Zadatak br.3

Grupa 4

Oktobar 2024

1 Klasifikacija neuređenih izbora elemenata

1.1 Uvod

U ovoj sekciji ćemo uvesti čitaoca u temu kombinatorike sa fokusom na neuređene izbore elemenata. Klasifikacija neuređenih izbora obuhvata m-kombinacije skupa i m-kombinacije multiskupa, uz osnovne metode prebrojavanja tih izbora. Ovi pojmovi igraju ključnu ulogu u rešavanju mnogih problema iz oblasti diskretne matematike, teorije brojeva i algoritama. Neuređeni izbori elemenata se razlikuju od uređenih time što redosled elemenata nije bitan. Kao što ćemo videti, m-kombinacije se mogu posmatrati kao specijalni slučajevi permutacija, gde se eliminacija redosleda uvodi korišćenjem odgovarajućih faktora.

1.2 Kombinacije bez ponavljanja

Kombinacije bez ponavljanja nazivamo m-kombinacijama skupa. To su neuređeni izbori m elemenata iz skupa S sa n elemenata, gde se svaki element može izabrati samo jednom. Broj takvih kombinacija se računa pomoću binomnog koeficijenta:

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

gde je n ukupan broj elemenata u skupu, a m broj elemenata koji se biraju. Binomni koeficijent daje broj različitih podskupova od m elemenata koji se mogu formirati iz skupa sa n elemenata.

1.3 Kombinacije sa ponavljanjem

Kombinacije sa ponavljanjem, poznate kao m-kombinacije multiskupa, dozvoljavaju biranje elemenata sa ponavljanjem. Broj ovih kombinacija se računa pomoću formule:

$$C(n + m - 1, m) = \frac{(n + m - 1)!}{m!(n-1)!}$$

Ova formula se koristi kada biramo m elemenata iz skupa od n elemenata, gde se svaki element može birati više puta.

2 M-kombinacija elemenata skupa

2.1 Uvod

Kada smo govorili o uređenim izborima m elemenata konačnog skupa A sa n elemenata, koristili smo preslikavanje kako bismo bili u mogućnosti da odredimo koji element je izabran prvi, koji drugi, a koji poslednji. Za razliku od toga, kada govorimo o m-kombinacijama elemenata skupa, tj. o neuređenom izboru elemenata skupa, nije nam bitan redosled izabranih elemenata, te nam preslikavanje neće biti potrebno. Kako u ovom slučaju razmatramo neuređene izbore elemenata bez ponavljanja, možemo videti da oni predstavljaju m-točlane podskupove skupa A. Kako bismo bolje razumeli o čemu govorimo, pogledajmo sledeći primer.

2.2 Intuicija

Zadatak: Grupa od 3 drugara igraju igru i treba da se napravi tim od 2 drugara. Koliko različitih timova se može napraviti?

Vidimo da kod ovog zadatka imamo skup od 3 drugara, $A = \{1, 2, 3\}$, i treba da odredimo njegove 2-člane podskupove. Uvodimo simbol $\binom{A}{2}$ da predstavimo broj načina za izbor 2-članih podskupova iz skupa A. U ovom konkretnom primeru $\binom{A}{2} = \{\{1, 2\}, \{1, 3\}, \{2, 3\}\}$. Vidimo da je rešenje ovog zadatka 3, tj. da možemo imati situaciju gde drugar 1 i 2 igraju zajedno ili drugar 1 i 3 igraju zajedno, ili drugar 2 i 3 igraju zajedno.

Ovakvo rešavanje zadatka je logično i relativno jednostavno, ali šta se dešava u slučaju kada imamo mnogo veći početni skup i veće podskupove? Da li ćemo u slučaju kada tražimo 8-člane podskupove 100-članog skupa ručno ispisivati sve mogućnosti? Odgovor je da naravno nećemo, i zato u sledećem delu uvodimo pojam binomnog koeficijenta kao i formulu za prebrojavanje kombinacija skupa elemenata.

2.3 Definicija

Neka su n I m nenegativni celi brojevi pri čemu važi n >= m . Binomni koeficijent $\binom{n}{m}$ je funkcija promenljivih n I r data pomoću:

$$\binom{n}{m} = \frac{n(n-1)(n-2)\cdots(n-m+1)}{m(m-1)\cdots 2 \cdot 1}$$

Binomni koeficijenti imaju izuzetno veliki broj primena I jedan su od najvažnijih kombinatornih pojmova. Binomni koeficijenti se najlakše predstavljaju pomoću faktorijela, kao što je pokazano u definciji iznad. Najvažnije svojstvo binomnih koeficijenata koristi se u Binomnom razvoju, tj. Razvoju stepena binoma.

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$$

2.4 Teorema

Broj m-kombinacija elemenata skupa A sa n elemenata jednak je:

$$C(n;m) = \frac{n(n-1)\dots(n-m+1)}{m!}$$

Dokaz. Neka je $n = |A|$. Prebrojaćemo uređene izbore m elemenata skupa A bez ponavljanja na dva načina.

Prvi način je koristeći prethodno uvedenu teoremu za uređene izbore bez ponavljanja i na osnovu nje znamo da je broj svih injektivnih preslikavanja $f : A \to B$ jednak $n(n-1)\dots(n-m+1)$.

Drugi način je da od svakog m-točlanog podskupa $B \in \binom{A}{m}$ uređivanjem njegovih elemenata možemo da dobijemo $m!$ različitih uređenih izbora m elemenata, i svaki uređeni izbor m elemenata može da se dobije samo iz jednog m-točlanog podskupa B na ovaj način.

