}$ biće:

123
132
213
231
312
321

4. Profesor je podelio 188 studenata u 4 jednake grupe od po 47 studenata. Svaka grupa bira po dva predstavnika da prezentuju rešenja zadataka. Na koliko načina može biti izabrano tih 8 predstavnika?

Rešenje. Broj načina da se izaberu dva predstavnika iz svake od četiri grupe može se izračunati korišćenjem binomnih koeficijenata. Za svaku grupu od 47 studenata, broj načina da se izaberu dva predstavnika je $\binom{47}{2}$. Pošto imamo četiri grupe, ukupan broj načina da se izaberu predstavnici je proizvod binomnih koeficijenata za svaku grupu:

$$\binom{47}{2} \times \binom{47}{2} \times \binom{47}{2} \times \binom{47}{2}$$

Izračunajmo vrednost binomnog koeficijenta $\binom{47}{2}$:

$$\binom{47}{2} = \frac{47!}{2!(47-2)!} = \frac{47 \times 46}{2 \times 1} = 1081$$

Dakle, ukupan broj načina da se izaberu predstavnici je:

$$1081 \times 1081 \times 1081 \times 1081 = 1081^4$$

Što daje konačan rezultat:

$$1081^4 = 1,217,696,081$$

Broj načina da se izabere 8 predstavnika iz 4 grupe po 47 studenata je 1,217,696,081.

$$C(47;2) \cdot C(47;2) \cdot C(47;2) \cdot C(47;2) = \binom{47}{2} \binom{47}{2} \binom{47}{2} \binom{47}{2}.$$

5. Neka je dat multiskup $M = \{a, a, b, c\}$.

(a) Napisati sve 3-permutacije multiskupa M .

(b) Napisati Java program koji generiše sve permutacije multiskupa M .

Rešenje.

(a) 3-permutacije multiskupa M su:

aaa, aab, aac, aba, abc, aca, acb, baa, bab, bac, bca, cab, cac

(b) Java program koji generiše sve permutacije multiskupa M :

```
1 // Java program to generate all permutations of a
  multiset
2 import java.util.ArrayList;
3 import java.util.List;
4
5 public class MultisetPermutations {
6
7     // Function to swap elements
8     private static void swap(char[] array, int i, int
9         j) {
10         char temp = array[i];
11         array[i] = array[j];
12         array[j] = temp;
13     }
14
15     // Recursive function to generate all permutations
16     private static void permute(char[] array, int l,
17         int r, List<String> results) {
18         if (l == r) {
19             results.add(new String(array));
20         } else {
21             for (int i = l; i <= r; i++) {
22                 swap(array, l, i);
23                 permute(array, l + 1, r, results);
24                 swap(array, l, i); // backtrack
25             }
26         }
27     }
28
29     // Main function
30     public static void main(String[] args) {
31         char[] M = {'a', 'a', 'b', 'c'};
32         List<String> results = new ArrayList<>();
33         permute(M, 0, M.length - 1, results);
34
35         // Print the results
36         for (String perm : results) {
37             System.out.println(perm);
38         }
39     }
40 }
```

37	}
38	}