пункті 5, тобто розміщувати посилання на матеріали конференції на інших сайтах. При розміщуванні посилань на «чужих» сайтах слід віддавати перевагу форумам на відомих, авторитетних сайтах. Адже посилання, розміщені на «сумнівних» сайтах, у Webometrics не враховуються. Авторитетними Вузівськими сайтами можна вважати web-сайти, що увійшли в опублікований рейтинг Webometrics (перші 4000 web-сайтів Університетів).

Список використаних джерел

1. Webometrics ranking of world's universities [Електронний ресурс]. – Режим доступу: http://www.webometrics.info/rank_by_country.asp? country=ua

2. Вебометричний рейтинг університетів світу [Електронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/Вебометричний_ рейтинг_університетів_світу.

3. Франчук В.М. Використання CMS Joomla! та LCMS MOODLE у ВНЗ. Тези міжнародної науково-практичної конференції "FOSS LVIV-2011": Львів, 1-6 лютого 2011р. – Львів: ЛНУ імені Івана Франка, 2011. – С. 165-167.

4. Кудін А.П., Кархут В.Я., Франчук В.М. Інформаційно-комунікаційні технології та управління діяльністю вищого навчального закладу: освітній портал. Збірник наукових праць Кам'янець-Подільського національного університету імені Івана Огієнка. Серія педагогічна / [редкол.: П.С.Атаманчук (голова, наук. ред.) та ін.]. – Кам'янець-Подільський: Кам'янець-Подільський національний університет імені Івана Огієнка, 2010. – Вип. 16: Формування професійних компетентностей майбутніх вчителів фізико-технологічного профілю в умовах євроінтеграції. С. 26-29.

Сейдаметова З.С.
Доктор педагогічних наук, професор
Кримський інженерно-педагогічний університет

Рандомізовані алгоритми та імовірнісний аналіз: як вчити майбутніх програмістів

Рандомізація і випадковість зустрічаються в різних ситуаціях в людському житті. Добре розроблений математичний апарат для їх вивчення, наприклад, описаний у роботах в підручнику з теорії ймовірностей і математичної статистики [5] в передмові наводиться невеликий, але ємний історичний екскурс в історію початку використання математичного підходу до вивчення різних випадкових явищ. У відповідності з цим екскурсом [5] факти вивчення випадкових подій були відомі ще до нашої ери в Стародавньому Китаї, Древньому Римі, Древній Греції [5, с. 3]. У передмові до підручника [5] зазначено, що передісторія теорії ймовірностей закінчується в XVI столітті і з середини XVII століття роботами Блеза Паскаля, П'єра Ферма, Християна Гюйгенса починається сама історія цього напрямку математики [5, с. 3-5]. Великий вплив на розвиток теорії ймовірностей і математичну статистику мали роботи українських та російських вчених дорадянського часу – В.Я. Буняковського, М.В. Остроградського, П.Л. Чебишова; а також радянських вчених – А.М. Колмогорова, А.А. Маркова, О.М. Ляпунова, Б.В. Гнеденка і ін [5]. Варто відзначити, що радянська школа математиків, що спеціалізуються в теорії ймовірностей і математичній статистиці, була однією з кращих у світі. Роботи [1] – [6] стимулювали мене звернути увагу на використання імовірнісного аналізу в підготовці майбутніх інженерів-програмістів, зокрема, в теорії алгоритмів, і на введення цих тем в зміст навчальної дисципліни «Алгоритми і структури даних».

Імовірнісний аналіз і рандомізація цікавить багатьох дослідників. У статтях Я. Кортсартса [7], М. Гудріча [8], Р. Монтавані і П. Рагхавана [9] розглядаються питання включення в навчальні матеріали університетів рандомізованих алгоритмів сортування. У статтях [10] – [11] описана методика застосування рандомізованих алгоритмів в задачах тестування працездатності програмних додатків, а також в мережевих програмах. Апарат дослідження рандомізованих алгоритмів представлений і в класичному підручнику з алгоритмів і структур даних Т. Кормена, Ч. Лейзерсон, Р. Ривеста, К. Штайна [12], [13].

В алгоритмічних задачах імовірнісний аналіз застосовується для розрахунку продуктивності алгоритму, в якому зроблено припущення про випадковий розподіл вхідних даних; також імовірнісний аналіз використовується як техніка розробки ефективних алгоритмів.