Prema tome:

$$n(n-1)\dots(n-m+1) = m!\left|\binom{A}{m}\right|$$

□

2.5 Primeri

Zadatak 1. *Restoran nudi 6 različitih vrsta jela. Na koliko načina možemo izabrati 2 jela za ručak, bez obzira na redosled?*

Rešenje. $C(6,2) = \frac{6!}{2!(6-2)!} = \frac{6*5}{2*1} = 15$ □

Zadatak 2. *Ako u grupi ima 8 muškaraca i 5 žena, na koliko načina možemo izabrati tim od 3 muškarca i 2 žene?*

Rešenje. Biramo 3 muškarca od 8: $C(8,3) = \frac{8*7*6}{3*2*1} = 56$ Biramo 2 žene od 5: $C(5,2) = \frac{5*4}{2*1} = 10$ Ukupan broj načina je proizvod ove dve vrednosti: $56 * 10 = 560$ □

2.6 Algoritam u jeziku Java

```
//Java program za generisanje m-kombacija skupa
 import java.util.ArrayList;
 import java.util.List;

 public class Kombinacije {

     public static List<List<Integer>> kombinacije(List<Integer> S,
         int m) {
         Set<Integer> jedinstveniSkup = new HashSet<>(S);
```

```
        List<Integer> jedinstvenaLista = new
            ArrayList<>(jedinstveniSkup);
        List<List<Integer>> rezultat = new ArrayList<>();
        kombinacijeHelper(jedinstvenaLista, m, 0, new ArrayList<>(),
            rezultat);
        return rezultat;
    }

    private static void kombinacijeHelper(List<Integer> S, int m,
        int pocetak, List<Integer> trenutna, List<List<Integer>>
        rezultat) {
        if (trenutna.size() == m) {
            rezultat.add(new ArrayList<>(trenutna));
            return;
        }

        for (int i = pocetak; i < S.size(); i++) {
            trenutna.add(S.get(i));
            kombinacijeHelper(S, m, i + 1, trenutna, rezultat);
            trenutna.remove(trenutna.size() - 1);
        }
    }

    public static void main(String[] args) {
        List<Integer> S = new ArrayList<>();
        S.add(1);
        S.add(2);
        S.add(3);
        S.add(4);

        int m = 2;
        List<List<Integer>> kombinacije = kombinacije(S, m);

        for (List<Integer> kombinacija : kombinacije) {
            System.out.println(kombinacija);
        }
    }
}
```

2.7 Objašnjenje algoritma i rezultati

Funkcija kombinacije prvo uklanja sve duplikate iz liste S (da ne bi doslo do ponavljanja) i smešta jedinstvene elemente u listu kedomstvenaLista, onda kreira listu svih mogućih rezultata zvanu rezultat. Listu jedinstvenaLista i rezultat zatim prosleđuje rekurzivnoj funckiji kombinacijaHelper. Ulazni parametru funkcije kombinacijeHelper su:

- S - originalna lista elemenata,

- m - velicina kombinacije,
- pocetak - trenutni indeks u listi odakle počinjemo da biramo elemente,
- trenutna - trenutna kombinacija koja se pravi,
- rezultat - lista svih generisanih kombinacija

Ako je trenutna kombinacija velicine m ona se dodaje u rezultat. Za svaki element u skupu, dodajemo ga u trenutnu kombinaciju, rekurzivno biramo sledeći element, a zatim ga uklanjamo (backtracking). Rezultati ovog algoritma za dati primer su:

- (1, 2)
- (1, 3)
- (1, 4)
- (2, 3)
- (2, 4)
- (3, 4)

Zadatak 3. *Neka je dat skup brojeva* $S = \{1, 2, 3, 4, 5\}$.

1. Napiši sve kombinacije veličine 3 iz skupa S.

Rešenje. Sve kombinacije veličine 3 iz skupa $S = \{1, 2, 3, 4, 5\}$ su:

$$\{1, 2, 3\}, \{1, 2, 4\}, \{1, 2, 5\}, \{1, 3, 4\}, \{1, 3, 5\}, \{1, 4, 5\}, \{2, 3, 4\}, \{2, 3, 5\}, \{2, 4, 5\}, \{3, 4, 5\}$$

Kako smo došli do ovih kombinacija?
Kreiramo listu brojeva i funkciji kombinacije je prosleđujemo kao i veličinu kombinacije 3. Ta funkcija zatim kreira HashSet od ulazne liste, da ne bi bilo ponavljanja elemenata, i zatim od toga kreira novu listu koju prosleđujemo, zajedno sa veličinom kombinacije, početnim indeksom (koji je 0), praznu trenutnu listu i listu rezultat, "backtracking" algoritmu [11]. Zatim algoritam kreće da kreira kombinacije:

