Kombinatorni objekti

16.10.2024.

1 Kombinatorna prebrojavanja

1.1 Definicija i podela kombinatornih objekata

Svaki kombinatorni objekat možemo definisati kao izdvajanje elemenata iz skupa / multiskupa svih zadatih objekata, gde pravimo razliku između toga da li jedan element možemo odabrati više puta i da li je redosled u kojem biramo elemente značajan ili ne.

	Redosled je bitan	Redosled nije bitan
Ne možemo birati element više puta	Permutacija skupa / permutacija bez ponavljanja	Kombinacija skupa / kombinacija bez ponavljanja
Možemo birati element više puta	Permutacija multiskupa / permutacija s ponavljanjem	Kombinacija multiskupa / kombinacija sa ponavljanjem

1.2 Permutacije skupa

Definicija: Pod **permutacijom** skupa A se podrazumeva svaka bijekcija $f : A \to A$ skupa A na samog sebe. Skup svih permutacija skupa A obično se obeležava sa $\mathrm{Sym}(A)$.

Za skup $\mathrm{Sym}(A)$ svih permutacija skupa A važi sledeća teorema, koja ilustruje algebarsku strukturu skupa $\mathrm{Sym}(A)$.

Teorema: Ako je dat skup A, tada za skup $\mathrm{Sym}(A)$ važe sledeće osobine:

a) Zatvorenost: $(\forall f, g \in \mathrm{Sym}(A)) \quad f \circ g \in \mathrm{Sym}(A)$

b) Asocijativnost: $(\forall f, g, h \in \mathrm{Sym}(A)) \quad f \circ (g \circ h) = (f \circ g) \circ h$;

c) Postojanje neutralnog elementa: $(\exists i_A \in \mathrm{Sym}(A)) \quad (\forall x \in A) \quad i_A(x) = x$;

d) Postojanje inverznog elementa: $(\forall f \in \mathrm{Sym}(A)) \quad (\exists f^{-1} \in \mathrm{Sym}(A)) \quad f \circ f^{-1} = f^{-1} \circ f = i_A$.

Algebarski, uređeni par $(Sym(A), \circ)$, gde je $\circ$ operacija kompozicije čini algebarsku grupu. Ova grupa se specijalno naziva **simetrijska grupa skupa** A

Definicija: k-permutacija skupa od n elemenata je svaka uređena k-torka različitih elemenata iz skupa od n elemenata. Permutacija skupa od n elemenata je svaka n-permutacija tog skupa.

Pored naziva k-permutacija koristi se i termin varijacija bez ponavljanja k-te klase od n elemenata.

Teorema: Neka su $P(n,k)$ i $P(n,n)$ redom broj k-permutacija skupa od n elemenata i broj permutacija skupa od n elemenata. Tada:

1. $P(n,k) = n \cdot (n-1) \cdot \ldots \cdot (n-k+1) = \frac{n!}{(n-k)!}$,
2. $P(n,n) = n \cdot (n-1) \cdot \ldots \cdot 2 \cdot 1 = n!$

Dokaz:

(1) Prvi član k-permutacije skupa od n elemenata može biti izabran na n načina. Pošto je jedan element izabran, a u permutaciji se nalaze samo različiti elementi, preostaje $n-1$ elemenat, tako da se sledeći član bira na $n-1$ način. Prema tome, prva dva člana mogu se izabrati na $n \cdot (n-1)$ načina. Postupak produžavamo, tako da se poslednji, k-ti član permutacije bira iz skupa koji sadrži $n-(k-1)$ elemenata, pa postoji upravo toliko načina da i on bude izabran. Prema pravilu proizvoda,

$$P(n,k) = n \cdot (n-1) \cdot \cdots \cdot (n-k+1).$$

(2) Direktna posledica (1) za $k = n$.

k-permutacija skupa predstavlja sve načine na koji iz skupa od n različitih elemenata možemo odabrati k elemenata, tako da je redosled kojim biramo te elemente značajan.

[ ]:
```python
#Rekurzivni algoritam za generisanje svih k-permutacija skupa
def generate_permutations(arr, length, current, used):
    if len(used) == min(length, len(arr)):
        print(current)
    else:
        for i in arr:
            if i not in used:
                newcurrent = current.copy()
                newcurrent.append(i)
                newused = used.copy()
                newused.add(i)
                generate_permutations(arr, length, newcurrent, newused)

def permute(arr, k):
    generate_permutations(arr, k, [], set())

permute([1, 2, 3, 4], 4)
print("####################")
permute([1, 2, 3, 4], 2)
```

1.3 Permutacije multiskupa

Teorema: Neka je dato k vrsta objekata, tako da i-te vrste ima n_i objekata, i neka je $n = \sum_{i=1}^{k} n_i$. Tada je broj permutacija tih n objekata jednak:

$$\frac{n!}{n_1! n_2! \ldots n_k!}.$$

Dokaz: Neka je x traženi broj permutacija. Za bilo koju permutaciju ovih n objekata, ako objekte prve vrste posmatramo kao različite, dobijamo $x \cdot n_1!$ permutacija.

