УДК 378

ТЕОРЕТИКО-ВЕРОЯТНОСТНЫЙ ПОДХОД ПРИ ОБУЧЕНИИ СТУДЕНТОВ АНАЛИЗУ СОРТИРУЮЩИХ АЛГОРИТМОВ

Д. В. Филимонов

Белорусский государственный университет,
пр. Независимости, 4, 2203030, г. Минск, Беларусь, dzfilimonau@gmail.com

В рамках обучения дисциплинам «Построение и анализ алгоритмов» и «Алгоритмы и структуры данных» предложен междисциплинарный подход, значительная роль в описываемых задачах отведена применению основополагающих сведений из теории вероятностей и математической статистики.

Ключевые слова: качество образования; анализ алгоритмов; теория вероятностей; математическая статистика.

Введение

Вопросам организации методов индексирования и анализа информации отведена одна из ключевых ролей в компьютерных науках, так как не представляется возможным решать многие тривиальные задачи, не обладая данными, содержащимися в упорядоченной последовательности.

Эффективность алгоритмов, использующихся с целью хранения и обработки информации, может быть оценена, исходя из расчета количества операций, требующихся для выполнения следующих действий:

– добавление нового элемента в структуру;
– достижение элемента, имеющего заданный индекс или ключ в структуре;
– поиск позиции элемента по значению;
– выполнение вспомогательных операций для завершения шага алгоритма.

При обучении методам программирования и алгоритмике, тем не менее, часто опускается вопрос об источнике данных, то есть законе, по которому информация возникает некоторым естественным образом – поэтому на модельных задачах часто предлагаются значения, сгенерированные с помощью датчика случайных чисел. Такое упрощение, конечно, позволяет быстро приступить к реализации некоего алгоритма, познакомить с базовой концепцией и идеей, стоящей за подходом.

Однако, после быстрого ознакомления с основами, дальнейшие исследования, предлагаемые студентам, обязаны преподносить более фундаментальные математические сведения, которыми ранее требовалось пренебречь: настоящие данные, с которыми в дальнейшем могут работать молодые специалисты, часто наделены характерными особенностями, которые возникают в условиях реальных процессов. В свою очередь, нельзя и не говорить о том, что эти данные, даже если не являются сгенерированными (как, например, датасеты, сведения которых отражают реальные наблюдения и служат прекрасным средством для обучения, так как результаты их исследований давно известны и опубликованы), то все так же могут быть рассмотрены как случайные.

К моменту более детального изучения алгоритмов студенты либо уже осведомлены о различных типах случайных величин, либо получают представления о них в процессе изучения теории вероятностей. Большой ошибкой будет рассказать подробности технического характера, касающиеся реализации алгоритмов, упорядочивающих (сортирующих) всевозможные данные, не затронув математических предпосылок для подобного рода вопросов. Таким образом, важность изучения данных понятий именно в курсе алгоритмов обуславливается тем, что

данная дисциплина, по сути, является одной из немногих, что естественным образом связывает компьютерные науки и теорию вероятностей, тем самым выполняя роль связующего звена, ведущего к анализу данных.

Корректность и применимость теоретико-вероятностного подхода

Программное обеспечение позволяет с легкостью генерировать сотни тысяч случайных значений, что и позволяет во многих случаях пренебречь значительным числом законов распределения. Тем не менее, характер, который носят именно неучтенные законы, обладает определенной ценностью именно в задачах исследовательского рода, направленных на оптимизацию алгоритмов сортировок.

Так как сами сортируемые наборы могут и не иметь десятков тысяч элементов, а процессы, на основе которых были получены данные, приводят к зависимости между величинами, условия центральных предельных теорем могут не выполняться. Следовательно, приближенное определение законов распределения может повысить эффективность алгоритмов. То же самое касается и категории, использующей «эталонный» (барьерный, pivotal) элемент – в терминологии теории вероятностей ему будет соответствовать одна из метрик (медиана, среднее арифметическое и т.д.). Исходя из данных заключений, можно выделить ряд математических понятий, которые необходимо ввести в рассмотрение при изучении данной темы, чтобы преподнести обучающимся более полное понимание явлений и причин, влияющих на внесение модификаций в предполагаемые решения (такие как выбор определенного сортирующего алгоритма или предложение собственного с учетом имеющихся данных).