- Počinje sa brojem 1. Sa brojem 1 može birati brojeve redom 2 i 3. Znači on prelazi na sledeći korak u kojem na drugom mestu bira 2 proverava da li je ta kombinacija jednaka sa 3, pošto nije prelazi na sledeći korak u kojem bira broj 3. Proverava da li je dužina te kombinacije jednaka sa 3 i ako jeste dodaje je u rezultat, zatim se vraća jedan korak
- Za sledeće kombinacije i dalje na prvom mestu ostaje broj 1, na drugom 2, a mogući koraci koje algoritam može da napravi jesu sa brojevima 3 i 4 pa se vraća na prethodni korak. Tako dobijamo kombinacije

$$\{1, 2, 4\}, \{1, 2, 5\}$$

- Dalje pošto sa brojem 2 na drugom mestu ne može da napravi više novih, jedinstvenih kombinacija, on broj 2 menja brojem 3. Zatim mogući koraci koje može da napravi su sa brojevima 4 i 5

$$\{1,3,4\},\{1,3,5\}$$

- Ovaj postupak nastavlja sve dok ne nestane kombinacija koje može da napravi

□

Zadatak 4. *U školi se organizuje takmičenje u matematici. Učestvuje 8 učenika: Ana, Marko, Jovana, Luka, Ivana, Stefan, Milica i Petar. Nastavnici žele da formiraju tim od 3 učenika koji će predstavljati školu na takmičenju. Koliko različitih timova može da se formira? [10]*

Rešenje. Mi u ovom slučaju imam skup od 8 takmičara, koji možemo da posmatramo kao da imamo skup od 8 brojeva. Želimo da ovih 8 brojeva, takmičara, raspodelimo na timove od po troje ljudi. Znači nama je početna lista 8 brojeva, svakog takmičara možemo predstaviti kao neki jedinstveni broj u skupu, a veličina kombinacije je 3. Tada dobijamo sledeće mogućnosti:

$$\{1,2,3\},\{1,2,4\},\{1,2,5\},\{1,2,6\},\{1,2,7\},\{1,2,8\},$$
$$\{1,3,4\},\{1,3,5\},\{1,3,6\},\{1,3,7\},\{1,3,8\},\{1,4,5\},$$
$$\{1,4,6\},\{1,4,7\},\{1,4,8\},\{1,5,6\},\{1,5,7\},\{1,5,8\},$$
$$\{1,6,7\},\{1,6,8\},\{1,7,8\},\{2,3,4\},\{2,3,5\},\{2,3,6\},$$
$$\{2,3,7\},\{2,3,8\},\{2,4,5\},\{2,4,6\},\{2,4,7\},\{2,4,8\},$$
$$\{2,5,6\},\{2,5,7\},\{2,5,8\},\{2,6,7\},\{2,6,8\},\{2,7,8\},$$
$$\{3,4,5\},\{3,4,6\},\{3,4,7\},\{3,4,8\},\{3,5,6\},\{3,5,7\},$$
$$\{3,5,8\},\{3,6,7\},\{3,6,8\},\{3,7,8\},\{4,5,6\},\{4,5,7\},$$
$$\{4,5,8\},\{4,6,7\},\{4,6,8\},\{4,7,8\},\{5,6,7\},\{5,6,8\},$$
$$\{5,7,8\},\{6,7,8\}$$

□

3 M-kombinacija multiskupa elemenata

3.1 Uvod

U definiciji M-kombinacija skupa elemenata pokazano je na koliko načina možemo izabrati r podskupova od skupa sa n elemenata, tako da nije bitan redosled elemenata. Matematički se prethodno naziva i kombinacija bez ponavljanja. Za slučaj da želimo rešiti problem zasnovan na kolekcijama koje podržavaju ponavljanja elemenata, sa multiskupovima, koristimo se drugim matematičkim sredstvima koja se nazivaju M-kombinacije multiskupa elemenata ili kombinacije sa ponavljanjem. [3]

3.2 Intuicija

Najbolji način za razumevanje m-kombinacija multiskupa elemenata je putem primera. Zadatak je sledeći: Kupili smo 4 iste loptice za tenis koje trebamo rasporediti u 3 torbe za trening. Svaki neuređeni izbor k elemenata sa ponavljanjem skupa X može se, na jedinstven način, predstaviti pomoću niza od k markera (u našem primeru teniske loptice) i $|X|-1$ ivica ($|X|$ je jednak broju torbi za trening). Ivice se koriste da označe pregrade koje odgovaraju pojedinim elementima skupa X, dok svaki marker označava izbor elementa koji odgovara pregradi u kojoj se marker nalazi.[2] Znači, moramo zamisliti da se između svake dve torbe nalazi pregrada koja ih odvaja, što će u našem slučaju biti 2 pregrade. Nasumično raspoređujemo 4 loptice u 3 torbe (između pregrada).

Prethodni niz označava da su jednom izabrane torbe 1 i 3, dok je torba 2 izabrana 2 puta. U ovom primeru nije slučaj međutim moguce je da se pojave torbe koje ostaju prazne, što je isto legitiman slučaj pri rešavanju problema. Kako smo rasporedili elemente sabiramo broj loptica i broj pregrada što će nam dati $4 + 2 = 6$, tj. $6!$ mogućnosti za raspored svih elemenata. Međutim, pošto je naznačeno da imamo 4 iste loptice znači da dolazi do viška prebrojenih elemenata, stoga tih $6!$ mogućnosti treba podeliti sa $4!$ što bi bio broj svih mogućnosti za raspored različitih loptica, isto tako i sa brojem mogućnosti za raspored dve iste pregrade što iznosi $2!$. To bi matematički zapisali sa $\frac{6!}{4!2!}$, gde bi gor Kada izračunamo prethodno dobijemo 15 načina da se rasporede 4 loptice u 3 torbe. U sledećem odeljku predstavićemo uopštenje m-kombinacija multiskupa [5]

3.3 Definicija

Neka je dat multiskup:

$$M = [b_1, \ldots, b_l]_{m,\ldots,m} = \left\{ \underbrace{b_1, \ldots, b_1}_{m} \quad \ldots \quad \underbrace{b_l, \ldots, b_l}_{m} \right\}$$

