Міністерство освіти і науки України
Нацональний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

Лабораторна робота № 6
з дисципліни
«Дискретна математика»

Виконав:
Студент групи КН-114
Кратко Денис

Викладач:
Мельникова Н.І.

Львів – 2019

Тема: Генерація комбінаторних конфігурацій

Мета роботи: набути практичних вмінь та навичок при комп'ютерній реалізації комбінаторних задач.

Теоретичні відомості

Головна задача комбінаторики – підрахунок та перелік елементів у скінчених множинах.

Правило додавання: якщо елемент – х може бути вибрано n способами, а у- іншими m способами, тоді вибір „ х або у" може бути здійснено (m+n) способами.

Правило добутку: якщо елемент – х може бути вибрано n способами, після чого у - m способами, тоді вибір упорядкованої пари (х, у) може бути здійснено (m*n) способами.

Набір елементів $x_{i1}, x_{i2}, ..., x_{im}$ з множини $X = \{x_1, x_2, ..., x_n\}$ називається вибіркою об'єму m з n елементів – *(n, m) – вибіркою*.

Упорядкована (n, m) – вибірка, в якій елементи не можуть повторюватися, називається *(n, m) – розміщеням*, кількість всіх можливих розміщень обчислюється за формулою:

$$A_n^m = \frac{n!}{(n-m)!}.$$

Упорядкована (n, m) – вибірка, в якій елементи можуть повторюватися, називається *(n, m) – розміщеням з повторюваннями*, кількість всіх можливих таких розміщень обчислюється за формулою:

$$\overline{A_n^m} = n^m.$$

Неупорядкована (n, m) – вибірка, в якій елементи не можуть повторюватися, називається *(n, m) – сполученням*, кількість всіх можливих сполучень обчислюється за формулою:

$$C_n^m = \frac{n!}{m!(n-m)!}.$$

Неупорядкована (n, m) – вибірка, в якій елементи можуть повторюватися, називається *(n,m)-сполученням з повторюваннями*, кількість всіх можливих таких сполучень обчислюється за формулою:

$$\overline{C_n^m} = C_{n+m-1}^m.$$

A_n^n – називається *перестановкою*, а кількість різних перестановок позначається та обчислюється за формулою:

$$P_n = n!.$$

Якщо в перестановках є однакові елементи, а саме перший елемент присутній n_1 разів, другий елемент – n_2 разів, ... , k-ий елемент – n_k разів, причому $n_1+n_2+....+n_k=n$, то їх називають *перестановками з повторенням* та кількість їх можна знайти за формулою

$$P(n_1, n_2, ..., n_k) = \frac{n!}{n_1!n_2!...n_k!}.$$

Нехай $X = \{X_1, X_2,...,X_k\}$ - *розбиття множини* X ($X = n$) на k підмножин таких, що: $\bigcup_{i=1}^{k} X_i = X$, $X_i \cap X_j = 0$ при $i \neq j$,

$|X_i| = n_i$.

Їх кількість при фіксованих n_i та *упорядкованих* $X_1, X_2,..., X_k$ обчислюється за формулою:

$$C_n^{n_1,n_2,\ldots,n_k}(n_1, n_2, \ldots, n_k) = \frac{n!}{n_1!n_2!\ldots n_k!}.$$

Якщо ж множину X ($|X| = n$) потрібно розбити на підмножини, серед яких для усіх i=1, …, n є $m_i \geq 0$ підмножин з i елементами, де $\sum_{i=1}^{n} i * m_i = n$, та при цьому набір підмножин в розбитті *не є упорядкованим,* тоді їх кількість обчислюється за формулою: $n!$

$$N(m_1, m_2, \ldots, m_n) = \frac{n!}{m_1!m_2!\ldots m_n!(1!)^{m1}(2!)^{m2}\ldots(n!)^{m_n}}.$$

Формула включень та виключень. Нехай X_i – скінчені множини, де i=1,…,n, тоді:

$$|X_1 \cup \ldots \cup X_n| = (|X_1| + \ldots + |X_n|) - (|X_1 \cap X_2| + \ldots + |X_{n-1} \cap X_n|) +$$

$$+ (|X_1 \cap X_2 \cap X_3| + \ldots + |X_{n-2} \cap X_{n-1} \cap X_n|) - \ldots + (-1)^{n+1}|X_1 \cap \ldots \cap X_n|.$$ Наслідок.

