Олексій Васильєв

# Алгоритми

Навчальний посібник

Київ
Видавництво Ліра-К
2022

УДК 510.5:004.4](075.8)
В19

*Рецензенти*

*Гаврюшенко Дмитро Анатолійович*, доктор фізико-математичних наук, професор кафедри молекулярної фізики фізичного факультету Київського національного університету імені Тараса Шевченка.

*Кондратенко Сергій Вікторович*, доктор фізико-математичних наук, професор кафедри оптики фізичного факультету Київського національного університету імені Тараса Шевченка.

*Решетняк Віктор Юрійович*, доктор фізико-математичних наук, професор, завідувач кафедри теоретичної фізики фізичного факультету Київського національного університету імені Тараса Шевченка.

**Васильєв О.М.**

**В19** Алгоритми : навч. посіб. Київ : Видавництво Ліра-К, 2022. 424 с.
ISBN 978-617-520-353-8

Посібник, який читач тримає в руках, присвячений алгоритмам. У ньому розглядається широке коло тем, так чи інакше пов’язаних з програмуванням і аналізом даних. Зокрема, розповідається, що таке алгоритми й навіщо вони потрібні, наводяться правила алгоритмічного псевдокоду, обговорюються підходи щодо аналізу складності алгоритмів. Крім цього, аналізуються такі типи даних, як масиви, списки, дерева та графи. Приділяється увага обчислювальним задачам. Окремі розділи книги присвячені нейронним мережам та квантовим обчисленням. Також книга містить додаток з прикладами програмних кодів мовою Python, в яких реалізуються алгоритми з книги.

Видання буде цікавим та корисним студентам та викладачам, а також усім, хто вивчає програмування.

**УДК 510.5:004.4](075.8)**

**ISBN 978-617-520-353-8**

# ЗМІСТ

**Вступ. Мистецтво думати та програмувати ..... 3**
Про що ця книга ..... 3
Для кого ця книга ..... 4
Структура книги ..... 5
Про автора ..... 6
Зворотній зв'язок ..... 6
Подяки ..... 6

**Розділ 1. Знайомство з алгоритмами ..... 7**
Алгоритми та життя ..... 7
Алгоритми та оптимізація ..... 9
Алгоритмічні задачі ..... 12
Обернені задачі ..... 15
Алгоритми та моделі ..... 20
Резюме ..... 22
Завдання для самостійної роботи ..... 23
Відповіді та розв'язки ..... 24

**Розділ 2. Алгоритмічний псевдокод ..... 29**
Знайомство з псевдокодом ..... 29
Умови й вибір ..... 32
Повторення команд ..... 35
Алгоритми та програми ..... 37
Приклади створення алгоритмів ..... 43
Резюме ..... 48
Завдання для самостійної роботи ..... 50
Відповіді й розв'язки ..... 51

**Розділ 3. Складність алгоритмів ..... 59**
Порівняння алгоритмів ..... 59
Асимптотичні оцінки ..... 63
Властивості асимптотичних оцінок ..... 66
Класи складності алгоритмів ..... 70
Оптимізація алгоритмів ..... 74
Резюме ..... 79
Завдання для самостійної роботи ..... 79
Відповіді й розв'язки ..... 80

**Розділ 4. Робота з масивами ..... 83**
Масив як тип даних ..... 83
Основні операції з масивами ..... 85

Пошук елементів ..... 90
Сортування масиву ..... 95
Двомірні масиви ..... 102
Резюме ..... 104
Завдання для самостійної роботи ..... 105
Відповіді та розв'язки ..... 106

**Розділ 5. Списки ..... 115**
Зв'язні списки ..... 115
Створення зв'язного списку ..... 117
Перебір елементів у списку ..... 119
Зміна складу списку ..... 121
Сортування зв'язних списків ..... 127
Списки з циклами ..... 132
Двонаправлені списки ..... 139
Стеки та черги ..... 142
Резюме ..... 145
Завдання для самостійної роботи ..... 146
Відповіді та розв'язки ..... 147

**Розділ 6. Дерева та графи ..... 158**
Знайомство з деревами ..... 158
Створення дерева ..... 161
Перегляд дерев ..... 164
Основні операції з деревами ..... 168
Впорядковані дерева ..... 171
Знайомство з графами ..... 177
Представлення графів ..... 180
Пошук у графі ..... 183
Використання матриці й списку суміжності ..... 185
Резюме ..... 189
Завдання для самостійної роботи ..... 189
Відповіді та розв'язки ..... 190

**Розділ 7. Числові алгоритми ..... 197**
Алгебраїчні рівняння ..... 197
Числове інтегрування ..... 206
Інтерполяційний поліном ..... 208
Апроксимація ..... 212
Диференціальні рівняння ..... 215
Резюме ..... 220
Завдання для самостійної роботи ..... 221
Відповіді та розв'язки ..... 222

**Розділ 8. Алгоритми та задачі ..... 232**
Генерування випадкових чисел ..... 232
Рекурсія ..... 235
Оптимізація розрахунків ..... 240
Ймовірнісні алгоритми ..... 243
Жадібні алгоритми ..... 246
Наближені алгоритми ..... 254
Резюме ..... 259
Завдання для самостійної роботи ..... 260
Відповіді та розв'язки ..... 261