Svaki m-točlani podmultiskup multiskupa M je m-kombinacija tog multiskupa. **Definicija m-kombinacija sa ponavljanjem**, ili **m-kombinacija multiskupa** kardinalnosti m, odabranih iz skupa M sa m elemenata, predstavlja neuredni izbor elemenata uzetih iz skupa M sa dozvoljenim ponavljanjem. [3] Broj m-kombinacija multiskupa M označavaćemo sa:

$$\overline{C} = (l; m)$$

Broj m kombinacija multiskupa $= [b_1, \ldots, b_l]_{m,\ldots,m}$ jednak je

$$\overline{C}(l; m) = \binom{l+m-1}{m} = \frac{(l+m-1)!}{m!(l-1)!}$$

Primer koji smo naveli u prethodnom odeljku rešava se po formuli za računanje kombinacija sa ponavljanjem, gde bi m predstavljalo broj teniskih loptica, a l bi predstavljao broj torbi u koje raspoređujemo loptice

Dokaz. Neka je:

$$\binom{M}{m} = \{M_1 : |M_1| = m \wedge M_1 \subseteq M\}$$

$$A = \left\{(a_1, \ldots, a_{m+l-1}) \in \{0,1\}^{m+l-1} : \{\{a_1, \ldots, a_{m+l-1}\}\} = [0,1]^{l-1,m}\right\}$$

Znači, $\binom{M}{m}$ je skup svih m-točlanih podmultiskupova od M, a A je SKup svih uređenih torki dužine m+l-1 koje imaju tačno m komponenti jednakih 1 i preostalih l - 1 komponenti jednakih 0.

Definišemo preslikavanje

$$\phi_{(b_1,\ldots,b_l)} : \binom{M}{m} \to A$$

na sledeći način

$$\phi_{(b_1,\ldots,b_l)}(M_1) = (c_1, \ldots, c_{m+l-1})$$

gde je $M_1 = [b_1, \ldots, b_l]_{m_1,\ldots,m_l}$ i za svako $i \in \{1, \ldots, l-1\}$

$$c_{(i-1)+(m_i+1)} = 0 c_{(i-1)+j} = 1, 0 < j \leq m_i$$

Kako je $\phi_{(b_1,\ldots,b_l)}$ bijektivno preslikavanje,

$$\overline{C}(l; m) = |\binom{M}{m}| = |A|$$

Broj načina da od $m + l - 1$ mesta izaberemo m za 1 i preostalih $l - 1$ za 0 jednak je

$$\overline{P}(m, l-1) = \frac{(m+l-1)!}{m!(l-1)!}$$

[6] □

3.4 Spona između m-kombinacija sa ponavljanjem i m-permutacija sa ponavljanjem

U raznoj matematičkoj literaturi i zadacima često se daje akcenat na formiranju veza između m-kombinacija i m-permutacija, neovisno o da li se radi o kombinatornim objektima sa ili bez ponavljanja. Pošto smo u prethodnim odeljcima pričali o m-kombinacijama sa ponavljanjem logičan sled bi bio da pričamo o njihovoj povezanosti sa m-permutacijama. Najlakši način da ispunimo prethodni zahtev bio bi da razumemo suštinu kombinacija i permutacija, i da ih tako analizirane povežemo u jednu celinu.

1. Odabir m-kombinacije: Prvo, izaberemo m-kombinaciju elemenata iz multiskupa. Na primer, iz multiskupa (M = a, a, b), možemo izabrati:
   - (aa) (2 puta (a))
   - (ab) (1 puta (a), 1 puta (b))
2. Permutacijom odabranih kombinacija: Nakon što dobijemo kombinacije, svaka kombinacija može biti raspoređena na različite načine, uzimajući u obzir ponavljanje elemenata.
   - Za kombinaciju (aa), postoji samo 1 permutacija: (aa).
   - Za kombinaciju (ab), postoje 2 permutacije: (ab, ba).
3. Kombinacija i Permutacija: Ukupni broj m-permutacija može se dobiti množenjem broja m-kombinacija sa ponavljanjem i brojem načina na koji te kombinacije mogu biti permutovane. Ova veza se može napisati kao:

   $$\text{Ukupan broj m-permutacija} = C(n + m - 1, m) \cdot m!$$

   gde je (C(n + m - 1, m)) broj m-kombinacija, a (m!) su različiti načini raspoređivanja m elemenata.

Na ovaj način, možemo razumeti kako se povezuju m-kombinacije sa ponavljanjem i m-permutacije sa ponavljanjem. Odabir kombinacija stvara osnovu za generisanje svih mogućih rasporeda tih kombinacija.

3.5 Algoritam za funkciju m-kombinaciju multiskupa

Algoritmi za m-kombinaciju multiskupa se koriste za pronalazenje svih mogucih kombinacija elemenata gde se elementi mogu ponavljati. Ova tema je znacajna u oblasti kombinatorike i ima primene u teoriji skupova, analizi podataka, kao i mnogim drugim matematickim disciplinama.