$$|X \setminus (X_1 \cup \ldots \cup X_n)| = |X| - (|X_1| + \ldots + |X_n|) +$$

$$(|X_1 \cap X_2| + \ldots + |X_{n-1} \cap X_n|) - \ldots + (-1)^n |X_1 \cap \ldots \cap X_n|.$$

Приведемо ще одну форму запису формули включень та виключень. Нехай X – скінчена множина з N елементів, $\alpha_1,\ldots,\alpha_n$ – деякі властивості, якими володіють чи ні елементи з X. Позначимо через $X_i = \{x \in X | \alpha_i(x)\}$ – множину елементів в X, які володіють властивістю α_i, а

$N(\alpha_{i1},\ldots,\alpha_{ik}) = |X_{i1} \cap \ldots \cap X_{ik}| = |\{x \in X | \alpha_{i1}(x) \wedge \ldots \wedge \alpha_{ik}(x)\}|$ – кількість

елементів в X, які володіють одночасно властивостями $\alpha_{i1},\ldots,\alpha_{ik}$, $N_0 = |X \setminus (X_1 \cup \ldots \cup X_n)|$ - кількість елементів, що не володіють жодною з властивостей $\alpha_{i1}, \ldots, \alpha_{ik}$. Тоді маємо формулу:

$N_0 = N - S_1 + S_2 - \ldots + (-1)^n S_n$, де $S_k = \sum_{1 \leq i_1,\ldots,i_k \leq n} N(\alpha_{i1},\ldots,\alpha_{ik})$ $k = 1,2,\ldots,n.$

Якщо треба знайти кількість елементів, які володіють рівно m властивостями, тоді використовують наступну формулу:

$$\hat{N}_m = \sum_{k=0}^{n-m} (-1)^k C_{m+k}^m S_{m+k}.$$

Лексикографічний порядок – це природний спосіб упорядкування послідовностей на основі порівняння індивідуальних символів. На множині всіх розміщень із *r* елементів означимо порівняння таким чином: $b_1b_2...b_r < a_1a_2...a_r$, якщо $\exists m:\left(b_i = a_i\ , i < m\right)\wedge(b_m < a_m)$.

У такому разі говорять, що перестановка $b_1b_2...b_r$ *менша* від перестановки $a_1a_2...a_r$, або перестановка $a_1a_2...a_r$ *більша* від перестановки $b_1b_2...b_r$.

Алгоритм побудови лексикографічно наступного розміщення з повтореннями за розміщенням $a_1a_2...a_r$

Алгоритм подібний до звичайного визначення наступного числа.

Крок 1. Знаходимо позицію *k* першого справа числа, відмінного від *n* $a_k < n$.

Крок 2. Збільшуємо елемент a_k на одиницю. Елементи a_i, де $i < k$ залишаються без змін. Елементи a_i, де $i > k$ стають рівними одиниці.

Приклад. Нехай $A = \{1, 2, 3\}$. Побудуємо 6 розміщень з повтореннями лексикографічно наступних після 1222. Наступне буде 1223, так як можливо збільшити останній елемент. Після нього буде 1231, оскільки 4-й елемент ми збільшити не можемо, то збільшуємо 3-й, а 4-й ставимо рівним одиниці. Як бачимо, маємо аналогію з переносом розряду, подібно до десяткового числення. Наступні елементи, відповідно будуть 1232,1233, 1311 та 1312.

Алгоритм побудови лексикографічно наступного розміщення без повторень за розміщенням $a_1a_2...a_r$

Алгоритм від попереднього відрізняється тим, що у розміщеннях не може бути повторів, і тому потрібно "оновлювати" елементи розміщення, не порушуючи цієї властивості.

Крок 1. Знайдемо множину *B* "вільних" чисел, яких немає у розміщенні $a_1a_2...a_r$: $B = A\setminus\{a_1a_2...a_r\}$.

Крок 2. Шукаємо перший справа елемент у розміщенні, який можна збільшити. Якщо у *B* є елементи, які більші за a_r, то вибираємо серед них такий, що: $b_r = \min\limits_{x\in B}\{x > a_r\}$.

Якщо у *B* немає елементів, більших за a_r, то додаємо до *B* елемент a_r, $B = B\cup\{a_r\}$ і шукаємо: $b_{r-1} = \min\limits_{x\in B}\{x > a_{r-1}\}$.

Якщо у *B* немає елементів, більших за a_{r-1}, то додаємо до *B* елемент a_{r-1}, і т.д. Продовжуємо цей процес, поки не знайдемо: $b_k = \min\limits_{x\in B}\{x > a_k\}$, або не дійдемо до початку розміщення. Якщо такого b_k не знайдено, то ми дійшли до максимального елементу, і алгоритм завершено. Якщо ні, то переходимо до кроку3.

