Романовский Иосиф Владимирович

ТЕМЫ ДИСКРЕТНОГО АНАЛИЗА В ЗАДАЧАХ ПО ПРОГРАММИРОВАНИЮ

Меня очень волнует, как и всех вас, я думаю, вопрос о сочетании обучения информационным технологиям (я боюсь говорить «информационной культуре») и обучения программированию.

Когда в школе учили только программированию, и при том на Бейсике, мне казалось, что это неправильно, что надо прежде всего учить современного человека пользоваться компьютером для своих разнообразных целей. Чем дальше компьютер внедряется в нашу жизнь, тем важнее становится эта часть образования. Уследить за всеми изменениями невозможно, и, например, я сам недавно для того, чтобы получить какие-то минимальные сведения об используемых сейчас системах распознавания текстов, взял книгу С. Симоновича, Г. Евсеева и А. Алексеева «Специальная информатика», в общем-то школьный учебник, и прочел в ней вполне достаточно для первого раза. С моей точки зрения такое направление образования безусловно важно и интересно.

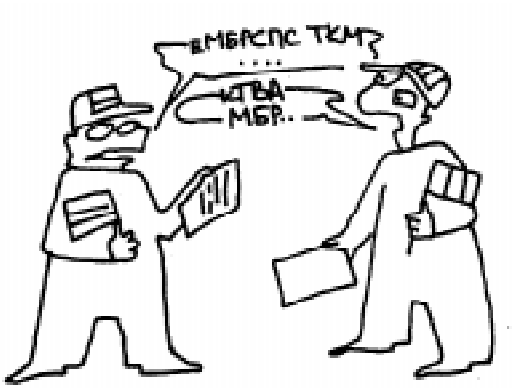
Вместе с тем, программирование ни в коей мере не должно вычеркиваться полностью, и навыки программирования должны получать все, кто их может воспринять. Сейчас нужно сосредоточиться на том, какие градации устанавливать между начальным уровнем знакомства с компьютером (такие, как навыки простейшей работы с текстами) и тем программированием, которое требуется вузам как начальная подготовка для будущих профессионалов. Я согласен с тем, что мы в России учим программированию хорошо, у нас есть много программистов очень высокого уровня, и при тех потребностях в программировании, которые сейчас есть в мире, выращивая новых программистов, мы увеличиваем золотой запас страны. Потому что эти люди, независимо от того, убегают ли они потом в Microsoft, или какую-нибудь другую иностранную фирму, или работают на нужды наших организаций, или зарабатывают валюту в зарубежных организациях, продолжая жить здесь, – это все нас обогащает, и таких людей мы должны учить. И мне кажется, что мы должны стараться, как можно больше людей научить программированию на таком уровне, чтобы они продолжали нашу традицию, а не просто пользовались пакетами.

Поэтому мне кажется важным, что мы должны добавлять к навыкам владения этими элементарными пакетами.

Допустим, мы учим текстовому редактору Word, электронной таблице Excel, учим корректировать графику, работать в Интернете. Разумеется при этом появляются и минимальные сведения об операционных системах, файлах, каталогах, о возможностях печати и отображения данных на экране и т.д., вырабатываются навыки работы с клавиатурой. Но все же, поверх этого, нужно давать разные познания в зависимости от способностей ученика их воспринять. Вот и хотелось бы поговорить о том, чему же учить школьников в программировании, начиная с новичков и кончая теми, из кого мы хотели бы сделать профессионалов.

Я в школе занятий не веду, а читаю лекции на мат-мехе Университета. Там, конечно, публика особенная (я имею в виду студентов). Мой курс[1] носит ввод-

[1] Курс «Дискретный анализ» издан в 1999 г. издательством «Невский диалект», которое сейчас готовит повторное издание.



Когда по международным каналам ... нужно было передавать тексты, мы со своим восьмым битом почувствовали это сразу...