Важно также понимать, что исследуемые данные на практике могут обладать свойствами, затрудняющими работу конкретного алгоритма. Этот факт дополнительно показывает, что не существует универсального алгоритма сортировки, и требуется комбинировать различные подходы, чтобы приходить к приемлемому времени работы и компенсировать недостатки самих алгоритмов. Проблемная задача, предлагающая обучающимся обработать данные такого рода, возникает естественным образом. Ее формулировка достаточно проста, а сама задача присутствует в любом курсе, посвященном алгоритмам:

Задача 0 («Эталон», Benchmark). Предлагается сравнить все изученные ранее сортировки, определив их затратность на массивах объемом 50 тыс. и 250 тыс. элементов. Массивы заполняются двумя способами: случайными равномерно распределенными (стандарт большинства языков) значениями от 1 до размерности массива; случайными значениями из промежутка от 1 до максимального целого значения, которое может поддерживать архитектура данного компьютера.

Такая задача возникает уже при первом знакомстве с сортирующими алгоритмами во время обучения дисциплине «Методы программирования» на первом курсе. К моменту более детального изучения сортировок Задача 0 является также своеобразным «бенчмарком» не только для исследуемых алгоритмов, но и для группы обучающихся: ею преподаватель может воспользоваться как инструментом диагностики, чтобы определить дальнейшие перспективы по изложению курса, отталкиваясь от уровня студентов. Ключевой подзадачей в данном случае является сравнение: формулировка, кроющаяся здесь, достаточно размыта, о пользе чего будет сказано ниже.

Наиболее важными результатами, которые могут быть выявлены при корректном проведении диагностики, являются:

– определение уровня математической подготовки обучающихся;

– выявление студентов, обладающих более развитым умением осуществления логического анализа. Обучающиеся с хорошей математической подготовкой трактуют «сравнение», придерживаясь составления шкалы «качества» данных. Обычно в понятие качества вкладывается разделение результатов выполнения на наибольшее время выполнения, наименьшее и результат, усредненный по остальным. При этом стоит отметить, что студенты-«математики» и

студенты-«аналитики», вообще говоря, могут прибегать к обоим методам, но тенденция, о которой здесь идет речь, сохраняется;

– оценка общего уровня подготовки студентов как программистов. Под данным понятием имеется в виду совокупность ЗУН и компетенций, позволяющих обучающимся быстро и эффективно описывать алгоритмы с учетом технических особенностей сред разработки и используемых языков.

Таким образом, преподаватель может оценить, насколько целесообразно в конкретной группе проводить занятия в классической форме или прибегать к другим подходам, например, Agile.

В случае, если проектная модель обучения применима в данной группе, каждая выделенная подгруппа должна получить собственные подзадачи в рамках общего проекта, помимо контроля качества разработки решения с позиций своего «отдела». Важную роль здесь и играют математические (в случае сортировок – чаще упомянутые теоретико-вероятностные) сведения, преподносимые на данном этапе обучения как факты, позволяющие углубить понимание реальных процессов и их математических моделей.

Постановка задач

Сообщить и обобщить все ранее упомянутые данные, учитывая специфику специальностей, можно последовательно, выстраивания «цепочки» взаимосвязанных задач, решение которых иллюстрирует важность учета математических соотношений и свойств, зачастую играющих определяющую роль в оптимальности алгоритма.

Возможность такого подхода обоснована характером задач, которые могут быть предложены к изучению: так, даже первый блок алгоритмических курсов, посвященный сортировкам, позволяет предложить задачи, решения которых могут быть неочевидными для студентов по причине различий в распределениях сортируемых величин. Приведем пример цепочки таких междисциплинарных задач:

Задача 1. Гибридной называется сортировка, которая упорядочивает набор данных, выбирая разделительный элемент для набора и затем сортируя элементы до него одним алгоритмом, а после – другим. Стоит задача: реализовав несколько гибридных сортировок, основанных на наиболее известных алгоритмах, отсортировать два массива вещественных чисел, предложенных преподавателем, и подсчитать количество элементарных операций (операций, которые компьютер выполняет за одно действие), произведенных реализациями.