**Розділ 9. Нейронні мережі ..... 273**
Загальні відомості про нейронні мережі ..... 273
Навчання нейронної мережі ..... 277
Структура нейронної мережі ..... 285
Мережі з багатьма прошарками ..... 288
Особливості використання нейронних мереж ..... 292
Резюме ..... 293
Завдання для самостійної роботи ..... 294
Відповіді і розв'язки ..... 296

**Розділ 10. Квантові обчислення ..... 299**
Квантові мікрооб'єкти ..... 299
Квантові формальності ..... 301
Знайомство з кубітами ..... 304
Обчислення на кубітах ..... 309
Принципи квантових обчислень ..... 317
Резюме ..... 321
Завдання для самостійної роботи ..... 321
Відповіді та розв'язки ..... 323

**Заключення. Фінальний акорд ..... 327**

**Додаток. Програмні коди ..... 328**
Розділ 2. Алгоритмічний псевдокод ..... 328
Розділ 3. Складність алгоритмів ..... 340
Розділ 4. Робота з масивами ..... 342
Розділ 5. Списки ..... 359
Розділ 6. Дерева та графи ..... 387
Розділ 7. Числові алгоритми ..... 399
Розділ 8. Алгоритми та задачі ..... 414
Розділ 9. Нейронні мережі ..... 420

Вступ

# Мистецтво думати та програмувати

*Думати завжди корисно. Особливо якщо за це ще й платять.*

Нерідко доводиться чути одне й те саме запитання: яка мова програмування найкраще підходить для того аби навчитися добре програмувати? Питання актуальне, але, відверто кажучи, не зовсім коректне. Мова програмування – це лише інструмент. А інструменти, як відомо, змінюються залежно від потреб. Інша справа – мистецтво програмування. Це те, що назавжди з тобою. Про такі речі й поговоримо в книзі.

**Історична довідка**

Від Марка Тулія Цицерона (106 до н.е. – 43 до н.е.) до нас дійшла фраза "Все своє ношу з собою". Сам Цицерон приписував її грецькому мудрецю Біанту Прієнському. Коли перси захопили місто Прієну, вони дозволили мешканцям піти й забрати з собою те, що зможуть нести в руках. Біант залишав стіни міста з легкою душею, без речей. Коли воїн-перс запитав його, невже він зовсім не нажив добра, Біант, показуючи на лоба, відповів: "Все своє ношу з собою".

## Про що ця книга

*Коли пояснюєш суть того, що відбувається, важливо не заплутати самого себе.*

Якщо коротко, то книга присвячена тому, як створювати, використовувати та аналізувати алгоритми. При цьому, безумовно, в першу чергу мається на увазі така сфера діяльності, як програмування. Дивного тут нічого немає, оскільки невід'ємна складова у підготовці гарного програміста пов'язана з розвитком алгоритмічного мислення. Важливо також розуміти, що в загальному випадку алгоритм як такий не пов'язаний з якоюсь конкретною мовою програмування. Тому не буде перебільшенням стверджувати, що книга, яку читач тримає в руках, про те, як програмувати без прив'язки до мови програмування (хоча ми і будемо для формалізації алгоритмів використовувати алгоритмічний псевдокод).

**До уваги**

Алгоритм – це душа програми. Тому, щоб навчитися програмувати, в першу чергу слід навчитися створювати алгоритми.

Взагалі ж у книзі розглядається досить багато тем. Почнемо ми з того, що обговоримо алгоритми як такі і дізнаємося, яке місце алгоритми займають в нашому житті. Ми розглянемо деякі історичні задачі, пов'язані з використанням

алгоритмів. Все це покликане не тільки переконати читача у важливості алгоритмів, але й показати наскільки це красива та "життєва" сфера діяльності.

Важливе значення у роботі з алгоритмами має аналіз їх складності. Адже часто необхідно оцінити ефективність того чи іншого алгоритму або вибрати з декількох алгоритмів найкращий. Для цього необхідно знати і розуміти, які характеристики і особливості алгоритмів необхідно враховувати. Ці питання обговорюються у книзі.

Як уже зазначалося, процес створення алгоритму не прив'язаний до певної мови програмування. При цьому для запису алгоритму (тобто для формального його представлення) потрібні якісь правила або спеціальні інструкції, котрі зазвичай називаються псевдокодом. Правилам запису алгоритмів з використанням псевдокоду також приділяється увага в книзі.

Якщо виходити з того, що алгоритми створюються для обробки даних (а в більшості випадків так і є), то спосіб реалізації або збереження даних є далеко не останнім фактором при виборі алгоритму. Тому в книзі коротко обговорюються основні способи організації даних (масиви, черги, стеки, бінарні дерева, графи) і особливості реалізації алгоритмів, призначених для обробки відповідних даних.

Є й інші теми, які в різній мірі згадуються у книзі. Серед них можна виділити алгоритми для математичних розрахунків, особливості створення алгоритмів для квантових комп'ютерів, принципи моделювання на основі нейронних мереж, та деякі інші питання. Хочеться вірити, що це буде не тільки корисно, але й цікаво.

