**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДВНЗ «Ужгородський національний університет»**

**Факультет інформаційних технологій**

**Кафедра інформаційних управляючих систем та технологій**

**Навчально-методичний посібник до курсу**

# **“ Теорія алгоритмів та математичні основи представлення знань”**

зі спеціальності

“інформаційні управляючі системи та технології”

***частина ІІІ***

**Ужгород-2015**

Копча-Горячкіна Галина Ернестівна

ТЕОРІЯ АЛГОРИТМІВ ТА МАТЕМАТИЧНІ ОСНОВИ ПРЕДСТАВЛЕННЯ ЗНАНЬ. Частина ІІІ: Навчально-методичний посібник для студентів факультету інформаційних технологій напряму „Комп'ютерні науки" спеціальності „Інформаційні управляючі системи та технології". – Ужгород: Видавництво ДВНЗ «Ужгородського національного університету», 2015 р.

Навчально-методичний посібник містить деякі теоретичні відомості, опис предмета навчальної дисципліни у відповідності до болонського процесу, навчально-тематичний план дисципліни «Теорія алгоритмів та математичні основи представлення знань», зміст лекційних тем курсу, тем практичних занять та тем самостійної та індивідуальної роботи студентів, перелік питань для модульних контролів, на іспит, літературу.

Друкується за рішенням кафедри інформаційних управляючих систем та технологій від . . 2015р., протокол №.

## Вступ

Інтенсивний розвиток математичної логіки в наш час супроводжується збільшенням її ролі в математиці.

Однією з основних задач математичної логіки є аналіз основ математики. Але в наш час вона вже вийшла з рамок цієї задачі і суттєво вплинула на розвиток самої математики. Із її ідей виникло точне визначення поняття алгоритму, що дозволило розв'язати багато питань, які без цього залишались би в принципі нерозв'язаними. Апарат, який виник в математичній логіці знайшов застосування в питаннях конструкцій обчислювальних машин і автоматичних пристроїв.

З переходом до впровадження ЕОМ в наукові розрахунки управління виникло намагання максимально використати розумові можливості людини шляхом впровадження ЕОМ в сам процес пізнання. Реалізація цієї ідеї пов'язана з необхідністю навчити ЕОМ мислити, наділити її інтелектом.

Сформувався новий науковий напрямок – штучний інтелект, засоби і методи якого широко застосовуються в автоматизованих системах управління.

Курс "Теорія алгоритмів та математичні основи представлення знань" складається з двох частин : "Математичні основи представлення знань" та "Теорія алгоритмів".

В першій частині розглядаються  проблеми представлення і обробки знань, основні напрямки представлення знань, способи формального задання алгоритмів, основні поняття формальної та математичної логіки, визначення і властивості алгебри і числення висловлювань, проблеми доведення теорем. Друга частина включає формальне визначення алгоритму та основні підходи до формального визначення, приклади різних алгоритмічних систем, їх повноту та еквівалентність.

## Мета та завдання дисципліни

Метою дисципліни є ознайомлення студентів з теорією алгоритмів (прикладами різних алгоритмічних систем) та математичними основами представлення знань (численням висловлювань, численням предикатів та ін.).

Завданням дисципліни є засвоєння базових навичок з числення висловлювань, числення предикатів, алгоритмічних систем Тюрінга, Маркова, Кліні, Черча.

В результаті вивчення даного курсу студент повинен

**знати:** основні означення та теореми теорії алгоритмів та математичних основ представлення знань

**вміти:** розв’язувати задачі з числення висловлювань, числення предикатів, алгоритмічних систем Тюрінга, Маркова, Кліні, Черча.

## Програма навчальної дисципліни

**Змістовий модуль 1. Алгоритмічні системи.**

**Тема 1.** Вступ. Предмет курсу. Короткий огляд історії розвитку теорії алгоритмів. Основні поняття, взаємозв'язок з іншими дисциплінами.

**Тема 2.** Конструктивні об'єкти. Класи конструктивних об'єктів

**Тема 3.** Конструктивні операції. Приклади конструктивних операцій та предикатів.

**Тема 4.** Ефективні нумерації. Ефективно нумеровані множини.

**Тема 5**. Елементи теорії графів.

**Тема 6**. Деякі алгебраїчні структури: підстановки, групи, кільця, поля, тіла, алгебри.

**Тема 7.** Інтуїтивне поняття алгоритму. Властивості алгоритмів.

**Тема 8.** Еквівалентність і повнота алгоритмічних систем.

**Тема 9**. Основні алгоритмічні системи. Машина Тюрінга. Елементарні машини Тюрінга.

**Тема 10**. Обчислення унарних функцій на множині слів за допомогою машини Тюрінга.

**Тема 11**. Алгоритмічна система Маркова.

**Тема 12**. Принцип нормалізації Маркова, асоціативне числення, дедуктивний інваріант, проблема слів.

**Тема 13**. Теорія рекурсивних функцій. Схеми примітивної рекурсії, суперпозиції і мінімізації.

**Тема 14**. Алгоритмічна система Кліні.

**Змістовий модуль 2. Логіка висловлювань та логіка предикатів.**

**Тема 1**. Алгебра висловлювань (АВ). Логічні операції. Закон двоїстості.

**Тема 2.** Види формул. Представлення довільної двозначної функції за допомогою формул АВ. Досконалі нормальні форми.

**Тема 3.** Числення висловлювань (ЧВ). Виводимість формул. Теорема дедукції.

**Тема 4**. Деякі правила числення висловлювань. Монотонність. Еквівалентні формули. Деякі теореми про виводимість.

**Тема 5**. Зв'язок між формулами алгебри висловлювань і числення висловлювань

**Тема 6**. Несуперечливість числення висловлювань. Повнота ЧВ.

**Тема 7**. Незалежність аксіом числення висловлювань.

**Тема 8**. Логіка предикатів (ЛП). Квантори. Рівносильні формули. Теоретико-множинний зміст предикатів.

**Тема 9**. Аксіоми. Несуперечливість і незалежність аксіом.

**Тема 10**. Нормальні формули і нормальні форми.

**Тема 11**. Логіка предикатів з однією змінною.

**Тема 12**. Числення предикатів (ЧП). Заміна змінних в формулах. Аксіоми ЧП.

**Тема 13**. Правила утворення виводимих формул.

**Тема 14**. Несуперечливість ЧП, повнота у вузькому значенні. Деякі теореми ЧП. Теорема дедукції.

## Структура навчальної дисципліни

| Назви змістових модулів і тем | Кількість годин | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | денна форма | | | | | заочна форма | | | | |
| | усього | у тому числі | | | | усього | у тому числі | | | |
| | | л | п | лаб | с.р. | | л | п | лаб | с.р. |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| **Змістовий модуль 1. Алгоритмічні системи** | | | | | | | | | | |
| Тема 1. Вступ. Предмет курсу. Короткий огляд історії розвитку теорії алгоритмів. Основні поняття, взаємозв’язок з іншими дисциплінами | 4 | 2 | | | 2 | | | | | |
| Тема 2. Конструктивні об'єкти. Класи конструктивних об’єктів | 4 | 2 | | | 2 | 6 | 2 | | | 4 |
| Тема 3. Конструктивні операції. Приклади конструктивних операцій та предикатів | 6 | | | | 6 | 6 | | | | 6 |
| Тема 4. Ефективні нумерації. Ефективно нумеровані множини | 4 | 2 | | | 2 | 5 | 1 | | | 4 |
| Тема 5. Елементи | 8 | 2 | | | 6 | 8 | | | | 8 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| теорії графів | | | | | | | | | | |
| Тема 6. Деякі алгебраїчні структури: підстановки, групи, кільця, поля, тіла, алгебри | 13 | 2 | 4 | | 7 | 10 | | | | 10 |
| Тема 7. Інтуїтивне поняття алгоритму. Властивості алгоритмів | 6 | | | | 6 | 8 | | | | 8 |
| Тема 8. Еквівалентність і повнота алгоритмічних систем | 6 | | | | 6 | 9 | 1 | | | 8 |
| Тема 9. Основні алгоритмічні системи. Машина Тюрінга. Елементарні машини Тюрінга | 10 | 2 | 4 | | 4 | 6 | 1 | | | 5 |
| Тема 10. Обчислення унарних функцій на множині слів за допомогою машини Тюрінга | 6 | 1 | 2 | | 3 | 6 | 1 | | | 5 |
| Тема 11. Алгоритмічна система Маркова | 6 | 1 | 2 | | 3 | 6 | 1 | | | 5 |
| Тема 12. Принцип нормалізації Маркова, асоціативне числення, | 5 | 1 | | | 4 | 7 | 1 | | | 6 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| дедуктивний інваріант, проблема слів | | | | | | | | | | |
| Тема 13. Теорія рекурсивних функцій. Схеми примітивної рекурсії, суперпозиції і мінімізації | 6 | 2 | 2 | | 2 | 8 | 1 | | | 7 |
| Тема 14. Алгоритмічна система Кліні | 4 | 1 | | | 3 | 5 | 1 | | | 4 |
| Разом за змістовим модулем 1 | 88 | 18 | 14 | | 56 | 90 | 10 | | | 80 |
| **Змістовий модуль 2. Логіка висловлювань та логіка предикатів** | | | | | | | | | | |
| Тема 1. Алгебра висловлювань (АВ). Логічні операції. Закон двоїстості | 6 | 1 | | | 5 | 7 | 1 | | | 6 |
| Тема 2. Види формул. Представлення довільної двозначної функції за допомогою формул АВ. Досконалі нормальні форми | 6 | 1 | | | 5 | 6 | | | | 6 |
| Тема 3. Числення висловлювань (ЧВ). Виводимість формул. Теорема дедукції | 10 | 2 | 4 | | 4 | 7 | 1 | | | 6 |
| Тема 4. Деякі правила числення | 8 | 2 | 2 | | 4 | 7 | 1 | | | 6 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| висловлювань. Монотонність. Еквівалентні формули. Деякі теореми про виводимість | | | | | | | | | | |
| Тема 5. Зв'язок між формулами алгебри висловлювань і числення висловлювань | 6 | 1 | | | 5 | 7 | | | | 7 |
| Тема 6. Несуперечливість числення висловлювань. Повнота ЧВ | 6 | 1 | | | 5 | 7 | | | | 7 |
| Тема 7. Незалежність аксіом числення висловлювань | 4 | 2 | | | 2 | 4 | | | | 4 |
| Тема 8. Логіка предикатів (ЛП). Квантори. Рівносильні формули. Теоретико-множинний зміст предикатів | 8 | 2 | 2 | | 4 | 5 | 1 | | | 4 |
| Тема 9. Аксіоми. Несуперечливість і незалежність аксіом | 6 | 2 | | | 4 | 6 | | | | 6 |
| Тема 10. Нормальні формули і нормальні | 6 | | | | 6 | 8 | | | | 8 |

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| форми | | | | | | | | | | |
| Тема 11. Логіка предикатів з однією змінною | 6 | | | | 6 | 6 | | | | 6 |
| Тема 12. Числення предикатів (ЧП). Заміна змінних в формулах. Аксіоми ЧП | 8 | 2 | 2 | | 4 | 5 | 1 | | | 4 |
| Тема 13. Правила утворення виводимих формул | 6 | 2 | 4 | | | 7 | 1 | | | 6 |
| Тема 14. Несуперечливість ЧП, повнота у вузькому значенні. Деякі теореми ЧП. Теорема дедукції | 6 | | | | 6 | 8 | | | | 8 |
| Разом за змістовим модулем 2 | 92 | 18 | 14 | | 60 | 90 | 6 | | | 84 |

## Теми практичних занять

| № п/п | Назва теми | К-сть годин |
|---|---|---|
| | Змістовий модуль1 | |
| 1. | Розклад підстановки в цикли, визначення парності; знаходження нейтральних та обернених елементів; визначення заданої алгебраїчної структури | 4 |
| 2. | Знаходження траєкторії та кінцевого стану послідовності елементарних машин Тюрінга | 2 |
| 3. | Побудова машин Тюрінга | 2 |
| 4. | Обчислення унарних функцій на множині слів за допомогою машини Тюрінга | 2 |
| 5. | Опис траєкторій нормальних алгоритмів | 1 |
| 6. | Побудова алгоритмів Маркова | 1 |
| 7. | Визначення примітивної та часткової рекурсивності заданої функції | 2 |
| | Всього за змістовий модуль 1 | 14 |
| | Змістовий модуль1 | |
| 8. | Доведення виводимості формул за правилами виводу та теоремою дедукції | 4 |
| 9. | Доведення монотонного зростання і спадання формул | 2 |
| 10. | Спрощення формули, знаходження рівносильної їй | 2 |
| 11. | Приклади операцій з кванторами (спрощення та побудова виразів) | 4 |
| 12. | Заміна змінних у формулах ЧП | 2 |
| | Всього за змістовий модуль 2 | 14 |

## Теми самостійної та індивідуальної роботи

| № п/п | Назва теми | К-сть годин |
|---|---|---|
| Змістовий модуль1 | | |
| 1. | Спрощення формул | 4 |
| 2. | Визначення числа інверсій в перестановках | 4 |
| 3. | Визначення гомоморфності відображення | 4 |
| 4. | Перевірка конструктивності об’єкта | 4 |
| 5. | Приклади алгоритмічно нерозв’язних проблем | 4 |
| 6. | Приклади ефективних та ефективно нумерованих множин | 5 |
| 7. | Основні структури для побудови алгоритмів; уточнення поняття алгоритму | 4 |
| 8. | Еквівалентність та повнота алгоритмічних систем | 4 |
| 9. | Інтерпретація програм Тюрінга на мові „стрічки”; запис програм Тюрінга у вигляді послідовності елементарних машин Тюрінга | 6 |
| 10. | Обчислення унарних функцій на множині слів за допомогою машин Тюрінга | 4 |
| 11. | Знаходження результату застосування алгоритмів Маркова до заданих слів | 4 |
| 12. | Обмежена та необмежена проблема слів | 2 |
| 13. | Перевірка часткової та загальної рекурсивності функції | 4 |
| 14. | Функція Кліні (*K*-функція) | 3 |
| Всього за змістовий модуль 1 | | 56 |
| Змістовий модуль 2 | | |
| 15. | Знаходження рівносильних формул | 5 |

| | | |
|---|---|---|
| 16. | Одержання ДНФ формули з її ДДНФ | 4 |
| 17. | Перевірка даних виразів на наявність ознак формули | 4 |
| 18. | Доведення виводимості і монотонності формул | 4 |
| 19. | Встановити зв’язок між формулами АВ і ЧВ | 4 |
| 20. | Показати тотожню істинність аксіом ЧВ | 5 |
| 21. | Перевірити незалежність аксіом групи IV | 4 |
| 22. | Теоретико-множинний зміст предикатів | 4 |
| 23. | Незалежність аксіом ЧП | 4 |
| 24. | Зведення формул до нормальної форми | 4 |
| 25. | Логіка Арістотеля | 5 |
| 26. | Заміна змінних у формулах | 4 |
| 27. | Перевірка виводимості формул | 5 |
| 28. | Порівняння повноти у вузькому значенні ЧВ і ЧП | 4 |
| Всього за змістовий модуль 2 | | 60 |

## Рекомендована література

### Базова


1. Вітенько І.В. Конструктивні операції. Ужгород: Вид-во УжДУ, 1992.
2. Новиков Ф.А. Дискретная математика для программистов. Учебник. Санкт-Петербург, 2001.
3. Новиков П.С. Элементы математической логики. Москва: Наука, 1993.


### Допоміжна


1. Вітенько І.В. Математична логіка (курс лекцій). – Ужгород, 1971. – 224с.
2. Нікольський Ю. В., Пасічник В. В., Щербина Ю. М. Дискретна математика. – К.: Видавнича група BHV, 2007. – 368 с.
3. Нефедов В. Н., Осипова В. А. Курс дискретной математики. – М.: Из-во МАЙ, 1992. – 264 с.
4. Андерсон Д. Дискретная математика и комбинаторика. — СПб.: Вильямс, 2003. — 958 с.


### Методичне забезпечення


1. Василенко Ю.А. Методичні вказівки до курсу "Основи дискретної математики". – УжДІІЕП, 1999.
2. Василенко Ю.А., Копча-Горячкіна Г.Е. Основи дискретної математики. Методичні рекомендації до курсу, частина І , 2002.
3. Василенко Ю.А., Копча-Горячкіна Г.Е. Основи дискретної математики. Методичний посібник, частина ІІ , 2006.
4. Копча-Горячкіна Г.Е., Повхан І.Ф. Теорія алгоритмів та математичні основи представлення знань. Методичні рекомендації до курсу, частина І. – Видавництво УжДІІЕП, 2002.
5. Копча-Горячкіна Г.Е. Теорія алгоритмів та математичні основи представлення знань. Методичний посібник, частина ІІ. – Ужгород: Видавництво ЗакДУ, 2006.
6. Копча-Горячкіна Г.Е. Фундаментальні алгоритми в комбінаториці та графах. Навчально-методичний посібник, частина І. – Ужгород, 2008.

# КОНСТРУКТИВНІ ОБ'ЄКТИ, КЛАСИ, ОПЕРАЦІЇ

## Вступ

В теорії алгоритмів, в основному, будемо мати справу з такими об'єктами як конструктивні функції та конструктивні об'єкти.

Функція, для якої існує алгоритм, що обчислює її значення, називається *конструктивною функцією (операцією).*

Алгоритм – це послідовність елементарних дій, по яких вхідній інформації ставиться у відповідність деяка вихідна інформація.

При вивченні конструктивних функцій виникає проблема: чи можна строго визначити поняття конструктивної функції. Цією проблемою займалися Гільберт, Кліні, Тюрінг.

Теорія, в якій строго визначається поняття конструктивної функції, називається *теорією рекурсивних функцій.* В цій теорії доводиться, що деякі масові проблеми не являються алгоритмічно розв'язними (наприклад, такою проблемою є проблема доведення теорем в арифметиці).

Проблема називається *алгоритмічно розв'язною*, якщо існує алгоритм для розв'язання будь-якої задачі із заданого класу задач. Алгоритмічна нерозв'язність тієї чи іншої проблеми означає, що єдиним засобом її розв'язання є винахідливість і творчість. Тому теорія рекурсивних функцій представляє інтерес не тільки для математиків, але і для всіх освічених людей. Але, в першу чергу, цю теорію варто знати математикам, і інженерам, які працюють в області обчислювальної математики, програмування, обчислювальної техніки, теорії автоматів та ін.

Адже, всі спеціальності даних областей весь час мають справу з алгоритмами.

Тому в них виникають такі загальні питання: що таке алгоритм, як строго визначити поняття алгоритмічного розв'язання проблеми, за допомогою яких

засобів можна довести, що та чи ін. проблема не являється алгоритмічно розв'язаною і т.ін.

На ці питання дає відповідь теорія рекурсивних функція.

Деякі терміни і позначення з теорії множин, які будуть використовуватися надалі:

Множина – сукупність, система, сімейство.

Символ ::= «означає»

$\chi \in A$ ::= елемент $\chi$ належить множинні $A$.

$\chi \notin A$ ::= елемент $\chi$ не належить множинні $A$.

$\varnothing$ ::= пуста множина.

$B(G)$ ::= множина всіх підмножин множини *G*.

$\{\chi\}$ :: = множина, в яку входить лише вказаний елемент.

$\{ \chi_1, \chi_2, ....., \chi_n \}$ ::= множина, членами якої являється вказані елементи.

$H_j (j \in Q)$ ::= індексоване сімейство об'єктів.

(можна представити, як відображення, яке кожному елементу *j* із *Q* ставить у відповідність об`єкт $H_j$).

Коли $H_j$ – множина , то $H_j (j \in Q)$ називається *індексованим сімейством множин.*

$A \cup B$ ::= об`єднання множин *A* і *B*.

$A \cap B$ ::= перетин множин *A* і *B*.

$\bigcap\limits_{\substack{j \in Q \\ j \in Q}} H_j$ ::= перетин індексованого сімейства множин $H_j (j \in Q)$.

$A - B$ ::= різниця множин *A* і *B* (сукупність всіх елементів, які належать A і не належать B).

$A \subset B$ ::= множина *A* включається в *B*. Вважається, що $A \subset A$.

$A \supset B$ ::= $B \subset A$.

$A \not\subset B$ ::= *A* не включається в *B*.

$B \subset A$, то $A - B$ називається *доповненням B в A.*

Якщо $B \subset A$ і $B \neq A$ ,то *B* називається власною підмножиною множини *A*.

