Matematyka dyskretna

Andrzej M. Borzyszkowski

Instytut Informatyki
Uniwersytet Gdański

sem. zimowy 2025/26

inf.ug.edu.pl/~amb/MaD

Wariacje, kombinacje z powtórzeniami

Rozmieszczenia i wybory

- na ile sposobów można rozmieścić k identycznych przedmiotów w n (rozróżnialnych) pudełkach ($n > 0$)?
  odp.: $\binom{n+k-1}{n-1}$ czyli $\binom{n+k-1}{k}$
- dw.: układamy wszystkie pudełka w rząd, przedmiotom przypisujemy 0, ściankom pudełek 1
  - rozmieszczeniu przedmiotów definiuje ciąg zer i jedynek długości $n + k - 1$ mający k zer i $n - 1$ jedynek
  - każdy taki ciąg odpowiada jednemu rozmieszczeniu
- na ile sposobów można wybrać k przedmiotów należących do n rozróżnialnych typów?
- wybór typu przedmiotu to umieszczenie żetonu w odpowiednim pudełku, ta sama odpowiedź $\binom{n+k-1}{n-1}$

Kombinacje z powtórzeniami

- wybieramy k-krotnie element ze zbioru n rozróżnialnych elementów,
- nie jest ważna kolejność – ile istnieje różnych kombinacji?
- gdy wybrany element *nie* uczestniczy w dalszej procedurze, jest to wybór podzbioru k-elementowego w zbiorze n-elementowym,
- tw.: jeśli element jest zwracany po wyborze (może być powtórnie wybrany), liczba kombinacji wynosi $\binom{n+k-1}{n-1} = \binom{n+k-1}{k}$
- dw.: problem umieszczenia k przedmiotów w n pudełkach czyli wybór k przedmiotów n typów
- ile jest rozwiązań w $\mathbb{N}$ równania $x_0 + \ldots + x_{n-1} = k$?
  k jedynek rozdzielamy na n zmiennych: $\binom{n+k-1}{n-1}$

Przykład konkretny

- na ile sposobów można wybrać 5 owoców mając do wyboru jabłka, gruszki i pomarańcze?
  - 3 typy, 5 przedmiotów, $\binom{7}{5} = 21$ sposobów
- pięciokrotnie wybieramy z koszyka owoc, w koszyku jest jabłko, gruszka i pomarańcza, notujemy wybór ale wybrany owoc zwracamy
  - 3 różne elementy, 5 wyborów, $\binom{7}{5} = 21$ sposobów
- ile liczb ze zbioru $\{1, \ldots, 1000\}$ ma sumę cyfr 5 ?

  pięć jedynek rozdzielamy na trzy miejsca, $\binom{7}{5} = 21$ liczb

  005, 014, 023, 032, 041, 050, 104
  113, 122, 131, 140, 203, 212, 221
  230, 302, 311, 320, 401, 410, 500

Różne wybory

wybór k elementów z n dostępnych	porządek się liczy (wariacja/permutacja)	porządek bez znaczenia (kombinacja)
bez zwracania, $n \geqslant k$	$\frac{n!}{(n-k)!}$	$\binom{n}{k}$
ze zwracaniem, $n \geqslant 1$	n^k	$\binom{n+k-1}{k}$

Wielozbiory

- zbiór $A \subseteq U$ ma funkcję charakterystyczną $\chi_A : U \to 2$
- wielozbiory pochodzą od funkcji $f : U \to \mathbb{N}$
  tzn. elementy mogą występować wielokrotnie
  ale kolejność wystąpień się nie liczy
- trzy kolejne poziomy abstrakcji:
  - ciąg $(a_0, a_1, \ldots, a_{k-1})$ długości k elementów z U
  - wielozbiór $[n_0 \cdot a_0, \ldots n_\ell \cdot a_\ell]$ liczności $\Sigma_{i=0}^{\ell} n_\ell$
  - zbiór $\{a_0, a_1, \ldots, a_\ell\}$ – wypisane elementy są różne
- kombinacja z powtórzeniami to wybór wielozbioru nad zbiorem wybieranych elementów – ile razy dany element
- wybór przedmiotów odróżnialnych według typu to wybór wielozbioru nad zbiorem typów – ile elementów danego typu

Generowanie różnych obiektów kombinatorycznych

Generowanie wszystkich podzbiorów zbioru n

- rozwiązanie 1: funkcje charakterystyczne podzbiorów traktujemy jako liczby w zapisie binarnym, wypisujemy je w naturalnym porządku
  - najmniejsza liczba to zero
  - w każdym ruchu dodajemy 1
  - czyli szukamy zera na najniższej pozycji, zamieniamy je na 1, a wszystkie wcześniejsze jedynki na zero
- 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111

```
int next(int mask[], int n){ //kolejny podzbior
  for (i=0; (i<n) && mask[i]; ++i) mask[i]=0;
  if (i<n) {mask[i]=1; return 1;}
  else return 0;
}
```

https://compprog.wordpress.com/2007/10/10/generating-subsets/

Generowanie podzbiorów – kod Graya

- rozwiązanie 2 (kod Graya, definicja rekursywna):
  - dla $n = 0$ ciąg pusty jest rozwiązaniem problemu
  - ustalamy porządek generowania dla n
  - dla $n + 1$ wypisujemy wszystkie ciągi dodając 0 z przodu
  - zamieniamy je na 1 i wypisujemy w porządku odwrotnym
- 0000, 0001, 0011, 0010, 0110, 0111, 0101, 0100, 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000
- dwa kolejne elementy różnią się jednym bitem, w przeciwieństwie do poprzedniego rozwiązania