Алгоритм вважається рандомізованим, якщо його функціонування пов'язане з випадковим набором вхідних даних або значеннями, які визначаються за допомогою генератора випадкових чисел.

Що саме треба знати студентам з теорії ймовірностей та математичної статистики при вивченні рандомізованих алгоритмів?

Викладач в рамках навчальної дисципліни «Алгоритми і структури даних» повинен нагадати студентам про основні аксіоми теорії ймовірностей; у разі необхідності, обговорити випадкові перестановки. Найбільш часто використовувані висловлювання в статтях про рандомізовані

алгоритми – «переставити елементи випадковим чином». У термінах теорії ймовірностей це означає наступне: створити ймовірнісний простір з усіма можливими перестановками елементів множини $V=\{\mathbf{v_1}, \ldots, \mathbf{v_n}\}$ в якості базових подій у просторі подій. При цьому вважається, що імовірність кожної події дорівнює 1/n!. Зрозуміло, що результуючий простір подій підпорядковується правилам теорії ймовірностей.

Оскільки тут обговорюються алгоритми, то слід зауважити, що наведений вище хід міркувань не може бути використаний на практиці, оскільки неможливо опрацювати експонентний розмір імовірнісного простору. Однак варто показати студентам, як можна легко сконструювати випадкові перестановки, вибираючи випадковим чином перший елемент, потім вибираючи випадковим чином другий елемент, і так до останнього. У результаті отримуються випадкові перестановки. Навчання дисципліни «Алгоритми і структури даних» дозволяє нагадати студентам ще раз основні правила теорії ймовірностей. Дійсно, нехай $\Pi=\{\mathbf{v_1},\ldots,\mathbf{v_n}\}$ деяка перестановка $\boldsymbol{V}=\{\mathbf{v_1},\ldots,\mathbf{v_n}\}$. Припустимо, що Π являє собою результуючу перестановку процесу ітеративного випадкового вибору, $\mathbf{v_{i1}}$ повинен бути вибраний першим, потім, з врахуванням, що $\mathbf{v_{i1}}$ обраний першим, вибирається другий – $\mathbf{v_{i2}}$, і т.д. Нехай подія $\mathbf{A_j}$ = {елемент $\mathbf{v_{ij}}$ обраний j-им}. Ймовірність події $\mathbf{A_1 \cap A_2 \cap \ldots \cap A_n}$ буде наступною:

$$\mathbf{Pr(A_1 \cap A_2 \cap \ldots \cap A_n) = Pr(A_1) \cdot Pr(A_2 \mid A_1) \ldots Pr(A_n \mid A_1 \cap A_2 \cap \ldots \cap A_{n-1});}$$

$$\mathbf{Pr(A_1 \cap A_2 \cap \ldots \cap A_n) = 1/(n \cdot (n-1) \ldots 2 \cdot 1) = 1/n!.}$$

Навіть якщо викладач не має можливості приділити достатньо часу на аналіз деталей, наведених вище, зауважимо, що описаний вище підхід формує у студента інтуїтивне уявлення про поняття випадкової перестановки.

Також варто звернути увагу студентів і на більш загальні питання. Нехай Π являє собою випадкову перестановку елементів множини $\boldsymbol{V}=\{\mathbf{v_1},\ldots,\mathbf{v_n}\}$ и нехай $\mathbf{P} \subseteq \boldsymbol{V}$, потужність $|\mathbf{P}|=\mathbf{k}$. Будемо вважати, що Π не порушує P, якщо $v_i, v_j \in P$ $(i<j)$ випливає, що v_i зустрінеться в Π раніше, ніж v_j. Це означає, що Π не змінює порядок елементів в P. Нехай O = {Π не порушує P}. Не складно вирахувати ймовірність цієї події Pr(O) за допомогою біноміальних коефіцієнтів. Розглянемо випадковий процес створення Π. Без втрати спільності будемо вважати, що $\mathbf{P}=\{\mathbf{v_1}, \mathbf{v_2},\ldots,\mathbf{v_k}\}$. Перестановка дає в $\boldsymbol{V} \setminus \mathbf{P}=\{\mathbf{v_{k+1}}, \mathbf{v_{k+2}},\ldots,\mathbf{v_n}\}$ такий же порядок, як і в P. Однак, зауважимо, що порядок в P не залежить ніяким чином від $V \setminus P$. Ми можемо проігнорувати не цікавий вибір під час побудови перестановки Π, вибір елемента $V \setminus P$. Отже, ми обмежуємося вибором в P. Питання: що є ймовірністю події, що елементи P зберігають їх первісний порядок у випадковій перестановці? Відповідь: 1/(n-k)!