$\prod_{j\in Q} A_i$ **::=** декартовий добуток індексованого сімейства множин $\mathrm{A}_j$.

**(**ця множина представляє собою сукупність всіх $\alpha_j (j \in Q)$, які задов. умові $\alpha_j \in \mathrm{A}_j$ при всіх $j \in Q$ ).

Якщо $\mathrm{A}_j = \mathrm{A}$ для всіх j із *Q*, то $\prod_{j\in Q} A_i$ позначається через $\mathrm{A}^{\mathrm{Q}}$, відповідно використовуються позначення:

$\mathrm{A}_1 \times \mathrm{A}_2 \times ....\mathrm{A}_n$ і $\mathrm{A}^{\mathrm{n}}$

Елемент із $\mathrm{A}_1 \times \mathrm{A}_2 \times ....\mathrm{A}_n$ є послідовністю виду $(\alpha_1, \alpha_2, ...\alpha_n)$, де $\alpha_i \in \mathrm{A}_i$.

Наприклад, елементи із $\mathrm{A}_1 \times \mathrm{A}_2$ представляють собою впорядковані пари виду $(\alpha_1, \alpha_2)$, де $\alpha_1 \in \mathrm{A}_1, \alpha_2 \in \mathrm{A}_2$.

$\{\chi / \mathrm{P}(\chi)\}$ ::= сукупність всіх елементів, для яких предикат $\mathrm{P}(\chi)$ являється істинним.

$f : \mathrm{A} \to \mathrm{B}$ ::= f представляє собою відображення множини *A* в множину *B*.

$\mathrm{A} \sim \mathrm{B}$ ::= множини *A* і *B* еквівалентні, тобто існує взаємо-однозначне відображення *A* на *B*.

$|\mathrm{A}|$ ::= потужність множини *A*.

I ::= істина.

Ф ::= фальш.

$R_1 \wedge R_2$ ::= *R1 і R2* (або $\mathrm{P}_1 \,\&\, \mathrm{P}_2$)

$R_1 \vee R_2$ ::= *R1* або *R1*.

$R_1 => R_2$ ::= якщо *R1*, то *R2;* із $R_1$ випливає *R2*.

$R_1 <=> R_2$ ::= *R1* істинна тоді і тільки тоді, коли являється істинним *R2*.

$\bar{R}_1$ ::= не *R1*.

$\forall \chi R_1(\chi)$ ::= $R_1(\chi)$ істинна для всіх $\chi$.

$\exists \chi R_1(\chi)$ ::= існує таке $\chi$, для якого $R_1(\chi)$ являється істинним.

$\forall (\chi \in G) R_1(\chi)$ ::= $R_1(\chi)$ істинне для всіх елементів множини *G*.

$\exists(\overline{\chi} \in G)R_1(\chi)$ ::= в множинні $G$ існує такий елемент $\chi$, для якого $R_1(\chi)$ являється істинним.

$R_1, R_2, R_1(\chi), R_2(\chi), R_2(\chi)$ – довільні висловлювання в цих позначеннях.

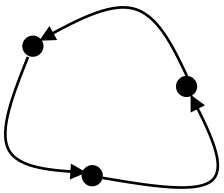
**Конструктивні об’єкти**

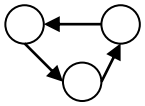
У математиці ми часто маємо справу з такими об’єктами як букви, цифри, формули, таблиці і т.д. Всі ці об’єкти називаються **конструктивними об’єктами**. Поняття конструктивного об’єкту являється первинним, інтуітивним поняттям. Тому строго описати всі властивості конструктивного об’єкту неможливо. Вкажемо на основні:

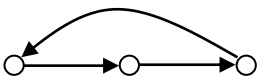
1. чітка означеність та зрозумілість (конструктивний об’єкт має бути чітко означеним та зрозумілим, щоб можна було пояснити навіть автомату, принципово простим, а не технічно, наприклад, як завгодно велика формула все одно буде конструктивним об'єктом);
2. скінченість структури (наприклад, *sinx* і *cosx* не являються конструктивними об’єктами, бо їх область визначення – $\boldsymbol{R}$ – нескінчена);
3. абстрактність (вона забезпечує чіткість і простоту конструктивного об’єкту).

Наприклад, розглянемо діаграму:

Не є суттєвим ні довжина стрілок , ні радіуси кружечків. Важливо лише те, що дана діаграма є певною конструкцією із вершин і стрілок. Тому її можна зобразити і так:







Тому дана діаграма являється конструктивним об’єктом.

Тобто кожен конструктивний об’єкт задається деякою системою правил. Більшість абстрактних об’єктів визначається за слідуючою схемою.

Нехай задано деякий клас об’єктів А. Для цього класу однозначно визначається новий об’єкт *а,* який називається *абстракцією* класу А.

Елементи із класу А називаються *конкретизаціями* або *представниками* об’єкту а.

Приклад. Діаграми, що попередньо розглядалися, є представниками вищевказаного конструктивного об’єкту.

Всі ці об’єкти називаються *конструктивними об’єктами*.

**Класи конструктивних об’єктів.**

### *І клас – натуральні числа*

Прикладом конструктивних об’єктів є натуральні числа. Ми розуміємо натуральне число як кількість елементів скінченої множини. Більш точно натуральне число можна визначити так.

Дві множини А і В називаються еквівалентними, коли між їх елементами можна встановити взаємно-однозначну відповідність.

Еквівалентні множини мають однакову потужність. Потужність, представники якої являються скінченими множинами, називається *натуральним числом*.

Наприклад, число 3 являється абстракцією класу всіх трьохелементних множин. Причому елементами цих множин можуть бути об’єкти якої завгодно природи. Візьмемо множину {місто Київ, місто Львів, місто Ужгород}. Елементами цієї множини являються складні і неконструктивні об’єкти. Але все ж таки число 3 є простим і конструктивним об’єктом завдяки абстракції.

### *ІІ клас – скінчені графи*

*Графом* називається пара *Г=(G,M),* де *G* – множина елементів довільної природи і *M* – деяка система пар елементів множини *G*, тобто $M \subset G^2$.

Коли *G* – скінчена множина, то граф *Г* називається *скінченим графом*.

Елементи множини $G$ називаються *вершинами*, а елементи множини $M$ – *дугами* графа $\Gamma$.

На нашій діаграмі, що представляє собою певний граф, в якому вершинами являються кільця, а дуги представляються за допомогою стрілок.

Два графи $\Gamma_1=(G_1,M_1)$ і $\Gamma_2=(G_2,M_2)$ називаються *ізоморфними*, коли існує таке взаємно-однозначне відображення $f: G_1 \to G_2$, що $(a,b)\in M_1 \Leftrightarrow (f(a),f(b))\in M_2$, де $a,b$ – довільні елементи із $G_1$.

Нагадаємо, що взаємно-однозначне (або бієктивне) відображення – це таке відображення, яке являється ін'єктивним та сюр'єктивним.

Відображення $f: A\to B$ називається ін'єктивним, коли

$$\forall(x\in A)\forall(y\in A)\quad (x\neq y \Rightarrow f(x)\neq f(y))$$

Відображення $f: A\to B$ називається сюр'єктивним, коли $f(A)=B$.

Із означення ізоморфізму випливає, що ізоморфні графи $\Gamma_1=(G_1,M_1)$ і $\Gamma_2=(G_2,M_2)$ можуть розрізнятися лише природою елементів множин $G_1$ і $G_2$, а в усьому іншому їх структури співпадають.

Абстракція певного класу всіх ізоморфних між собою графів називається *абстрактним графом.*

Якщо представники абстрактного графу являються скінченими графами, то даний граф називається *скінченим абсрактним графом.*

Очевидно, що діаграми, які розглядалися раніше, є ізоморфними між собою і вони представляють конструктивний об'єкт, що являється скінченим абстрактним графом.

### *III клас – навантажені скінчені абстрактні графи*

*Навантаженим графом* називається четвірка об'єктів $\Gamma=(G,M,f,H)$, де $G$ і $H$ – довільні множини, $M$ – множина пар елементів із $G$, тобто $M\subset G^2$ і $f$ – сюр'єктивне відображення $G$ на $H$.

Елементи множин $G$ і $M$ відповідно називаються *вершинами* і *дугами* графа $\Gamma$.

Елементи множини $H$ будемо називати *вершинними елементами* графа $\Gamma$.

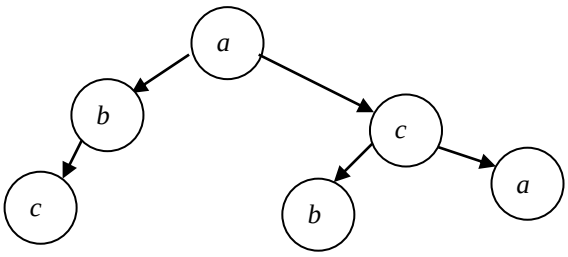
Два навантажені графи $\Gamma_1 = (G_1, M_1, f_1, H_1)$ і $\Gamma_2 = (G_2, M_2, f_2, H_2)$ називаються *ізоморфними,* коли існує таке бієктивне відображення $\varphi$: $G_1 \rightarrow G_2$, що $(a,b) \in M_1 \Leftrightarrow (\varphi(a), \varphi(b)) \in M_2$ і $f_1(a) = f_2(\varphi(a))$, де $a,b$ – довільні елементи із $G_1$. В ізоморфних навантажених графах $H_1 = H_2$.

Абстракція певного класу всіх ізоморфних між собою навантажених графів називається *абстрактним навантаженим графом.*

Якщо представники абстрактного навантаженого графа являються скінченими, то даний граф називається *скінченим абстрактним навантаженим графом.*

Кожному абстрактному навантаженому графові відповідає певна множина вершинних елементів.

Приклад. Нехай задано такий навантажений граф:



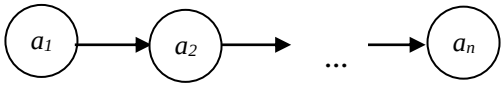

Тобто навантажений – це граф, вершини якого "навантажені" деякими елементами в даному випадку – елементами *c, b, a, b, a, c.*

Множина вершин цього графа $G = \{c,b,a,b,a,c\}$, множина вершинних елементів даного графа $H = \{a,b,c\}$.

!Конструктивним об'єктом являється той і тільки той скінчений навантажений абстрактний граф, в якому всі вершинні елементи представляють собою конструктивні об'єкти.

### *IV клас – слова*

*Слово* – це скінчений абстрактний навантажений граф, який має вигляд:




Елементи $a_1, ..., a_n$ називаються *буквами* слова, $a_1$ називається першою, $a_n$ – останньою буквою слова.

Слово часто представляється у вигляді рядка $a_1a_2...a_n$. Цей рядок називають *n*-кою, або скінченою послідовністю.

! Слово $a_1 a_2 ... a_n$ тоді і тільки тоді являється конструктивним об'єктом, коли всі букви $a_i$ цього слова представляють собою конструктивні об'єкти.

Наприклад, довільна скінчена послідовність натуральних чисел є конструктивним об'єктом.

Слово найбільш часто застосовується в математиці завдяки простоті своєї конструкції.

Наприклад, формула $log\ (xy) = log\ x + log\ y$ представляє собою слово, в якому в ролі букв виступають значки $l, o, g, (, ), x, y, +, =$.

### *V клас – скінчені абстрактні сімейства*

Кожне сюр'єктивне відображення $f: G \to H$ можна інтерпретувати як сімейство елементів множини $H$, які проіндексовані елементами $G$.

Тобто $f: G \to H$ можна представити у вигляді $h_i\ (i \in G)$, $h_i \in H$ і $h_i = f(i)$.

Коли $G$ – скінчена множина, тоді сімейство $h_i\ (i \in G)$ називається *скінченим*.

Два сімейства називаються *ізоморфними*, коли $H_1 = H_2$ і існує таке бієктивне відображення $\varphi: G_1 \to G_2$, що

$$f_1(a) = f_2(\varphi(a)),$$

де $a$ – довільний об'єкт із $G_1$.

Абстракція певного класу всіх ізоморфних між собою сімейств називається *абстрактним сімейством*.

!Абстрактне сімейство тоді і тільки тоді представляє собою конструктивний об'єкт, коли представники цієї абстракції є скінченими сімействами і всі об'єкти $h_i$ – конструктивні.

Сімейство часто зображається у вигляді певної сукупності.

Наприклад, сукупність $\{a,a,a,b,b\}$ представляє собою сімейство, в якому 3 рази повторюється буква $a$ і 2 рази — буква $b$.

Це сімейство являється конструктивним об'єктом.

Легко переконатись, що сімейство представляє собою частковий випадок навантаженого графа, в якому немає дуг.

### *VI клас – скінчені матриці*

В різних областях математики важливу роль відіграють об'єкти, які називаються матрицями.

Матриця – це таблиця вигляду:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} = ( a_{ij} ),$$

де $a_{ij}$ – об'єкти довільної природи. Вони називаються елементами матриці.

Матриця скінчена, якщо числа $m$ і $n$ – скінчені (тобто якщо матриця містить скінчену кількість рядків і стовпців).

!Матриця тоді і тільки тоді є конструктивним об'єктом, коли вона скінчена і всі її елементи $a_{ij}$ є конструктивними.

В понятті матриці теж присутня доля абстракції: її можна записати на дошці, на папері ... Але ми абстрагуємося від цих форм запису – нас цікавить лише взаємне розташування елементів $a_{ij}$.

### *VII клас – букви, цифри і різні значки.*

Особливе місце в математиці (і не тільки) займають конструктивні об'єкти, які представляють собою букви, цифри і різні значки.

Із цих об'єктів конструюються слова, речення і формули.

Цифри, скінчені послідовності цифр і формули служать для позначення різних математичних обєктів: чисел, множин, функцій і т. ін.

Цифри, букви і значки теж є абстрактними об'єктами.

Наприклад: цифру «2» можна написати олівцем, чорнилом, краскою, будь-яким шрифтом і довільної величини.

Але при оперуванні цією цифрою, ми абстрагуємось від конкретних форм її представлення. Точніше, цифра «2» – це абстракція класу всіх її конкретних записів.

Це ж стосується довільної букви і значка.

**Загальні властивості конструктивних операцій.**

**Приклади конструктивних операцій.**

Нам потрібно зафіксувати деякий клас $U$ конструктивних об'єктів.

Клас $U$ повинен відігравати роль універсального класу, в рамках якого розглядають різні операції над конструктивними об'єктами.

Визначимо даний клас за допомогою певних умов:

а) класу $U$ належать всі натуральні і раціональні числа. Причому, під раціональними числами розуміють невід'ємні раціональні числа.

б) класу $U$ належать цифри всі абстрактні скінчені графи.

в) класу $U$ належать цифри, різні математичні значки, букви українського, латинського, грецького і інших алфавітів.

Будемо вимагати, щоб він був замкнутий відносно операцій утворення слів, скінчених множин, матриць і скінчених навантажених графів.

Це означає, що кожне слово (матриця, скінчена множина, скінчений навантажений граф), буквами (елементами, вершинними елементами) якого являється об`єкт із $U$, теж представляє собою об`єкт класу $U$.

*Операцією* будемо називати довільну функцію $f(\chi_1, \chi_2, ....., \chi_n)$, область визначення і область значень якої включаються в клас $U$.

Операція $f(\chi_1, \chi_2, ....., \chi_n)$ називається *конструктивною***,** коли існує інтуїтивно ясна процедура (алгоритм), за допомогою якої для довільного набору $\chi_1^{\circ}, \chi_2^{\circ}, ..., \chi_{n^{\circ}}$ із області визначення $f$ знаходиться значення $f(\chi_1^{\circ}, \chi_2^{\circ}, ..., \chi_n^{\circ})$.

Конструктивні операції бувають елементарними і неелементарними.

Елементарні конструктивні операції – це такі операції, які ми виконуємо легко, не задумуючись.

Для реалізації неелементарних конструктивних операцій потрібно застосувати певну послідовність елементарних операцій.

Прикладом елементарних дій являється додавання і множення одноцифрових чисел, записаних в десятковій системі числення. Більш складною конструктивною операцією являється операція додавання двох багатоцифрових чисел.

Введемо два конструктивні об'єкти *I* і *Ф,* які, відповідно, означають «істину» і «фальш» (хибність).

Операція $\mathrm{P}(\chi_1, \chi_2, ....., \chi_n)$, яка приймає значення із множини $\{I, \Phi\}$, називається *предикатом*.

При $n \geq 2$ предикат $\mathrm{P}(\chi_1, \chi_2, ....., \chi_n)$ виражає певне відношення між об'єктами, а при $n = 1$ – властивості об'єктів.

На практиці зустрічаються наступні предикати:

$< \chi = y >$ ::= $\chi$ дорівнює *y* (має сеанс на довільному класі об'єктів).

$\chi < y$ ::= *x* менше *y*.

$\chi / y$ ::= *x* ділиться на *y*.

$g(\chi)$ ::= *x* просте.

Оскільки P являється частковим випадком операції, то можна ставити питання про його конструктивність.

**Приклади найважливіших конструктивних операцій і предикатів.**

1. Операція вибору аргумента $\mathrm{I}_m^{\,n}(\chi_1, \chi_2, ....., \chi_n) = \chi_m$, де $1 \leq m \leq n, n \geq 1$.
2. Операція константи $C(\chi_1, \chi_2, ...\chi_n) = \alpha$, де *a* – довільний об'єкт *U*.
3. Операція відповідності скінченої множини $m(\chi_1, \chi_2, ....., \chi_n) = \{\chi_1, \chi_2, ....., \chi_n\}$

Ця операція кожному набору $\chi_1, \chi_2, ..., \chi_n$ об'єктів із $U$ ставить у відповідність скінчену множину $\{\chi_1, \chi_2, ....., \chi_n\}$.

4. $< \chi = y >$ ::= предикат рівності.

5. $M(\chi)$ ::= $x$ – скінчена множина об'єктів із $U$.

6. $T(\chi)$ ::= $x$ – скінчена матриця, елементами якої являються об'єкти із $U$.

7. $S(\chi)$ ::= $x$ – слово, буквами якого являються об'єкти із $U$.

8. $\Delta(\chi)$ ::= $x$ – скінчений навантажений граф, в вершинах якого знаходяться об'єкти із $U$.

9. $M_n(\chi)$ ::= $x$ – множина, в яку входить не більше, ніж $n$ елементів.

10. $S_n(\chi)$ ::= $x$ – слово, яке має не більше, ніж $n$ входжень букв.

11. $\Delta_n(\chi)$ ::= $x$ – навантажений граф, який має не більше, ніж $n$ вершин.

12. $T_m^{\,n}(\chi)$ ::= $x$ – матриця, яка має $m$ рядків і $n$ стовпців.

13. $\Delta^*(\chi)$ ::= $x$ – скінчений абстрактний граф.

14. $N(\chi)$ ::= $x$ – натуральне число.

15. $R(\chi)$ ::= $x$ – раціональне число.

16. $Укр(\chi)$ ::= $x$ – буква українського алфавіту.

17. $Lat(\chi)$ ::= $x$ – буква латинського алфавіту.

Всі ці операції і предикати означені на класі $U$.

Конструктивність операцій 1-3 очевидна.

Розглянемо предикат 4. $< \chi = y >$

Якщо припустимо, що він конструктивний, то це буде означати, що для довільної пари об'єктів $\chi$ і $y$ із $U$ можна сказати чи рівні вони, чи ні.

Але це не завжди можливо.

Візьмемо, наприклад число 5 і одноелементну матрицю (5). Чи рівні ці об'єкти?

Відповідь залежить від того, як ми домовимось.

Отже, однією з передумов конструктивності предикату $< \chi = y >$ є те, що для довільної пари об'єктів $x$ і $y$ із $U$ для якої інтуїтивно важко встановити істинність предикату $< \chi = y >$ є чітка зумовленість про їх рівність.

Так і само для предикатів 5-17.

Отже, припущення конструктивності предикатів 4-17 представляє собою певну систему умов, які накладаються на клас $U$.

Нехай задано деякий клас $\mathrm{K}(\mathrm{K} \subset U)$ .

Предикат $$\mathrm{P}_K(\chi) = \begin{cases} I, коли\ \ x \in K \\ \Phi, коли\ \ x \notin K \end{cases}$$

називається *характеристичним предикатом* класу $K$.

Предикат

$$\mathrm{P}_k^{\,*}(\chi) = \begin{cases} I, коли\ \ x \in K \\ \pi, коли\ \ x \notin K \end{cases}$$

називається *частково – характеристичним предикатом* класу $K$.