Крок 3. Обчислюємо наступне розміщення. Записати в k-ту позицію розміщення знайдене число b_k і вилучити його з множини *B*. Записати у висхідному порядку число b_{k+1}, b_{k+2}, b_{k+3} … b_r – найменші з чисел у множині *B*, розмістивши їх на позиціях *k + 1, k + 2,..., r*.

Приклад. Нехай $A = \{1, 2, 3, 4\}$. Побудуємо 4 розміщень без повтореннями лексикографічно наступних після 123. Наступне буде 124, так як можливо збільшити останній елемент (3).

Обчислимо наступне розміщення після 124. Оскільки 3-й елемент ми збільшити не можемо, то збільшуємо 2-й елемент, тобто заміняємо "2" на "3". На останньому місці не можна ставити одиницю, бо вона вже є у розміщенні, і ставимо найменший елемент, такий що його немає в розміщенні , тобто "2". Отже отримуємо 132 .

Наступне розміщення після 132 рівне 134, тому що можливо збільшити останній елемент. Яке розміщення буде після 134? Останній елемент ми не можемо збільшити, але можемо збільшити передостанній. Тому наступне розміщення 142.

Алгоритм побудови лексикографічно наступної перестановки за перестановкою $a_1a_2...a_n$

Наведемо кроки алгоритму.

Крок 1. Знайти такі числа a_j і a_{j+i}, що $(a_j<a_{j+1},)\ \widehat{}\ (a_{j+i} \geq a_{j+2} \geq ... \geq a_n)$. Для цього треба знайти в перестановці першу справа пару сусідніх чисел, у якій число ліворуч менше від числа праворуч.

Крок 2. Записати в *j*-ту позицію таке найменше з чисел $a_{j+1}, a_{j+2},..., a_n$, яке водночас більше, ніж a_j.

Крок 3. Записати у висхідному порядку число a_j і решту чисел $a_{j+1}, a_{j+2},...,a_n$ у позиції *j + 1,..., n*.

Приклад. Побудуємо 2 перестановки, наступні в лексикографічному порядку за 34521. Згідно першого кроку *j=2*, бо 4<5>2>1. Отже, перше число (3) залишається на місці, а збільшується друге число(4). Розглянемо послідовність чисел 521. Серед них найменше число, більше від 4, це 5. Тепер на другому місці 5, а решту чисел розміщуємо у вихідному порядку: 35124.

Побудуємо наступну перестановку після 35124. Згідно першого кроку *j =4,* і щоби отримати наступну перестановку, треба збільшити "2", поставивши замість нього "4", так як справа немає іншого числа більше "2". Переставивши місцями два останніх числа, ми отримаємо 35142.

Зауважимо також, що для перебору перестановок з повтореннями чи без повторень алгоритм не відрізнятиметься. Єдина відмінність полягатиме у тому, що для перестановок без повторень рівності у кроці 1 будуть строгими.

Алгоритм побудови лексикографічно наступного сполучення з повтореннями за сполученням $a_1a_2...a_r$

Алгоритм подібний до алгоритмів генерування розміщень, але має одну особливість, яка полягає в наступному: якщо сполучення впорядковане у висхідному порядку, то кожен наступний елемент сполучення не менший за попередній.

Крок 1. Знаходимо позицію *k* першого справа числа, відмінного від *n*: $a_k < n$.

Крок 2. Збільшуємо елемент a_k на одиницю $b_k = a_k + 1$.

Елементи зліва a_i залишаються без змін $b_i = a_i$, де $i < k$.

Елементи справа a_i, де $i > k$ стають рівними b_k, $b_i = b_k$, де $i > k$.

Приклад. Нехай A = {1, 2 , 3, 4}. Побудуємо 7 сполучень з повтореннями лексикографічно наступних після 1233. Наступне буде 1234, так як можливо збільшити останній елемент.
При пошуку наступного сполучення після 1234 бачимо, що збільшувати можна "3". Отже отримаємо 124. Останній елемент теж повинен бути рівний "4", бо не може бути менший за попередній. Отже отримуємо 1244.
Після нього буде 1333 - оскільки ми можемо збільшити тільки "2", а інші елементи мають бути такими самими. Аналогічно знаходимо наступні елементи: 1334, 1344, 1444 та 2222.