Algoritam za m-kombinaciju multiskupa koristi rekurziju za generisanje svih mogucih kombinacija elemenata. Ovaj algoritam koristi matematicke koncepte iz kombinatorike, posebno izracunavanje binomnih koeficijenata sa ponavljanjem. Broj m-kombinacija multiskupa sa beskonacnim brojem ponavljanja je dat binomnim koeficijentom sa ponavljanjem:

$$\binom{r + k - 1}{k - 1}$$

- Algoritam u jeziku Java:

```java
// Java program za generisanje m-kombinacije multiskupa
import java.util.*;

public class MultiCombination {
    public static void main(String[] args) {
        // Definisemo niz elemenata (multiskup)
        String[] elements = {"a", "b", "a", "c", "a"};
        // Definisemo broj elemenata u kombinaciji (m)
        int m = 3;
        // Dobijamo sve kombinacije i cuvamo ih u listi
        List<List<String>> combinations = getCombinations(elements,
            m);
        // Ispisujemo sve kombinacije
        for (List<String> combination : combinations) {
            System.out.println(combination);
        }
    }

    // Metoda za generisanje svih m-kombinacija multiskupa
    public static List<List<String>> getCombinations(String[]
        elements, int m) {
        // Lista za cuvanje rezultata
        List<List<String>> result = new ArrayList<>();
        // Sortiramo niz elemenata
        Arrays.sort(elements);
        // Pozivamo rekurzivnu metodu za generisanje kombinacija
        combine(elements, m, 0, new ArrayList<String>(), result);
        return result;
    }

    // Rekurzivna metoda za generisanje kombinacija
    private static void combine(String[] elements, int m, int
        start, List<String> combination, List<List<String>>
        result) {
        // Ako kombinacija ima m elemenata, dodajemo je u rezultat
        if (combination.size() == m) {
            result.add(new ArrayList<>(combination));
            return;
        }
        // Iteriramo kroz sve elemente pocevsi od 'start'
        for (int i = start; i < elements.length; i++) {
            // Izbegavamo duplikate
            if (i > start && elements[i].equals(elements[i - 1]))
                continue;
            // Dodajemo element u trenutnu kombinaciju
            combination.add(elements[i]);
            // Rekurzivno pozivamo metodu sa sledecim elementima
            combine(elements, m, i + 1, combination, result);
```

```
            // Uklanjamo poslednji element kako bismo istrazili
                druge mogucnosti
            combination.remove(combination.size() - 1);
        }
    }
}
```

Objasnjenje algoritma i rezultati
Rezultati algoritma prikazuju sve moguce 3-kombinacije elemenata iz skupa {"a", "b", "a", "c", "a"}. Primeri generisanih kombinacija su:

{"a", "a", "a"}
{"a", "a", "b"}
{"a", "a", "c"}
{"a", "b", "c"}
{"a", "a", "a"}
{"a", "a", "b"}

Algoritam funkcioniše tako što prvo sortira elemente multiskupa, zatim koristi rekurzivnu funkciju za generisanje kombinacija. Kombinacije se dodaju u listu rezultata kada dostignu potrebnu velicinu (m). Ovaj pristup osigurava da se sve moguce m-kombinacije pronađu bez ponavljanja.

Specifični koraci algoritma su sledeći:

1. Sortiranje niza elemenata radi lakšeg upravljanja i eliminisanja duplikata.
2. Korišćenje rekurzivne metode koja prolazi kroz sve elemente počevši od zadatog indeksa.
3. U svakom koraku dodavanje elementa u trenutnu kombinaciju i rekurzivni poziv za dodavanje sledećeg elementa.
4. Kada kombinacija dostigne velicinu m, dodavanje kombinacije u rezultujuću listu.
5. Uklanjanje poslednjeg elementa iz trenutne kombinacije kako bi se omogućilo istraživanje drugih mogućnosti.

Ovakav metod omogućava generisanje svih mogucih kombinacija bez ponavljanja, što je posebno korisno u analizi podataka i rešavanju optimizacionih problema.

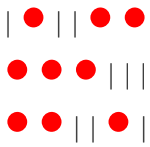
3.6 Primeri

Zadatak 5. *Napišite listu svih kombinacija multiskupova veličine 3, koji se mogu odabrati iz skupa* $\{1, 2, 3, 4\}$ *i uporedite konačan broj elemenata sa rešenjem do koga smo došli matematičkom formulom [3]*

Rešenje. Bitno je napomenuti da ako želimo da rešimo zadtak sa što manje grešaka trebamo postavljati elemente u multiskupove tako da su u neopadajućem poretku.

$$\begin{array}{llll} [1,1,1]; & [1,1,2]; & [1,1,3]; & [1,1,4] \\ [1,2,2]; & [1,2,3]; & [1,2,4]; & \\ [1,3,3]; & [1,3,4]; & [1,4,4]; & \\ [2,2,2]; & [2,2,3]; & [2,2,4]; & \\ [2,3,3]; & [2,3,4]; & [2,4,4]; & \\ [3,3,3]; & [3,3,4]; & [3,4,4]; & \\ [4,4,4] & & & \end{array}$$

Broj 3-kombinacija sa ponavljanjem jednak je broju 20.
Primetićemo da je ovaj proces veoma zahtevan, vremenski i fizički, što se najbolje primećuje kod rešavanja sličnih problema sa skupovima velikih kardinalnosti. Efikasniji način dolaska do rešenja zasniva se na posmatranju brojeva iz skupa $\{1,2,3,4\}$ kao kategorija, od kojih biramo 3 nasumične, neovisno o tome da li se ponavljaju ili ne. Rezultati nekoliko takvih odabira bi ovako izgledali:



Kao što se i da uočiti vertikalne linije odvajaju četiri kategorije, dok tačke predstavljaju izabran broj u toj kombinaciji. Uopštavanjem prethodnih slučajeva uočavamo da je broj tačaka uvek 3, a pregrada koje odvajaju kategorije isto 3, što kad ubacimo u formulu m-kombinacija sa ponavljanjem izgleda ovako:

$$\binom{6}{3} = \frac{6!}{3!(6-3)!} = 20$$

Pošto nam se rešenja poklapaju znači da prethodna formula odgovara sumi svih ispisanih elemenata.