Клас $K$ називається *конструктивним*, коли предикат $\mathrm{P}_K(\chi)$ представляє собою конструктивний предикат.

Тобто клас $K$ – конструктивний, коли існує процедура, за допомогою якої для довільного об`єкту $\chi$ із класу $U$ можна встановити, чи входить $\chi$ в клас $K$ чи ні.

Клас $K$ називається *частково – конструктивним*, коли предикат $\mathrm{P}_k^{\,*}(\chi)$ конструктивний.

Кожен конструктивний клас являється частково – конструктивним. Але не навпаки.

Класи, для яких предикати $\mathrm{M}(\chi), \mathrm{T}(\chi), S(\chi), \Delta(\chi)$ являються характеристичними, позначимо, відповідно, через $\mathrm{M}, \mathrm{T}, S, \Delta$.

Всі ці класи є конструктивними.

На цих класах можна визначити конструктивні операції.

Розглянемо клас $S$.

Кожен об`єкт з $S$ представляє собою скінчену послідовність із $\chi_1, \chi_2, .....\chi_n$, де $\chi_i$ – об`єкт класу $U$. Число $n$ називається *довжиною слова* $\chi_1, \chi_2, .....\chi_n$.

Довжину слова $S$ будемо позначати $\|S\|$ .

$\|S\|$ є конструктивною операцією, яка кожному слову $S$ однозначно ставить у відповідність натуральне число. Це перша конструктивна операція.

Пусте слово позначається через $e$ (слово, яке не має ні однієї букви), $e \in S$ і $\|e\| = 0$.

Розглянемо другу конструктивну операцію – *операцію композиції слів* $S_1 \times S_2$.

Ця операція визначається умовою:

Якщо $S_1 = \chi_1\chi_2.....\chi_n$ і $S_2 = y_1y_2.....y_m$, то

$$S_1 * S_2 = \chi_1\chi_2.....\chi_n y_1y_2.....y_m.$$

(тобто слово $S_1$ дописується зліва до $S_2$).

Операцію композиції можна застосувати до будь-якої кількості слів.

Результат застосування цієї операції до слів $S_1, S_2, .....S_k$ позначається через $S_1 * S_2 * ..... * S_k$.

На $S$ можна визначити ще такі конструктивні операції:

$L(\chi_1\chi_2.....\chi_n) = \chi_2\chi_3.....\chi_n$ – викреслюється перша зліва буква.

$R(\chi_1\chi_2.....\chi_n) = \chi_1\chi_2....\chi_{n-1}$ – викреслюється перша справа буква.

$L(e) = R(e) = e$.

Розглянемо конструктивний предикат, означений на $S$.

Предикат $s_1 \nabla s_2 = \exists s_3 \exists s_4 (s_2 = s_3 * s_1 * s_4)$

тобто слово $S_1$ входить в слово $S_2$.

Пусте слово входить в кожне слово класу $S$.

В класі $M$ можна привести такі приклади конструктивних операцій і предикатів.

$A \cup B, A \cap B, A - B, A \subset B$, де $A, B \in M$.

Конструктивність цих операцій випливає із скінченності множин *A* і *B*.

В класі *T* матриць відома операція транспортування. При цій операції матриця:

$$\begin{matrix} a_{11} & a_{12}... & a_{13}.... & a_{1n} \\ a_{21} & a_{22}.... & a_{23}.... & a_{2n} \\ ... & .... & .... & .... \\ a_{m1} & a_{m2} & ..... & a_{mn} \end{matrix} \qquad (1)$$

Переходить в матрицю вигляду:

$$\begin{matrix} a_{11} & a_{21} & a_{31}.... & a_{m1} \\ a_{12} & a_{22} & a_{32....} & a_{m2} \\ .... & .... & .... & .... \\ a_{1n} & a_{2n} & a_{3n}.... & a_{mn} \end{matrix}$$

Ще одна конструктивна операція на *T*:

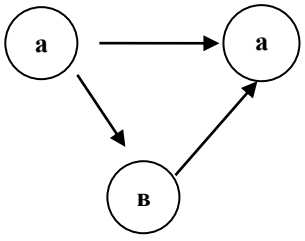
перетворення матриці в слово: матриця (1) переходить в слово вигляду:

$a_{11}a_{12}.....a_{1n}a_{21}a_{22}...a_{2n}..a_{m1}a_{m2}.....a_{mn}$

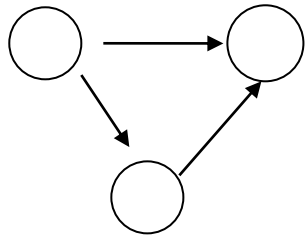
В класі $\Delta$ навантажених графів існує така операція, яка переводить навантажений граф $(G, M, f, H)$ в граф $(G, M)$.

При цій операції в графі $(G, M, f, H)$ «стираються» всі його вершинні елементи.

Наприклад навантажений граф:




Переходить в граф виду:



Із операцій і предикатів за допомогою певних схем можна утворити нові операції і предикати.

Наприклад, такі схеми:

1) схема підстановки операцій в операцію:

$$d(\chi_1, \chi_2, ..., \chi_m) = f_0(f_1(\chi_1, \chi_2, ..., \chi_n), ... f_m(\chi_1, \chi_2, ... \chi_n))$$

2) схема підстановки операцій в предикат:

$$Q(\chi_1, \chi_2, ....., \chi_m) = \mathrm{P}(f_1(\chi_1, \chi_2, ..., \chi_n), ..., f_m(\chi_1, \chi_2, ..., \chi_n)).$$

3) схема кон'юнкції, диз'юнкції і заперечення предикатів:

$$\mathrm{P}_1 \,\&\, \mathrm{P}_2, \mathrm{P}_1 \vee \mathrm{P}_2, \overline{\mathrm{P}}$$

Дані схеми обчислюються за допомогою таблиць:

| $\mathrm{P}_1$ | $\mathrm{P}_2$ | $\mathrm{P}_1 \,\&\, \mathrm{P}_2$ | $\mathrm{P}_1 \vee \mathrm{P}_2$ |
|---|---|---|---|
| $\Phi$ | $\Phi$ | $\Phi$ | $\Phi$ |
| $\pi$ | $\Phi$ | $\Phi$ | $\pi$ |
| $I$ | $\Phi$ | $\Phi$ | $I$ |
| $\Phi$ | $\pi$ | $\Phi$ | $\pi$ |
| $\pi$ | $\pi$ | $\pi$ | $\pi$ |
| $I$ | $\pi$ | $\pi$ | $I$ |
| $\Phi$ | $I$ | $\Phi$ | $I$ |
| $\pi$ | $I$ | $\pi$ | $I$ |
| $I$ | $I$ | $I$ | $I$ |

| P | $\overline{P}$ |
|---|---|
| $\Phi$ | $I$ |
| $\pi$ | $\pi$ |
| $I$ | $\Phi$ |

4) схема – альтернатор. По даній схемі із операцій $f_1(\chi_1,....,\chi_n)$ і $f_2(\chi_1,\chi_2,...,\chi_n)$ і предикату $P(\chi_1,\chi_2,...,\chi_n)$ утворюється нова операція $\varphi(\chi_1,\chi_2,...,\chi_n)$.

$$\varphi(\chi_1,\chi_2,...,\chi_n)=\begin{cases} f_1(\chi_1,\chi_2,.....\chi_n), коли\ \ P(\chi_1,...\chi_n)=I \\ f_2(\chi_1,\chi_2,.....\chi_n), коли\ \ P(\chi_1,...,\chi_n)=\Phi \\ \pi, коли\ \ P(\chi_1,...,\chi_n)=\pi \end{cases}$$

5) Схема – ітерація. По даній схемі із операції $f(\chi)$ і предикату $P(\chi)$ утворюється новий предикат $D(\chi)$:

$$D(\chi)=\begin{cases} I, коли\ існує\ таке\ натуральне\ число\ m, що\ P(f^0(\chi))=...P(f^{m-1}(\chi))=\Phi, \\ \qquad P(f^m(\chi)=I \\ \pi\ в\ іншому\ випадку \end{cases}$$

$f^i(\chi)$ означає $f(....f(f(\chi))....)$, $f^0(\chi)=\chi$

Всі ці схеми є конструктивними.

Тобто із конструктивних операцій і предикатів за даними схемами знову можна утворити конструктивні операції і предикати.

Нехай класи $K_1$ і $K_2$ конструктивні. А це означає, що характеристичні предикати $P_{K1}$ і $P_{K2}$ являються *конструктивними.*

Звідси випливає, що предикати $P_{K1}\vee P_{K2}$, $P_{K1}\&P_{K2}$ і $P_{K1}\&\overline{P_{K_2}}$, які представляють собою відповідні характерні предикати для класів $K_1\cup K_2$, $K_1\cap K_2$, і $K_1-K_2$ теж є конструктивні.

Отже, маємо таке твердження:

Твердження 1.1. Якщо класи $К_1$ і $К_2$ – конструктивні, то класи $К_1К_2$, $К_1 \wedge К_2$, $K_1 – K_2$ теж конструктивні

Нехай $K$ – довільний клас конструктивних об'єктів. Через $S(K)$, $M(K)$, $T(K)$ і $\Delta(K)$ відповідно позначимо класи всіх слів, скінчених множин, матриць і навантажених графів, буквами (елементами, вершинними елементами) яких являються об'єкти із класу $K$.

Твердження 1.2. Якщо клас К – конструктивний, то класи S(K), M(K), T(K) і Δ(K) теж конструктивні

Введемо поняття регулярного алгоритму.

Четвірку об'єктів $O=(\varphi_0, f, \varphi_1, P)$, де $\varphi_0, f, \varphi_1$ всюдиозначенні конструктивні функції і $P$ – всюдиозначений конструктивний предикат, назвемо *регулярним алгоритмом.*

Під всюдиозначеністю розуміємо означеність на всьому класі $U$.

Операції $\varphi_0, f, \varphi_1$ відповідно називають *кодуючою, головною і декодуючою* операціями алгоритму $O$.

Предикат $P$ називають *предикатом зупинки* алгоритма $O$.

Кожен регулярний алгоритм $O=(\varphi_0, f, \varphi_1, P)$ обчислює певну конструктивну операцію $O(x)$.

Спочатку знаходиться значення $y_0=\varphi_0(x)$. Потім послідовно обчислюються елементи $y_0, y_1, y_2, \ldots$, де $y_{i+1}=f(y_i)$ $(i=1,2\ldots,)$

Дані об'єкти обчислюються до тих пір, поки не знайдеться таке натуральне число $n$, що $P(y_0)=P(y_1)=P(y_{n-1})=Ф$ і $P(y_n)=I$.

Після цього покладається $O(x)=\varphi_1(y_n)$.

Якщо для всіх натуральних чисел $i$ $P(y_i)=Ф$, то $O(x)=\pi$ і будемо говорити, що алгоритм при даному $x$ зациклюється.

Коли $O(x)\neq\pi$, то будемо говорити, що алгоритм $O$ при даному $x$ зупиняється.

Очевидно, що ця умова, має місце тоді і тільки тоді, коли в послідовності $y_0, y_1, \ldots,$ знайдеться таке $y_n$, що $P(y_n)=I$.

Постулат — Кожна конструктивна операція обчислюється за допомогою деякого регулярного алгоритму.

Цей постулат підтверджується на практиці.

Наприклад, при обчисленнях на ЕОМ ми спочатку вхідну інформацію кодуємо. Це відповідає дії операції $\varphi_0(x)$. Після цього запускаємо машину.

В процесі своєї роботи машина послідовно застосовує деякий оператор $f$ і обчислює значення $y_0, y_{1,\ldots}$

Даний оператор $f$ діє до тих пір, поки не спрацює предикат зупинки $P$, тобто поки не виконається умова $P(y_n)=I$.

Після зупинки інформація, яка знаходиться в середині машини, декодується. Це відповідає дії операції $\varphi_1(y_n)$.

Твердження 1.3. (Теорема Поста) — Якщо класи К і К – частково-конструктивні, то вони є конструктивні

Доведення. Нехай $O_1$ і $O_2$ – відповідні регулярні алгоритми, які обчислюють частково – характеристичні предикати для класів $K$ і $\overline{K}$. Нехай $x$ – довільний об'єкт із $U$. Запускаємо при даному $x$ обидва алгоритми $O_1$ і $O_2$. Через те, що $K \cup \overline{K} = U$ і $K \cap \overline{K} = \varnothing$, то один з цих алгоритмів зупиниться, а другий зациклиться. Коли зупиниться $O_1$, то $xK$, коли зупиниться $O_2$, то $x \in \overline{K}$. Отже, для довільного об'єкта $x$ із $U$ алгоритмічно розпізнається належність його до класів $K$ і $\overline{K}$.

Твердження 1.4. — Коли $K$ – частково-конструктивний, але не конструктивний, то $\overline{K}$ не являється частково-конструктивним класом

Твердження 1.5. Область визначення конструктивної операції φ(x) являється частково-конструктивним класом.

Нехай *К* – деякий клас конструктивних об'єктів. Операція (або предикат) *f*(*x*) називається *конструктивною в класі К*, коли існує такий алгоритм О, що *O(x)=f(x)* для всіх *x* із *К*.

Клас $K_1$ називається *конструктивним в К* (*частково-конструктивним*), коли $K_1 \subset K$ і предикат $P_{K1}(P_{K1}^*)$ являється конструктивним в класі *К*.

Для довільного класу *К* мають місце аналоги тверджень 1.3 і 1.4.

Твердження 1.3[1]. Коли класи $K_1$ і $K\text{-}K_1$ частково-конструктивні в класі *К,* то вони є конструктивні в даному класі.

Твердження 1.4[1]. Якщо клас $K_1$ – частково-конструктивний, але не конструктивний в класі *К,* то $K\text{-}K_1$ не являється частково-конструктивним в класі *К*.

Твердження 1.6. Нехай клас *К* – конструктивний і $K_1 \subset K$. Для того, щоб $K_1$ був конструктивним (частково-конструктивним) в класі *К,* необхідно і достатньо, щоб $K\text{-}K_1$ був конструктивним (частково-конструктивним).

Зауваження: між поняттями конструктивного об'єкту і поняттями конструктивної операції і конструктивного класу є суттєва різниця – конструктивні операції і класи в більшості випадків представляють собою нескінчені об'єкти, в той час, як конструктивні об'єкти, повинні мати скінчену структуру.

## Ефективні нумерації

Множину *{0,1,……}* будемо позначати через *N*.

*Ефективною нумерацією* називається конструктивна функція *f(n)*, яка визначена на всій множині *N*.

Наприклад, $x+10$, $2x$, $x^2$ , $x^x$ – ефективні нумерації.

Наприклад, функція, яка натуральному числу *n* ставить у відповідність слово *в в….в* , в якому буква *в* взята *n* разів, теж ефективна нумерація.

Нехай *f* – деяка ефективна нумерація. Через *f(N)* позначимо множину всіх образів натуральних чисел при даній нумерації, тобто $f(N)=\{f(0),f(1),\ldots\}$

Будемо говорити, що множина *G ефективно нумерується*, коли існує така ефективна нумерація *f*, що $G=f(N)=\{f(0),f(1),\ldots\}$.

Наприклад, множини всіх парних і всіх непарних натуральних чисел ефективно нумеруються або порожня множина.

Твердження 2.1. Кожна скінчена множина $\{a_0,a_1,\ldots,a_n\}$ ефективно нумерується.

Твердження 2.2. Якщо множина *G* ефективно нумерується, то *G* – частково-конструктивний клас.

Відображення $f: A\to B$ називають *конструктивним,* коли функція *f* являється конструктивною функцією.

Коли відображення $f_1:A\to B$ і $f_2:B\to C$ є конструктивні, то конструктивним буде відображення $f_2(f_1)$.

Кожна ефективна нумерація являється конструктивним відображенням вигляду $N\to U$

Ефективна нумерація *f* називають *ін'єктивною,* коли $\forall i\,\forall j(i\neq j\Rightarrow f(i)\neq f(j))$.

*f* тоді і тільки тоді являється *ін'єктивною нумерацією* множини *G*, коли відображення $f:N\to G$ являється ін'єктивним і конструктивним.

Твердження 2.3. Коли $f:N\to G$ ін'єктивна нумерація множини $G$, то $f^{-1}:G\to N$ – конструктивне відображення.

Доведення. Нехай $x$ – довільний елемент із $G$. Послідовно порівнюючи $x$ з $f(0), f(1),\ldots$ знайдемо таке $n$, що $f(n)=x$ і $f(i)\neq x$ при $0\leq i<n$. Після цього покладемо $f^{-1}(x)=n$. Отже, ми одержали певний алгоритм обчислення функції $f^{-1}(x)$.Отже, $f^{-1}(x)$–конструктивне.

Твердження 2.4. Коли $G$ – нескінчена і ефективно нумерується, то $G$ – ін'єктивно нумерується.

Твердження 2.5. Нехай множини $G_1$ і $G_2$ – нескінчені і ефективно нумеруються. Тоді існує таке бієктивне конструктивне відображення $\varepsilon:G_1\to G_2$ що $\varepsilon^{-1}:G_2\to G_1$, теж конструктивне.

Позначимо через $M(G)$ систему всіх скінчених підмножини множини $G$. Множина $G$ *ефективно апроксимується,* коли існує така ефективна нумерація $\varphi(i)$, що $\varphi(i)\subset M(G)$, $\varphi(i)\subset\varphi(i+1)$ і $G=\bigcup_{i=0}^{\infty}\varphi(i)$.

Множина $G$ *задовільняє принципу конструктивного вибору,* коли на $M(G)$ існує така конструктивна функція $l(x)$, що $l(x)\in x$ для всіх непустих множин $x$ із $M(G)$.

Твердження 2.6. Множина $G$ тільки тоді ефективно нумерується, коли $G$ ефективно апроксимується і задовільняє принцип конструктивного вибору

Твердження 2.7. Підмножина $H$ ефективно нумерованої множини $G$ тоді і тільки тоді ефективно нумерується, коли $G$ ефективно апроксимується.

Твердження 2.8. Якщо $G_1$ і $G_2$ ефективно нумеруються, то $G_1 \cup G_2$ теж ефективно нумерується.

Твердження 2.9. Якщо $G_1$ і $G_2$ ефективно нумеруються, то $G_1 \cap G_2$ теж ефективно нумерується.

Твердження 2.10. Коли $G_1$ ефективно нумерується і $G_2$ – конструктивний клас, то $G_1$-$G_2$ теж ефективно нумерується.

Твердження 2.11. Нехай $f$:G→N ін'єктивне конструктивне відображення, G ефективно апроксимується і N ефективно нумерується. Тоді G ефективно нумерується

Покажемо ефективну нумерованість множини всіх простих чисел

Через $\boldsymbol{p_n}$ позначимо $\boldsymbol{n}$-е по порядку просте число.

Тобто $p_0=2, p_1=3, p_2=5, p_3=7...$

$\boldsymbol{p_n}$ представляє собою конструктивну функцію, яка кожному натуральному числу $\boldsymbol{n}$ ставить у відповідність $\boldsymbol{n}$-е по порядку просте число.

Для обчислення $\boldsymbol{p_n}$ можна застосувати таку *рекурсивну процедуру:* покладаємо $p_0=2$.

Нехай уже обчислені $p_0, p_1, \ldots p_n$. Послідовно перевіряємо подільність чисел $p_n+1, p_n+2, \ldots$ - на числа $p_0, p_1, \ldots p_n$.

Перше з чисел $p_n+1, p_n+2, \ldots$ яке не ділиться без залишку на жодне з чисел $p_0, p_1, \ldots p_n$ покладаємо рівним $\boldsymbol{p_{n+1}}$.

Нехай $\boldsymbol{N^\infty}$ - множина всіх скінчених послідовностей натуральних чисел. Тобто $\boldsymbol{N^\infty}$ - це множина всіх слів, буквами яких являються натуральні числа.

Твердження 2.12. Множина $N^\infty$ ефективно нумерується.

Через Z позначимо множину всіх скінчених матриць, елементами яких являються натуральні числа.

Твердження 2.12*. Множина Z ефективно нумерується.

Твердження 2.13. Якщо G ефективно нумерується і H – конструктивний клас, то G∩N теж ефективно нумерується.

Твердження 2.14. Якщо G ефективно нумерується, H⊂G і H - конструктивний клас, то H теж ефективно нумерується.