Сами сортировки и отдельные их свойства изучаются на 1-2 курсе, в то время как дисциплины по анализу алгоритмов в целом преподаются на 3 курсе. Следовательно, обычная задача о сортировке уже не может представлять интерес для студентов. В то же время, введение в рассмотрение параметров, о которых ранее ничего не было известно (а именно: распределения случайных величин), создает проблему и мотивирует поисковую деятельность. В ее процессе студенты знакомятся с некоторыми понятиями высшей математики, являющимися фундаментальными в реальных задачах.

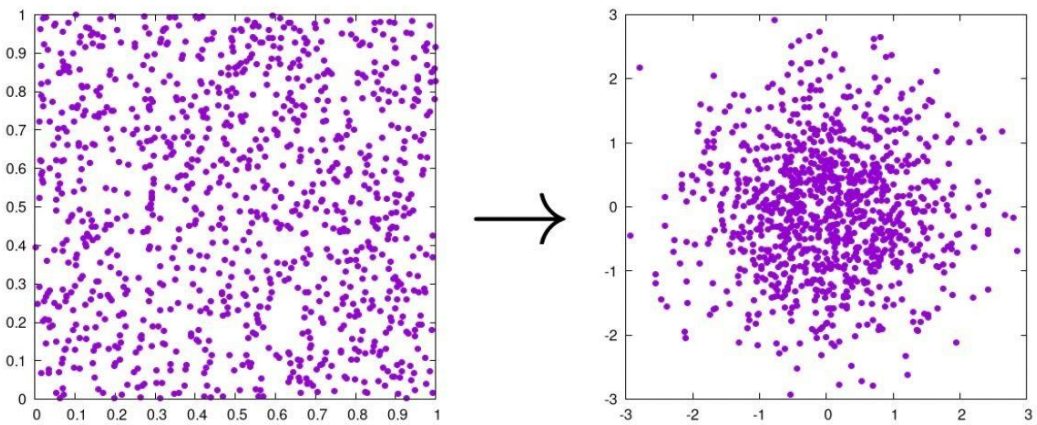
Говоря о результатах, важно отметить, что, во-первых, такая задача позволяет оценить лучшие, худшие и средние случаи для алгоритмов. Во-вторых, массивы, используемые для оценки, представляют собой наборы случайных величин, имеющих различные распределения – например, нормальное и равномерное. В ходе занятия студенты могут построить гистограммы, чтобы визуализировать причины таких различий в подсчетах, проведенных алгоритмами. Это приведет их к открытию важности определения характера распределения – одной из основополагающих задач Data Science, которая обычно возлагается на компьютер. Возможно также, что студенты смогут установить, что не на все сортировки влияет характер распределения случайных величин (например, такой является QuickSort).

Задача 2. Алгоритм быстрой сортировки одинаково хорошо справляется с данными, имеющими различные распределения. Требуется сравнить его модификации на данных двух случайных массивов – теперь уже генерируемых самой программой. Один из них использует

стандартную случайную генерацию (с равномерным распределением), другой – генерацию нормально распределенных величин (если язык или пакет не имеет такой возможности – реализовать преобразование Бокса-Мюллера).

В ходе решения задания студенты знакомятся с некоторыми математическими преобразованиями, позволяющими получать случайные величины иным образом, нежели это делает компьютер в обычном режиме. Таким образом, не только углубляется знание о принципах, заложенных в вычислительные машины (например, о том, что в действительности они работают с псевдослучайными числами), но и о математических моделях, описывающих случайности и причины, по которым применяется несколько распределений.

При выполнении этой задачи студенты обнаружат, что некоторые особенности результатов работы алгоритмов они обосновать не могут. В свою очередь, преподавателю предлагается предоставить теоретико-вероятностные доказательства или хотя бы обоснования для этих результатов. Данная задача является одной из наиболее важных в данном цикле с точки зрения фундаментальной математики и ее приложений. Желательно и полезно сопроводить ее решение дополнительными средствами визуализации.