Алгоритм побудови лексикографічно наступного сполучення без повторень за сполученням $a_1a_2...a_r$

Перед тим як розглянути алгоритм, розглянемо одну особливість. Якщо сполучення впорядковане у висхідному порядку і не має повторів, то кожен наступний елемент сполучення більший від попереднього принаймі на одиницю.
Тоді максимальне значення, яке може набувати його останній елемент, рівне n. Максимум для передостаннього елементу рівний $n-1$, а не n. Доведемо це від зворотнього. Припустимо. що останній елемент рівний n, тоді наступний елемент має бути рівний $n+1$, але такого елементу немає в множині.

Отже максимум передостаннього елементу $n-1$. Аналогічно можна довести, що максимум елементу на k-ій позиції рівний $n-(r-k)$. Мінімум елементу – попереднє число сполучення, збільшене на одиницю.
Крок 1. Знайдемо перший справа елемент сполучення, який можна збільшувати. Він має бути менший за свій допустимий максимум, тобто $a_k < n - r + i$.

Крок 2. Збільшимо елемент a_k на одиницю $b_k = a_k + 1$.

Крок 3. Елементи зліва від a_i не змінюємо $b_i = a_i$, $i<k$.

Крок 4. Елементи справа змінюємо на мінімальні, тобто такі, що на одиницю більші від попереднього:
$b_i = b_{i-1} + 1 = a_k + i - k$, $i>k$.
Приклад. Нехай A={1,2,3,4,5}. Знайдемо сполучення, наступне за {1,2,5,6} у лексикографічному порядку. Це сполучення подамо рядком 1256. Маємо n = 6, r= 4. Перший справа з таких елементів, що $a_i \neq 6 - 4 + i$, — це a_2=2. Для обчислення наступного більшого сполучення збільшуємо a_2 на 1 та одержуємо a_2 = 3. Тепер нехай a_3 = 3+1=4 і a_4 = 3 + 2 = 5. Отже, наступне в лексикографічному порядку сполучення — те, що зображене рядком 1345, тобто {1, 3, 4, 5}.

Біномом Ньютона називають формулу для обчислення виразу $(a+b)^n$ для натуральних n.

$$(a+b)^n = \sum_{k=0}^{n} C_n^k a^k b^{n-k}$$

Завдання (12 варіант)

Додаток 1

Умова

Використовуючи теоретичні відомості, розв'язати наступні комбінаторні задачі за своїм варіантом.

Варіант № 12

1. В дитячому садку 10 хлопчиків. Скільки є способів одягнути їх в новорічні костюми: а) якщо є 10 різних костюмів; б) є 2 костюми зайців, 5 - ведмежат і 3 - білочок.
2. Скільки різних чотирицифрових чисел можна скласти з цифр
1, 2, 3, 4, 5, 6, якщо кожну з них використовувати при записи числа лише один раз?
3. У вазі стоїть пронумеровані 10 червоних і 5 рожевих гвоздик. Скількома способами можна вибрати з вази три квітки?
4. У чемпіонаті України з футболу грає 18 команд. Скількома способами можуть розподілити місця, якщо відомо, що команди «Динамо», «Дніпро», «Шахтар», «Чорноморець» і «Таврія» займуть перші п'ять місць?
5. Скількома способами можна поділити 15 однакових цукерок між п'ятьма дітьми?
6. Дванадцять атлетів треба розподілити на 2 групи по 3 атлета, та 3 групи по 2 атлета для змагань на різні дистанції, при цьому кожна з цих груп може поїхати на змагання в одне з трьох можливих міст. Скількома способами можна розподілити атлетів на необхідні групи та для кожної з них вибрати місто для змагання?
7. На одній з кафедр університету працює 13 чоловік, кожен з яких знає хоча б одну іноземну мову. 10 чоловік знають англійську, 7 – німецьку, 6 – французьку, 5 – англійську та німецьку, 4 – англійську та французьку, 3 – німецьку та французьку. Скільки чоловік: а) знають всі три мови; б) знають тільки дві мови; в) знають лише англійську?