□

Pokušajmo sada da rešimo sledeći zadatak, ali ovoga puta koristeći stečeno znanje o kombinacijama sa ponavljanjem.

Zadatak 6. *Koliki je broj neopadajućih nizova dužine l, čiji članovi su izabrani iz skupa $1, 2, \ldots, k$?*

Rešenje. Neopadajući nizovi koji se broje mogu se dobiti prvo izborom r-kombinacije multiskupa $S = \{\infty \cdot 1, \infty \cdot 2, \ldots, \infty \cdot k\}$, a zatim raspoređivanjem elemenata u neopadajućem redosledu. Dakle, broj takvih nizova je jednak broju r-kombinacija multiskupa S, i stoga je jednak:

$$\binom{l+k-1}{l}$$

□

[4]

Zadatak 7. *Koliko načina postoji da se izabere pet novčanica iz kase koja sadrži novčanice od 1$, 2$, 5$, 10$, 20$, 50$ i 100$? Pretpostavite da redosled u kojem su novčanice izabrane nije bitan i da su novčanice istovetne, i da ima najmanje pet novčanica svake vrste. [1]*

Rešenje. Pošto redosled kojim su novčanice izabrane nije bitan i sedam različitih vrsta novčanica može biti izabrano do pet puta, ovaj problem uključuje brojanje 5-kombinacija sa ponavljanjem iz skupa sa sedam elemenata. Navođenje svih mogućnosti bilo bi skupo, jer postoji veliki broj rešenja. Umesto toga, ilustrovaćemo upotrebu tehnike za brojanje kombinacija sa ponavljanjem. Pretpostavimo da kasa ima 7 odeljaka, jednu za svaku vrstu novčanica. Pošto ima 7 odeljaka, pregrada kojih se nalaze između njih ima n = 7 - 1 = 6. Nasumično odaberemo 5 novčanica, što označavamo sa kvadratićem. To bi u ovom slučaju izgledalo:

Broj načina da se izabere pet novčanica odgovara broju načina da se poređaju šest linija i pet zvezdica u red sa ukupno 11 pozicija. Shodno tome, broj načina da se izabere pet novčanica odgovara broju načina da se izaberu pozicije za pet kvadratića iz 11 pozicija. Ovo odgovara broju neuređenih izbora 5 objekata iz skupa od 11, što zapisujemo:

$$C(11,5) = \frac{11!}{5!6!} = 462$$

Znači postoji 462 načina da izaberemo 5 novčanica iz kase koja sadrži 7 vrsta novčanica □

4 Linearne jednačine sa n nepoznatih

4.1 Opsta forma linearne jednačine

Pocinjemo sa linearnom jednačinom:

$$a_1\ x_1\ + a_2\ x_2\ + \ldots + a_k\ x_k\ = b$$

4.2 Kombinatorni pristup

U ovom kontekstu, možemo zamisliti x_i kao broj "resursa" ili "jedinica" koje dodeljujemo različitim kategorijama, gde je a_i težina ili vrednost svakog resursa. Naš cilj je da raspodelimo resurse tako da ukupna vrednost bude jednaka b.

Zamislimo da imamo b resursa koje želimo da raspodelimo među k kategorija, pri čemu svaka kategorija i zahteva a_i jedinica resursa da bi se zadovoljila.

S obzirom na to, možemo postaviti problem:

$$x_1 + x_2 + \cdots + x_k = n$$

gde je n broj resursa koje imamo. Takođe, možemo povezati ovu jednacinu sa uslovom da x_i predstavlja nenegativne celine koje zadovoljavaju težinske koeficijente a_i.

4.3 Rešenje kao kombinacija

Rešenje ovog problema može se smatrati kao kombinacija načina na koje možemo raspodeliti n resursa. Ako posmatramo:

$$\sum_{i=1}^{n} a_i\ x_i\ = b$$

mi možemo koristiti kombinatoriku da izračunamo koliko načina možemo raspodeliti resurse.

4.4 Primena kombinatornih formula

Ako pretpostavimo da su svi x_i neodređeni i da njihovi koeficijenti a_i predstavljaju koliko resursa svaka kategorija zahteva, možemo zamisliti da imamo b resursa koje raspoređujemo u k kategorija.

Korišćenjem osnovne kombinatorne formule, broj načina na koji možemo raspodeliti b identičnih resursa u k kategorija je dat formulom:

$$\binom{k+n-1}{n}$$

Ova formula predstavlja broj kombinacija sa ponavljanjem.