# Для кого ця книга

*Люди поділяються на тих, хто любить читати, і тих, хто читає.*

Безумовно, книга в першу чергу призначена для тих, хто хоче навчитися програмувати. Але тим, хто вже програмує, вона також буде корисною. Адже програмувати можна по-різному. При цьому не вимагається, аби читач володів якимись спеціальними знаннями в області програмування, математики чи чогось схожого на ці захопливі сфери людської діяльності. Простіше кажучи, книга стане в нагоді навіть новачкам у всьому, що має відношення до програмування, розрахунків, математики.

**До уваги**

Сьогодні в це, мабуть, важко повірити, але ще відносно недавно програмуванню навчали не те що без застосування будь-якої мови програмування, а взагалі без комп'ютера. Безумовно, до такої ситуації можна ставитися по різному, але якщо відверто, то доброму, якісному вивченню програмування це не заважає. Скоріше, навіть навпаки.

Незважаючи на те, що спеціальна підготовка для читача не вимагається, в книзі зустрічається досить багато "спеціальних моментів", пов'язаних як з математикою, так і з іншими нетривіальними темами. В усіх таких ситуаціях ми

будемо виходити з того, що читач не має відповідної підготовки і всі складні чи незрозумілі питання будуть коментуватися і пояснюватися у деталях.

Нарешті, хочеться вірити, що книга виявиться корисною й цікавою навіть тим, хто не планує серйозно займатися програмуванням. Адже вона містить багато інформації всебічного характеру, котра дозволить по-новому подивитись на різні речі та події.

# Структура книги

*Важливим є не тільки те, що ти кажеш, але і як ти це робиш.*

Основна частина книги складається з декількох розділів. Кожен розділ присвячений певній темі або вивченню якогось питання.

У *Розділі 1* обговоримо найбільш загальні питання, пов'язані з алгоритмами. Там ми дізнаємось, що таке алгоритми, навіщо вони потрібні, як алгоритми використовують і яка їх роль у повсякденному житті.

*Розділ 2* присвячений правилам представлення алгоритмів за допомогою псевдокоду.

У *Розділі 3* обговоримо підходи, які дозволяють аналізувати складність та ефективність алгоритмів.

*Розділ 4* присвячений питанням використання масивів. Там розглянемо базові прийоми обробки масивів.

*Розділ 5* містить базові відомості щодо зв'язних списків. В контексті цієї теми ми розглянемо низку різних задач.

У *Розділі 6* описано дерева та графи. Цей розділ дасть змогу скласти певне уявлення щодо специфіки використання зазначених нетривіальних конструкцій, зрозуміти їх переваги та недоліки.

*Розділ 7* містить матеріал, присвячений числовим алгоритмам. Там стисло розглянемо розрахункові задачі, які мають стосунок до розв'язання алгебраїчних та диференціальних рівнянь, інтерполяції, апроксимації та інтегрування.

У *Розділі 8* зібрано і розглянуто деякі задачі, що з різних причин не увійшли до попередніх розділів. Серед іншого, там можна знайти корисну інформацію щодо генерування випадкових чисел, ймовірнісних та наближених алгоритмів, та ще дещо.

*Розділ 9* присвячений використанню нейронних мереж (в контексті роботи з алгоритмами). Цей розділ буде корисний тим, хто цікавиться сучасними підходами до розв'язання складних задача.

У *Розділі 10* стисло проаналізуємо принципи квантових розрахунків. Це досить нова область знань, але не виключено, що за нею майбутнє.

У кінці кожного розділу є *резюме* та *список задач* для самостійного розв'язання. Наводяться також і можливі розв'язки цих задач. Разом з тим, перед ознайомленням з цими розв'язками рекомендується все ж таки докласти зусиль для їх самостійного пошуку.

*Додаток* у кінці книги містить приклади реалізації алгоритмів мовою Python. Ця мова вибрана в силу двох причин. По-перше, вона популярна і широко використовується на практиці. По-друге, синтаксис мови в багатьох моментах корелює з конструкціями псевдокоду, який використовується в книзі. Але це все ж скоріше доповнення до основної частини книги, своєрідний бонус. Для успішного оволодіння матеріалом книги зовсім не обов'язково мати досвід в області програмування.

# Про автора

*Навіть у народної творчості є автор. Просто народу про нього нічого не відомо.*

*Васильєв Олексій Миколайович*, доктор фізико-математичних наук, професор кафедри теоретичної фізики фізичного факультету Київського національного університету імені Тараса Шевченка. Автор книг з програмування мовами C, C++, C#, Java, JavaScript, Python, PHP. Сфера наукових інтересів: фізика рідин і рідких кристалів, біофізика, математична лінгвістика, моделювання економічних та соціально-політичних процесів.

# Зворотній зв'язок

*Страшно уявити, як це люди у 19 столітті обходилися без інтернету та електронної пошти.*

Свої зауваження, побажання та коментарі можна направляти за електронною поштою `oleksii@vasyliev.kyiv.ua`. Також корисну інформацію про автора та його книги можна знайти на сайті `www.vasyliev.kyiv.ua` та на YouTube-каналі `www.youtube.com/channel/UClyL2-UbbWXd7S1PHPefM0g`.