ный характер, в нем должны даваться общие представления о самых разнообразных вещах, большое внимание уделяется связям разных областей математики и информатики. На самом деле, хотелось бы, чтобы часть из того, что я читаю, студенты уже знали при поступлении на мат-мех, но так не получается, мне приходится читать и достаточно элементарные вопросы. Так что недавно, когда я заменял заболевшего коллегу и прочел несколько лекций по информатике на факультете международных отношений (ФМО) (это контингент, «бежавший от математики подальше»; что им нужно рассказывать о математике и информатике?), то я выбрал несколько тем из своего матмеховского курса и увидел, что из этой выборки может получиться хорошая надстройка над исходным фундаментом «информационной культуры».

Первое, что я рассказывал студентам ФМО, – кодирование чисел. Казалось бы, что они должны были пройти этот материал в школе, но надеяться не приходилось. Я быстро рассказал им про двоичную систему счисления, но действиям с двоичными числами не учил, а ограничился сопоставлением последовательностей нулей и единиц с натуральными числами. Попутно в этом сопоставлении появился байт, попутно появились коды ASCII и особенности (в частности, многообразие) кодировок кириллицы. Появилась возможность заглянуть в недалекое будущее и рассказать о переходе на двухбайтовую кодировку Unicode, которая уже начала появляться в типовых пакетах (например, обратите внимание! – в свежих версиях того же Word). С этой «двоичной» темой связано упражнение, которое мне очень понравилось; я могу его всячески рекомендовать.

Несколько лет назад появилась потребность в переводе двоичных файлов в «видимый» формат, чтобы двоичный файл записывался как текстовый. Эта специфическая задача появилась в связи с электронной почтой. Когда по международным каналам, которые тогда были 7-битовые (восьмой был контрольный), нужно было передавать тексты, мы со своим восьмым битом почувствовали это сразу, а за границей это вначале почувствовалось на сжатых файлах, на программах... Не так важно, в связи с чем, но задача появилась.

Один такой формат есть, это шестнадцатеричная запись, она всем известна и проста, но при ее использовании размер файла увеличивается ровно вдвое. Нужно сделать что-либо более экономное.

Итак, есть двоичный файл, в котором все 8 битов используются, и нужно его передать по текстовому каналу. Как бы мы с вами это сделали? Мы бы взяли 3 байта, а это 24 бита, составили из них одну 24-строку и разделили бы ее на 4 шестерки. Каждая такая шестерка битов может рассматриваться как двоичное представление целого числа в диапазоне от 0 до 63, а, стало быть, как цифра в некоей 64-ричной системе счисления. Именно так все и делается. Мы преобразуем три байта в четыре.

11111111	00000000	11111111

111111110000000011111111

111111	110000	000011	111111
63	48	3	63

Есть две системы. Первая из них, Uuencode, появилась давно, на заре электронной почты, вторая, более простая и