Твердження 2.15. Множина Δ* всіх абстрактних скінчених графів ефективно нумерується.

Нехай ***G*** – довільна множина об'єктів із класу ***U.*** Через *S(G)* позначимо множину всіх слів, буквами яких являються елементи множини ***G.***

Твердження 2.16. Якщо множина ***G*** ефективно нумерується, то множини *S(G)* і *M(G)* теж ефективно нумерується.

Твердження 2.16*. Коли множини $A_1, A_2, \ldots A_K$ теж ефективно нумеруються, то множина $A_1 \times A_2 \times \ldots \times A_K$ теж ефективно нумерується.

Через *M(G)* позначимо множину всіх скінчених підмножин множини *G*.

Через *T(G)* позначимо множину всіх матриць, елементами яких являються об'єкти із множини *G*.

Твердження 2.10. Якщо *G* ефективно нумеруються, то *T(G)* теж ефективно нумерується.

Нехай *D* – довільна множина слів. Через *Б(D)* позначимо множину всіх букв, із яких будуються слова множини *D*.

Зокрема, через *Б*(*S*) позначимо множину всіх букв слова *s* . Наприклад, *Б*(*ааввса*)={*а,в,с*}

Твердження 2.18. Якщо *D* – ефективно нумерується, то *Б(D)* теж ефективно нумерується.

Твердження 2.19. Частково-конструктивна підмножина *H* ефективно нумерованої множини *G* ефективно нумерується.

Твердження 2.20. Підмножина *H* ефективно нумерованої множини *G* тоді і тільки тоді ефективно нумеруються, коли *H* – частково-конструктивна множина.

Таким чином, ми розглянули поняття нумерованості для множин, елементами яких являються об'єкти класу *U*.

Можна також визначити поняття ефективної *нумерованості для множин, елементами яких являються функції або предикати.*

Система *Q* конструктивних функцій (предикатів) *ефективно нумерується,* коли виконуються такі умови:

*а)* $Q=\{f_i(x_1,x_2,\ldots,x_{n_i})/i\in N\}$

(1) *б)* існує алгоритм *A*, який для кожного натурального числа і обчислює $n_i$ і для кожного набору $i,x_1,x_2,\ldots,x_{n_i}$, де $x_1,x_2,\ldots,x_{n_i}\in U$, обчислює значення $f_i(x_1,x_2,\ldots,x_{n_i})$.

Умова *а)* означає, що систему *Q* можна представити у вигляді послідовності $f_0(x_1,x_2,\ldots,x_{n_0})$, $f_1(x_1,x_2,\ldots,x_{n_1})$.

Із умови *б)* випливає, що всі функції із *Q* обчислюються за допомогою єдиного алгоритму *A*.

Даний алгоритм називається *універсальним алгоритмом* для системи *Q*.

Коли в умовах *(1)* $n_i=n \ \forall i=0,1,2,\ldots$, тоді система $Q$ представляється за допомогою однієї конструктивної функції $F(i,x_1,\ldots,x_n)= f_i(x_1,\ldots,x_n)$.

В даному випадку функція $F(i,x_1,\ldots,x_n)$ називається *універсальною функцією* для системи $Q=\{f_i(x_1,x_2,\ldots,x_n)/i\in N\}$.

Ефективно нумеровану систему функцій можна також визначити на довільному конструктивному підкласі $K$ класу $U$. Для цього в умові *(1) б)* замість $U$ треба підставити $K$.

Нехай $\Sigma$ - деяка система класів. Через $Q_\Sigma$ і $Q_\Sigma$* відповідно позначимо системи характеристичних і частково-характеристичних предикатів для класів системи $\Sigma$, тобто:

$Q_\Sigma=\{P/P$- характеристичний предикат деякого класу із $\Sigma\}$.

$Q_\Sigma$*$=\{P$*$/P$*- частково-характеристичний предикат деякого класу із $\Sigma\}$.

Будемо говорити, що система $\Sigma$ конструктивних (частково-конструктивних) класів *ефективно нумерується*, коли ефективно нумерується система $Q_\Sigma(Q_\Sigma$*$)$.

Прикладом ефективно нумерованої системи класів являється така система: $\Sigma_0=\{K_i/i\in N\}$, де $K_i$ клас всіх натуральних чисел, які націло діляться на число $i+1$.

Універсальним предикатом для системи $\Sigma_0$ являється предикат $B(i,x)$, який визначається за умовою: натуральне число націло ділиться на $i+1$.

## Аксіоматизація елементарних теорій.

Множина $G$ називається *ефективною*, коли $G$ – конструктивна і ефективно нумерується.

Ефективними являються такі множини:

***1)*** Довільна скінчена множина об'єктів із $U$.

***2)*** Множина $N$ натуральних чисел.

***3)*** Множина парних і простих натуральних чисел.

***4)*** Множина $R$ раціональних чисел.

***5)*** Множина $\Delta^*$ скінчених абстрактних графів

Із тверджень попереднього параграфа випливають такі твердження:

Твердження 3.1. Якщо $G_1$ і $G_2$ ефективні , то множини $G_1 \cup G_2$, $G_1 \cap G_2$, і $G_1 - G_2$ теж ефективні.

Твердження 3.2. Якщо множина G ефективна, то множини S(G), M(G) і T(G) теж ефективні.

Тобто ефективною являється, наприклад, множина всіх слів, які будуються із букв українського (латинського, грецького та ін.) алфавіту.

Ефективними будуть також множини $N^\infty$, $N^n$ і $Z$.

Твердження 3.3. Коли $A_1, A_2, \ldots A_n$ – ефективні, то множина $A_1 \times A_2 \times \ldots \times A_n$ теж ефективні.

Із означення ефективної множини випливає, що нескінчена ефективна множина за своїми властивостями близька до множини $N$ натуральних чисел.

Тому часто елементи тої чи іншої ефективної множини $G$ представляються (нумеруються) за допомогою натуральних чисел і, навпаки, натуральні числа зображаються за допомогою елементів різних ефективних множин.

Наприклад, запис натурального числа $n$ в десятковій системі числення є не, що інше як представлення даного числа у вигляді слова, буквами якого являються цифри *0, 1, 2, …,9.*

Ефективність багатьох множин часто проявляються в тому, що ми наочно представляємо собі загальний вигляд об'єктів, які належать даним множинам.

Прикладом складної ефективної множини являється множина всіх простих натуральних чисел.

Хоча ефективні множини можуть розрізнятись за "технічною" складністю, але принципово вони належать до найбільш простих множин, які використовуються в математиці.

Будемо говорити, що функція $f(x_1,\ldots,x_n)$ *всюди означена на множині* $G$, коли виконується умова. $x_i \in G(1\le i\le n)\Rightarrow f(x_1,\ldots,x_n)\ne\pi$.

Коли має місце умова $f(x_1,\ldots,x_n)\ne\pi\Leftrightarrow x_i\in G(1\le i\le n)$, то будемо говорити, що $f(x_1,\ldots,x_n)$ *точно означена на множині* $G$.

Конструктивна функція (предикат) $f(x_1,x_2,\ldots,x_n)$ називається *ефективною,* коли $f(x_1,\ldots,x_n)$ точно означена на деякій ефективній множині $G_1$ і приймає значення із деякої ефективної множини $G_2$.

Зокрема, конструктивне відображення $\varepsilon$: $G_1\to G_2$, де $G_1$ і $G_2$ – ефективні множини, будемо називати *ефективним відображенням.*

Прикладом ефективних функцій являються функції $x+y$, $xy$, $x^y$, які визначені на $N$.

Розглянемо застосування теорії ефективно нумерованих множин до елементарних теорій.

*Елементарною теорією* називається пара $T=(G,H)$, де $G$ – ефективна множина і $H$ – підмножина множини $G$.

$G$ – це множина всіх висловлювань теорії $T$, а $H$ – множина теорем даної теорії.

Предикат $P(x_1,x_2,\ldots,x_n,y)$, точно означений на множині $G$, називається *правилом доведення в теорії* $T=(G,H)$, коли виконується така умова:

$x_i\in H(1\le i\le n)\wedge P(x_1,x_2,\ldots,x_n,y)=I\Rightarrow y\in H$.

Зокрема, коли $n=0$, то умова *(1)* має вигляд: $P(y)=I \Rightarrow y \in N$.

Правило вигляду $P(y)$ будемо називати *унарним правилом доведення в теорії.*

Якщо $n \geq 1$, то правило $P(x_1,x_2,\ldots,x_n,y)$, в теорії $T=(G,H)$, називають *ефективним,* коли предикат $P(x_1,x_2,\ldots,x_n,y)$ являється ефективним предикатом.

Система $Q$ правил доведення в $T=(G,H)$, називається *ефективною*, коли $Q$ – скінчена система і всі правила із $Q$ являються ефективними.

Нехай $Q$ – довільна (не обов'язкова ефективна) система правил доведення в теорії $T=(G,H)$. Нехай $P_1(y),P_2(y),\ldots,P_m(y)$ – всі унарні правила системи $Q$.

Система $Z_Q=\{y/y \in G \wedge (P_1(y) \vee P_2(y) \vee \ldots \vee P_m(y)=I)\}$

називається *системою Q-аксіом.*

Очевидно, що $Z_Q \subset H$. Система $Q_2=Q-\{P_1,P_2,\ldots,P_m\}$ представляє собою деяку систему правил доведення в теорії.

Послідовність $y_1,y_2,\ldots y_к$ елементів із $G$ називають $Q$ – <u>*доведенням,*</u> в теорії $T=(G,H)$, коли для кожного $y_i (1 \leq i \leq n)$ виконується хоча б одна з таких умов:

*а)* $y_i \in Z_Q$

*б)* знайдеться таке правило $P(x_1,x_2,\ldots,x_n,y_i)$ із $Q_2$ і такі елементи $y_{j_1}, y_{j_2},\ldots, y_{j_n}$, що

$1 \leq j_1 < i,\ 1 \leq j_2 < i,\ \ldots,\ 1 \leq j_n < i$ і $P(y_{j_1}, y_{j_2},\ldots, y_{j_n}, y_i)=I$

Коли $y_1,y_2,\ldots,y_к$ – $Q$ – доведення, то кожен лівий відрізок $y_1,y_2,\ldots,y_i$ $(1 \leq i \leq к)$ теж являється $Q$ – доведенням.

$Q$ – доведення називається $Q$ – *доведенням елемента* $y_к$.

Будемо говорити, що елемент $y$ із $G$ <u>*Q-доводиться*</u>, коли існує $Q$– доведення елемента $y$.

Через $L_Q$ позначимо множину всіх $Q$ – доведень.

Нехай множину всіх букв, із яких будуються слова множини $L_Q$ позначимо $Б(L_Q)$. Нехай $T_Q=Б(Q)$.

Через те, що кожен лівий відрізок $Q$ – доведення теж являється $Q$ – доведенням, то $T_Q$ представляє собою сукупність тих і тільки тих елементів множини $G$, які $Q$ – доводяться. Тобто справедливо $T_Q \subset H$.

Система правил $Q$ – доведення в теорії $T=(G,H)$, називають *повною* в цій теорії, коли $T_Q = H$.

Говорять, що теорія *$T$ аксіоматизується,* коли в даній теорії існує повна і ефективна система правил доведення.

Коли $Q$ – ефективна система правил доведення в $T=(G,H)$, тоді $Z_Q$ являється ефективною множиною.

Отже, коли в теорії $T=(G,H)$ існує повна і ефективна система правил доведення, то всі теореми даної теорії виводяться із ефективної системи аксіом $Z_Q$ за допомогою скінченої системи $Q_2$ власних і ефективних правил доведення.

Теорема Гермеса Твердження 3.4. Теорія $T=(G,H)$ тоді і тільки тоді аксіоматизується, коли множина її теорем ефективно нумерується.

Коли $H$ – скінчена множина, то дане твердження тривіальне. Тому будемо вважати, що $H$ – нескінчена.

Із твердження 3.4 випливає такий результат:

якщо теорія $T=(G,H)$ аксіоматизується, то вона аксіоматизується за допомогою однієї аксіоми і одного бінарного правила.

Твердження 3.5. Якщо в теорії $T=(G,H)$ існує ефективно нумерована і повна система $Q$ правил доведення, то множина $H$ ефективно нумерується.

Теорія $T=(G,H)$ називається *розв'язаною*, коли $H$ ефективна система.

В даному випадку система $Q=\{P_H(x)\}$, де $P_H$ – характеристичний предикат системи $H$, представляє собою ефективну і повну систему правил доведення в $T$.

Отже, в даному випадку непотрібно ніяких власних правил доведення.

В розв'язній теорії $T=(G,H)$ доведення висловлювання $x$ зводиться до алгоритмічної перевірки умови $x \in H$. Тому проблема доведення в розв'язній теорії принципово не являється творчою справою.

Твердження Кожна розв'язна теорія $T$ аксіоматизується.

Але існують аксіоматизовані теорії, які не являються розв'язними.

Коли теорія $T=(G,H)$ аксіоматизується, але не являється розв'язною, тоді існує деякий алгоритм $A^*$, що обчислює частково-характерний предикат $P_H^*(x)$ і не існує алгоритму, який би обчислював характерний предикат $P_H(x)$.

Часто алгоритм $A^*$ являється дуже складним і на практиці не використовується.

Існує зручний спосіб перевірки, чи аксіоматизована теорія розв'язна.

В багатьох елементарних теоріях $T=(G,H)$ існує ефективний оператор, який кожному $x$ із $G$ однозначно ставить у відповідність деяке висловлювання $\bar{x}$ ($\bar{x} \in G$).

Висловлювання $\bar{x}$, називається *запереченням висловлювання x.*

Елементарна теорія $T=(G,H)$ називається *несуперечливою* відносно заперечення $\bar{x}$, коли для кожного $x$ із $G$ використовується умова:

$$x \notin H \vee \bar{x} \notin H \quad (*)$$

Умова (*) говорить, що не існує в $G$ такого висловлювання $x$, для якого б $x$ і $\bar{x}$ були теоремами в $T$.

Елементарна теорія $T=(G,H)$ називається *повною відносно заперечення* $\bar{x}$, коли $x \in H \vee \bar{x} \in H$ (**)

Умова (**) говорить, що для кожного $x$ хоча б одн із висловлювань $x$ або $\bar{x}$ являється теоремою в $T$.

Твердження 3.6. Нехай заперечення $\bar{x}$ в $T=(G,H)$ являється ефективним і, крім того, задовільняє (*) і (**). При цих умовах теорія $T$ тоді і тільки тоді являється розв'язаною, коли $T$ аксіоматизується.

## Основи теорії графів

Теорія графів – це математичний апарат для формалізації (моделювання) реальних завдань по дослідженню властивостей скінченних множин із заданими відносинами між їх елементами. У їх числі завдання з області адміністрування мереж, інформаційних потоків, планування, проектування та управління різними системами.

Задачі на графах зручно перекладати на мови програмування, тобто вирішувати з використанням сучасної обчислювальної техніки.

Уміння вирішувати завдання на графах дозволить майбутньому фахівцю придбати досвід розробки технологій і методів теорії операцій для вирішення завдань при наукових дослідженнях і проектно-конструкторській діяльності.

Графи допомагають описувати і досліджувати різні системи об'єктів та їх зв'язки. Наприклад, у графі на рис.1 точки (вершини графа) можна інтерпретувати як міста, а лінії, що з'єднують вершини (ребра), як дороги, що з'єднують ці міста.

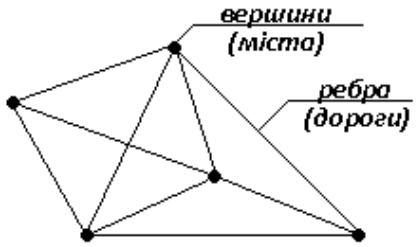



Рис.1

### 1. Загальні поняття

Формальне визначення графа таке.

Графом Г = (V, X) називається пара множин: V – множина, елементи якої називаються вершинами, X – множина невпорядкованих пар вершин, званих ребрами. Якщо $v, w \in V, x=(v,w) \in X$ , то кажуть, що ребро x з'єднує вершини v і w або *x інцидентне v і w*. Таким чином, {v, w} - позначення ребра. Якщо X являє собою впорядковані пари (тобто X - підмножина декартового добутку $V \times V$), то граф називається *орієнтованим*, а пари {v, w} називають *дугами*. Якщо множині X належать пари $v = w$, то такі ребра $(v, v)$ називають *петлями*.

Існування однакових пар {v, w} відповідає наявності паралельних або кратних ребер (дуг), а кратністю ребер називають кількість таких однакових пар.

Наприклад, кратність ребра $\{v_1, v_2\}$ в графі, зображеному на рис. 2, дорівнює двом, кратність ребра $\{v_3, v_4\}$ - трьом.

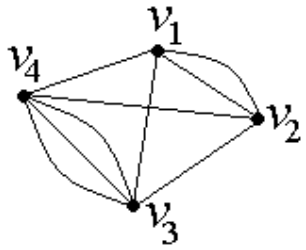



Рис.2

*Псевдограф* - граф, в якому є петлі і / або кратні ребра.

*Мультіграф* - псевдограф без петель.

Зауважимо, що графом також називають мультиграф, в якому жодна пара не зустрічається більше одного разу.

Отже, використовувані далі позначення:

*V* - множина вершин;

*X* - множина ребер або дуг;

*v* (або $v_i$) - вершина або номер вершини;

G, $G_0$ - неорієнтовані граф;

D, $D_0$ - орієнтований;

{v, w} - ребра неорієнтованого графа;

{v, v} - позначення петлі;

(v, w) - дуги в орієнтованому графі;

v, w - вершини, x, y, z - дуги і ребра;

n (G), n (D) кількість вершин графа;

m (G) - кількість ребер, m (D) - кількість дуг.

Приклади

1) Орієнтований граф D = (V, X), V = $\{v_1, v_2, v_3, v_4\}$,

$X=\{x_1=(v_1,v_2), x_2=(v_1,v_2), x_3=(v_2,v_2), x_4=(v_2,v_3)\}$.

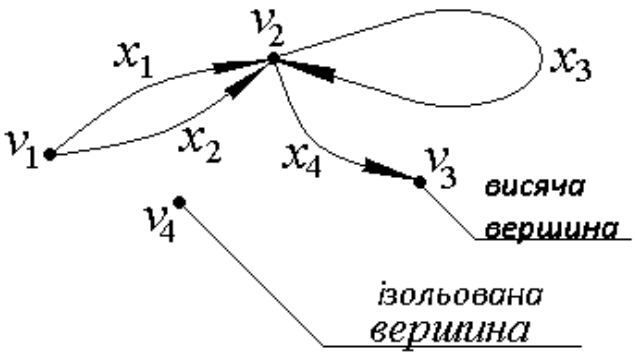



Рис. 3.

2) неорієнтовані графів, зображений на рис. 4:

$G=(V, X)$, $V=\{v_1, v_2, v_3, v_4, v_5\}$,
$X=\{x_1=\{v_1,v_2\}, x_2=\{v_2,v_3\}, x_3=\{v_2,v_4\}, x_4=\{v_3,v_4\}\}$.


Рис. 4.

**2. Поняття суміжності, інцидентності, степені**

Якщо $x$ = {v, w} – ребро, то v і w – кінці ребра $x$.

Якщо $x$ = (v, w) – дуга орієнтованого графа, то v – початок, w – кінець дуги.

Вершина v і ребро x неорієнтованого графа (дуга x орієнтованого графа) називаються *інцидентними*, якщо v є кінцем ребра x (початком або кінцем дуги x).

Вершини v, w називаються *суміжними*, якщо $\{v,w\}\in X$.

*Степінню* вершини v графа G називається число $\delta(v)$ ребер графа G, інцидентних вершині v.

Вершина графа, що має ступінь 0, називається ізольованою, а ступінь 1 – висячою.

*Напівстепінню* виходу (заходу) вершини $v$ орієнтованого графа D називається число $\delta^+(v)$ $(\delta^-(v))$ дуг орієнтованого графа D, що виходять з v (заходять в v).

Слід зауважити, що у разі орієнтованого псевдографа внесок кожної петлі інцидентної вершині v дорівнює 1 як в $\delta^+(v)$, так і в $\delta^-(v)$.