Точки, координаты которых – равномерно (слева) и нормально распределенные случайные числа (получены преобразованием Бокса-Мюллера). Важно отметить, что преобразование тем лучше генерирует большие последовательности, чем выше точность (32-битная архитектура не так эффективна, как 64-битная)

Задача 3. Сравнить наиболее эффективный из алгоритмов сортировки, определенных в предыдущих двух задачах и наиболее эффективную реализацию быстрой сортировки, с карманной сортировкой (BucketSort), использующей быструю в качестве вспомогательной, с сортировкой кучей (HeapSort) или с сортирующим деревом. Массивы генерируются, как в Задаче 2, и сливаются в один (набор равномерно распределенных следует за набором нормально распределенных).

Данная задача ставит важный вопрос о выборе целесообразной конкретной задаче структуры для хранения данных: хотя Heap и Tree\List возникают в дисциплинах-пререквизитах, курс «Алгоритмы и структуры данных» позволяет оценить их эффективность в условиях большого объема значений. Оценивается так же, как и в случае сортирующих алгоритмов, количество операций, необходимых для работы с динамической памятью. Таким образом, задачу можно рассматривать и как опережающую в контексте компонента компьютерных наук для данной дисциплины: в курсах-пререквизитах структура Tree обычно не используется для реализации сортировок, но может быть имплементирована для достижения поиска значений с высокой скоростью. В свою очередь, Tree найдет другие применения далее в этом разделе.

Из всей цепочки предложенных задач эта является менее остальных нацеленной на математическое моделирование явлений, это компенсируется алгоритмической составляющей. Так

или иначе, причина снижения эффективности на равномерно распределенных величинах кроется в том, «карманы» какой вместимости создает BucketSort.

В качестве диагностической задачи для завершения темы можно предложить Задачу 4.

Задача 4. Используя данные из некоторого открытого источника (например, датасеты из репозитория Kaggle) создать набор экземпляров структуры, описываемой различными полями значений. Провести сортировку полученных моделей явления по всем доступным полям и подсчеты производительности (число элементарных операций, затраты памяти)

Важность поставленной задачи состоит, во-первых, в переходе к комбинированным типам данных – студенты знакомы с ними с изучения ООП, но ранее в этом курсе такие типы были необязательны. Хотя обычно поставленную проблему решают, применяя базы данных, не всегда присутствует явное обоснование преимущества именно индексированных структур над структурами случайного или последовательного доступа. Во-вторых, свобода выбора любого набора данных позволяет преподавателю использовать Задачу 4 в том числе для диагностики аналогично Задаче 0. Кроме того, если рассматривать в качестве источников датасеты, очень удобно в дальнейшем проводить проверку, так как большинство доступных наборов данных подвергались исследованиям специалистами, все особенности данных могут быть подробно задокументированы. Тем не менее, важно иметь в виду, что Задача 4 не подается как введение в машинное обучение или анализ больших данных: ее роль заключается в моделировании определенных условий для алгоритмов.

Выводы

Таким образом, говоря о математическом содержании темы сортировок, можно выделить понятия и концепции, содержание которых в полной мере будет раскрыто только в курсе теории вероятностей и статистики, однако их включение в данную дисциплину позволяет строить более точные модели. К данным сведениям относятся:

– неединственность закона распределения и их отличительные особенности;

– расширение понятия метрик для последовательностей (от средних величин до моды и медианы);

– существование преобразований, позволяющих изменить закон распределения;

– понятие степени выраженности зависимости (корреляции) случайных величин;

– обоснование сходимости различных непрерывных распределений со слабой корреляцией величин к нормальному (в силу центральной предельной теоремы);

Библиографические ссылки

1. Филимонов Д.В., Бровка Н.В. О междисциплинарных задачах и их роли в модернизации курса «Построение и анализ алгоритмов» // Математические методы в технологиях и технике. 2023. № 12. С. 93-97.

2. Филимонов Д.В. О развитии вычислительного мышления и Agile-практиках в образовательном процессе учреждений высшего образования // Университетский педагогический журнал. 2022. № 2. С. 61-65.