Розв'язки

1. А) $P_{10} = 10! = 1 * 2 * 3 * 4 * 5 * 6 * 7 * 8 * 9 * 10 = 3628800$

   Б) $P(2;5;3) = \frac{10!}{2!5!3!} = \frac{5!*6*7*8*9*10}{2*5!*6} = \frac{7*8*9*10}{2} = 2520$

2. $A_6^4 = \frac{6!}{(6-4)!} = \frac{6!}{2!} = 3 * 4 * 5 * 6 = 360$

3. $C_{10+5}^3 = C_{15}^3 = \frac{15!}{3!(15-3)!} = \frac{15!}{3!12!} = \frac{13*14*15}{6} = 455$

4. $P_5\ P_{18-5} = P_5 P_{13} = 5! * 13! = 120 * 6227020800 = 747242496000$

5. Кожній дитині має дістатись принаймні по цукерці. Спочатку роздамо по 1 цукерці кожному з 5 дітей, залишиться 10 цукерок. Тепер для кожної з 10 цукерок є 5 варіантів, якій дитині її дати. Отже, способів $\overline{A_5^{10}} = 5^{10} = 9765625.$

6. $C_{12}^3 C_9^3 C_6^2 C_4^2 C_2^2 \overline{A_3^5} = \frac{12!}{3!9!} * \frac{9!}{3!6!} * \frac{6!}{2!4!} * \frac{4!}{2!2!} * \frac{2!}{2!0!} * 3^5 = \frac{12!}{3!} * \frac{1}{3!} * \left(\frac{1}{2!}\right) * \frac{1}{2!} * \frac{1}{2!0!} * 3^5 = 3^5 * \frac{1}{8} * \frac{12!}{36} = 3^5 * 1663200 = 243 * 1663200 = 404157600$

7.

$|A \cup N \cup F| = 13, |A| = 10, |N| = 7, |F| = 6, |A \cap N| = 5, |A \cap F| = 4, |N \cap F| = 3$

$|A \cup N \cup F| = |A| + |N| + |F| - (|A \cap N| + |A \cap F| + |N \cap F|) + |A \cap N \cap F|$

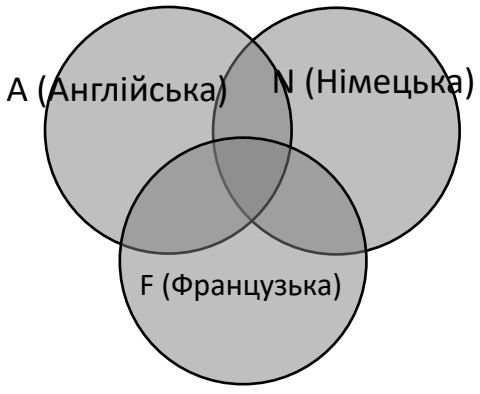
$A)\ 13 = 10 + 7 + 6 - (5 + 4 + 3) + |A \cap N \cap F|$

$13 = 23 - 12 + |A \cap N \cap F|$

Отже, $|A \cap N \cap F| = 2$ (знають всі три мови).

Б) Знають тільки дві мови $(((|A \cap N| + |A \cap F| - |A \cap N \cap F|) + |N \cap F|) - |A \cap N \cap F|) - |A \cap N \cap F|) = |A \cap N| + |A \cap F| + |N \cap F| - 3|A \cap N \cap F| = 5 + 4 + 3 - 3 * 2 = 12 - 6 = 6$ людини.

В) $|N \cup F| = |N| + |F| - |N \cap F| = 7 + 6 - 3 = 10$



Знають лише англійську $|A \cup N \cup F| - |N \cup F|$
$= 13 - 10 = 3$ людини.

Додаток 2

Запрограмувати за варіантом обчислення кількості розміщення (перестановок, комбінацій, алгоритму визначення наступної лексикографічної сполуки, перестановки) та формулу Ньютона і побудувати за допомогою неї розклад за варіантом.

Задане додатне ціле число n. Розташувати у лексикографічному порядку всі перестановки множини {1, 2, …, n}. Побудувати розклад $(a+b)^{10}$.