Слід звернути увагу студентів на те, що рандомізація в алгоритмах може з'являтися за рахунок використання генератора випадкових чисел. Зазвичай для цього використовується процедура Random(a,b), що повертає ціле число, яке належить інтервалу (a,b) і рівномірно розподілене в цьому інтервалі.

При аналізі рандомізованих алгоритмів, зокрема, алгоритму швидкого сортування Randomized-Quicksort, використовується індикаторна випадкова величина I. Нехай A – деяке подія, тоді $I\{A\}$ визначається наступним чином:

$$I\{A\} = \begin{cases} 1, & \text{подія } A \text{ відбулася,} \\ 0, & \text{подія } A \text{ не відбулася.} \end{cases}$$

Продуктивність (час роботи) T_n алгоритмів безпосередньо залежить від вхідного масиву, а оскільки вхідних варіантів може бути багато (наприклад, в n-елементному масиві – n!), то T_n буде залежати від усіх початкових варіантів. У статті [7] сказано, що в кінці 1970-х були введені в розгляд рандомізовані алгоритми, більш прості у використанні і з деякою ймовірністю швидші від звичайних алгоритмів. Наприклад, рандомізований алгоритм швидкого сортування Randomized-Quicksort є й більш простий, і приводить до результату O(n logn) час при будь-якому вході, в той час як детермінований алгоритм швидкого сортування Quicksort працює за $\Theta(n^2)$ час в гіршому випадку ([12, с. 122-130, 130-146], [13, с. 141-171], [14, с. 707-794]).

Опис алгоритму швидкого сортування поданий в псевдокоді в таблиці 1, там же наведені коментарі та інваріант алгоритму. Алгоритм Quicksort містить поділ масиву щодо опорного елемента, а також саме сортування. Для цього алгоритму час роботи залежить від поділу. Якщо поділ збалансовано, то час роботи T_n=O(n logn). Доданням в алгоритм швидкого сортування рандомізації можна отримати час роботи як для збалансованого поділу.

Рандомізований алгоритм Randomized-Quicksort представлений в таблиці 2. Рандомізація здійснюється за допомогою методу випадкової вибірки: опорний елемент вибирається в підмасиві A[p..q] випадковим чином, що забезпечує рівну імовірність опинитися опорним будь-якого елемента подмасиву A[p..q]. Математичне сподівавання часу роботи рандомізованого алгоритму швидкого сортування різних за величиною елементів O(n logn).

Таблиця 1

Алгоритм швидкого сортування Quicksort		
PARTITION(*A*,*p*,*q*) *x*←*A*[*p*] *i*←*p* **for** *j*←*p*+1 **to** *q* **do if** *A*[*j*]≤*x* **then** *i*←*i*+1 exchange *A*[*i*]↔*A*[*j*] exchange *A*[*p*]↔*A*[*i*] **return** i	▷ *A*[*p* . . *q*] ▷ вибір опорного елемента **Інваріант** 	Час роботи O(n) для n-елементного входу
QUICKSORT(*A*,*p*,*r*) **if** *p*<*r* **then** *q*←PARTITION(*A*,*p*,*r*) QUICKSORT(*A*,*p*,*q-1*) QUICKSORT(*A*,*q+1*,*r*)	▷ *A*[*p* . . *r*]	Для сортування всього масиву *A*[*1..n*] викликається: QUICKSORT(*A*,*1*,*n*) Час роботы в гіршому випадку $\Theta(n^2)$ для n-елементного входу; в кращому випадку Θ(n logn)