# Подяки

*Іноді так багато хочеться сказати, що краще взагалі нічого не говорити.*

Користуючись нагодою хочеться висловити щиру подяку своїм студентам та читачам. Це саме та вдячна та терпляча аудиторія, завдяки якій вдається робити книги кращими та цікавішими.

Розділ 1

# Знайомство з алгоритмами

*Безумовно, правила поведінки є, але на жаль, не всім про них відомо.*

У цьому розділі ми ознайомимось з концепцією алгоритму. Ми дізнаємось, як і де алгоритми використовуються і що потрібно зробити для того, щоб уміти застосовувати алгоритми в потрібному місці в потрібний час.

## Алгоритми та життя

*У будь якій важливій справі головне – мати план дій.*

Так що ж таке *алгоритм*? З цього приводу є різні формулювання, але якщо особливо не заглиблюватися в деталі, то алгоритм – це певний набір інструкцій чи команд, виконання яких приводить до розв'язання певної задачі чи вирішення певного завдання.

**Історична довідка**

Вважається, що слово "алгоритм" походить від імені вченого із Середньої Азії Аль-Хорезмі (783-850), математика, астронома, перекладача та філософа, "батька алгебри".

Іншими словами, якщо нам необхідно вирішити якусь задачу, і ми для вирішення цієї задачі розробляємо детальний план з конкретними інструкціями, виконуючи які однозначно отримуємо потрібний результат, то це й буде алгоритм. Зазвичай про алгоритми говорять у контексті задач з програмування. Але насправді алгоритми оточують нас скрізь, в тому числі й у повсякденному житті. Просто ми не завжди усвідомлюємо, що маємо справу саме з алгоритмами.

**До уваги**

Алгоритмічне мислення – надзвичайно важлива річ не тільки в програмуванні, але й у житті. Тому наївно вважати, що алгоритми – це тільки про програмування. У дійсності алгоритми – це про все. І ми навіть не завжди уявляємо, наскільки сильно алгоритми присутні в нашому повсякденному житті й нашій діяльності.

У нашому повсякденному житті ми постійно (і зазвичай неявно) використовуємо алгоритми. Причому часто цього навіть не помічаємо. Так, у багатьох людей існує чітка послідовність дій, яких вони дотримуються збираючись на роботу. Наприклад, алгоритм може бути таким:

[1] Прокинутися в певний час.

[2] Умитися, почистити зуби, привести себе до ладу.
[3] Поснідати.
[4] Одягтися відповідним чином.
[5] Дійти до зупинки громадського транспорту.
[6] Сісти в правильний тролейбус/автобус.
[7] Вийти на певній зупинці.
[8] Дійти до офісу.

Якщо хтось на шляху до роботи виконує декілька пересадок або користується власним транспортом, то алгоритм буде іншим, але важливо, що він буде.

**Теорія**

Алгоритмічний тип поведінки є невід'ємною складовою нашого життя, так би мовити, на фундаментальному рівні. Як приклад, можна навести економічну теорію в тій її частині, котра пояснює поведінку економічних агентів. У класичній економічній теорії прийнято вважати, що люди в процесі своєї діяльності оптимізують власну вигоду. Іншими словами, кожного разу, коли індивід приймає рішення, він розв'язує певну оптимізаційну задачу. Така точка зору в економічній науці історично склалася першою. Але на сьогоднішній день існують переконливі докази, що це далеко не завжди так. Один з перших каменів у город раціональності економічних агентів був кинутий Торстейном Вебленом. Мова про ефект Веблена, або ефект демонстративної поведінки. Суть ефекту в тому, що деякі індивіди за інших рівних умов купують найбільш дорогі товари з метою підкреслити свій соціальний статус.
Ефект приєднання до більшості (ефект наслідування) полягає в тому, що індивіди роблять такий самий вибір, як і більшість інших агентів (тобто купують те, що купують всі).
Ефект сноба відповідає поведінці індивіда, за якої обираються товари, найменш затребувані іншими економічними агентами (щоб було не так, як у всіх).
У кожному із зазначених випадків індивіди замість того, щоб вирішувати задачу оптимізації, використовують певні алгоритми (або набори інструкцій) для прийняття рішень.

**Історична довідка**

Торстейн Веблен (1857-1929) – американський економіст та соціолог, засновник інституціонального напрямку в економічній теорії. Одна з основних та найбільш відомих праць – "Теорія бездіяльного класу".

Запропонований алгоритм досить умовний. Практично кожна позиція з цього алгоритму може бути реалізована своїм внутрішнім алгоритмом. Скажімо, пункт, пов'язаний з посадкою в тролейбус, може бути реалізований таким алгоритмом:

[1] Очікувати прибуття тролейбусу.
[2] Якщо тролейбус прибув, перевірити його номер.
[3] Якщо номер "правильний" – сісти в тролейбус.
[4] Інакше (це не наш тролейбус) – чекати на наступний тролейбус (див. пункт [1]).