Napomena 1. *Prema Uslovu simetricnosti imamo da je broj resenja jednacine* $x_1 + x_2 + \cdots + x_k = n$ *jednak* $\binom{k+n-1}{n} = \binom{k+n-1}{k-1}$

4.5 Primeri

Zadatak 8. *Na polici se nalazi n knjiga. Na koliko nacina se moze izabrati k knjiga sa police, tako da nikoje dve izabrane knjige nisu bile susedne na polici? [2]*

Rešenje. Neka x_1 oznacava broj knjiga na polici ispred prve izabrane knjige x_i, $2 \leq i \leq k$, broj knjiga na polici koje se nalaze između $(i-1)$-ve i i-te izabranje knjige, a $x_k + 1$ broj knjiga na polici iza poslednje izabrane knjige. S obzirom da brojevi x_i, $1 \leq i \leq k+1$ prebrojavaju sve neizabrane knjige sa police, vazi da je:

$$x_1 + x_2 + \ldots + x_k + x_{k+1} = n - k.$$

S druge strane, iz uslova da izabrane knjige nisu bile susedne vidimo da vazi:

$$x_2, x_3, \ldots, x_k > 1,$$

dok je

$$x_1, x_{k+1} > 0.$$

Da bismo izbegli ovakvo izdvajanje posebnih slucajeva, zamislimo da smo na policu dodali jos dve nove knjige: jednu ispred prve knjige na polici i jednu iza poslednje knjige na polici. Ako dozvolimo da x_1 i $x_k + 1$ prebrojavaju i ove dve nove knjige onda ce vaziti:

$$x_1 + x_2 + \ldots + x_k + x_{k+1} = n - k + 2,$$

$$x_1, x_2, \ldots, x_{k+1} > 1.$$

Neka je sada $y_i = x_i - 1$, $1 \leq i \leq k+1$, Tada su y_i nenegativni celi brojevi za koje vazi da je

$$y_1 + y_2 + \ldots + y_{k+1} = (n - k + 2) - (k + 1) = n - 2k + 1.$$

Iz definicije linearnih jednacina sa n nepoznatih vidimo da je broj resenja poslednje jednacine jednak

$$\binom{n \breve{} k+1}{n \breve{} 2k+1}$$

Ovo je ujedno i broj mogucih izbora k nesusednih knjiga sa police, s obzirom da niz brojeva y_i, $1 \leq i \leq k+1$ na jedinstven nacin određuje niz brojeva x_i, $1 \leq i \leq k+1$ a da on dalje na jedinstven nacin određuje izbor nesusednih knjiga. □

Zadatak 9. *Odrediti broj celobrojnih resenja jednacine [6]*

$$x_1 + x_2 + x_3 + x_4 + x_5 = 30,\ x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0,\ x_4 \geq 0,\ x_5 \geq 0$$

tako da vazi
a) $x_1 > 1,\ x_2 > 1,\ x_3 > 1,\ x_4 > 1,\ x_5 > 1$
b) $x_1 \leq 4$
c) $x_1 < 8,\ x_2 > 8$

Rešenje. **a)**Uvedimo smenu:

$$y_1 = x_1 - 2, y_2 = x_2 - 2, y_3 = x_3 - 2, y_4 = x_4 - 2, y_5 = x_5 - 2.$$

Kada je $x_1 > 1,\ x_2 > 1,\ x_3 > 1,\ x_4 > 1,\ x_5 > 1$ za nove promenljive vazi

$$y_1 \geq 0,\ y_2 \geq 0,\ y_3 \geq 0,\ y_4 \geq 0,\ y_5 \geq 0$$

Nakon uvođenja smene jednacina postaje

$$(y_1 + 2) + (y_2 + 2) + (y_3 + 2) + (y_4 + 2) + (y_5 + 2) = 30$$

odnosno

$$y_1 + y_2 + y_3 + y_4 + y_5 = 20.$$

Dobijena jednacina ima $\binom{20+5-1}{5-1} = \binom{24}{4}$ resenja, a svako njeno resenje na jedinstven nacin određuje resenje polazne jednacine.
b) Ako je $x_1 \leq 4$ to znaci da $x_1 \in \{0, 1, 2, 3, 4.\}$. Odredicemo posebno broj jednacina za svako $x_1 \in \{0, 1, 2, 3, 4.\}$
(i) za $x_1 = 0$ jednacina je

$$x_2 + x_3 + x_4 + x_5 = 30$$

Njeno resenje je $\binom{33}{3}$.
(ii) za $x_1 = 1$ jednacina je

$$x_2 + x_3 + x_4 + x_5 = 29$$

Njeno resenje je $\binom{32}{3}$.
(iii) za $x_1 = 2$ jednacina je

$$x_2 + x_3 + x_4 + x_5 = 28$$

Njeno resenje je $\binom{31}{3}$.
(iv) za $x_1 = 3$ jednacina je

$$x_2 + x_3 + x_4 + x_5 = 27$$

Njeno resenje je $\binom{30}{3}$.
(v) za $x_1 = 4$ jednacina je

$$x_2 + x_3 + x_4 + x_5 = 26$$

Njeno resenje je $\binom{29}{3}$.
Znaci konacan broj resenja polazne jednacine je

$$\binom{33}{3} + \binom{32}{3} + \binom{31}{3} + \binom{30}{3} + \binom{29}{3}$$

c) Ako prvo uvedemo smenu $y_2 = x_2 - 9 \geq 0$, onda je broj resenja zadate jednacine jednak broju resenja jednacine

$$x_1 + y_2 + x_3 + x_4 + x_5 = 21,\ x_1 \leq 7, y_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0.$$

Broj resenja prethodne jednacine mozemo dobiti ako od broja svih resenja jednacine

$$x_1 + y_2 + x_3 + x_4 + x_5 = 21,\ x_1 \geq 0, y_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0.$$

oduzmemo broj resenja jednacine

$$x_1 + y_2 + x_3 + x_4 + x_5 = 21,\ x_1 \geq 8, y_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0.$$