Таблиця 2

Рандомізований алгоритм швидкого сортування Randomized-Quicksort	
RANDOMIZED-PARTITION(*A*,*p*,*q*) *i*←Random(*p*,*q)* exchange *A*[*p*]↔*A*[*i*] **return** PARTITION(*A*,*p*,*q*)	
RANDOMIZED-QUICKSORT(*A*,*p*,*r*) **if** *p*<*r* **then** *q*←RANDOMIZED-PARTITION(*A*,*p*,*r*) RANDOMIZED-QUICKSORT(*A*,*p*,*q-1*) RANDOMIZED-QUICKSORT(*A*,*q+1*,*r*)	Час роботи в середньому випадку – Θ(n logn)

Одним із класичних рандомізованих алгоритмів є метод Монте-Карло. Уявлення про метод Монте-Карло дано в навчальних посібниках М.І. Жалдака [2, с. 148-153], [5, с. 466-470]. Метод Монте-Карло використовується у фізиці, хімії, математиці, економіці, в теоріях оптимізації та управління. Фактично, це не один метод, а група чисельних методів в задачах, в яких необхідно визначити імовірнісні характеристики стохастичних процесів. Для ілюстрації застосування імовірнісного аналізу необхідно включити в навчальні програми дисциплін підготовки майбутніх програмістів розгляд методу Монте-Карло. У статті [7, с. 197-198] Ян Кортсартс вважає, що алгоритм Монте-Карло представляє собою дуже хорошу ілюстрацію рандомізованого алгоритму. На цьому алгоритмі можна показати ефективність імовірнісного аналізу та рандомізації. Імовірнісний аналіз часу роботи рандомізованих алгоритмів дозволяє викладачеві продемонструвати практичне застосування теорії ймовірностей і математичної статистики.

Важливим інструментом реалізації цілого ряду програмних засобів є рандомізація як ефективний спосіб побудови алгоритмів. Для педагогічних цілей може бути зручна додана нижче класифікація рандомізованих алгоритмів. Ця класифікація залежить від способу побудови рандомізованого алгоритму, а також використовуваного імовірнісного аналізу.

1. Foiling an Adversary. «Долаючий суперника». – У класичному аналізі гірших випадків детермінованих алгоритмів термін «adversary» використовується при визначенні нижньої межі часу роботи. Вхідні дані можуть бути різним для кожного детермінованого алгоритму. У теоретичній інтерпретації теорії ігор можна розглянути рандомізований алгоритм як випадковий вибір із множини детермінованих алгоритмів. Найчастіше такі задачі зустрічаються там, де необхідно вибрати виграшну стратегію. У статтях [10, с. 34] та [15, с. 167] наведені приклади завдань, які розв'язуються за допомогою цієї техніки.