У цьому випадку в алгоритмі з'являється умова. Ми перевіряємо, чи той тролейбус приїхав, і якщо так, то сідаємо у нього. Інакше продовжуємо чекати наступний тролейбус. Фактично це означає, що ми перестрибуємо на початок списку з інструкціями. Тобто наш алгоритм виконується з урахуванням циклічності. Ми повторюємо команди допоки не приїде потрібний тролейбус.

**До уваги**

Слід відмітити дві важливі обставини. По-перше, в алгоритмі важливі не тільки інструкції самі по собі, але й порядок їхнього виконання. По-друге, алгоритм неявно передбачає наявність виконавця. Іншими словами, алгоритм призначений для когось. Зазвичай це або людина, або комп'ютер. Якщо ми маємо справу з комп'ютером, алгоритм доведеться перекласти на мову, зрозумілу для комп'ютера – тобто написати одною з мов програмування програму, яка реалізує алгоритм. У випадку, коли алгоритм призначений для людини, його можна сформулювати в звичайній словесній або текстовій формі, причому суттєво економивши на строгості інструкцій, оскільки для людини, на відміну від комп'ютера, цілком прийнятні натяки і недомовленості.

Подібних ситуацій багато. Ми несвідомо використовуємо алгоритмічний підхід для вирішення найрізноманітніших повсякденних задач. Це важлива частина нашого життя. Разом з тим, у книзі ми будемо обговорювати алгоритми дещо іншого роду. Мова піде про застосування алгоритмів для вирішення конкретних прикладних задач, більша частина з яких стосується програмування.

**До уваги**

Щодо застосування алгоритмів, грань між звичайним життям та програмуванням не завжди легко визначити. Як приклад можна навести алгоритми, які використовуються в криптографії (шифрування) або при майнінгу криптовалют. З одного боку, це, безумовно, програмування. З іншого боку, долі цілих країн залежали і залежать від такої діяльності.

# Алгоритми та оптимізація

*Краще мати грамотний план дій і програти, ніж виграти випадково й без будь-якого плану.*

Власне, в книзі ми будемо обговорювати способи розв'язання задач. Але не будь яких задач, а лише тих, котрі допускають алгоритмічне розв'язання. В такому випадку розв'язком задачі буде набір інструкцій (алгоритм), виконання яких дозволить отримати бажаний результат.

**Теорія**

Алгоритми мають дві важливі властивості: конкретність і масовість (або універсальність). Конкретність алгоритму пов'язана з тим, що дії або інструкції, які виконуються в рамках алгоритму, мають бути чіткими, зрозумілими та інтерпретуватись однозначно. Тому різні виконавці будуть виконувати один і той самий алгоритм однаково.

Масовість означає, що один і той самий алгоритм можна багатократно використовувати для розв'язання однотипних задач. Як ілюстрацію до останнього твердження розглянемо невеликий приклад. Припустимо, нам необхідно розв'язати рівняння $3x = 6$. Нескладно здогадатися, що в цьому випадку існує єдиний розв'язок $x = 2$. Ми цей розв'язок отримали, поділивши 6 на 3. Варто зазначити, що розв'язком є число. Цей розв'язок "одноразовий" в тому сенсі, що якщо ми змінимо коефіцієнти в рівнянні, то й розв'язок буде іншим.

Тепер розглянемо більш загальний випадок: будемо розв'язувати рівняння вигляду $Ax = B$. Природнім чином на думку спадає відповідь $x = B/A$. Але проблема в тому, що ділити на $A$

можна не завжди, а тільки якщо $A \neq 0$. За нульового значення параметра $A$ можливі два варіанти. Якщо $B \neq 0$, то рівняння розв'язків не має. Якщо ж $B = 0$, то розв'язком рівняння може бути довільне число.
Отже, як "розв'язок" рівняння вигляду $Ax = B$ можна запропонувати такий алгоритм.
[1] Якщо параметр $A$ відмінний від нуля, то розв'язок $x = B/A$.
[2] Інакше (тобто за умови $A = 0$), якщо параметр $B$ відмінний від нуля, розв'язків немає.
[3] Інакше (тобто якщо $A = 0$ та $B = 0$) розв'язок рівняння – довільне число.
Цей алгоритм можна застосовувати для отримання розв'язку довільного рівняння, в тому числі й розглянутого на самому початку.

Алгоритмічні задачі мають свої особливості, а результати, які отримують при застосуванні алгоритмів, не є такими прогнозованими та однозначними, як може видатись на перший погляд. Звернемося до невеликого прикладу. У теорії ігор ця задача отримала назву "дилема в'язня".

**Деталі**

У рамках теорії ігор вивчаються процеси, в яких беруть участь не менше двох учасників (гравців), котрі намагаються реалізувати свої інтереси (в тому числі й за рахунок інших гравців). Більш конкретно, предметом вивчення теорії ігор є оптимальні стратегії поведінки учасників, які дозволяють їм досягати потрібного результату. Методи теорії ігор знаходять широке застосування в економіці, соціології, психології, політології.
Одна з фундаментальних робіт у цій області – книга "Теорія ігор і економічна поведінка" Джона фон Неймана та Оскара Моргенштерна. Лауреатами Нобелевської премії з економіки (премії з економіки пам'яті Альфреда Нобеля) є й дослідники в області теорії ігор. Серед них Джон Неш, про якого знято фільм "Ігри розуму".