**3. Маршрути та шляхи**

Послідовність $v_1x_1v_2x_2v_3\ldots x_kv_{k+1}$, (де $k\geq 1$, $v_i\in V$, $i=1,\ldots,k+1$, $x_i\in X$, $j=1,\ldots,k$), в якій чергуються вершини і ребра (дуги) і для кожного j = 1, ..., k ребро (дуга) $x_j$ має вигляд $\{v_j,v_{j+1}\}$ (для орієнтованого графа $(v_j,v_{j+1})$), називається *маршрутом*, що з'єднує вершини $v_1$ і $v_{k+1}$ (*шляхом* із $v_1$ в $v_{k+1}$).

Приклад

У графі, зображеному на рис.4, $v_1x_1v_2x_2v_3x_4v_4x_3v_2$ – маршрут, $v_2x_2v_3x_4v_4$ – підмаршрут;

маршрут можна відновити і за таким записом $x_1x_2x_4x_3$;

якщо кратності ребер (дуг) дорівнюють 1, можна записати і так $v_1v_2v_3v_4v_2$.

*Ланцюг* – незамкнутий маршрут (шлях), в якому всі ребра (дуги) попарно різні.

*Цикл* – замкнутий ланцюг (в неорієнтованому графі).

*Контур* – замкнутий шлях (в орієнтованому графі).

*Простий шлях* (*ланцюг*) – шлях (ланцюг, цикл, контур), в якому жодна дуга / ребро не зустрічається двічі.

*Простий цикл* (*контур*) – цикл (контур), в якому всі вершини попарно різні.

*Гамильтоновий ланцюг* (*шлях*, *цикл*, *контур*) - простий ланцюг (шлях, цикл, контур), через всі вершини.

*Ейлеровий ланцюг* (шлях, цикл, контур) – ланцюг (шлях, цикл, контур), що містить всі ребра (дуги) графа по одному разу.

*Довжина маршруту* (*шляху*) – число ребер в маршруті (дуг в шляхові).

Твердження 1. Для того, щоб зв'язний псевдограф G володів ейлеровим циклом, необхідно і достатньо, щоб степені всіх його вершин були парними.

Твердження 2. Для того, щоб зв'язний псевдограф G володів ейлеровим ланцюгом, необхідно і достатньо, щоб він мав рівно 2 вершини непарної степені.

## 4. Матриці суміжності та інцидентності

Нехай $D = (V, X)$ орієнтований граф, $V=\{v_1,...,v_n\}$, $X=\{x_1,...,x_m\}$.

*Матриця суміжності* орієнтованого графа $D$ – квадратна матриця $A(D)=[a_{ij}]$ порядку $n$, де

$$a_{ij} = \begin{cases} 1, & (v_i, v_j) \in X \\ 0, & (v_i, v_j) \notin X \end{cases}$$

*Матриця інцидентності* – матриця $B(D)=[b_{ij}]$ порядку $n \times m$, де

$$b_{ij} = \begin{cases} 1, & v_i - \textit{кінець дуги } x_j, \\ -1, & v_i - \textit{початок дуги } x_j, \\ 0, & v_i - \textit{неінцидентна } x_j. \end{cases}$$

*Матрицею суміжності* неорієнтованого графа G = (V, X) називається квадратна симетрична матриця $A(G)=[a_{ij}]$ порядку $n$, де

$$a_{ij} = \begin{cases} 1, & \{v_i, v_j\} \in X \\ 0, & \{v_i, v_j\} \notin X \end{cases}$$

Для орієнтованого графа

$$a_{ij} = \begin{cases} 1, & (v_i, v_j) \in X \\ 0, & (v_i, v_j) \notin X \end{cases}$$

*Матрицею інцидентності* графа G називається матриця $B(G)=[b_{ij}]$ порядка $n \times m$, де

$$b_{ij} = \begin{cases} 1, v_i - \textit{инцидентна ребру } x_j, \\ 0, v_i - \textit{неинцидентна ребру } x_j. \end{cases}$$

**5. Зв'язність. Компоненти зв'язності**

*Підграфом* графа G (орієнтованого графа D) називається граф, всі вершини і ребра якого містяться серед вершин і ребер графа G (D).

Підграф називається *власним*, якщо він відмінний від самого графа.

Кажуть, що *вершина w* орієнтованого графа D (графа G) *досяжна з вершини v*, якщо або w = v, або існує шлях (маршрут) з *v* в *w*.

Граф (орієнтований граф) називається *зв'язним* (*сильно зв'язним*), якщо для будь-яких двох його вершин *v*, *w* існує маршрут (шлях), що з'єднує *v* і *w*.

*Компонентою зв'язності* графа G (*сильної зв'язності* орієнтованого графа D) називається його зв'язний (сильно зв'язний) підграф, який не є власним підграфом жодного іншого зв'язного (сильно зв'язного) підграфа графа G (орієнтованого графа D).

**6. Матриці досяжності і зв'язності**

Нехай $A(D)$ – матриця суміжності орієнтованого псевдографа $D=(V,X)$ (або псевдографа $G=(V,X)$), де $V=\{v_1,\ldots, v_n\}$. Позначимо через $A^k=[a^{(k)}_{ij}]$ $k$-ву степінь матриці суміжності $A(D)$.

Елемент $a^{(k)}_{ij}$ матриці $A^k$ орієнтованого псевдографа $D=(V,X)$ (псевдографа $G=(V,X)$) дорівнює числу всіх шляхів (маршрутів) довжини $k$ із $v_i$ в $v_j$.

*Матриця досяжності* орієнтованого графа $D$ – квадратна матриця $T(D)=[t_{ij}]$ порядку $n$, елементи якої рівні

$$t_{ij}=\begin{cases}1, & v_j \text{ досяжна із } v_i,\\ 0, & \text{в противному випадку.}\end{cases}$$

*Матриця сильної зв'язності* орієнтованого графа $D$ – квадратна матриця $S(D)=[s_{ij}]$ порядку $n$, елементи якої рівні

$$s_{ij}=\begin{cases}1, & v_j \text{ досяжна із } v_i \text{ і } v_i \text{ досяжна із } v_j\\ 0, & \text{в противному випадку.}\end{cases}$$

*Матриця зв'язності* графа $G$ – квадратна матриця $S(G)=[s_{ij}]$ порядку $n$, елементи якої рівні

$$s_{ij}=\begin{cases}1, & \text{якщо } \exists \text{ маршрут, зєднуючий } v_j \text{ і } v_i,\\ 0, & \text{в противному випадку.}\end{cases}$$

Твердження 3. Нехай $D=(V,X)$ – орієнтований граф, $V=\{v_1,\ldots, v_n\}$, $A(D)$ – його матриця суміжності. Тоді

1) $T(D)=\text{sign}[E+A+A^2+A^3+\ldots A^{n-1}]$,

2) $S(D)=T(D)\&T^T(D)$ ($T^T$ – транспонована матриця, $\&$ – поелементне множення).

Нехай $G = (V, X)$ – граф, $V=\{v_1,\ldots, v_n\}$, $A(G)$ – його матриця суміжності. Тоді

$S(G)=\text{sign}[E+A+A^2+A^3+\ldots A^{n-1}]$ ($E$ – одинична матриця порядку $n$).

## 7. Відстані в графі

Нехай $G=(V,X)$ – граф (або псевдограф). *Відстанню* між вершинами $d(v,w)$ називається мінімальна довжина шляху між ними, при цьому $d(v,v)=0$, $d(v,w)=\infty$, якщо не існує шляху.

Відстань у графі задовольняє аксіомам метрики

1) $d(v,w)\ge 0$, $d(v,w)=0 \Leftrightarrow v=w$
2) $d(v,w)=d(w,v)$ (в неорієнтованому графі)

3) $d(v,w) \le d(v,w_1) + d(w_1,v)$

4) $d(v,w) < \infty$ в зв'язному неорієнтованому графі.

Нехай $G = (V, X)$ зв'язний граф (або псевдограф).

*Діаметром* графа G називається величина

$$d(G) = \max_{v,w \in V} d(v,w).$$

Нехай $v \in V$.

*Максимальним віддаленням (ексцентриситетом)* у графі G від вершини $v$ називається величина $r(v) = \max_{w \in V} d(v,w)$.

*Радіусом* графа G називається величина $r(G) = \min_{v \in V} r(v)$

*Центром* графа G називається будь-яка вершина $w \in V$ така, що

$r(w) = r(G)$.

## 8. Образ і прообраз вершини і безлічі вершин

Нехай $D = (V, X)$ орієнтований граф, $v \in V$ - деяка вершина, $V_1 \subseteq V$.

Позначимо $D(v) = \{w \in V \mid (v,w) \in X\}$ - образ вершини $v$;

$D^{-1}(v) = \{w \in V \mid (w,v) \in X\}$ - прообраз вершини $v$;

$D(V_1) = \bigcup_{v \in V_1} D(v)$ - образ множини вершин V1;

$D^{-1}(V_1) = \bigcup_{v \in V_1} D^{-1}(v)$ - прообраз множини вершин V1.

## 9. Навантажені графи

*Навантажений граф* - орієнтований граф D = (V, X), на множині дуг якого визначена деяка функція $l : X \to R$, яку називають ваговою функцією. Цифра над дугою (див. рис. 5) - вага дуги (ціна дуги).

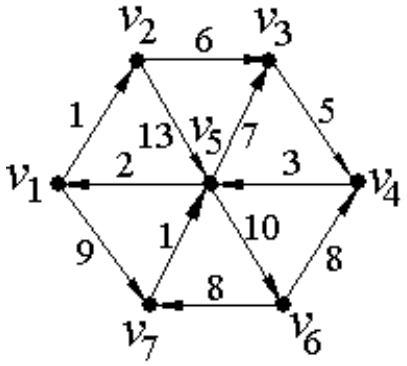



Рис. 5.

Позначимо для будь-якого шляху П навантаженого орієнтованого графа D через $l$ (П) суму довжин дуг, що входять в шлях П. (Кожна дуга рахується стільки разів, скільки вона входить в шлях П).

Величина $l$ називається *довжиною* шляху.

Якщо вибрати ваги рівними 1, то прийдемо до ненавантаженого графу.

*Шлях* в навантаженому орієнтованому графі з вершини $v$ у вершину $w$, де $v \neq w$, називається *мінімальним*, якщо він має найменшу довжину.

Аналогічно визначається *мінімальний шлях* в навантаженому графі.

Введемо матрицю довжин дуг $C(D)=[c_{ij}]$ порядку $n$, причому

$$c_{ij} = \begin{cases} l(v_i, v_j), & (v_i, v_j) \in X, \\ \infty & (v_i, v_j) \notin X. \end{cases}$$

*Властивості мінімальних шляхів в навантаженому орієнтованому графі*

1) Якщо для $\forall$ дуги $x \in X$ $l(x) > 0$, то довільний мінімальний шлях (маршрут) є простим ланцюгом;

2) якщо $v_1, v_2, ..., v_k$ - мінімальний шлях (маршрут) то для $\forall\ i,j\ :\ 1 \le i < j \le k$ шлях (маршрут) $v_i v_{i+1} ... v_{j-1} v_j$ теж є мінімальним;

3) якщо $v...uw$ - мінімальний шлях (маршрут) серед шляхів (маршрутів) з $v$ в $w$, що містять не більше k +1 дуг (ребер), то $v...uw$ - мінімальний шлях (маршрут) з $v$ в $u$ серед шляхів (маршрутів), що містять не більше $k$ дуг (ребер).

## 10. Дерева і цикли

Граф G називається *деревом,* якщо він є зв'язним і не має циклів.

Граф G називається *лісом*, якщо всі його компоненти зв'язності - дерева.

*Властивості дерев*:

Наступні твердження еквівалентні:

1) Граф G є деревом.

2) Граф G є зв'язним і число його ребер рівно на 1 менше числа вершин.

3) $\forall$ дві різні вершини графа G можна з'єднати єдиним (і при цьому простим) ланцюгом.

4) Граф G не містить циклів, але, додаючи до нього будь-яке нове ребро, отримуємо рівно один і притому простий цикл.

Твердження 4. Якщо у дерева G є, принаймні, 1 ребро, то у нього знайдеться висяча вершина.

Твердження 5. Нехай G зв'язний граф, а $v$ - висяча вершина в G, граф $G'$ одержується з G в результаті видалення вершини $v$ і інцидентного їй ребра. Тоді $G'$ теж є зв'язним.

*Остовним деревом* зв'язного графа G називається будь-який його підграф, що містить всі вершини графа G і який є деревом.

Нехай G - зв'язний граф. Тоді остовне дерево графа G повинно містити $n(G)$-1 ребер. Значить, для отримання остовного дерева з графа G потрібно видалити $m(G)-(n-1)$ ребер.

Число $v(G)=m(G)-n(G)+1$ називається цикломатичним числмо графа G.

# ОСНОВНІ АЛГОРИТМИ В КОМБІНАТОРИЦІ

На практиці люди часто зустрічаються із задачами, в яких необхідно підрахувати число всіх можливих способів розміщення деяких предметів скінченої множини або число всіх можливих способів виконання певної дії зі скінченої множини таких дій.

Наприклад.

1. Скількома способами можуть розподілитися глядачі на чемпіонаті світу з футболу серед 24-ти команд-учасниць фінальної групи?

2. З пункту *A* міста Києва в пункт *B* можна доїхати трьома видами транспорту:

- тролейбусом (Т),
- автобусом (А),
- метро (М);

з пункту *B* в пункт *C* – лише двома видами транспорту: тролейбусом (Т) і автобусом (А). Скількома способами можна доїхати з пункту *A* в пункт *C* міста Києва?

3. Агентство нерухомості має базу даних з *n* записів, причому кожен запис містить одну пропозицію (що є в наявності) і один запит (що потрібно). Потрібно знайти всі такі пари записів, в яких пропозиція першого запису співпадає з запитом другого запису і навпаки, пропозиція другої співпадає з запитом першої (тобто це називається підбором варіантів обліку).

4. Скількома різними способами можна розставити дужки у виразі $a+b+c+d+e+f$, якщо операція + асоціативна, а букви $a,b,c,d,e,f$ – деякі числа?

На перший погляд здається, що формули для розв’язання цих задач корисні лише для розв'язання олімпіадних задач і не мають відношення до практичного програмування.

Насправді, це зовсім не так. Обчислення на дискретних скінчених математичних структурах, які часто називають комбінаторними обчисленнями, потребують комбінаторного аналізу для встановлення властивостей і виявлення оцінки застосовності алгоритмів, що використовуються.

Для прикладу розглянемо задачу 3.

Нехай СУБД даного агентства нерухомості дозволяє перевірити варіант за одну мілісекунду. Тоді при “лобовому” алгоритмі пошуку варіантів (кожен запис порівнюється з кожним) потрібно $n(n-1)/2$ порівнянь.

Якщо $n=100$, то відповідь буде одержана за 4,95 секунди.

Але якщо $n=100\ 000$ (що досить реально), то відповідь буде одержано за 4 999 950 секунд – а це 1389 годин!

І це ми оцінили лише трудомісткість підбору прямих варіантів, а ще існують варіанти, коли кількість учасників угоди більше двох …

Отже, як бачимо, комбінаторні обчислення потребують попереднього аналізу і кількісної оцінки вихідних задач і алгоритмів, що використовуються.

Зазвичай задачі оцінюються з точки зору розміру, тобто загальної кількості різних варіантів, серед яких треба знайти розв'язок, а алгоритми оцінюються з точки зору складності.

При цьому розрізняють складність за часом (часову складність), тобто кількість необхідних кроків алгоритму, і складність за пам'яттю (або об'ємну складність), тобто об'єм пам'яті, необхідний для роботи алгоритму.

На практиці виникає необхідність підрахувати кількість можливих комбінацій об'єктів, що задовольняють певним умовам. Задачі, в яких визначають всі можливі різні комбінації, складені зі скінченого числа елементів за деяким правилом, називають комбінаторними (наші задачі **1-4**), а розділ

математики, в якому вивчається їх розв’язання, – комбінаторикою, методи їх розв'язку – методами комбінаторного аналізу.

Таким чином, у комбінаториці (комбінаторному аналізі) вивчають об'єкти зі скінченої множини $A=\{a_1,a_2,a_n\}$ та їх властивості, а також визначають кількість об'єктів із певними властивостями.

Розглядають також принципи, що використовуються в різних задачах, на яких ґрунтуються важливі методи математичного доведення, широко застосовні в теорії скінчених автоматів та інших розділах.

Для формулiровки і розв'язку комбінаторних задач використовують різні моделі комбінаторних конфігурацій.

Найбільш популярні такі:

1.Дано *n* предметів. Їх треба розмістити по *m* ящиках так, щоб виконувались задані обмеження. Скількома способами це можна зробити?

2.Розглянемо множину функцій $F: X \to Y$, де $|X|=n$, $|Y|=m$, $X=\{1,..,n\}$.

Не обмежуючи загальності, можна вважати, що

$Y=\{1,..,m\}$, $F =< F(1),..., F(n) >$, $1 \le F(i) \le m$.

Скільки існує функцій *F*, що задовольняють заданим обмеженням?

Є два основні правила комбінаторики.

1.Правило суми. Якщо об'єкт *x* можна вибрати $n_1$ способами, а інший об'єкт $y-n_2$ способами, то можна вибрати або *x*, або $y$ $n_1+n_2$ способами.
Наприклад.

Студент має вибрати тему курсової роботи зі списку, розміщеного на трьох аркушах. Аркуші містять відповідно 20,15 і 17 тем. З якої кількості можливих тем студент робить свій вибір?

За правилом суми ця кількість: 20+15+17=52.

2.Правило добутку. Якщо об'єкт *x* можна вибрати $n_1$ способами та після кожного такого вибору об'єкт *y* можна вибрати $n_2$ способами, то пару об'єктів *(x,y)* у зазначеному порядку можна вибрати $n_1 n_2$ способами.

Або інакше: нехай деяку процедуру можна виконати, розв'язавши два завдання. Якщо є $n_1$ спосіб розв'язати перше завдання та $n_2$ способи розв'язати після цього друге завдання, то всю процедуру можна виконати $n_1 n_2$ способами.

Наприклад.

1. Розглянемо задачу **1**. Золоту медаль може одержати будь-яка з 24-х команд, тобто є 24 можливості. Срібну медаль може виграти одна з 23-х команд, бронзову – одна з 22-х команд.

Отже, за правилом добутку загальне число способів розподілу медалей 24·23·22=12144.

2. Розглянемо задачу **2**. Розв'язок цієї задачі зводиться до підрахунку числа елементів в декартовому добутку множин $\{A,T,M\}\times\{A,T\}$.

Кількість таких елементів дорівнює добутку числа елементів першої множини на число елементів другої множини, тобто 3·2=6.

Отже, існує шість способів доїхати з пункту *A* в пункт *C*.

3. Наступна задача.

Скільки тризначних чисел можна скласти з цифр 0,1,2,3,4,5, якщо:

1) цифри можуть повторюватись;
2) жодна з цифр не повторюється двічі;
3) цифри непарні і можуть повторюватися.

Розв'язок.

1) Першою цифрою може бути одна з цифр 1,2,3,4,5, оскільки 0 не може бути першою цифрою, бо в цьому випадку число не буде тризначним.

Якщо перша цифра вибрана, то друга може бути вибрана шістьма способами, як і третя.

Отже, загальне число тризначних цифр 5·6·6=180.

2) Першою цифрою може бути одна з цифр – 1,2,3,4,5. Якщо перша цифра вибрана, то другою може бути теж одна з п'яти цифр (тут уже враховується 0), а третя може бути вибрана чотирма способами з 4-х цифр, що залишились. Отже, загальна кількість 5·5·4=100.

3) Першою цифрою може бути одна з трьох цифр: 1,3,5 (непарні).

Другою теж може бути одна з цих трьох цифр. Аналогічно і третя може бути вибрана трьома способами.

Отже, загальна кількість таких чисел 3·3·3=27.

4. В одній із версій мови БЕЙСІК ім'я змінної – це рядок з одного чи двох символів, якими можуть бути 26 букв латинського алфавіту та 10 цифр. Першим символом має бути буква. Крім того, неможна використовувати п'ять двосимвольних рядків, які зарезервовані для спеціального використання.

Скільки різних імен змінних є в цій версії мови БЕЙСІК?

Розв’язок.

Нехай $V$ – величина, яку треба обчислити, $V_1$ – кількість односимвольних імен, $V_2$ – двосимвольних.

За правилом суми всього імен $V = V_1 + V_2$.

Очевидно $V_1$=26, за правилом добутку $V_2$=26·36-5=931.

Отже, $V$=26+931=957.

## Основні комбінаторні об'єкти.

Спочатку визначимо важливе поняття вибірки.