немного более эффективная, MIME64[2], сравнительно недавно. Я с этой проблемой столкнулся сам, когда получил по электронной почте первое письмо в кодировке MIME64 и не имел в машине никаких средств его расшифровки. Тогда я стал думать, как бы я сам делал такую кодировку, и оказалось, что как я бы это сделал, так и они сделали. Вопрос только в том, какие выбрать 64 символа в качестве «цифр» в упомянутой системе счисления. Естественно было бы взять прописные латинские буквы, затем строчные, затем цифры. Получилось 62 символа, и еще два символа можно добавить какие угодно. В MIME64 добавляются косая черта и двоеточие, так что мы получаем строку:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefg
hijklmnopqrstuvwxyz0123456789/:
```

С помощью этой строки четыре числа из примера превращаются в странный текст `:wD:` Представьте себе радость от такой догадки. Я думаю, и детей хорошо учить на подобном положительном подкреплении.

Обратите внимание: *каждый* человек, который пользуется электронной почтой (а ею пользуемся мы все), может получить по e-mail какую-то тарабарщину. И мы его можем научить расшифровке сообщения, и попутно он начинает осваивать такие операции, как выбор символов, перевод их в двоичную систему. Его уже не пугает 64-ричная система, и поэтому 16-ричная – это уже совсем просто.

Вот об этом я и рассказывал студентам факультета международных отношений, которые к математике и программированию относятся сдержанно. Студенты все поняли, это для них оказалось не очень сложно. Но рассказывать можно всем, ожидая различного эффекта: для кого-то, например, не очень сложно создать свой кодировщик и проверить, правильно ли он работает.

Из области кодировок можно назвать и другие темы. Например:

– *штриховые коды* – коды, которые пишутся сейчас на заграничных конвертах. Эти коды предназначены для записи десятичных цифр, каждая цифра кодируется пятью вертикальными черточками, из которых ровно две черточки длиннее. Число таких наборов как раз равно числу сочетаний из 5 по 2, мы получили прямое практическое использование этой элементарной комбинаторики.

– детали двухбайтовой кодировки Unicode и промежуточный формат UTF-8 для экономной пересылки текстов в Unicode[3].

Еще один сюжет на темы кодирования. У нас почти каждый, кто работает на компьютере, сталкивается с нехваткой дисковой памяти и практически сразу начинает пользоваться архиваторами (`zip, arj, rar` и т.п.). Я считаю, что в обучение информатике тему об архиваторах нужно включать обязательно: каждый пользователь должен уметь свои файлы сохранять, а попутно мы можем этой же темой пользовать-

[2] MIME – Multipurpose Internet Mail Exchange.

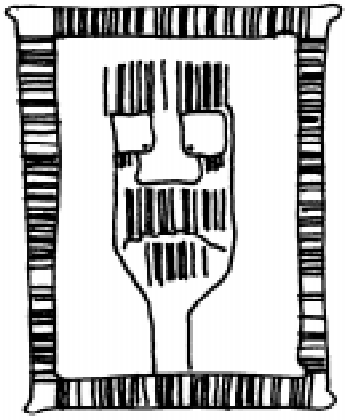
[3] Пользуясь случаем, представляю вашему вниманию книгу «Практика программирования» Б.Кернигана и Р.Пайка, перевод которой скоро выйдет. Тот, кто читал книгу Кернигана и Ритчи по языку Си, знают программу «Hello, world!», с которой они рекомендуют начинать обучение программированию. Речь идет о том, что надо вначале на изучаемом языке написать хоть что-то – например, печать одной фразы. Мне очень понравился этот дух обучения программированию.

Когда они разрабатывали ОС UNIX, то делали одновременно три продукта: сам UNIX, язык программирования С и полиграфическую систему TROFF. Все три системы были написаны на языке С. Меня очаровала в то время такая системность подхода, как они сразу поняли, что надо делать все вместе. Впоследствии я познакомился с созданным при участии Кернигана языком AWK, который позволяет быстро делать простые вещи. Ни на одном языке я не написал столько программ, сколько на AWK. Обычно программы в 1-2 строки – это замечательное дополнение к текстовому редактору.

«Практика программирования» отражает богатый опыт авторов и их коллег. Вместе с тем, она использует и свежие идеи – примеры программирования приведены на Си, C++ и Java. Одна из тем, которые там рассматриваются, как раз взаимодействие кодировок – старой кодировки ASCII и UTF-8.

ся для того, чтобы учить ребят тому, как это делается.