**Історична довідка**

Джон фон Нейман (1903-1957) – видатний математик, фізик та дослідник, який зробив фундаментальний внесок у розвиток математики, фізики, інформатики, економіки. Народився в Угорщині, згодом переїхав до США.

Оскар Моргенштерн (1902-1977) – економіст, один із засновників теорії ігор. Народився в Німеччині, потім емігрував до США.

Джон Неш (1928-2015) – американський математик, який, крім іншого, зробив значний внесок у розвиток теорії ігор.

У класичній версії дилему в'язня формулюють таким чином. Припустимо, поліція затримала двох злочинців (умовно назвемо їх `А` та `Б`) та бажає, щоб кожен дав свідчення на спільника. Ось можливі варіанти:

● Якщо обидва мовчать (не дають свідчень), то як покарання кожен отримує `1` рік тюремного ув'язнення.

● Якщо обидва зізнаються (дають свідчення на спільника), то як покарання вони отримують по `5` років тюремного ув'язнення.

● Якщо зізнається тільки один зі злочинців, то його відпускають (покарання у вигляді `0` років тюремного ув'язнення), а іншого (того, що не зізнався) карають строком у `10` років.

Яким буде алгоритм поведінки злочинців і яким буде результат (яке покарання отримає кожен зі злочинців)? Зрозуміло, що кожен зі злочинців

може обрати одну з двох стратегій: мовчати або говорити (давати свідчення). Два злочинці, й у кожного по дві стратегії – всього виходить чотири комбінації. Для зручності вони наведені в табл. 1.1.

**Таблиця 1.1.** Стратегії в задачі "дилема в'язня"

| | **Б мовчить** | **Б говорить** |
|---|---|---|
| **А мовчить** | (1:1) | (10:0) |
| **А говорить** | (0:10) | **(5:5)** |

Поля таблиці позначають стратегії, котрі обирає кожен зі злочинців, а на перетині відповідного рядка і стовпця вказана пара чисел, яка визначає результат. Наприклад, (10:0) означає, що злочинець А отримує як покарання 10 років, а злочинець Б отримує як покарання 0 років. Пара (1:1) означає, що обидва злочинці отримують покарання в 1 рік, і так далі.

**До уваги**

У теорії ігор учасників прийнято називати гравцями, а результат – виграшом. Хоча в цьому конкретному випадку це скоріше програш. Мета гравців полягає в тому, щоб максимізувати власний виграш (мінімізувати програш).

Яку стратегію виберуть гравці (будемо так називати учасників)? Наприклад, як себе буде поводити гравець А? У нього дві стратегії – мовчати або говорити. Якщо Б мовчить, то А краще говорити: в цьому випадку його програш дорівнює 0 замість 1 у випадку, якщо А буде мовчати. Якщо Б говорить, то А також краще говорити. Така стратегія дає програш 5 замість програшу 10 у випадку, коли А буде мовчати. Таким чином, незалежно від стратегії гравця Б, гравцю А вигідніше говорити. Аналогічні міркування приводять до висновку, що гравцю Б також вигідніше говорити. Як наслідок, реалізується стратегія, при якій обидва гравці говорять (обидва злочинці дають свідчення), і результатом є комбінація (5:5). Цей результат вочевидь гірший за результат (1:1), який реалізується у випадку, якщо обидва гравці обирають "мовчазну" стратегію.

Виходить, що оптимізуючи власний результат гравці в решті решт отримують не найкращий фінал. Для отримання іншого результату гравці повинні скооперуватися: наприклад, домовитися про те, що обидва притримуються стратегії "мовчати". Але така сумісна стратегія нестійка в тому сенсі, що у кожного з гравців буде спокуса порушити домовленості й тим самим покращити свій власний результат.

**До уваги**

Алгоритми, засновані на пошуку локального (частинного) оптимального розв'язку, називають жадібними (їх ми ще будемо обговорювати). Застосування таких алгоритмів далеко не завжди дозволяє отримати загальний оптимальний розв'язок. У цьому випадку гравці оптимізують власну вигоду, програючи в цілому.

Більш складний варіант гри передбачає, що вона повторюється знову і знову з тими самими гравцями. Зазвичай така гра називається "повторювана дилема в'язня". Тепер гравці роблять вибір, знаючи про попередні результати гри. У цьому випадку задача ускладняється кардинально, і зазвичай все зводиться до вибору тої чи іншої стратегії з урахуванням попередніх дій суперника. Фактично, йдеться про змагання алгоритмів. Причому перевірка виконується експериментальним шляхом.

**Деталі**

Проходять чемпіонати з повторюваної дилеми в'язня. Учасниками є програми, які реалізують той чи інший алгоритм поведінки гравця. Найкращі результати демонструє алгоритм, який можна назвати "ти мені – я тобі". У цьому випадку гравець обирає таку стратегію, яку його супротивник обрав на попередньому етапі.

Які висновки з усього зазначеного? Тут ми стикаємося з прикладом задачі, в якій пошук оптимального рішення в класичному розумінні або не приводить до прийнятного результату, або взагалі неможливий. Тому розв'язок пропонується у вигляді алгоритму, який регламентує стратегію учасників.