2. Random Sampling. «Випадкова вибірка». – Ідея використання в рандомізованих алгоритмах взята з соціології: невелика випадкова вибірка я, обрана з генеральної сукупності. Наприклад,в рандомізованому алгоритмі Флойда і Рівеста [16] про знаходження k-го за значенням елемента з n-елементної множини використовується цей підхід що дозволяє зменшити кількість порівнянь до 1.5n + o(n). Відзначимо, що за наявними детермінованими алгоритмами виконується такий вибір за 2n порівнянь.
3. Abundance of Witnesses. «Велика кількість свідків». – Часто в обчислювальних завданнях може бути прискорений пошук свідків або підтвердження того, що висунута гіпотеза може бути ефективно підтверджена. Наприклад, щоб показати, що число не є простим, достатньо знайти будь-який нетривіальний дільник цього числа. Ця техніка часто використовується в теорії чисел, завданнях тестування. Відомі рандомізовані алгоритми Соловея і Страссена (1978) [12], за якими результат отримується за поліноміальний час.
4. Fingerprinting and Hashing. «Сліди і хешування». – Використовується у випадках, коли необхідно визначити ідентичність двох відбитків, в програмах (pattern-matching application), за допомогою яких аналізують об'єкти на наявність в них заданих шаблонів. Також ця техніка використовується в хешуванні, де елементи множини S, де $S \subseteq U$ (U – універсальна множина) майже рівномірно розподіляються в слоти хеш-таблиці (хеш-таблиця – структура даних, яка використовується для реалізації словників, і математичне сподівання часу пошуку елемента в ній становить O(1)) [12], [13]. При хешуванні використовується припущення, що для кожного ключа в якості послідовності досліджень різноймовірні всі m! перестановок множини {0, 1, …, m-1}. Для обчислення послідовності досліджень в випадку відкритої адресації можна використовувати метод лінійного, квадратичного досліджень і подвійне хешування [13]. Рандомізовані алгоритми визначення ідентичності (random fingerprint) використовуються в генерації псевдовипадкових числах і теорії складності обчислень.
5. Random Reordering. «Випадкове переупорядкування». – Великий клас задач має властивість, що відносно наївний алгоритм A може бути реалізований за допомогою добре організованого вхідного потоку даних, поданих у випадковому порядку. Хоча алгоритм A може мати дуже поганий випадок реалізації, випадкове переупорядкування вхідних даних забезпечує малу ймовірність то, що цей вхід буде наявний в упорядкуванні. Один з прикладів цього явища може бути пов'язаний з алгоритмом швидкого сортування Quicksort [12]. Підхід випадкового переупорядкування успішно застосовується в структурах даних і обчислювальній геометрії. Огляд рандомізованих геометричних алгоритмів можна дати при вивченні дисципліни «Обчислювальна геометрія».
6. Load Balancing. «Збалансоване навантаження». – У випадках, коли доводиться вибирати між різними ресурсами (наприклад, посилання в мережевих комунікаціях або коли виконується завдання з використанням паралельних процесорів), рандомізація може бути використана для рівномірного розподілу навантаження серед ресурсів. Ця парадигма має застосування у паралельному і розподіленому комп’ютінгові, де рішення про утилізацію ресурсів має бути прийняте локально без залучення глобальних знань. Такі алгоритми можуть зустрічатися при вирішенні мережевих проблем [11].
7. Rapidly Mixing Markov Chains. «Швидко зміщуваючи ланцюги Маркова». – У завданнях підрахунку метою є визначення числа комбінаторних об'єктів зі специфічними властивостями. Коли простір об'єктів великий, кращим рішенням може бути використання підходу Монте-Карло, за яким аналізуються тільки об'єкти, які цікавлять в випадкових вибірках повного простору. У ряді випадків може бути показано, що вибір універсальної випадкової вибірки є такою ж важкою задачею, як і завдання підрахунку. Успішним методом для генерації майже універсальних випадкових вибірок є ланцюг Маркова. Цей метод успішно використовується в алгоритмах статистичної фізики, а також в алгоритмах для оцінювання число ідеальних порівнянь в графах. У статтях [7], [9] дані численні приклади застосування цього методу в задачах теорії графів.
8. Isolation and Symmetry Breaking. «Порушення ізолювання і симетрії». – У обчисленнях, що зустрічаються в асинхронних розподілених процесорах, часто необхідно в багатозадачному середовищі шукати вихід із ситуації взаємних блокувань або симетрії і зробити спільний вибір. Рандомізація є потужним інструментом, що дає можливість уникати випадків взаємних блокувань. Такий підхід до побудови алгоритму – протокол координації вибору (або розподілений консенсус) – наведений у статті Майкла Рабіна [17].
9. Probabilistic Methods and Existence Proofs. «Імовірнісні методи і доведення існування». – за допомогою імовірнісних методів можна встановити існування специфічних типів комбінаторних об'єктів за умови, що випадкові об'єкти з відповідно заданих просторів мають наперед задані властивості з ненульовою ймовірністю. Цей метод зазвичай використовується для гарантованості

існування алгоритму розв'язування задачі. Це дозволяє вважати, що алгоритм існує у випадках, коли невідомо, як він виглядає, і як його побудувати [12].

Аналіз рандомізованих алгоритмів проводиться на основі імовірнісного аналізу або аналізу випадку роботи за алгоритмом в середньому (average-case) в припущенні, що всі набори вхідних параметрів однакового розміру зустрічаються з однією і тією самою імовірністю.

З досвіду навчання дисципліни «Алгоритми і структури даних» можна стверджувати, що тематика рандомізованих алгоритмів викликає у студентів достатній інтерес, дозволяє згадати вивчений раніше матеріал з теорії ймовірностей та математичної статистики. Крім того, виконання завдань, в яких зустрічаються рандомізовані алгоритми, дозволяє удосконалювати програмістський досвід в рамках студентських проектів. Наприклад, це можуть бути завдання перемножування ланцюжка матриць [13], проблеми побудови оптимального бінарного дерева пошуку та ін.