Isto ponavljamo za objekte druge i svih ostalih vrsta, što daje:

$$x \cdot n_1! \cdot n_2! \cdot \cdots \cdot n_k! = P(n, n) = n!.$$

Stoga je traženi broj jednak:

$$x = \frac{n!}{n_1! \cdot n_2! \cdot \cdots \cdot n_k!}.$$

[ ]:
```python
class ListElement:
    def __init__(self, value, next):
        self.value = value
        self.next = next
    def nth(self, n):
        o = self
        i = 0
        while i < n and o.next is not None:
            o = o.next
            i += 1
        return o

def init(multiset):
    multiset.sort() # ensures proper non-increasing order
    h = ListElement(multiset[0], None)
    for item in multiset[1:]:
        h = ListElement(item, h)
    return h, h.nth(len(multiset) - 2), h.nth(len(multiset) - 1)

def visit(h):
    """Converts our bespoke linked list to a python list."""
    o = h
    l = []
    while o is not None:
        l.append(o.value)
        o = o.next
    return l

def permutations(multiset):
    """Generator providing all multiset permutations of a multiset."""
    h, i, j = init(multiset)
    yield visit(h)
    while j.next is not None or j.value < h.value:
        if j.next is not None and i.value >= j.next.value:
            s = j
```

```
        else:
            s = i
        t = s.next
        s.next = t.next
        t.next = h
        if t.value < h.value:
            i = t
        j = i.next
        h = t
        yield visit(h)

if __name__ == '__main__':
    import sys
    multiset = [1, 1, 2, 2, 3, 3, 3]
    if multiset != []:
        for permutation in permutations(multiset):
            for item in permutation:
                print(item)
            print()
    else:
        print("usage", sys.argv[0], "<multiset>")
```

Teorema: Neka je dato n objekata. Broj k-permutacija sa ponavljanjem je jednak n^k.

Dokaz: Pošto u svakom od k koraka izbora možemo izabrati jedan od n elemenata, rezultat neposredno sledi prema pravilu proizvoda. Dakle, ukupan broj mogućih izbora je:

$$n \cdot n \cdot \cdots \cdot n \quad (k \text{ puta}) = n^k.$$

1.4 Permutacije "oko okruglog stola" / kružne permutacije

Teorema: Broj načina da se n objekata rasporedi oko okruglog stola je $(n-1)!$.

Dokaz (prvi način): 1. Fiksiramo jedan objekat (npr. A) da izbegnemo dupliranje zbog rotacija. 2. Preostali $n-1$ objekata možemo rasporediti na $(n-1)!$ načina.

Dokaz (drugi način): Primetimo da za svako raspoređivanje n elemenata u nizu postoji tačno n raspoređivanja za okruglim stolom koja se razlikuju samo po tome od kojeg mesta smo počeli raspored, tj. postoji n rasporeda koji se od jednog mogu generisati samo rotacijom stola. Kako je ukupan broj permutacija $n!$ i svaka permutacija se za okruglim stolom ponavlja n puta, ukupan broj kružnih permutacija je $\frac{n!}{n} = (n-1)!$

1.5 Primeri

[ ]:
```
#Fisher-Yates algoritam za nasumično permutovanje

import random

def random_permutation(n):
```

```
    elements = list(range(1, n + 1))

    for i in range(n - 1, 0, -1):
        j = random.randint(0, i)
        elements[i], elements[j] = elements[j], elements[i]

    return elements

n = 5
perm = random_permutation(n)
print(f"Nasumična permutacija skupa {{1, 2, ..., {n}}}: {perm}")
```

```
Nasumična permutacija skupa {1, 2, …, 5}: [4, 1, 5, 3, 2]
```

[ ]:
```
#Kod za generisanje određene permutacije po leksikografskom poretku, u ovom
↪primeru izračunata je 5. permutacija

import math

def permutacija_po_redu(k, m, n, a):
    a[m] = 1 + (k - 1) // math.factorial(n - m)

    if m < n - 1:
        k0 = k - (a[m] - 1) * math.factorial(n - m)
        permutacija_po_redu(k0, m + 1, n, a)

        for i in range(m + 1, n):
            if a[i] > a[m]:
                a[i] += 1

n = 4
k = 5
a = [0] * n

permutacija_po_redu(k, 0, n, a)
print("Permutacija sa rednim brojem", k, ":", a)
```

```
Permutacija sa rednim brojem 5 : [1, 1, 5, 1]
```