Нехай задано скінченну непорожню множину $A=\{a_1,a_2,..,a_n\}$ і виконано *r* таких кроків.

Крок 1. З множини *A* вибирають деякий елемент $a_{i_1}$.

Крок 2. З множини *A* чи $A\setminus\{a_{i_1}\}$ вибирають деякий елемент $a_{i_2}$.

.....

Крок *r*. Якщо $a_{i_1}, a_{i_2} \ldots, a_{i_{r-1}}$ – елементи, які вибрані на перших *r-1* кроках $(r \geq 3)$, то на цьому кроці вибирають деякий елемент $a_{i_r}$ з множини *A* або $A\setminus\bigcup_{k=1}^{r-1}\{a_{i_k}\}$. Тоді елементи $a_{i_1}, a_{i_2} \ldots, a_{i_r}$ утворюють вибірку обсягом *r*, або *r*-вибірку з множини *A*.

Вибірку називають впорядкованою, якщо задано порядок її елементів.

Якщо ні – невпорядкованою.

Тобто впорядкована вибірка – це кортеж (вектор) з *r* компонентів і позначається $(b_1,b_2,..,b_r)$, $b_i \in A$, $i$=1,.., *r*.

Невпорядковану вибірку будемо позначати $(b_1,b_2,..,b_r)$, $b_i \in A$, $i$=1,.., *r*.

Впорядковані вибірки з *n*-елементної множини називаються розміщеннями з *n* елементів по *r*.

Невпорядковані вибірки з *n*-елементної множини називаються сполученнями з *n* елементів по *r*.

Існують два способи вибору елементів.

1 спосіб. На кожному кроці вибирають елемент з усієї множини *A*.

Отже, один і той самий елемент з множини *A* може зустрітись у вибірці декілька разів.

Такі вибірки називаються вибірками з повтореннями.

2 спосіб. Вибраний елемент вилучають з множини $A$. Це означає, що на кожному $i$-му кроці $(1 < i \le k)$ вибирають елемент з множини $A \setminus \bigcup_{k=1}^{r-1} \{a_{i_k}\}$, і вибірка не містить однакових елементів.

Такі вибірки називають вибірками без повторень.

Наприклад.

Задано множину $A = \{a,b,c\}$, тобто $n$=3.

Наведемо розміщення без повторень з трьох елементів по два, тобто $r$=2: *(a,b), (a,c), (b,c), (b,a), (c,a), (c,b).*

Розміщення з повтореннями з трьох елементів по два: *(a,b), (a,c), (b,c), (b,a), (c,a), (c,b), (a,a), (b,b), (c,c).*

Сполучення без повторень з трьох елементів по два:

$[a,b]$, $[a,c]$, $[b,c]$.

Сполучення з повтореннями з трьох елементів по два:

$[a,b]$, $[a,c]$, $[b,c]$, $[a,a]$, $[b,b]$, $[c,c]$.

Тобто сполучення без повторень з $n$ елементів по $r$ – це просто $r$-елементні підмножини з $n$ елементів, отже, їх можна записати так: $\{a,b\}$, $\{a,c\}$, $\{b,c\}$.

Сполучення з повтореннями – це не є множина у звичайному розумінні, її елементи можуть повторюватись, тобто зустрічатись більше одного разу.

Обчислення кількості розміщень і сполучень.

Кількість всіх розміщень без повторень з $n$ елементів по $r$ позначають $A_n^r$ (або $A(n,r)$), де $r$ і $n$ - невід'ємні цілі числа, $r \le n$.

Кількість різних розміщень з повтореннями з $n$ елементів по $r$ позначають $\tilde{A}_n^r$ (або $\tilde{A}(n,r)$), де $r$ і $n$ – будь-які невід'ємні цілі числа.

Кількість всіх сполучень без повторень з $n$ елементів по $r$ позначають $C_n^r$ (або $C(n,r)$), де $r$ і $n$ – невід'ємні цілі числа, $r \le n$.

Кількість всіх сполучень з повтореннями з $n$ елементів по $r$ позначають $H_n^r$ (або $H(n,r)$), де $r$ і $n$ – будь-які невід'ємні цілі числа.

Числа $C_n^r$ називаються біноміальними коефіцієнтами.

Твердження.

$$A_n^r = n(n-1)...(n-r+1) = \frac{n!}{(n-r)!} \qquad \textbf{(1)}$$

$$\tilde{A}_n^r = n^r \qquad \textbf{(2)}$$

$$C_n^r = \frac{A_n^r}{r!} = \frac{n!}{r!(n-r)!} \qquad \textbf{(3)}$$

$$H_n^r = C_{n+r-1}^r \qquad \textbf{(4)}$$

## Перестановки.

Перестановка з $n$ елементів – це особливий випадок розміщення без повторень з $n$ елементів, коли в розміщення входять усі елементи.

Перестановки з $n$ елементів ще називають $n$-перестановками.

Окремі $n$-перестановки різняться лише порядком елементів.

Кількість таких перестановок позначається $P_n$. Формулу для $P_n$ одержують з формули (1) для кількості розміщень без повторень.

$$P_n = A_n^n = n!$$

Розглянемо задачу про перестановки з $n$ елементів при умові, що не всі елементи різні (перестановки з повтореннями).

Точніше, нехай є $n$ елементів $k$ різних типів, а число $n_j (j = 1,..,k)$ – кількість елементів $j$-го типу.

Очевидно, що $n_1 + n_2 + .. + n_k = n$.

Перестановки з $n$ елементів за такої умови називають перестановками з повтореннями.

Кількість таких перестановок позначається $P_n(n_1, n_2,.., n_k)$.

Щоб знайти явний вираз для $P_n(n_1, n_2,.., n_k)$, візьмемо окрему перестановку та замінимо в ній усі однакові елементи різними.

Тоді кількість різних перестановок, котрі можна отримати з узятої однієї перестановки $n_1! n_2! .. n_k!$

Якщо зробити це для кожної перестановки, то одержимо $n!$ перестановок.

Отже, $P_n(n_1, n_2,.., n_k)\ n_1! n_2! .. n_k! = n!$

Звідси одержуємо формулу для кількості розміщень з повтореннями:

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

Приклад 1.

Знайти кількість слів (рядків), які можна утворити, переставляючи букви слова PRODUCT.

В даному слові жодна буква не повторилася.

Отже, можна утворити $P_7 = 7! = 5040$ слів.

Приклад 2.

Знайти кількість слів, які можна утворити, переставляючи букви слова SUCCESS.

У цьому слові є повторні входження букв, тому використовують формулу для перестановок з повторенням.

$$P_7(3,2,1,1) = \frac{7!}{3!2!1!1!} = 420 \text{ слів.}$$

Отже, перестановки деяких елементів – це всі можливі способи, якими ці елементи (частіше – числа) можна вистроїти в ряд.

Підстановка – це операція, яка міняє порядок елементів в перестановці. Або можна означити по-іншому:

підстановка *n*-ї степені – це взаємно-однозначне відображення множини із *n* елементів на себе.

Для задання підстановки необхідно:

1) задати область визначення, тобто всю сукупність елементів, над якими проводиться підстановка;
2) задати алгоритм підстановки, тобто для кожного елемента з області визначення вказати елемент, в який він перейде під дією підстановки, причому різні елементи мають переходити в різні.

Введемо операцію множення множення ($\times$) над підстановками.

Добутком підстановок називається підстановка, яка одержується в результаті послідовного виконання спочатку першої, а потім другої із перемножуваних підстановок.

Наприклад, якщо $a=\begin{pmatrix} x_1 & x_2 & x_3 \\ x_2 & x_1 & x_3 \end{pmatrix}$, $b=\begin{pmatrix} x_1 & x_2 & x_3 \\ x_1 & x_3 & x_2 \end{pmatrix}$, то

$$a\times b=\begin{pmatrix} x_1 & x_2 & x_3 \\ x_3 & x_1 & x_2 \end{pmatrix}.$$

Множення підстановок не є комутативним, але є асоціативним.

Одиницею групи підстановок є тотожня підстановка *I,* тобто така підстановка, кожен елемент якої переходить сам в себе

$$I=\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ x_1 & x_2 & x_3 & x_4 \end{pmatrix}.$$

Для кожної підстановки $P$ існує обернена $P^{-1}$, яка знаходиться заміною верхнього рядка на нижній. Справедливо співвідношення:

$$P\times P^{-1}=P^{-1}\times P=I.$$

Підстановки можна розкладати в добуток циклів.

Наприклад, розглянемо таку підстановку

$$P=\begin{pmatrix}0\,1\,2\,3\,4\,5\,6\,7\,8\,9\\3\,5\,8\,7\,2\,0\,9\,1\,4\,6\end{pmatrix}.$$

Запишемо її в такому вигляді:

$$P=\begin{pmatrix}0\,3\,7\,1\,5\,2\,8\,4\,6\,9\\3\,7\,1\,5\,0\,8\,4\,2\,9\,6\end{pmatrix}.$$

Або у вигляді множення:

$P=\begin{pmatrix}03715\\37150\end{pmatrix}\begin{pmatrix}284\\842\end{pmatrix}\begin{pmatrix}69\\96\end{pmatrix}$. Ці підстановки можна виконувати в довільному порядку, оскільки вони незалежні, тобто не мають спільних елементів.

Кожну зі складових підстановок можна записати у вигляді циклічної підстановки (циклу), тобто

$$P=(03715)(284)(69).$$

Будь-яку підстановку можна розкласти в добуток незалежних циклів.

Алгоритм цього розкладу: шукаємо, в який елемент перейде перший, тоді результат записуємо на 2-ге місце, шукаємо, в який елемент перейде цей 2-й елемент, записуємо на 3-тє місце, ..., продовжуємо цей процес, поки результат останньої дії не буде дорівнювати першому елементу, – це 1-й цикл. Аналогічно – інші цикли.

Цикли довжини 2 називаються транспозиціями.

Основною їх властивістю є те, що вони співпадають з оберненими до себе.

Всі підстановки можна представити у вигляді добутку транспозицій.

Наприклад, (12345) = (12)(13)(14)(15).

Або по-іншому: (12345) = (23)(24)(25)(21).

Можна дописати в розклад пару однакових транспозицій, оскільки ця пара є тотожною підстановкою (обернена до транспозиції дорівнює їй самій):

(12345) = (12)(13)(14)(15)(23)(23).

Можна в підстановку вписати елемент, який не входить в підстановку, і, переставивши елементи, виключити його:

(12345) = (01)(02)(03)(04)(05)(01).

Отже, будь-яку підстановку можна розкласти в добуток транспозицій багатьма способами, і їх в розкладі буде завжди парна або непарна кількість.

Означення 1. Якщо число транспозицій в будь-якому розкладі підстановок завжди парне, то підстановка називається парною. В противному випадку – непарною.

Означення 2. Підстановка називається парною, якщо сума інверсій її верхнього і нижнього рядків є число парне. В противному випадку – непарною.

Нехай $k_1,...,k_i,...,k_j,...,k_n$ – деяка перестановка.

Числа $k_i$ та $k_j$ утворюють інверсію, якщо $i<j$ і $k_i > k_j$.

Теорема. Будь-яку підстановку можна представити у вигляді добутку транспозицій сусідніх елементів.

Наслідок. Будь-яке сортування може бути виконано перестановкою сусідніх елементів.

Метод сортування, в основі якого лежить ця теорема, відомий як „метод бульбашки” (або „бульбашковий метод”).

При переміщенні елементів на своє місце транспозиціями сусідніх елементів всі елементи залишаються на своїх місцях, крім елемента, що переміщується, і того елемента, що стоїть на цільовому місці (тобто на місці, куди переміщуємо наш елемент), ці два елементи міняються місцями.

Даний метод може бути виражений у вигляді такого алгоритму:

**Алгоритм 1. Сортування елементів „бульбашковим методом”**

*Вхід*: масив *A*: array [1..*n*] of *B*, де значення елементів масиву розміщені в довільному порядку і для значень типу *B* задано відношення <.

*Вихід*: масив *A*: array [1..*n*] of *B*, де значення елементів масиву розміщені в порядку зростання.

```
For i from 1 to n-1 do
      m:=I {індекс кандидата в мінімальні елементи}
       for j from i+1 to n do
        if A[j]<A[m] then
          m:=j {новий кандидат в мінімальні елементи}
        end if
       end for
   A[j]↔A[m] {ставимо мінімальний елемент на місце}
End for
```

Щоб поставити мінімальний елемент на місце, використовуємо введення додаткової змінної.

## II метод сортування – заміною сусідніх елементів

Даний алгоритм містить максимально *n*-1 крок. Якщо два сусідні елементи стоять „неправильно” (спочатку більший, а потім менший), то міняємо їх місцями.

Алгоритм заключається в перегляді вихідної послідовності справа наліво, і при кожному кроці менший з двох сусідніх елементів переміщується до лівої позиції (лівіше).

В результаті першого перегляду найменший елемент буде знаходитися в крайній лівій позиції.

Після цього повторюємо описаний вище процес, розглядаючи в якості вихідної послідовності масив, починаючи з другої позиції і т.д.

## Алгоритм 2. Сортування методом транспозиції

```
var i,j: index, x:item;
begin
  for i:=2 to n do
```

```
    for j:=n downto i do
      if a[j-1]>a[j] then
        { x:=a[j-1];
          a[j-1]:=a[j];
          a[j]:=x
        }
end;
```

Розглянемо першу комбінаторну конфігурацію, яка ще називається задачею розкладання в ящики.

Дано $n$ різних предметів і $k$ ящиків. Треба розкласти в перший ящик $n_1$ предметів, у 2-й – $n_2$ предметів,.., $k$-й – $n_k$ предметів, де $n_1+n_2+..+n_k=n$, $n_1,n_2,..,n_k$ – фіксовані числа. Скількома способами можливо це зробити?

Можна розкласти так. Серед $n$ предметів візьмемо довільну $n_1$-підмножину і покладемо її в перший ящик (це можна зробити $C_n^{n_1}$ способами).

Серед $n-n_1$ предметів, що залишились, візьмемо $n_2$-підмножину й покладемо її в 2-й ящик (це можливо зробити $C_{n-n_1}^{n_2}$ способами) і продовжимо цей процес.

За правилом добутку загальна кількість розкладань дорівнює:

$$C_n^{n_1}\cdot C_{n-n_1}^{n_2}\cdot..\cdot C_{n-n_1-n_2-..-n_{k-1}}^{n_k}=\frac{n!}{n_1!(n-n_1)!}.\frac{(n-n_1)!}{n_2!(n-n_1-n_2)!}.....\frac{(n-n_1-..-n_{k-1})!}{n_k!(n-n_1-..-n_k)!}=$$

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

Отже, розкладань у ящики стільки, скільки перестановок з повтореннями. Розглянемо зв'язок між цими двома задачами. Для цього занумеруємо $n$ місць, які можуть займати предмети.

Кожній перестановці відповідає розподіл номерів місць на $k$ класів: в $i$-й клас потрапляють номери тих місць, на які покладено предмети $i$-го типу.

Отже, знайдено відповідність між перестановками з повтореннями та розкладанням номерів місць у ящики.

Тому формули розв'язань обох задач збігаються.

Біном Ньютона.

Числа $C_n^r = \frac{n!}{r!(n-r)!}$ називається біноміальними коефіцієнтами, тобто це кількість сполучень з $n$ елементів по $r$.

Зміст цієї назви випливає з теореми, яка відома як формула бінома Ньютона: $(x+y)^n = \sum_{k=0}^{n} C(n,k) x^k y^{n-k}$.

Розглянемо властивості біномних коефіцієнтів.

1. Нехай $n$ і $r$ – невід'ємні цілі числа, $r \le n$.

Тоді $C_n^r = C_n^{n-r}$.

Дійсно, $C_n^{n-r} = \frac{n!}{(n-r)!\left[n-(n-r)\right]!} = \frac{n!}{(n-r)!r!} = \frac{n!}{r!(n-r)!} = C_n^r$.

2. Рівність Паскаля: $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$.

Позначимо через $S_{n,k}$ множину всіх сполучень з елементів $\{a_1,..,a_{n-1},a_n\}$ по $k$ елементів.

$S_{n-1,k}$ – з елементів $\{a_1,..,a_{n-1}\}$ по $k$.

$S_{n-1,k-1}$ – з елементів $\{a_1,..,a_{n-1}\}$ по $k$-1.

Кожному сполученню з $S_{n,k}$, яке містить елемент $a_n$, відповідає сполучення з $S_{n-1,k-1}$.

Якщо сполучення з $S_{n,k}$ не містить $a_n$, то йому відповідає сполучнення з $S_{n-1,k}$.

Отже, існує бієкція між множинами $S_{n,k}$ і $S_{n-1,k} \cup S_{n-1,k-1}$.

Оскільки очевидно, що $S_{n-1,k} \cap S_{n-1,k-1} \neq \varnothing$, то $\left|S_{n,k}\right| = \left|S_{n-1,k}\right| + \left|S_{n-1,k-1}\right|$, тобто $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$.

Отримане співвідношення дає змогу побудувати таблицю для чисел $C_n^k$, яке називають трикутником Паскаля.

Записують так:

1 1 $n$ =1

1 2 1 $n$ =2

1 3 3 1 $n$ =3

1 4 6 4 1 $n$ =4

……

В цьому рівнобедреному трикутнику кожне число (крім одиниць на бокових сторонах) являється сумою двох чисел, які стоять над ним.

Число сполучень $C_n^r$ знаходиться в *(n+1)*-му рядку на *(r+1)*-му місті.

Або Трикутник Паскаля записують так:

| $n$ \ $k$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | .. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | | | | | | | | | | .. |
| 1 | 1 | 1 | | | | | | | | | .. |
| 2 | 1 | 2 | 1 | | | | | | | | .. |
| 3 | 1 | 3 | 3 | 1 | | | | | | | .. |
| 4 | 1 | 4 | 6 | 4 | 1 | | | | | | .. |
| 5 | 1 | 5 | 10 | 10 | 5 | 1 | | | | | .. |
| 6 | 1 | 6 | 15 | 20 | 15 | 6 | 1 | | | | .. |
| 7 | 1 | 7 | 21 | 35 | 35 | 21 | 7 | 1 | | | .. |
| 8 | 1 | 8 | 28 | 56 | 70 | 56 | 28 | 8 | 1 | | .. |
| 9 | 1 | 9 | 36 | 84 | 126 | 126 | 84 | 36 | 9 | 1 | .. |
| .. | .. | .. | .. | .. | .. | .. | .. | .. | .. | .. | .. |

3. Послідовність ($p_n$) дійсних чисел називається унімодальною, якщо є такий унімодальний номер $m$, що $p_0 < p_1 < ... < p_m$, $p_{m+1} > p_{m+2} > ... > p_n$, тобто:

- послідовність строго зростає на відрізку $[0, m]$, $m > 0$,

- послідовність строго спадає на відрізку $[m+1, n]$, $m+1 < n$,

- максимальне значення досягається не більше, ніж у двох точках: $m$ і, можливо, $m+1$.

Через $\lfloor x \rfloor$ позначимо найбільше ціле число, яке менше або дорівнює $x$ (цілу частину числа $x$).

Наприклад, $\lfloor 3{,}14 \rfloor = 3$, $\lfloor -3{,}14 \rfloor = -4$.

Теорема. При фіксованому $n$ послідовність біноміальних коефіцієнтів $(C_n^k)$, $k$=0,1,..,$n$, унімодальна, $m = \left\lfloor \frac{n}{2} \right\rfloor$.

Якщо $n$ – парне, то максимум досягається в точці $m = \left\lfloor \frac{n}{2} \right\rfloor = \frac{n}{2}$, якщо непарне, то – у двох точках $m = \left\lfloor \frac{n}{2} \right\rfloor = \frac{n-1}{2}$ і $m+1 = \frac{n+1}{2}$.

**4**. Рівність Вандермонда.

Нехай $m,n,r$ – невід'ємні цілі числа, причому $r \le \min\{m,n\}$. Тоді $C_{n+m}^r = \sum_{k=0}^{r} C_m^{r-k} C_n^k$.

Теорема біноміальна.

Нехай $x$ та $y$ – змінні, $n$ – додатне ціле число.

Тоді $(x+y)^n = \sum_{j=0}^{n} C_n^j x^j y^{n-j} = \sum_{j=0}^{n} C_n^j x^{n-j} y^j$.

Цю рівність легко отримати.