Література

1. Жалдак М.І. Використання міжпредметних зв'язків та аналогій у процесі навчання теорії ймовірностей майбутніх учителів математики / М.І. Жалдак, Г.О. Михалін, І.М. Біляй // Науковий часопис НПУ ім. М.П.Драгоманова. Серія №2. Комп'ютерно-орієнтовані системи навчання. Зб. наукових праць. – К.: НПУ ім. М.П. Драгоманова, 2012. – Випуск 12(19). – С. 3-15.
2. Жалдак М.И. Теория вероятностей с элементами информатики: Практикум: Учеб. Пособие / Под общ. ред. М.И. Ядренко / М.И. Жалдак, А.Н. Квитко. – К.: Выща школа, 1989. – 263 с.
3. Жалдак М.І. Теорія ймовірностей і математична статистика з елементами інформаційної технології / М.І. Жалдак, Н.М. Кузьміна, С.Ю. Берлінська. – К.: Вища школа, 1996. – 352 с.
4. Жалдак М.І. Елементи стохастики з комп'ютерною підтримкою. Посібник для вчителів / М.І. Жалдак, Г.О. Михалін. – К.: Шкільний світ, 2002. – 120 с.
5. Жалдак М.І. Теорія ймовірностей і математична статистика. Підручник для студентів фізико-математичних спеціальностей пед. університетів. – Вид. 2, перероб. і доп. / М.І. Жалдак, Н.М. Кузьміна, Г.О. Михалін. – Полтава: «Довкілля-К», 2009. – 500 с.
6. Жалдак М.І. Збірник задач і вправ з теорії ймовірностей і математичної статистики. Навч. посібник для студентів фіз.-мат. спец-тей пед. університетів / М.І. Жалдак, Н.М. Кузьміна, Г.О. Михалін. – К.: НПУ імені М.П. Драгоманова, 2009. – 610 с.
7. Kortsarts Y. How (and why) to introduce Monte Carlo randomized algorithms into a basic algorithms course? / Yana Kortsarts // Journal of Computing Sciences in Colleges, vol. 21, issue 2, Dec. 2005. – USA: Consortium for Computing Sciences in Colleges, 2005. – P. 195-203.
8. Goodrich M.T. Randomized Shellsort: A Simple Data-Oblivious Sorting Algorithm / Michael T. Goodrich // Journal of the ACM (JACM), vol. 58, issue 6, Dec. 2011, Article No. 27. – 26 p.
9. Motwani R. Randomized Algorithms / R. Motwani, P. Raghavan // ACM Computing Surveys, vol. 28, No. 1, March 1996. – USA: CRC Press, 1996. – P. 33–37.
10. Arcuri A. A practical guide for using statistical tests to assess randomized algorithms in software engineering / A. Arcuri, L. Briand // Proceedings of the 33rd International Conference on Software Engineering. – New York, NY, USA: ACM, 2011. – P. 1-10.
11. Khabbazian M. Time-Efficient Randomized Multiple-Message Broadcast in Radio Networks / M. Khabbazian, D. R. Kowalski // Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing. – New York, NY, USA: ACM, 2011. – P. 373-380.
12. Cormen T.H. Introduction to Algorithms / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein; Third Edition. – The MIT Press, 2009. – 1312 p.
13. Кормен Т.Х. Алгоритмы: построение и анализ, 2-е издание. : Пер. с англ. / Т.Х. Кормен, Ч.И. Лейзерсон, Р.Л. Ривест, К. Штайн. – М.: Изд. дом «Вильямс», 2005. – 1296 с.
14. Randomized algorithms / In Kleinberg J. Algorithm Design / J. Kleinberg, É. Tardos. – USA: Addison-Wesley, 2005. – 864 p.
15. Karp R.M. An introduction to randomized algorithms / Richard M. Karp // Discrete Applied Mathematics, vol. 34, issue 1-3, Nov. 1991. – P. 165-201.
16. Floyd R.W. Expected time bounds for selection / R.W. Floyd, R.L. Rivest // Commun. ACM, vol. 18, issue 3, March 1975. – P. 165-172.
17. Rabin M.O. The choice coordination problem / Michel O. Rabin // Acta Informatica, vol. 17, 1982. – P. 121-134.