Способы сжатия текста, которые лежат в основе работы архиваторов – это замечательное открытие нашего времени, и учить детей (хотя бы на нескольких принципах) тому, на каких принципах основывается работа архиваторов, в частности методам Хаффмена и Зива-Лемпеля, очень полезно: можно знакомить их с идеями использования статистического анализа и адаптивности. Нужно только подобрать уровень математической строгости изложения, например, в алгоритме Хаффмена, вместо вероятностей, вполне можно обходиться частотами. А на современных способах сжатия (например, JPEG и MPEG) мы можем учиться и учиться. Причем, вы можете варьировать эту тематику от рассказа о том, какой результат достигается, до программирования соответствующих алгоритмов, если дети на это способны.



Например ... штриховые коды...

Можно к этой теме добавлять и вопросы защиты информации, а также в какой-то мере криптографию.

Следующая важная тема, связанная с обработкой данных, – сортировка. Это замечательный пример задачи, которую можно объяснить менее чем за минуту, которая совершенно ясна, у которой одно решение совсем очевидно, и которой при всем при этом продолжают заниматься до сих пор. Имеется масса сортировок с теми или иными специфическими свойствами, и на этой задаче очень легко объяснить, почему мы продолжаем заниматься такой простой задачей. И почему это непросто – придумать эффективную сортировку, удовлетворяющую некоторым особым условиям.

Когда говорят, например, о способах вычисления специальной функции, это требует очень много объяснений, много математики, а здесь все условие на виду, а задача неочевидная.

В связи с сортировкой появляется возможность говорить о способах сравнения записей, о лексикографическом сравнении. Я очень мучался с этим вопросом, когда принимал зачеты у студентов ФМО: они спокойно пользовались алгоритмом, сравнивая конкретные слова, и почти никто не мог объяснить, как правильный результат получается.

Третья тема – поиск информации и специальные структуры поиска. Я не знаю, удается ли вам рассказать школьникам о поисковых системах. Я ограничусь одной темой – «Способы организации быстрого доступа к информации». У нас есть бинарное дерево. И, конечно, после массивов и цепных списков я обычно рассказываю про замечательную конструкцию – хеш-функцию. Речь идет вот о чем. Представьте себе, что у вас много элементов информации, и каждый обладает уникальным ключом[4]. Для меня привычный пример – студенты Университета и номера их зачетных книжек. Номер девятизначный. Быстрее всего было бы организовать доступ так: каждый девятизначный ключ – это число. Если бы мы могли хранить массив такого размера, нам ничего другого не было бы нужно. А мы не можем. И есть такой промежуточный способ доступа. Мы создаем массив доста-

[4] Вполне понятные примеры возможных информационных систем можно брать прямо из газет, примеры, поражающие беспомощностью органов, которые принято называть компетентными. Подумать только, в Министерстве обороны нет информационно-справочной системы, позволяющей опознать военнослужащего по его индивидуальному жетону («Известия», 20 января 2000 г.)! Несколько лет назад газеты писали, что в МПС СССР была служба потерянных и найденных вагонов, где бумажки с информацией о таких вагонах вывешивались на бельевых веревках в большой комнате и люди ходили вдоль веревок и сопоставляли номера.

Способы сжатия текста ... — это замечательное открытие нашего времени...

точно большого размера K, это будет массив *корзин* (корзина – *bucket*). Сделаем функцию h, которая отображает множество ключей в множество корзин, так что с помощью этой функции по номеру зачетной книжки мы находим номер корзины. Какие требования к h? Чтобы она распределяла ключи по корзинам возможно более равномерно. На самом деле выбор этой хеш-функции (от английского слова to hash – «перемалывать») должен быть таким, чтобы как можно сильнее перемешать ключи, чтобы при близких значениях ключа записи попадали в разные корзины. Сделать это можно по-разному. Типичная хеш-функция:

$$h(s) = h(s')\times\alpha+\mathrm{Ord}(\omega),$$

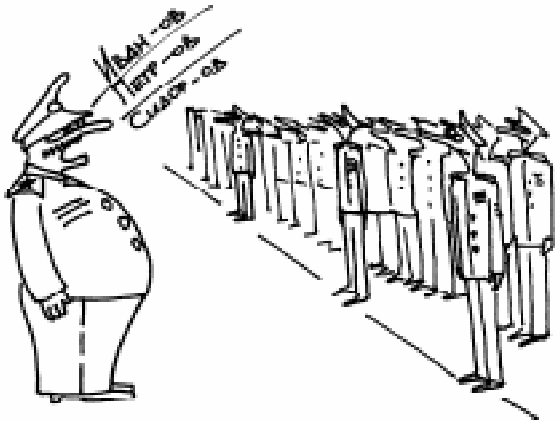
где s – строка, ω – ее последний символ, а s' – строка без последнего символа. Число α – это множитель. Результат вычисляется по модулю K, так что получается число в диапазоне от 0 до K-1. Хорошее упражнение в информатике – сравнить результаты, получающиеся при выборе различных значений α и K.