Generowanie podzbiorów k-elementowych zbioru n

- ze wzoru na przekątną trójkąta Pascala $\Sigma_{i=k-1}^{n-1} \binom{i}{k-1} = \binom{n}{k}$
- najpierw wybór największego elementu, potem generowanie podzbiorów złożonych z mniejszych liczb

```
subsets(int n, int k){ if (n>0)
    for (i=k-1;i<n;i++) subsets_max(i,k-1);
}
//subsets_max(i,k-1) dodaje element i
//do zbiorow utworzonych przez subsets(i,k-1)
```

- subsets(7,5) wypisze po kolei (notacja bez przecinków i nawiasów)

  43210, 53210, 54210, 54310, 54320, 54321, 63210, 64210, 64310, 64320, 64321, 65210, 65310, 65320, 65321, 65410, 65420, 65421, 65430, 65431, 65432.

Podziały nieuporządkowane

- każdy z elementów podziału będzie wypisany w/g zasady: wybór największego elementu, potem generowanie podzbiorów złożonych z mniejszych liczb
  - dodatkowo podzbiory w podziale też będą generowane od największych elementów
- np. podział sześciu wierzchołków na pary $\frac{6!}{(2!)^3 \cdot 3!} = 15$ podziałów

65 43 21	65 42 31	65 41 32	64 53 21	64 52 31,
64 51 32	63 54 21	63 52 41	63 51 42	62 54 31,
62 53 41	62 51 43	61 54 32	61 53 42	61 52 43

Reprezentacje wielozbiorów

- przykład: dwa jabłka, trzy gruszki, zero pomarańczy

1. elementy (numerowane) wielozbioru (wybór ze zwracaniem): 22111 – funkcja nierosnąca $f : 5 \to 3$
2. elementy zbioru (sztuczne „zaostrzenie" nierówności): 65321 – funkcja malejąca $f : 5 \to 7$
3. funkcja charakterystyczna wielozbioru (zliczanie liczby wyborów): 230 – funkcja $f : 3 \to \mathbb{N}$ taka, że $\Sigma f = 5$
4. funkcja charakterystyczna czy wybór elementu, czy zmiana typu: 1101110 – funkcja $f : 7 \to 2$ taka, że $\Sigma f = 5$

- przykład: jabłko i cztery pomarańcze, 20000, 63210, 104, 1001111

Generowanie wielozbiorów

- najpierw wybór największego elementu, potem generowanie wielozbiorów złożonych z liczb *nie większych*

```
multisets(int n, int k){ if (k>0)
    for (i=0;i<n;i++) multisets_max(i,k-1);
}
//multisets_max(i,k-1) dodaje element i
//do zbiorow utworzonych przez multisets(i+1,k-1)
```

- przykład: 5 losowań ze zwracaniem w zbiorze 3-elementowym można wykonać na $\binom{7}{2} = 21$ sposobów
- multisets(3,5) wypisze po kolei (notacja bez przecinków i nawiasów)

Generowanie wielozbiorów

1. typ dla każdego wyboru:
   00000, 10000, 11000, 11100, 11110, 11111, 20000,
   21000, 21100, 21110, 21111, 22000, 22100, 22110,
   22111, 22200, 22210, 22211, 22220, 22221, 22222.
2. funkcja malejąca zamiast nierosnącej
   43210, 53210, 54210, 54310, 54320, 54321, 63210
   64210, 64310, 64320, 64321, 65210, 65310, 65320
   65321, 65410, 65420, 65421, 65430, 65431, 65432.
3. liczba elementów danego typu
   005, 014, 023, 032, 041, 050, 104,
   113, 122, 131, 140, 203, 212, 221,
   230, 302, 311, 320, 401, 410, 500.
4. wybór elementu czy zmiana typu
   0011111, 0101111, 0110111, 0111011, 0111101, 0111110, 1001111
   1010111, 1011011, 1011101, 1011110, 1100111, 1101011, 1101101
   1101110, 1110011, 1110101, 1110110, 1111001, 1111010, 1111100.

Generowanie permutacji – algorytm Dijkstry

- zadanie: wypisać wszystkie permutacje zbioru n
- rozwiązanie: mamy do dyspozycji „cyfry" $0, 1, \ldots, n-1$
  permutacje traktujemy jako liczby w układzie liczenia o podstawie n
  wypisujemy je w kolejności rosnącej
  - najmniejsza liczba to cyfry w kolejności rosnącej
  - dla każdej liczby szukamy ostatniej cyfry, po której dalszy ciąg cyfr jest malejący
  - by mieć liczbę większą niż bieżąca zmieniamy początek jej zapisu – zamieniamy znalezioną cyfrę na kolejną dostępną w końcówce zapisu liczby
  - końcówkę porządkujemy w kolejności rosnącej
- 0123, 0132, 0213, 0231, 0312, 0321,
  1023, 1032, 1203, 1230, 1302, 1320,
  2013, 2031, 2103, 2130, 2301, 2310,
  3012, 3021, 3102, 3120, 3201, 3210.

Generowanie permutacji, kod

```
int Value[];
void getNext(int N)
{
   int i = N-1;
   while(Value[i-1] >= Value[i]) i--; //ogon maleje
   int j = N;
   while(Value[j-1] <= Value[i-1]) j--;
   //kolejna wartosc z pozostalych
   swap(i-1, j-1);
   i++; j = N;
   while(i < j){  //odbicie zwierciadlane ogona
     swap(i-1, j-1); i++; j--; }
}
```

https://www.cut-the-knot.org/do_you_know/AllPerm.shtml