Оскільки $x^j y^{n-j}$ одержано внаслідок $j$-кратного вибору $x$ і $(n-j)$- кратного вибору $y$ з $n$ співмножників у виразі $(x+y)^n$, то коефіцієнт при $x^j y^{n-j}$ дорівнює кількості способів $j$-кратного вибору $x$ з $n$ співмножників, тобто $C_n^j$.

Друга рівність випливає з того, що $C_n^j = C_n^{n-j}$.

Легко перевірити, що $(x-y)^n = \sum_{j=0}^{n} (-1)^j C_n^j x^{n-j} y^j$.

Приклад 1.

Знайти розклад виразу $(x+y)^4$.

За біноміальною теоремою одержимо:

$$(x+y)^4 = C_4^0 x^4 + C_4^1 x^3 y + C_4^2 x^2 y^2 + C_4^3 x y^3 + C_4^4 y^4 = x^4 + 4x^3 y + 6x^2 y^2 + 4xy^3 + y^4.$$

Біноміальні коефіцієнти можна брати з трикутника Паскаля або обчислювати за формулою (3) (за формулою для кількості сполучень).

Приклад 2.

Визначити коефіцієнт при $x^{12} y^{13}$ в розкладі $(x+y)^{25}$.

Очевидно, цей коефіцієнт визначається за формулою

$$C_{25}^{13} = \frac{25!}{13!12!} = 5200300.$$

За допомогою біноміальної теореми можна одержати ще дві властивості біноміальних коефіцієнтів:

5. $\sum_{k=0}^{n} C_n^k = 2^n$.

Дійсно, $2^n = (1+1)^n = \sum_{k=0}^{n} C_n^k 1^{n-k} 1^k = \sum_{k=0}^{n} C_n^k$.

Або по-іншому можна показати цю рівність. Оскільки $C_n^k$ – число різних $k$-елементних підмножин $n$-елементної множини, то сума всіх таких чисел становить число всіх підмножин даної $n$-елементної множини.

6. $\sum_{k=0}^{n} (-1)^k C_n^k = 0$.

Аналогічно $0 = [1+(-1)]^n = \sum_{k=0}^{n} C_n^k 1^{n-k} (-1)^k = \sum_{k=0}^{n} (-1)^k C_n^k$.

Поліноміальна теорема.

Як узагальнення бінома розглянемо вираз

$$(x_1 + x_2 + ... + x_k)^n.$$

Теорема.

Вираз $(x_1 + x_2 + ... + x_k)^n$ дорівнює сумі всіх можливих доданків $P_n(n_1, n_2, .., n_k)x_1^{n_1}x_2^{n_2}...x_k^{n_k}$, де $n_1 + n_2 + ... + n_k = n$, тобто

$$(x_1 + x_2 + ... + x_k)^n = \sum_{\substack{n_1 \geq 0, ..., n_k \geq 0 \\ n_1 + ... + n_k = n}} P_n(n_1, .., n_k)x_1^{n_1}x_2^{n_2}...x_k^{n_k}.$$

Покажемо це. Запишемо $(x_1 + x_2 + ... + x_k)^n$ у вигляді добутку $n$ співмножників і розкриємо дужки.

Коефіцієнт при $x_1^{n_1}x_2^{n_2}...x_k^{n_k}$ дорівнює кількості перестановок із повтореннями, таких, що елемент $x_1$ міститься в кожній з них $n_1$ разів, $x_2$ – $n_2$ разів,..., $x_k$ входить $n_k$ разів, а всього елементів $n_1 + n_2 + ... + n_k = n$. Очевидно, що цей коефіцієнт дорівнює $P_n(n_1, .., n_k)$.

Отриману формулу називають поліноміальною.

Вона дає змогу доводити деякі властивості чисел $P_n(n_1, .., n_k)$. Зазначимо дві з них.

**1**. Нехай $x_1 = x_2 = ... = x_k = 1$, тоді $\sum_{\substack{n_1 \geq 0, ..., n_k \geq 0 \\ n_1 + ... + n_k = n}} P_n(n_1, n_2 .., n_k) = k^n$.

**2**. Помножимо обидві частини поліноміальної формули для $n-1$ на $(x_1 + x_2 + ... + x_k)$ та порівняємо коефіцієнти при однакових доданках. Одержимо таке співвідношення:

$$P_n(n_1, n_2, .., n_k) = P_{n-1}(n-1, n_2, .., n_k) + P_{n-1}(n_1, n_2 - 1, .., n_k) + ... + P_{n-1}(n_1, n_2, .., n_k - 1).$$

Задача про цілочислові розв'язки.

Формулюється так:

Знайти кількість розв’язків рівняння $x_1 + x_2 + ... + x_r = n$ у цілих невід'ємних числах, де $n$ – ціле , невід'ємне число.

Взявши такі невід'ємні числа $x_1, x_2, ..., x_r$, що $x_1 + x_2 + ... + x_r = n$, можна одержати сполучення з повтореннями з $r$ елементів по $n$, а саме: елементів першого типу – $x_1$ одиниць, другого – $x_2$,..., $r$-го – $x_r$.

Навпаки, якщо є сполучення з повтореннями з $r$ елементів по $n$, то кількості елементів кожного типу задовольняють рівнянню $x_1 + x_2 + ... + x_r = n$ у цілих невід'ємних числах.

Отже, кількість цілих невід'ємних розв'язків цього рівняння $H_r^n = C_{r+n-1}^n = \frac{(r+n-1)!}{n!(r-1)!}$.

Приклад 1

Знайти кількість невід'ємних цілих розв'язків рівняння $x_1 + x_2 + x_3 = 11$.

За формулою: $H_3^{11} = C_{3+11-1}^{11} = C_{13}^{11} = \frac{13!}{11!2!} = \frac{12 \cdot 13}{2} = 78$.

Кількість розв'язків рівняння $x_1 + x_2 + ... + x_r = n$ у цілих невід'ємних числах можна визначити і тоді, коли на ці змінні накладено певні обмеження.

Приклад 2. Знайти кількість невід'ємних цілих розв'язків рівняння $x_1 + x_2 + x_3 = 11$ за умов $x_1 \geq 1$, $x_2 \geq 2$, $x_3 \geq 3$.

Очевидно, що ця задача еквівалентна рівнянню $x_1 + x_2 + x_3 = 5$ без обмежень.

Дійсно, треба взяти щонайменше один елемент першого типу, два елементи другого типу, три елементи третього – разом 1+2+3=6 елементів.

Отже, 11-5=6 елементів залишається для довільного вибору.

$H_3^5 = C_{4+11-1}^{11} = C_{14}^{11} = \frac{14!}{11!3!} = 364$.

Властивості біноміальних коефіцієнтів.

1. Формула симетрії $C_{n+m}^n = C_{n+m}^m$.

2. Формула додавання $C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$.

3. Формула суми всіх біноміальних коефіцієнтів

$C_0^0 - C_n^1 + C_n^2 - ... + (-1)^n C_n^n = 0$.

4. Формула винесення за дужки $C_n^k = \frac{n}{k} \cdot C_{n-1}^{k-1}$.

5. $C_n^0 + C_{n-1}^1 + ... + C_{n+k}^n = C_{n+k+1}^n$.

6. $C_{n-1}^{r-1} + C_{n-2}^{r-1} + ... + C_{r-1}^{r-1} = C_n^r$.

7. $C_n^0 + 2C_n^1 + ... + nC_n^n = n2^{n-1}$.

8. $C_n^0 C_m^k + C_n^1 C_m^{k-1} + ... + C_n^k C_m^0 = C_{n+m}^k$.

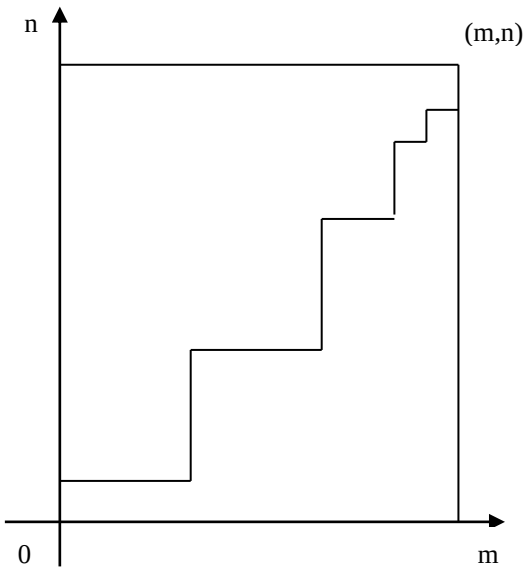
9. $C_n^m C_k^0 + C_{n-1}^{m-1} C_{k+1}^1 + ... + C_{n-m}^0 C_{k+m}^m = C_{n+k+1}^m$.

10. $(C_n^0)^2 + (C_n^1)^2 + ... + (C_n^n)^2 = C_{2n}^n$.

Це не всі властивості біноміальних коефіцієнтів. Є велика їх кількість і виявляються все нові і нові.

Геометрична інтерпретація біноміальних коефіцієнтів.

Нехай маємо прямокутну шахову дошку *m* на *n*, яка розміщена на координатній площині.




Ця дошка складається з *m-n* елементарних квадратів, розділених *n-1* горизонтальною і *m-1* вертикальною лініями.

Визначимо, скількома різними найкоротшими шляхами можна потрапити з точки (*0,0*) в точку (*m,n*) на цій дошці.

Кожен найкоротший шлях, який веде з точки (*0,0*) в точку (*m,n*), складається з *m+n* сторін елементарних квадратів, серед яких *m* горизонтальних і *n* вертикальних.

Ці шляхи розрізняються лише числом вертикальних і горизонтальних сторін.

Отже, загальна кількість шляхів дорівнює числу способів, якими з *m+n* сторін можна вибрати *n* вертикальних, тобто це число рівно $C_{m+n}^{n}$.

Можна було підрахувати не за вертикальними сторонами, а за горизонтальними.

Тобто існує $C_{m+n}^{m}$ різних найкоротших шляхів, і отже, справджується рівність $C_{m+n}^{n} = C_{m+n}^{m}$ – формула симетрії.

Приклади.

**1**. Збірна команда університету з волейболу налічує 15 чоловік. Скільки різних варіантів повинен розглянути тренер перед грою, щоб заявити список гравців на гру?

Розв’язок.

Кількість гравців волейбольної команди 6. Отже, число всіх можливих варіантів – це число різних підмножин, які складаються з шести елементів у множині з 15-ти елементів, тобто

$$C_{15}^{6} = \frac{15!}{6!(15-6)!} = \frac{15 \cdot 14 \cdot 13 \cdot 12 \cdot 11 \cdot 10}{6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = 5 \cdot 7 \cdot 13 \cdot 11 = 5005.$$

**2**. У скількох точках перетинаються діагоналі випуклого десятикутника, якщо жодні три з них не перетинаються в одній точці?

Розв’язок.

Кожній точці перетину двох різних діагоналей відповідають чотири вершини десятикутника, а кожним чотирьом вершинам – одна точка перетину.

Таким чином, число всіх точок перетину дорівнює числу способів, якими з 10-ти вершин можна вибрати чотири вершини, тобто

$$C_{10}^{4} = \frac{10!}{4!6!} = \frac{10 \cdot 9 \cdot 8 \cdot 7}{4 \cdot 3 \cdot 2} = 210.$$

**3**. В деяких видах спортивних змагань результатом являється визначення учасників, які зайняли 1-ше, 2-ге і 3-тє місця.

Скільки існує різних результатів, якщо в змаганні бере участь $n$ учасників?

Розв’язок.

Кожен можливий результат відповідає функції $F:\{1,2,3\} \to \{1...n\}$ (аргумент – номер призового місця, результат – номер учасника).

Таким чином, всього можливо $A_n^3 = n(n-1)(n-2)$ різних результатів.

**4**.На початку гри в доміно кожному учаснику гри видається 7 костів з 28 різних, що є в наявності.

Скільки існує різних комбінацій костів, які учасник може отримати на початку гри?

Розв’язок.

Потрібна кількість дорівнює кількості 7-елементних підмножин 28-елементної множини :

$$C_{28}^{7} = \frac{28!}{7!(28-7)!} = \frac{28 \cdot 27 \cdot 26 \cdot 25 \cdot 24 \cdot 23 \cdot 22}{7 \cdot 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = 1184040.$$

**5**.Скількома способами можна розсадити *n* нових гостей серед *m* гостей, які вже сидять за круглим столом?

Розв’язок.

Між *m* гостями, що вже сидять існує *m* проміжків, в які можна розсадити нових. Тобто, це можна зробити

$$H_m^n = C_{m+n-1}^n = \frac{(m+n-1)!}{n!(m-1)!}$$ способами.

## Генерація перестановок.

Проблемі систематичної побудови всіх *n!* перестановок *n*-елементної множини присвячено багато публікацій.

Ця проблема має давню історію.

Її появу можна віднести до початку XVII ст., коли в Англії виникло особливе мистецтво дзвонарства. Воно полягало у вибивані на *n* різних дзвонах усіх *n!* перестановок. Це треба було робити по пам'яті.

Тому шанувальники цього мистецтва розробили перші прості методи систематичної побудови всіх перестановок без повторень.

Деякі з цих методів було знову відкрито в наш час у зв'язку з появою комп'ютерів.

“Книга рекордів Гіннеса” містить інформацію про вибивання всіх 8!=40320 перестановок на восьми дзвонах у 1963р.

Для цього було потрібно 17 год. 58 хв. 30 с.

З використанням комп'ютерів генерування перестановок здійснюється дуже швидко.

Кожній *n*-елементній множині *A* можна поставити у взаємно-однозначну відповідність множину $A' = \{1,2,..,n\}$.

Зручно спочатку генерувати перестановки *n* перших натуральних чисел, а потім замінити кожне число відповідним елементом множини *A*.

Внаслідок цього отримаємо всі перестановки елементів деякої множини *A*.

**Генерування перестановок в лексикографічному порядку.**

Існують різні алгоритми побудови всіх перестановок множини $A' = \{1,2,..,n\}$.

Розглянемо один з них.

Цей алгоритм ґрунтується на послідовній побудові перестановок множини $A'$ у лексикографічному порядку.

Перестановку $(a_1, a_2, .., a_n)$ будемо позначати $a_1 a_2 .. a_n$.

На множині всіх перестановок (в загальному – на множині всіх кортежів довжиною *n* з елементами з множини $A' = \{1,2,...,n\}$) визначимо лексикографічний порядок: $a_1 a_2 ... a_h < b_1 b_2 ... b_h$,

якщо для деякого $k$, $1 \le k \le n$, виконуються співвідношення $a_1 = b_1$, $a_2 = b_2, \ldots, a_{k-1} = b_{k-1}$ але $a_k < b_k$.

В цьому випадку кажуть, що перестановка $a_1 a_2 ... a_n$ менше від перестановки $b_1 b_2 ... b_n$ або перестановка $b_1 b_2 ... b_n$ більше від перестановки $a_1 a_2 ... a_n$.

Якщо замість чисел 1,2,…,*n* взяти букви *a,b,…,z* з природним порядком $a < b < ... < z$, то лексикографічний порядок – стандартна послідовність, у якій слова довжиною *n* наведено в словнику.

Перестановку $b_1 b_2 ... b_n$ називають лексикографічно наступною за $a_1 a_2 ... a_n$, якщо не існує такої перестановки $c_1 c_2 ... c_n$, що $a_1 a_2 ... a_n < c_1 c_2 ... c_n$ і $c_1 c_2 ... c_n < b_1 b_2 ... b_n$.

Приклад 1.

Перестановка 23415 множини $\{1,2,3,4,5\}$ менша від перестановки 23514.

Алгоритм генерування перестановок множини $A' = \{1,2,...,n\}$ ґрунтується на процедурі, що будує перестановку, лексикографічно наступну за даною перестановкою $a_1 a_2 .. a_n$.

Покажемо, як це можна зробити.

Спочатку припустимо, що $a_{n-1} < a_n$.

Поміняємо місцями $a_{n-1}$ і $a_n$.

Одержимо більшу перестановку.

Вона лексикографічно наступна, бо жодна інша підстановка не більша за дану перестановку й не менше за отриману.

Приклад 2.

Нехай 234156 – задана перестановка, тоді перестановка 234165 лексикографічно наступна.

Розглянемо випадок, коли $a_{n-1} > a_n$. Переглянемо останні три члени перестановки.

Якщо $a_{n-2} < a_{n-1}$, то останні три члени можна переставити для отримання наступної перестановки.

Поставимо менше з двох чисел $a_{n-1}$ і $a_n$, яке, однак, більше, ніж $a_{n-2}$, на позицію *n*-2.

Потім розмістимо число, яке залишилося, і $a_{n-2}$ на останніх двох позиціях у висхідному порядку.

Приклад.3.

Нехай 234165 – задана перестановка і тоді перестановка 234516 лексикографічно наступна.

Отже, маємо такий алгоритм.

Алгоритм побудови лексикографічно наступної перестановки за перестановкою $a_1 a_2 .. a_n$.

Кроки алгоритму:

Крок 1. Знайти такі числа $a_j$ і $a_{j+1}$, що $(a_j < a_{j+1}) \wedge (a_{j+1} > a_{j+2} > ... > a_n)$.

Для цього треба знайти в перестановці першу справа пару сусідніх чисел, у якій число, що ліворуч, менше від числа, що 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 $j+1, ..., n$.

Обґрунтування.

Покажемо, що не існує перестановки, яка водночас більша від $a_1 a_2 .. a_n$, але менша від побудованої за цим алгоритмом.

Це означає, що побудована перестановка дійсно лексикографічно наступна за даною перестановкою $a_1 a_2 .. a_n$.

Справді, за наведеним алгоритмом нова перестановка збігається зі старою в позиціях 1, ..., *j*-1.

У *j*-й позиції нова перестановка містить $a_k$, а стара – $a_j$, причому $a_k > a_j$.

Отже, нова перестановка лексикографічно більша від старої.

Крім того, вона перша в лексикографічному порядку з $a_1, a_2, .., a_{j-1}, a_k$ у позиціях з 1 до *j*.

Стара перестановка остання з $a_1, a_2, .., a_{j-1}, a_j$ у цих самих позиціях.

Згідно з алгоритмом $a_k$ вибирають найменшим з $a_{j+1}, a_{j+2}, ..., a_n$, але більшим, ніж $a_j$.

Отже, не існує жодної перестановки між старою та новою.

Приклад. 4.

Побудуємо наступну в лексикографічному порядку за перестановкою 362541.

Остання пара чисел, у якій перше число менше за друге, – 25.

Отже, розглянемо послідовність чисел 541.

Серед них найменше число, більше від 2, це – 4.

Тепер 4 запишемо на місце 2, а решту чисел 251 розмістимо на останніх трьох позиціях у висхідному порядку: 364125.

Щоб побудувати всі $n!$ перестановок множини $A' = \{1,2,...,n\}$, починаємо з лексикографічної найменшої перестановки $123\ldots n$ і послідовно $n!$-1 разів виконуємо алгоритм побудови лексикографічно наступної перестановки.

$n!=4!=1\cdot2\cdot3\cdot4=24 \rightarrow$ 24 перестановки.

| | | | | | |
|---|---|---|---|---|---|
| 1) | 1234 | 12) | 2431 | 23) | 4312 |
| 2) | 1243 | 13) | 3124 | 24) | 4321 |
| 3) | 1324 | 14) | 3142 | | |
| 4) | 1342 | 15) | 3214 | | |
| 5) | 1423 | 16) | 3241 | | |
| 6) | 1432 | 17) | 3412 | | |
| 7) | 2134 | 18) | 3421 | | |
| 8) | 2143 | 19) | 4123 | | |
| 9) | 2314 | 20) | 4132 | | |
| 10) | 2341 | 21) | 4213 | | |
| 11) | 2413 | 22) | 4231 | | |

**Генерування перестановок в антилексикографічному порядку.**

Інакше кажучи, на множині перестановок природнім чином можна визначити впорядкованість на основі впорядкованості елементів.

Тобто:

Кажуть, що перестановка $< a_1,...,a_n >$ лексикографічно передує перестановці $< b_1,...,b_n >$, якщо $\exists k \le n \quad (a_k < b_k) \wedge (\forall i < k \ ) \quad a_i = b_i$.

Кажуть, що перестановка $< a_1,...,a_n >$ антилексикографічно передує перестановці $< b_1,...,b_n >$, якщо $\exists k \le n \quad (a_k > b_k) \wedge (\forall i > k) \quad a_i = b_i$.