Розв'язок

```cpp
#include <bits/stdc++.h>

using namespace std;

struct drib {
    unsigned long int a;
    unsigned long int b;
};

drib multiply_drib_by(drib d, unsigned long int a, unsigned long int b) {
    d.a *= a;
    d.b *= b;
    unsigned long int m = max(d.a, d.b);
    while (m > 0) {
        if (!(d.a % m) && !(d.b % m)) {
            d.a = d.a / m;
            d.b = d.b / m;
        }
        m--;
    };
    return d;
};

void swap(unsigned long int* a, unsigned long int* b) {
    unsigned long int temp = *a;
    *a = *b;
    *b = temp;
};

void my_sort(unsigned long int* arr, unsigned long int n) {
    // bubblesort from https://www.geeksforgeeks.org/bubble-sort/
    unsigned long int i, j;
    for (i = 0; i < n-1; i++) {
        // Last i elements are already in place
        for (j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                swap(arr + j, arr + (j+1));
            };
        };
    };
};

bool next_permutation(unsigned long int* arr, unsigned long int n) {
    if (n > 1) {
        unsigned long int j, i, min_i;
        for (j = n - 2; (arr[j] >= arr[j + 1]); j--) {
            if (j == 0) {
```

```
                return false;
            };
        };
        min_i = j + 1;
        for (i = j + 1; i < n; i++) {
            if ((arr[i] < arr[min_i]) && (arr[i] > arr[j])) {
                min_i = i;
            };
        };
        swap(arr + j, arr + min_i);
        my_sort(arr + (j + 1), n - (j + 1));
        return true;
    }
    else {
        return false;
    };
};

void print_array(unsigned long int* arr, unsigned long int n, unsigned long int
counter) {
    cout << "Permutation " << counter << ": [";
    if (n > 0) {
        cout << arr[0];
        for (unsigned long int i = 1; i < n; i++) {
            cout << ", " << arr[i];
        };
    }
    cout << "]" << endl;
};

void print_binom(unsigned long int n) {
    unsigned long int k;
    drib c = {1, 1};
    for (k = 0; k < n; k++) {
        cout << "(" << c.a << "/" << c.b << ")";
        cout << "*(a^" << k << ")*(b^" << (n - k) << ")";
        cout << " + ";

        c = multiply_drib_by(c, n - k, k + 1);
    };

    cout << "(" << c.a << "/" << c.b << ")";
    cout << "*(a^" << k << ")*(b^" << (n - k) << ")";
    cout << endl;
};

int main() {
    unsigned long int i, n;
    do {
        cout << "Enter n=";
        cin >> n;
    } while (n == 0);
    // 1 2 3 4 5 6 7 8 9 10
    // ...
    // 10 9 8 7 6 5 4 3 2 1
    unsigned long int* a = (unsigned long int*)calloc(n, sizeof(unsigned long int));
    for (i = 0; i < n; i++) {
        a[i] = i + 1;
    };

    unsigned long int counter = 0;
    do {
```

```
        counter++;
        print_array(a, n, counter);
    } while (next_permutation(a, n));

    free(a);

    cout << endl << "(a + b)^10 = ";
    print_binom(10);

    return 0;
};
```