# Алгоритмічні задачі

*Інтуїція – це коли серед нерозв'язаних задач знаходиш ту, яку можна розв'язати.*

Існує клас задач, розв'язок яких необхідно сформулювати у вигляді правила дій або алгоритму, що дозволяє отримати необхідний результат. Практично кожен з нас стикався з подібними ситуаціями. У повсякденному житті відповідні задачі називають головоломками. Вони бувають різними. Як приклад, розглянемо невелику ігрову задачу.

Двоє грають у таку гру. Вони по черзі називають числа в діапазоні від `1` до `4` включно. Усі названі в процесі гри числа підсумовуються. Виграє той, після ходу якого сума чисел стане рівною `50`. Необхідно визначити виграшну стратегію – алгоритм дій, який дозволяє гарантовано досягти успіху, незалежно від того, які числа називає супротивник.

Щоб розібратися в тому, яка стратегія виграшна, дещо змінимо опис задачі, не змінюючи при цьому її суті. А саме, процес будемо кодувати послідовністю чисел, кожне з яких більше попереднього на число в межах від `1` до `4`. Це ті суми, які ми отримуємо в процесі гри після того, як черговий гравець назвав своє число. Таким чином, останнім числом у послідовності буде `50`, і це число означає перемогу одного з гравців. Наприклад, якщо перший гравець називає число `3`, потім другий гравець називає число `4`, перший називає число `1`, а другий – число `2`, то цей фрагмент гри можна представити у вигляді послідовності чисел `3`, `7`, `8`, `10` – сума чисел на кожному кроці гри. Хто ж виграє? Вочевидь той, хто першим отримає суму `45`. Чому? Тому що після

того, як сума чисел стала рівна 45, наступний гравець назве число від 1 до 4 і сума буде в діапазоні від 46 до 49. Значить, у суперника з'являється можливість назвати число в діапазоні від 1 до 4 і отримати суму 50. А хто першим отримає суму 45? Той, хто першим отримає суму 40. Логіка така сама, як і в попередньому випадку. Якщо після нашого ходу сума рівна 40, то після ходу суперника вона буде у діапазоні від 41 до 44. Отже, ми зможемо назвати таке число, щоб сума дорівнювала 45.

Дотримуючись того самого принципу, легко приходимо висновку, що виграшна стратегія полягає у тому, щоб після нашого ходу сума чисел дорівнювала 5, 10, 15, 20, 25, 30, 35, 40 і 45. Але цю стратегію ми можемо реалізувати тільки в тому випадку, якщо наш супротивник ходить першим. Дійсно, той, хто називає число першим, може назвати число від 1 до 4. Тоді другий гравець має можливість назвати число таке, що в сумі з першим воно дає 5.

**До уваги**

Ми виходимо з того, що наш супротивник не помиляється. Іншими словами, розв'язок шукаємо виходячи з припущення, що якщо виграшна стратегія відома нам, то вона автоматично відома й нашому супротивнику.

Таким чином, розв'язок задачі можна сформулювати у вигляді двох тверджень.

- Виграє гравець, який називає число другим.
- Виграшна стратегія полягає у тому, щоб називати числа, які роблять суму кратною п'яти (тобто щоб сума чисел була 5, 10, 15 і так далі).

**Деталі**

Важливо, які числа може назвати гравець і яку суму необхідно набрати для перемоги. Наприклад, якщо змінити умову так, що переможцем стає гравець, після ходу якого сума дорівнює 51, то при правильній стратегії виграє першій учасник (який називає число першим), а сама виграшна стратегія полягатиме у тому, щоб сума приймала значення 1, 6, 11, 16, і так далі до 46 та 51 (тобто числа, які при діленні на 5 дають у залишку 1). Особливо варто відмітити, що перший гравець на самому початку повинен назвати число 1. Якщо він назве будь яке інше число, то тим самим він передає можливість реалізувати виграшну стратегію своєму супернику.

Наступна задача така. Існує 7 зовні однакових монет. Відомо, що серед них може виявитись фальшива. Фальшива монета легша за справжню. Необхідно за допомогою терезів з чашами за мінімальну кількість зважувань визначити, чи є серед монет фальшива, і якщо є, то необхідно її визначити.

Алгоритм, або стратегія, що дозволяє розв'язати задачу, тепер мають точки розгалуження – на деяких етапах наступні дії залежать від результату, отриманого на попередньому кроці. Зокрема, для пошуку фальшивої монети ми можемо діяти таким чином. Відкладаємо одну з монет у бік, а ті 6 монет, що

залишились, розділяємо на дві групи по 3 монети в кожній і зважуємо їх на терезах. Це перше зважування. Тут можливі два варіанти.

[1] Вага трьох монет на одній чаші терезів така сама як вага трьох монет на іншій чаші терезів. Це означає, що всі 6 монет – справжні. Але фальшивою може виявитися та монета, яку ми відклали з самого початку. Щоб перевірити цю монету, ми кладемо її на одну чашу терезів, а на іншу чашу кладемо справжню монету (будь яка з 6 перевірених на попередньому етапі монет). Це друге зважування. Якщо чаші терезів врівноважені – відкладена монета теж справжня. Якщо ні – то ні.