Za poslednju jednacinu mozemo uvesti smenu $y_1 = x_1 - 8 \geq 0$ cime se ona svodi na

$$y_1 + y_2 + x_3 + x_4 + x_5 = 21,\ y_1 \geq 0, y_2 \geq 0, x_3 \geq 0, x_4 \geq 0, x_5 \geq 0.$$

Tako je broj resenja zadate jednacine jednak

$$\binom{25}{4} - \binom{17}{4}$$

□

5 Zadaci

Backtracking algoritmi pokušavaju da reše računarske probleme postepeno, jedan po jedan mali komad. Kad god algoritam treba odlučiti između više alternativa za sledeću komponentu rešenja, rekurzivno procenjuje svaku alternativu i zatim bira najbolju. Primer koji ovaj algoritam rešava je problem sume podnizova.

Zadatak 10. *Pronaći sve podnizove niza $X = 8, 6, 7, 5, 3, 10, 9$ za koje važi da je zbir njihovih elemenata jednak $T = 15$. Elementi se mogu ponavljati.*

Rešenje. Primer Java algoritma :

```
public class SubsetSum {
public static List<List<Integer>> getSumSubsets(int[] array, int
    target) {
List<List<Integer>> res = new ArrayList<>();

search(0, new ArrayList<>(), 0, array, target, res);

return res;
}

private static void search(int i, List<Integer> curr, int
    currTotal, int[] array, int target, List<List<Integer>> res) {
if (currTotal == target) {
   res.add(new ArrayList<>(curr));
   return;
}

if (currTotal > target || i >= array.length) {
   return;
}

curr.add(array[i]);
search(i, curr, currTotal + array[i], array, target, res);

curr.remove(curr.size() - 1);

search(i + 1, curr, currTotal, array, target, res);
}


public static void main(String[] args) {
int[] array = {8, 6, 7, 5, 3, 10, 9};
int target = 15;

List<List<Integer>> result = getSumSubsets(array, target);

for (List<Integer> subset : result) {
   System.out.println(subset);
```

```
}
```

U ovom slučaju pomaže nam to što koristimo kombinacije a ne permutacije. Ne gledamo skupove 8,7 i 7,8 kao različita rešenja pa na taj način štedimo resurse. □

Zadatak 11. *U kriptografiji možemo izdvojiti Shamir's secret sharing. Radi tako što poruku koju treba sakriti šifruje i podeli na n delova. Zatim se odredi k, odnosno broj delova koji je dovoljan za rekonstruisanje originala. Iako se primarno oslanja na polinomsku interpolaciju, uočavamo da se izbor delova ključa vrši formiranjem (n/k) kombinacija.*

Rešenje. Kod u Javi sa visim nivoom apstrakcije :

```
public class ShamirSecretSharingExample {
public static void main(String[] args) {
int totalShares = 5; // Broj delova (shares)
int threshold = 3; // Broj delova potreban za rekonstrukciju

// tajna koju delimo (pretvaramo u niz bajtova)
byte[] secret = "nekitajnikod1234".getBytes();

Scheme scheme = new Scheme(new Random(), totalShares, threshold);

// delimo tajnu na delove (shares)
Map<Integer, byte[]> shares = scheme.split(secret);

System.out.println("Podeljeni delovi (shares):");
for (Map.Entry<Integer, byte[]> share : shares.entrySet()) {
    System.out.println("Delovi #" + share.getKey() + ": " +
    Hex.toHexString(share.getValue()));
}

Map<Integer, byte[]> selectedShares = new HashMap<>();

selectedShares.put(1, shares.get(1));
selectedShares.put(2, shares.get(2));
selectedShares.put(3, shares.get(3));

byte[] reconstructedSecret = scheme.join(selectedShares);
System.out.println("\nRekonstruisana tajna: " + new
     String(reconstructedSecret));

if (new String(reconstructedSecret).equals(new String(secret))) {
    System.out.println("\nTajna je uspesno rekonstruisana!");
} else {
    System.out.println("\nGreska pri rekonstrukciji tajne.");}
```

□

Reference

[1] Kenneth H. Rosen, *Discrete mathemathics and its appliacations Seventh Edition*, McGraw-Hill, 2012

[2] Dragan Stevanović i Miroslav Ćirić i Slobodan Simić i Vladimir Baltić, *DISKRETNA MATEMATIKA OSNOVE KOMBINATORIKE I TEORIJE GRAFOVA*, Matematički institut u Beogradu, 2007

[3] Sussana S. Epp, *Discrete Mathematics with Applications*, Cengage Learning, Inc. 2019

[4] Richard A. Brualdi, *Introductory Combinatorics, Fifth Edition*, Pearson Education,2009

[5] Chris Peach, *Stanford CS109 Probability for Computer Scientists I Combinatorics I 2022 I Lecture 2*, *Video*, Stanford University, 2023

[6] Jovanka Pantović, *Skrita: Kombinatorni objekti*, Fakultet Tehničkih Nauka, Univerzitet u Novom Sadu

[7] Art of the Problem, *Secret Sharing Explained Visually*, YouTube, 2019.

[8] Aman N., *Feature Selection in Machine Learning*, AnalyticsVid, 2020.

[9] Jeff Erickson, *Algorithms*, University of Illinois, 2019.

[10] James A. Anderson, *Diskretna matematika sa kombinatorikom, Prevod Drugog Izdanja*, Računarski Fakultet, 2005

[11] Petlja *Petlja*