Можна визначити так: кажуть, що перестановка $a_1a_2...a_n$ антилексикографічно передує перестановці $в_1в_2...в_n$ , якщо для деякого $k$ виконуються співвідношення:

$a_k > b_k$ , $a_{k+1} = b_{k+1}$, ..., $a_n = b_n$.

Наприклад,

(1,2,3) антилексикографічно передує (2,1,3);

(2,1,3) антилексикографічно передує (1,3,2);

(1,3,2) антилексикографічно передує (3,1,2);

(3,1,2) антилексикографічно передує (2,3,1);

(2,3,1) антилексикографічно передує (3,2,1).

Розглянемо алгоритм, який генерує всі перестановки елементів $1,...,n$ в антилексикографічному порядку.

Масив *P*: *array* [1..*n*] *of* [1..*n*] є глобальним і призначений для зберігання перестановок.

**Алгоритм 2. Генерація перестановок в антилексикографічному порядку**

*Вхід*: *n* – кількість елементів,

*Вихід*: послідовність перестановок елементів $1,\ldots,n$ в анти-лексикографічному порядку.

```
for i from 1 to n  do
P[i]:=1{ініціалізація}
end for
Antilex (n) {виклик рекурсивної процедури Antilex}
```

Основна робота по генерації перестановок виконується рекурсивною процедурою *Antilex*.

*Вхід*: $m$ – параметр процедури – кількість перших елементів масиву $P$, для яких генеруються перестановки.

*Вихід*: послідовність перестановок $1,\ldots,m$ в антилексикографічному порядку.

```
if m=1 then
  yield P {чергова перестановка}
else
  for i from 1 to m do
    Antilex (m-1) {рекурсивний виклик}
    if  i<m then
       P[i] ↔ P[m] {наступний елемент}
      Reverse (m-1) {зміна порядку елементів}
    end if
  end for
end if
```

Допоміжна процедура *Reverse* переставляє елементи заданого відрізку масиву $P$ в оберненому порядку.

*Вхід*: $k$ – номер елемента, що задає відрізок масиву $P$, який має бути переставлений в оберненому порядку.

*Вихід*: перші $k$ елементів масиву $P$, переставлені в оберненому порядку $j := 1$ {нижня границя діапазону обернення}.

*while* $j<k$ *do*

$P[j] \leftrightarrow P[k]$

$j := j+1$

$k := k-1$

*end while*

Обґрунтування:

Шукану послідовність перестановок $n$ елементів можна отримати з послідовності перестановок $n$-1 елемента таким чином.

Треба виписати $n$ блоків по ($n$-1)! перестановок в кожному, що відповідають послідовності перестановок ($n$-1) елемента в анти-лексикографічному порядку. Потім до всіх перестановок в першому блоці треба приписати справа $n$, у другому ($n$-1) і аналогічно в інших у спадному порядку.

Тоді в кожному з блоків (крім першого), до перестановок якого справа приписаний елемент $i$, потрібно в перестановках блоку замінити всі входження елемента $i$ на елемент $n$.

В одержаній послідовності всі перестановки різні, і їх $n$($n$-1)!=$n$!, тобто перераховані всі перестановки.

При цьому антилексикографічний порядок збережений для послідовностей всередині одного блоку, бо цей порядок був збережений у вихідній послідовності, а для послідовностей на границях двох блоків, тому що відбувається зменшення крайнього правого елемента.

Звернемося до процедури *Antilex* – легко бачити, що в ній реалізована вказана побудова.

В основному циклі спочатку будується черговий блок – послідовність перестановок перших $m$-1 елементів масиву $P$ (при цьому елементи $P[m],..,P[n]$ залишаються незмінними).

Потім елемент $P[m]$ міняється місцями з черговим елементом $P[i]$.

Виклик допоміжної процедури *Reverse* необхідний, оскільки остання перестановка в блоці являється оберненням першої, а для генерації наступного блоку на черговому кроці циклу потрібно відновити вихідний порядок.

Приклад.

Послідовність перестановок в антилексикографічному порядку для *n=3: (1,2,3), (2,1,3), (1,3,2), (3,1,2), (2,3,1), (3,2,1).*

**Генерування підмножин.**

Елементи множини $\{1,..,m\}$ впорядковані. Тому кожна $n$-елементна підмножина теж може бути розглянута як впорядкована послідовність.

На множині таких послідовностей природнім чином визначається лексикографічний порядок.

Розглянемо алгоритм, що генерує всі $n$-елементні підмножини $m$-елементної множини в лексикографічному порядку.

Алгоритм 3. Генерація $n$-елементних підмножин $m$-елементної множини.

*Вхід*: $n$ – потужність підмножини, $m$ – потужність множини, $m \ge n > 0$.

*Вихід*: послідовність всіх $n$-елементних підмножин $m$-елементної множини в лексикографічному порядку.

```
for I  from 1 to m do
   A[i] := i  {ініціалізація вихідної множини}
end for
if  m=n  then
```

*yield* $A[1..n]$ {єдина підмножина}

*exit*

*end if*

*p:=n* {*p* – номер першого елемента, що змінюється}

*while* $p \geq 1$ *do*

*yield* $A[1..n]$ {чергова підмножина в перших *n* елементах масиву *A*}

*if* $A[n] = m$ *then*

*p:=p* -1 {неможливо збільшити останній елемент}

*else*

*p:=n* {можна збільшити останній елемент}

*end if*

*if* $p \geq 1$ *then*

*for i from n dowhto p do*

$A[i] := A[p] + i - p + 1$ {збільшення елементів}

*end for*

*end if*

*end while.*

Обґрунтування.

У вихідній послідовності *n*-елементних підмножин (кожна з яких являється зростаючою послідовністю *n* чисел із діапазона 1..*m*) за послідовністю $< a_1,..,a_n >$ слідує послідовність

$$< b_1,..,b_n >=< a_1,..,a_{p-1}, a_p + 1, a_p + 2,.., a_p + n - p + 1 >,$$

де *p* – максимальний індекс, для якого $b_n = a_p + n - p + 1 \leq m$.

Іншими словами, наступна послідовність одержується із попередньої заміною деякої кількості елементів у “хвості” послідовності на цілі числа, що слідують підряд один за одним, але так, щоб останній елемент не перевищував *m*, а перший елемент, що змінюється, був на 1 більше, ніж відповідний елемент в попередній послідовності.

Таким чином, індекс $p$, починаючи з якого потрібно змінити “хвіст послідовності”, визначається за значенням елемента $A[n]$.

Якщо $A[n] < m$, то потрібно змінити лише $A[n]$, і при цьому $p:=n$.

Якщо вже $A[n] = m$, то потрібно зменшувати індекс $p:=p$ -1, збільшуючи довжину “хвоста”, що змінюється.

Приклад.

Послідовність $n$-елементних підмножин $m$-елементної множини в лексикографічному порядку для $n$=3 і $m$=4:

(1,2,3), (1,2,4), (1,3,4), (2,3,4).

## Генерування сполучень

Знову будемо розглядати, множину A={1,2,…,$n$}.

Генерування сполучень аналогічно генеруванню підмножин.

Розглянемо алгоритм побудови лексикографічно наступного сполучення.

Алгоритм побудови лексикографічно наступного сполучення.

Кроки алгоритму:

Крок 1.Знайти в рядку перший справа елемент $a_i$ такий, що $a_i \neq n\text{-}r\text{+}i$.

Крок 2.Для знайденого елемента виконати присвоєння $a_i := a_i + 1$.

Крок 3.Для $j=i+1, i+2,..., r$ виконати: $a_j:=a_i+j-i$ (або, що те саме, $a_j=a_{j-1}+1$).

Приклад.

Нехай A’ = {1,2,3,4,5,6}. Знайти сполучення, наступне за {1,2,5,6} в лексикографічному порядку.

Подаємо це сполучення рядком 1256, маємо $n$=6, $r$=4.

Перший справа з таких елементів, що $a_i \neq 6\text{-}4\text{+}i$ , це $a_2$=2.

Для обчислення наступного більшого сполучення збільшуємо $a_2$ на 1 і 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}.

Лексикографічний порядок – такий, як і для перестановок:

$b_1 b_2 ... b_n$ наступне за $a_1 a_2 ... a_n$, якщо $\exists k \le n \quad b_k > a_k, \forall i < k \quad a_i = b_i$.

Це визначення можливе, оскільки сполучення – це підмножина і порядок запису елементів множини є неістотний, тому будемо записувати елементи в кожному сполученні у висхідному порядку, наприклад

{3,5,1} будемо записувати {1,3,5}.

Отже, сполучення $\{a_1, a_2, ..., a_r\}$ розглядатимемо як рядок чисел $a_1 a_2 ... a_r$, причому $a_1 < a_2 < ... a_r$.

Розглянемо лексикографічний порядок.

Нехай $n$=5, $r$=3

Якщо можна збільшити останню цифру, то так і будемо робити.

Тому маючи рядок 123, його можна замінити на 124.

Якщо ж маємо 125, останнє число збільшити не можна.

Тому переходимо до наступного (справа) числа і дивимось, чи можна його збільшити.

У даному випадку це можна зробити: потрібно замінити 2 на 3.

Проте ми прагнемо побудувати найменший рядок із тих, які більші від 125.

Тому збільшуємо останнє число (тобто 3) на 1 і записуємо результат у наступну позицію.

Отже, перші два числа – 1 і 3, тому наступний рядок – 134.

Припустимо, що є рядок 145. Останнє і передостаннє числа збільшити неможна.

Проте перше число можна збільшити, тому замість 1 пишемо 2.

Щоб зробити рядок мінімальним, в якості останніх чисел візьмемо 3 та 4, отже, отримаємо рядок 234.

Узагальнюємо.

Значення останнього числа в рядку – максимально можливе, якщо воно дорівнює $n=n-r+r$.

Якщо останнє число – максимально можливе, то передостаннє – максимально можливе, якщо воно дорівнює $n-r+(r-1)$ або $n-r+i$, де $i=r-1$ – позиція цього числа.

Тобто, значення кожного $i$–го числа максимально можливе, якщо числа праворуч від нього – максимально можливі, і це значення дорівнює $n-r+i$.

Отже, проглядаємо рядок справа наліво і визначаємо, чи дорівнює значення $i$- го елемента $n-r+i$ (це максимальне значення, яке може бути в $i$-й позиції).

Перше значення, яке не задовольняє цю умову, можна збільшити.

Нехай, наприклад, це значення дорівнює $m$ і займає $j$-ту позицію. Збільшуємо $m$ на 1, а значення кожного елемента, що стоїть після $j$-го, дорівнює значенню попереднього елемента плюс 1.

## Приклади.

1.Обчислити значення:

$A_5^3; A_6^5, A_8^1, A_8^5, A_8^8, A_{10}^9, A_{n+4}^{n-2}$.

$A_n^r = \frac{n!}{(n-r)!}$ – кількість розміщень без повторень з $n$ елементів по $r$ .

$$A_5^3 = \frac{5!}{(5-3)!} = \frac{1\cdot 2\cdot 3\cdot 4\cdot 5}{2!} = \frac{1\cdot 2\cdot 3\cdot 4\cdot 5}{1\cdot 2} = 3\cdot 4\cdot 5 = 60$$

$$A_6^5 = \frac{6!}{(6-5)!} = \frac{1\cdot 2\cdot 3\cdot 4\cdot 5\cdot 6}{1!} = 6\cdot 20\cdot 6 = 36\cdot 20 = 720$$

$$A_8^1 = \frac{8!}{(8-1)!} = \frac{1\cdot 2\cdot 3\cdot 4\cdot 5\cdot 6\cdot 7\cdot 8}{1\cdot 2\cdot 3\cdot 4\cdot 5\cdot 6\cdot 7} = 8$$

$$A_8^5 = \frac{8!}{(8-5)!} = \frac{8!}{3!} = \frac{3!\cdot 4\cdot 5\cdot 6\cdot 7\cdot 8}{3!} = 20\cdot 8\cdot 42 = 160\cdot 42 = 6720$$

$$A_8^8 = \frac{8!}{0!} = 40320$$

$$A_{10}^9 = \frac{10!}{(10-9)!} = \frac{10!}{1} = 10!$$

2.Обчислити значення:

$C_5^1; C_5^3, C_8^4, C_{12}^6$

$C_n^r = \frac{n!}{r!(n-r)!}$ – кількість розміщень без повторень з $n$ елементів по $r$ .

$$C_5^1 = \frac{5!}{1!(5-1)!} = \frac{5!}{4!} = \frac{4*!5}{4!} = 5$$

$$C_5^3 = \frac{5!}{3!(5-3)!} = \frac{3!\cdot 4\cdot 5}{3!\cdot 2!} = \frac{20}{2} = 10$$

$$C_8^4 = \frac{8!}{4!(8-4)!} = \frac{4\cdot!5\cdot 6\cdot 7\cdot 8}{4!\cdot 4!} = \frac{10\cdot 7\cdot 2}{2} = 70$$

$$C_{12}^6 = \frac{12!}{6!(12-6)!} = \frac{6!\cdot 7\cdot 8\cdot 9\cdot 10\cdot 11\cdot 12}{6!6!} = \frac{7\cdot 8\cdot 9\cdot 10\cdot 11\cdot 12}{1\cdot 2\cdot 3\cdot 4\cdot 5\cdot 6} = \frac{7\cdot 8\cdot 9\cdot 11}{3\cdot 2} = 7\cdot 4\cdot 3\cdot 11 = 924$$

3.Нехай M= $\{1,2,3,4,5\}$.Навести всі розміщення та сполучення без повторень з елементів множини M по 3 елементи.

4.Скількома способами можна визначити призові місця (1-ше, 2-ге, 3-тє) у забігу 12 коней?

$$F\{1,2,3\} \to \{1.....12\}$$

$$\mathrm{A}_{12}^3 = \frac{12!}{(12-3)!} = \frac{12!}{9!} = \frac{9!\cdot 10\cdot 11\cdot 12}{9!} = 110\cdot 12 = 1320\,.$$

5.Скільки різних слів (рядків) можна утворити зі слова MISSISSIPPI, використовуючи всі букви?

В цьому слові є повторні входження букв, тому використовують. формулу для перестановок з повтореннями

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

$$P_{11}(4,4,1,2) = \frac{11!}{4! \cdot 4! \cdot 1! \cdot 2!} = \frac{4! \cdot 5 \cdot 6 \cdot 7 \cdot 8 \cdot 9 \cdot 10 \cdot 11}{4! \cdot 1 \cdot 2 \cdot 3 \cdot 4 \cdot 2} = 5 \cdot 7 \cdot 9 \cdot 10 \cdot 11 = 35 \cdot 9 \cdot 110 = 34650.$$

6.Знайти лексикографічно наступну перестановку для кожної з перестановок

1432; 54123; 12453; 45231; 6714235; 31528764.

$\underbrace{1432}$ $54\underbrace{123}$ $12\underbrace{453}$ $45\underbrace{231}$ $6714\underbrace{235}$ $315\underbrace{28764}$

2134 54132 12534 45312 6714253 31542678.

7.Розмістити наведені перестановки елементів множини $\{1,2,3,4,5,6\}$ у лексикографічному порядку

234561, 231456, 165432, 156423, 543216,

541236, 231465, 314562, 432561, 654321,

654312, 435612.

Відповідь: 1) 156423; 2) 165432; 3) 231456; 4) 231465; 5) 234561; 6)314562; 7) 432561; 8) 435612; 9) 541236; 10) 543216; 11) 654312; 12) 654321

8.За допомогою алгоритму побудови лексикографічно наступної перестановки записати перші 132 перестановки елементів множини $\{1,2,3,4,5,6\}$.

$123\underbrace{456}$ $126\underbrace{435}$ $135\underbrace{624}$ $145\underbrace{236}$ $153\underbrace{426}$

$123\underbrace{465}$ $126\underbrace{453}$ $135\underbrace{642}$ $145\underbrace{263}$ $153\underbrace{462}$

$123\underbrace{546}$ $126\underbrace{534}$ $136\underbrace{245}$ $145\underbrace{326}$ $153\underbrace{624}$

123564 126543 136254 145362 153642

123645 132456 136425 145623 154236

123654 132465 136452 145632 154263

124356 132546 136524 146235 154326

124365 132564 136542 146253 154362

124536 132645 142356 146325 154623

124563 132654 142365 146352 154632

124635 134256 142536 146523 156234

124653 134265 142563 146532 156243

125346 134526 142635 152346 156324

125364 134562 142653 152364 156342

125436 134625 143256 152436 156423

125463 134652 143265 152463 156432

125634 135246 143526 152634 162345

125643 135264 143562 152643 162354

126345 135426 143625 153246 162435

126354 135462 143652 153264 162453

162534 164523 213645

162543 164532 213654

163245 165234 214356

163254 165243 214365

163425 165324 214536

163452 165342 214563

163524 165423 214635

163542 165432 214653

164235 213456

164253 213465

164325 213546

164352 213564

9.Задати взаємно-однозначну відповідність між елементами множини M={*a,b,c,d,e*} і X={1,2,3,4,5}.

Побудувати перші 28 перестановок елементів множини M у лексикографічному порядку.

Задамо взаємно-однозначну відповідність між елементами даних множин таким чином:

$$a \Leftrightarrow 1$$
$$b \Leftrightarrow 2$$
$$c \Leftrightarrow 3$$
$$d \Leftrightarrow 4$$
$$e \Leftrightarrow 5$$

| | |
|---|---|
| 12345 | *abcde* |
| 12354 | *abced* |
| 12435 | *abdce* |
| 12453 | *abdec* |
| 12534 | *abecd* |
| 12543 | *abedc* |
| 13245 | *acbde* |
| 13254 | *acbed* |
| 13425 | *acdbe* |
| 13452 | *acdeb* |
| 13524 | *acebd* |
| 13542 | *acedb* |
| 14235 | *adbce* |
| 14253 | *adbec* |
| 14325 | *adcbe* |
| 14352 | *adceb* |
| 14523 | *adebc* |
| 14532 | *adecb* |
| 15234 | *aebcd* |
| 15243 | *aebdc* |
| 15324 | *aecbd* |
| 15342 | *aecdb* |
| 15423 | *aedbc* |
| 15432 | *aedcb* |
| 21345 | *bacde* |
| 21354 | *baced* |

Приклади на генерування підмножин.

Їх кількість $C_m^n = \frac{m!}{n!(m-n)!}$

Приклад 1.

Знайти всі чотирьохелементні підмножини множини з шести елементів.

Використовуємо формулу:

$$C_6^4 = \frac{6!}{2!\cdot 4!} = \frac{5\cdot 6}{2} = 15$$

| | | | | | |
|---|---|---|---|---|---|
| 1) | 1234 | 6) | 1256 | 11) | 2345 |
| 2) | 1235 | 7) | 1345 | 12) | 2346 |
| 3) | 1236 | 8) | 1346 | 13) | 2356 |
| 4) | 1245 | 9) | 1356 | 14) | 2456 |
| 5) | 1246 | 10) | 1456 | 15) | 3456 |

→ п'ятнадцять підмножин.

Приклад 2.

Знайти всі 3-елементні підмножини 6-елементної множини.

Їх кількість $C_6^3 = \frac{6!}{3!3!} = \frac{4 \cdot 5 \cdot 6}{6} = 20$

| | | | | | |
|---|---|---|---|---|---|
| 1) | 123 | 8) | 145 | 15) | 246 |
| 2) | 124 | 9) | 146 | 16) | 256 |
| 3) | 125 | 10) | 156 | 17) | 345 |
| 4) | 126 | 11) | 234 | 18) | 346 |
| 5) | 134 | 12) | 235 | 19) | 356 |
| 6) | 135 | 13) | 236 | 20) | 456 |
| 7) | 136 | 14) | 245 | | |

двадцять підмножин.

**Зміст**

**Копча-Горячкіна Галина Ернестівна** – старший викладач кафедри інформаційних управляючих систем та технологій факультету інформаційних технологій ДВНЗ «Ужгородський національний університет»

**ТЕОРІЯ АЛГОРИТМІВ ТА МАТЕМАТИЧНІ ОСНОВИ ПРЕДСТАВЛЕННЯ ЗНАНЬ**

Навчально-методичний посібник. Частина ІІІ для студентів факультету інформаційних технологій напряму „Комп'ютерні науки" зі спеціальності „інформаційні управляючі системи та технології" ДВНЗ «Ужгородський національний університет»

Відповідальний за випуск: **Міца О.В.** – кандидат тех. наук, завідувач кафедри інформаційних управляючих систем та технологій.