МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

Лабораторна робота № 6

з дисципліни
«Дискретна математика»

Виконав:
студент групи КН-114
Долінський А.Г.
Викладач:
Мельникова Н.І.

Львів – 2019р.

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 $\exists m : (b_i = a_i\ , i < m) \wedge (b_m < a_m)$.

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}, $b_{k+3} \dots b_r$ – найменші з чисел у множині *B*, розмістивши їх на позиціях $k + 1, k + 2,..., r$.

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 $(a_j < a_{j+1},) \wedge (a_{j+i} \geq a_{j+2} \geq ... \geq a_n)$. 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 $j + 1,..., n$.

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 $i > k$.

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: $b_i = b_{i-1} + 1 = a_k + i - k,\ i>k$.

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

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

Додаток №1

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

Варіант №7

1. Учасники шахового турніру грають у залі, де є 8 столів. Скількома способами можна розмістити 16 шахістів, якщо учасники всіх партій відомі?
2. Скільки трицифрових чисел можна утворити з дев'яти цифр 1, 2, 3, 4, 5, 6, 7, 8, 9?
3. Скільки можна побудувати різних прямокутних паралелепіпедів, довжини ребер яких виражають натуральними числами від 1 до 10?
4. У вищій лізі чемпіонату України з футболу грають 16 команд. Скільки існує способів розподілення I, II, та III місця та вибору двох команд які перейдуть у першу лігу (дві останні команди)?
5. З цифр 1, 2, 3, 4, 5, 6, 7, 8, 9 утворюють різні п'ятицифрові числа, що не мають однакових цифр. Визначити кількість чисел, у яких зустрічається цифри 5, 3, 4 одночасно, якщо вони не стоять поруч?
6. У шаховому турнірі беруть участь 18 шахістів. Визначити кількість різних розкладів першого туру (розклади вважаються різними, якщо вони відрізняються учасниками, колір та номер столу не враховується).

7.Знайти кількість цілих додатних чисел, які змінюються від 101 до 1000 та діляться рівно на два з чисел
3, 6 і 7.

1. $P_8 = 8! = 40320$ (способи).

2. $\overline{A_9^3} = 9^3 = 729$ (чисел).

3. $\overline{A_{10}^3} = 10^3 = 1000$ (паралелепіпедів).

4. $A_{16}^3 = \frac{16!}{13!} = 3360$; $C_{13}^2 = \frac{13!}{11!*2!} = 78$;
   $A_{16}^3 * C_{13}^2 = 3360 * 78 = 262080$ (способів).

5. $P_3 = 3! = 6$; $A_6^2 = \frac{6!}{4!} = 30$; $P_3 * A_6^2 = 6 * 30 = 180$ (чисел).

6. $C_{18}^2 = \frac{18!}{16!*2!} = 153$ (різних розкладів).

7. N = 900
   $S_1 = 608$
   $S_2 = 113$

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.

Варіант №7

Визначити лексикографічно наступну перестановку для кожної з перестановок: 1432, 54123, 12453, 45231, 6714235 і 31528764. Побудувати розклад $(x-y)^8$.

```
#include <iostream>
using namespace std;
void next(int* n, int m);
void NewtonBinom(char op, int pow);
int Combination(int pow, int n);
int factorial(int n);
int main() {
    int counter = 0;
    while (counter < 6 ) {
        int m;
        cout << "Enter a number of digits: ";
        cin >> m;
        int* number = new int[m];
        cout << "Enter that digits" << endl;
        for (int i = 0; i < m; i++) {
            cin >> number[i];
        }
        next(number, m);
        counter++;
    }
    NewtonBinom('-',8);
    system("pause");
    return 0;
}

void next(int* n, int m) {
    int temp1, temp2, temp3;
    bool BOOL = false;
    for (int i = m - 1; i >= 0; i--) {
        if (BOOL == false) {
```

```
            if (n[i-1] < n[i]) {
                temp1 = i - 1;

                BOOL = true;
            }
        }
    }
    cout << temp1 << endl;
    BOOL = false;
    for (int c = m - 1; c > temp1; c--) {
        if (BOOL == false) {
            if (n[c] > n[temp1]) {
                temp3 = n[c];
                n[c] = n[temp1];
                n[temp1] = temp3;
                BOOL = true;
            }
        }
    }
    cout << n[temp1] << endl;
    for (int i = temp1+1; i < m; i++) {
        for (int j = temp1+1; j < m -1; j++) {
            if (n[j] > n[j+1]) {
                temp2 = n[j];
                n[j] = n[j+1];
                n[j+1] = temp2;
            }
        }
    }
    cout << "The next one: ";
```

```
    for (int i = 0; i < m;  i++) {
        cout << n[i];
    }
    cout << endl;
}

void NewtonBinom(char op, int pow) {
    cout << "x^" << pow << " ";
    if (op == '-') {
        for (int i = 1; i < pow; i++) {
            if (i % 2 == 0) {
                cout << " + " << Combination(pow, i) << "x^" << pow - i << "*" << "y^" << i << " ";
            }
            else {
                cout << " - " << Combination(pow, i) << "x^" << pow - i << "*" << "y^" << i << " ";
            }


        }
        if (pow % 2 == 0) {
            cout << "+ y^" << pow << endl;
        }
        else {
            cout << "- y^" << pow << endl;
        }
    }
    else if (op == '+') {
        for (int i = 1; i < pow; i++) {
                cout << " + " << Combination(pow, i) << "x^" << pow - i << "*" << "y^" << i << " ";
        }
        cout << "+ y^" << pow << endl;
```

```cpp
    }
    else {
        cout << "Wrong operator!" << endl;
    }
}

int Combination(int pow, int n) {
    int result;
    result = factorial(pow) / (factorial(pow - n) * factorial(n));
    return result;
}
int factorial(int n)
{
    if (n > 1)
        return n * factorial(n - 1);
    else
        return 1;
}
```

Скрін-шот коду програми на мові C++.

```
C:\Users\Admin\source\repos\Lab 6 (math)\Debug\Lab 6 (math).exe
Enter a number of digits: 4
Enter that digits
1 4 3 2
The next permutation: 2134
Enter a number of digits: 5
Enter that digits
5 4 1 2 3
The next permutation: 54132
Enter a number of digits: 5
Enter that digits
1 2 4 5 3
The next permutation: 12534
Enter a number of digits: 5
Enter that digits
4 5 2 3 1
The next permutation: 45312
Enter a number of digits: 7
Enter that digits
6 7 1 4 2 3 5
The next permutation: 6714253
Enter a number of digits: 8
Enter that digits
3 1 5 2 8 7 6 4
The next permutation: 31542678
x^8  - 8x^7*y^1  + 28x^6*y^2  - 56x^5*y^3  + 70x^4*y^4  - 56x^3*y^5  + 28x^2*y^6  - 8x^1*y^7 + y^8
Press any key to continue . . .
```

Скрін-шот тесту програми.

Висновок: Я набув практичних вмінь та навичок у комп’ютерній реалізації комбінаторних задач.