[2] Одна з чаш терезів з трьома монетами переважує іншу чашу терезів. Це означає, що серед 3 монет на тій чаші, що має меншу вагу, є фальшива монета. Тому далі діємо так. Кладемо на кожну чашу терезів по одній монеті з тих 3, серед яких є фальшива. Це друге зважування. Якщо чаші терезів урівноважені, то фальшивою є та монета з 3, що залишилась (тобто монета, яку не зважували). Якщо чаші терезів не урівноважені, то фальшива монета знаходиться на тій чаші, де менша вага.

Таким чином, для визначення фальшивої монети або встановлення того факту, що фальшивої монети немає, достатньо виконати два зважування.

Ще одна невелика задача формулюється так. Є дві ємності – одна об'ємом 3 літри, інша об'ємом 5 літрів. За допомогою цих ємностей необхідно набрати рівно 1 літр води (виходимо з того, що ресурси води необмежені). Алгоритм дій може бути таким:

[1] Набираємо 3 літри води в меншу ємність.

[2] Переливаємо воду з меншої ємності у більшу ємність. Тепер у більшій п'ятилітровій ємності 3 літри води.

[3] Знову набираємо 3 літри води у меншу ємність.

[4] Доливаємо воду з меншої ємності у більшу ємність так, щоб остання була заповнена повністю. Оскільки там було 3 літри води, а об'єм ємності 5 літрів, то буде долито 2 літри. Відповідно, в меншій ємності залишиться 1 літр води.

Таких задач, насправді, дуже багато, частина з них більшості знайомі ще з дитинства – як, наприклад, задача про перевезення через річку вовка, кози та капусти.

**Деталі**

Про всяк випадок, нагадаємо умову головоломки та її розв'язок. Отже, через річку човняр перевозить вовка, козу та капусту. У човен можна взяти лише одного "пасажира", козу не можна залишати з капустою, а вовка – з козою.
Розв'язок такий. Спочатку на протилежний берег перевозиться коза. Потім на протилежний берег перевозиться, наприклад, вовк (капуста), а назад забирається коза. Потім на протилежний берег перевозиться капуста (вовк), після чого на протилежний берег перевозиться коза.

Список можна продовжувати, і він постійно поповнюється новими задачами. Але ці задачі не є нашою головною метою, хоча як ілюстрація вони цілком показові.

# Обернені задачі

*Іноді за одною лише відповіддю можна здогадатися, яке було запитання.*

Далі обговоримо клас задач, які умовно можна було би назвати "оберненими". Причому цей термін стосується не тільки і не стільки алгоритмічних задач. Обернені задачі зустрічаються тут і там, у математиці, фізиці, соціології, та багатьох інших наукових областях і сферах людської діяльності. Про що йдеться? Пояснимо на прикладах.

Почнемо з операцій, які є протилежними до обернених. Їх називають "прямими". Для таких операцій існує набір інструкцій або правил, які, власне, і виконуються в контексті цієї операції. Наприклад, в математиці є така операція, як розрахунок похідної. Для цієї операції існує чітке визначення, дотримуючись якого можна розрахувати похідну практично для будь якої функції. В теорії ймовірностей є таке поняття, як математичне сподівання. Якщо відомий закон розподілу випадкової величини, то, застосовуючи правило розрахунку математичного сподівання, отримаємо потрібний результат. У фізиці, знаючи потенціал взаємодії частинок, можемо розрахувати, як розсіюється потік частинок. Існують і більш "приземлені" приклади. Скажімо, рецепт борщу або іншої страви. Ми зазвичай сприймаємо все це в "кулінарному" контексті, але якщо дивитися в корінь, то рецепт являє собою алгоритм, виконання якого приводить до певного результату.

**До уваги**

Алгоритм дій для розв'язання прямої задачі зазвичай досить універсальний і дозволяє отримати потрібний результат у будь-якій або практично будь-якій ситуації. Тобто в основному до прямих належать задачі, які розв'язуються без особливої інтриги (хоча вони можуть бути й складними з технічної точки зору).

З оберненими операціями ситуація принципово інша. Необхідність у виконанні таких операцій виникає, якщо відомий результат деякої прямої операції й необхідно відновити початкову "картину". Зазвичай для обернених операцій не існує простих рецептів, і результат далеко не завжди гарантований. Іноді задача має декілька розв'язків або не має їх зовсім. На загал, обернені операції нетривіальні, ексклюзивні і вимагають наявності досвіду, ерудиції та везіння.

Якщо повернутися до згаданого раніше списку прямих задач, то для них існують і обернені. Так, оберненою для операції розрахунку похідної є операція з розрахунку інтеграла. У цьому випадку відома похідна функція, а розрахувати необхідно первісну – тобто таку функцію, похідна від якої дорівнює вказаній відомій функції.

Якщо відома реалізація випадкової величини, то за цими результатами можна оцінити закон розподілу цієї випадкової величини. Це ще одна обернена задача, як і більшість задач статистики. За картиною розсіяння частинок відновлюють