Результат виконання програми

1 випадок

```
"D:\Dyskretna laboratorni\6\cmake-build-debug\6.exe"
Enter n=0
Enter n=0
Enter n=5
Permutation 1: [1, 2, 3, 4, 5]
Permutation 2: [1, 2, 3, 5, 4]
Permutation 3: [1, 2, 4, 3, 5]
Permutation 4: [1, 2, 4, 5, 3]
Permutation 5: [1, 2, 5, 3, 4]
Permutation 6: [1, 2, 5, 4, 3]
Permutation 7: [1, 3, 2, 4, 5]
Permutation 8: [1, 3, 2, 5, 4]
Permutation 9: [1, 3, 4, 2, 5]
Permutation 10: [1, 3, 4, 5, 2]
Permutation 11: [1, 3, 5, 2, 4]
Permutation 12: [1, 3, 5, 4, 2]
Permutation 13: [1, 4, 2, 3, 5]
Permutation 14: [1, 4, 2, 5, 3]
Permutation 15: [1, 4, 3, 2, 5]
Permutation 16: [1, 4, 3, 5, 2]
Permutation 17: [1, 4, 5, 2, 3]
Permutation 18: [1, 4, 5, 3, 2]
Permutation 19: [1, 5, 2, 3, 4]
Permutation 20: [1, 5, 2, 4, 3]
Permutation 21: [1, 5, 3, 2, 4]
Permutation 22: [1, 5, 3, 4, 2]
Permutation 23: [1, 5, 4, 2, 3]
Permutation 24: [1, 5, 4, 3, 2]
Permutation 25: [2, 1, 3, 4, 5]
Permutation 26: [2, 1, 3, 5, 4]
Permutation 27: [2, 1, 4, 3, 5]
Permutation 28: [2, 1, 4, 5, 3]
Permutation 29: [2, 1, 5, 3, 4]
Permutation 30: [2, 1, 5, 4, 3]
Permutation 31: [2, 3, 1, 4, 5]
Permutation 32: [2, 3, 1, 5, 4]
Permutation 33: [2, 3, 4, 1, 5]
Permutation 34: [2, 3, 4, 5, 1]
Permutation 35: [2, 3, 5, 1, 4]
Permutation 36: [2, 3, 5, 4, 1]
Permutation 37: [2, 4, 1, 3, 5]
Permutation 38: [2, 4, 1, 5, 3]
Permutation 39: [2, 4, 3, 1, 5]
Permutation 40: [2, 4, 3, 5, 1]
Permutation 41: [2, 4, 5, 1, 3]
```

```
Permutation 42: [2, 4, 5, 3, 1]
Permutation 43: [2, 5, 1, 3, 4]
Permutation 44: [2, 5, 1, 4, 3]
Permutation 45: [2, 5, 3, 1, 4]
Permutation 46: [2, 5, 3, 4, 1]
Permutation 47: [2, 5, 4, 1, 3]
Permutation 48: [2, 5, 4, 3, 1]
Permutation 49: [3, 1, 2, 4, 5]
Permutation 50: [3, 1, 2, 5, 4]
Permutation 51: [3, 1, 4, 2, 5]
Permutation 52: [3, 1, 4, 5, 2]
Permutation 53: [3, 1, 5, 2, 4]
Permutation 54: [3, 1, 5, 4, 2]
Permutation 55: [3, 2, 1, 4, 5]
Permutation 56: [3, 2, 1, 5, 4]
Permutation 57: [3, 2, 4, 1, 5]
Permutation 58: [3, 2, 4, 5, 1]
Permutation 59: [3, 2, 5, 1, 4]
Permutation 60: [3, 2, 5, 4, 1]
Permutation 61: [3, 4, 1, 2, 5]
Permutation 62: [3, 4, 1, 5, 2]
Permutation 63: [3, 4, 2, 1, 5]
Permutation 64: [3, 4, 2, 5, 1]
Permutation 65: [3, 4, 5, 1, 2]
Permutation 66: [3, 4, 5, 2, 1]
Permutation 67: [3, 5, 1, 2, 4]
Permutation 68: [3, 5, 1, 4, 2]
Permutation 69: [3, 5, 2, 1, 4]
Permutation 70: [3, 5, 2, 4, 1]
Permutation 71: [3, 5, 4, 1, 2]
Permutation 72: [3, 5, 4, 2, 1]
Permutation 73: [4, 1, 2, 3, 5]
Permutation 74: [4, 1, 2, 5, 3]
Permutation 75: [4, 1, 3, 2, 5]
Permutation 76: [4, 1, 3, 5, 2]
Permutation 77: [4, 1, 5, 2, 3]
Permutation 78: [4, 1, 5, 3, 2]
Permutation 79: [4, 2, 1, 3, 5]
Permutation 80: [4, 2, 1, 5, 3]
Permutation 81: [4, 2, 3, 1, 5]
Permutation 82: [4, 2, 3, 5, 1]
Permutation 83: [4, 2, 5, 1, 3]
Permutation 84: [4, 2, 5, 3, 1]
Permutation 85: [4, 3, 1, 2, 5]
Permutation 86: [4, 3, 1, 5, 2]
Permutation 87: [4, 3, 2, 1, 5]
Permutation 88: [4, 3, 2, 5, 1]
Permutation 89: [4, 3, 5, 1, 2]
Permutation 90: [4, 3, 5, 2, 1]
Permutation 91: [4, 5, 1, 2, 3]
Permutation 92: [4, 5, 1, 3, 2]
Permutation 93: [4, 5, 2, 1, 3]
Permutation 94: [4, 5, 2, 3, 1]
Permutation 95: [4, 5, 3, 1, 2]
Permutation 96: [4, 5, 3, 2, 1]
Permutation 97: [5, 1, 2, 3, 4]
Permutation 98: [5, 1, 2, 4, 3]
Permutation 99: [5, 1, 3, 2, 4]
Permutation 100: [5, 1, 3, 4, 2]
Permutation 101: [5, 1, 4, 2, 3]
Permutation 102: [5, 1, 4, 3, 2]
Permutation 103: [5, 2, 1, 3, 4]
```

```
Permutation 104: [5, 2, 1, 4, 3]
Permutation 105: [5, 2, 3, 1, 4]
Permutation 106: [5, 2, 3, 4, 1]
Permutation 107: [5, 2, 4, 1, 3]
Permutation 108: [5, 2, 4, 3, 1]
Permutation 109: [5, 3, 1, 2, 4]
Permutation 110: [5, 3, 1, 4, 2]
Permutation 111: [5, 3, 2, 1, 4]
Permutation 112: [5, 3, 2, 4, 1]
Permutation 113: [5, 3, 4, 1, 2]
Permutation 114: [5, 3, 4, 2, 1]
Permutation 115: [5, 4, 1, 2, 3]
Permutation 116: [5, 4, 1, 3, 2]
Permutation 117: [5, 4, 2, 1, 3]
Permutation 118: [5, 4, 2, 3, 1]
Permutation 119: [5, 4, 3, 1, 2]
Permutation 120: [5, 4, 3, 2, 1]

(a + b)^10 = (1/1)*(a^0)*(b^10) + (10/1)*(a^1)*(b^9) + (45/1)*(a^2)*(b^8) +
(120/1)*(a^3)*(b^7) + (210/1)*(a^4)*(b^6) +
(252/1)*(a^5)*(b^5) + (210/1)*(a^6)*(b^4) + (120/1)*(a^7)*(b^3) + (45/1)*(a^8)*(b^2)
+ (10/1)*(a^9)*(b^1) + (1/1)*(a^10)
*(b^0)

Process finished with exit code 0
```

2 випадок

```
"D:\Dyskretna laboratorni\6\cmake-build-debug\6.exe"
Enter n=1
Permutation 1: [1]

(a + b)^10 = (1/1)*(a^0)*(b^10) + (10/1)*(a^1)*(b^9) + (45/1)*(a^2)*(b^8) +
(120/1)*(a^3)*(b^7) + (210/1)*(a^4)*(b^6) +
(252/1)*(a^5)*(b^5) + (210/1)*(a^6)*(b^4) + (120/1)*(a^7)*(b^3) + (45/1)*(a^8)*(b^2)
+ (10/1)*(a^9)*(b^1) + (1/1)*(a^10)
*(b^0)

Process finished with exit code 0
```

3 випадок

```
"D:\Dyskretna laboratorni\6\cmake-build-debug\6.exe"
Enter n=2
Permutation 1: [1, 2]
Permutation 2: [2, 1]

(a + b)^10 = (1/1)*(a^0)*(b^10) + (10/1)*(a^1)*(b^9) + (45/1)*(a^2)*(b^8) +
(120/1)*(a^3)*(b^7) + (210/1)*(a^4)*(b^6) +
(252/1)*(a^5)*(b^5) + (210/1)*(a^6)*(b^4) + (120/1)*(a^7)*(b^3) + (45/1)*(a^8)*(b^2)
+ (10/1)*(a^9)*(b^1) + (1/1)*(a^10)
*(b^0)

Process finished with exit code 0
```

4 випадок

```
"D:\Dyskretna laboratorni\6\cmake-build-debug\6.exe"
Enter n=4
Permutation 1: [1, 2, 3, 4]
Permutation 2: [1, 2, 4, 3]
Permutation 3: [1, 3, 2, 4]
Permutation 4: [1, 3, 4, 2]
Permutation 5: [1, 4, 2, 3]
Permutation 6: [1, 4, 3, 2]
Permutation 7: [2, 1, 3, 4]
Permutation 8: [2, 1, 4, 3]
Permutation 9: [2, 3, 1, 4]
Permutation 10: [2, 3, 4, 1]
Permutation 11: [2, 4, 1, 3]
Permutation 12: [2, 4, 3, 1]
Permutation 13: [3, 1, 2, 4]
Permutation 14: [3, 1, 4, 2]
Permutation 15: [3, 2, 1, 4]
Permutation 16: [3, 2, 4, 1]
Permutation 17: [3, 4, 1, 2]
Permutation 18: [3, 4, 2, 1]
Permutation 19: [4, 1, 2, 3]
Permutation 20: [4, 1, 3, 2]
Permutation 21: [4, 2, 1, 3]
Permutation 22: [4, 2, 3, 1]
Permutation 23: [4, 3, 1, 2]
Permutation 24: [4, 3, 2, 1]

(a + b)^10 = (1/1)*(a^0)*(b^10) + (10/1)*(a^1)*(b^9) + (45/1)*(a^2)*(b^8) +
(120/1)*(a^3)*(b^7) + (210/1)*(a^4)*(b^6) +
(252/1)*(a^5)*(b^5) + (210/1)*(a^6)*(b^4) + (120/1)*(a^7)*(b^3) + (45/1)*(a^8)*(b^2)
+ (10/1)*(a^9)*(b^1) + (1/1)*(a^10)
*(b^0)

Process finished with exit code 0
```