Если учащиеся более продвинуты в математике, их можно учить и более сложным способам хранения информации. В частности, я недавно (с некоторым опозданием) узнал, что есть такое понятие, как *приоритетная очередь*. После того, как они познакомятся со стеком, цепным списком, можно приучить их к чему-нибудь «более шикарному». Приоритетная очередь – это такая конструкция хранения данных, в которую вы можете складывать элементы в любом порядке, а вынимать каждый раз минимальный из имеющихся (например, элемент с наименьшим номером).

Существует много компьютерных моментов, которые связаны не с математикой, а с внешним миром. Я сейчас отклонюсь от темы дискретной математики и скажу, что всячески пропагандирую язык PostScript, на котором можно давать школьникам возможность рисовать, проектировать рисунки в цвете, причем имеются свободно распространяемые программные средства, которые вы можете установить на своем компьютере. Я однажды в Педагогическом институте им. Герцена, чтобы пропагандировать эти идеи, даже прочел спецкурс по PostScript. Зачет состоял в том, чтобы нарисовать свою картинку. Были нарисованы очень разные картинки, среди них были настоящие наглядные пособия, например, «Пересечение куба с плоскостью».

Чем еще хорош PostScript? Он по своим геометрическим установкам предельно согласован со школьной геометрией. Поле деятельности – это лист бумаги, у которого начало координат в левом нижнем углу. Вы можете его использовать для геометрических чертежей, а можете нарисовать орнамент. И в Школе современного программирования одно занятие целиком посвящено PostScript.

PostScript был разработан американской фирмой Adobe, чтобы управлять



Следующая важная тема ... — сортировка...

принтерами высокого разрешения. Чтобы компьютер не посылал принтеру все точки, а посылал программу, и интерпретатор, встроенный в принтер, выполняя эту программу, рисовал очередную страницу.

Но не у всех есть мощные принтеры со встроенным интерпретатором PostScript. Поэтому некоторая фирма с интригующим названием Alladyn Enterprises создала интерпретатор, который работает на самом компьютере. Он называется GhostScript и может прочитать вашу программу на Postscript и ее выполнить. Для того, чтобы в Windows было удобно с ней работать, есть надстройка, которая называется GSView. И то и другое размещено в Internet.

Кроме того, имеется очень много PostScript-овских шрифтов. Практически все достояние мировой полиграфической культуры переведено в PostScript. Кроме того, имеется редактор шрифтов FontLab, который позволяет самостоятельно создавать шрифты.

Язык PostScript очень специфический, красивый, похож на FORTH, и кто-то сказал, что PostScript больше FORTH, чем сам FORTH (имеется в виду работа со стеками), на русском языке почти ничего про эту науку нет.

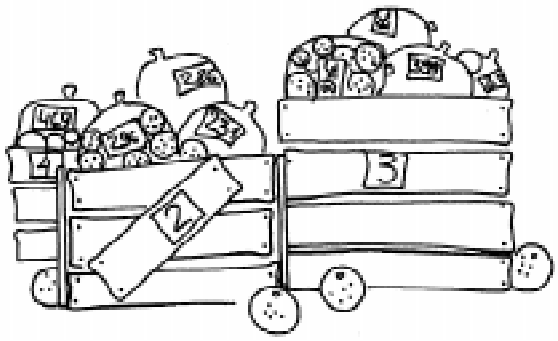
Теперь хотелось бы поговорить про детей, которые могут научиться программированию и которых нужно этому учить. Я долгое время добивался, чтобы на матмехе обучение программированию начинали с Паскаля. Недавно я познакомился с одной паскалевской системой, которая называется Delphi. Delphi хорош тем, что позволяет писать объекты на Паскале, в Windows-приложениях, причем избавляет вас от огромного количества технических деталей. Я не раз устраивал себе такие «разминки» – на глазах у человека составлял какую-нибудь программу. Например, нажимаешь кнопку, и появляется картинка. Получается сочетание программирования такого «изначального», когда сам что-то пишешь, и «сборочного», когда собираешь что-то из компонент. Я думаю, что выгодно начинать со сборочного программирования, а потом постепенно прибавлять вычислительные детали. Для многих это оказывается проще.

Но вернемся к вопросу, который уже затрагивался: кого, когда и чему надо начинать учить? Моя точка зрения, о которой я уже упоминал: надо сначала учить Паскалю. Известен такой эффект: цыпленок, когда он только что вылупился, первый движущийся предмет считает своей мамой. И человек первый выученный им язык программирования считает основным. С этой точки зрения Паскаль получился лучше, поскольку он дает лучшую базу. Многие мои коллеги считают, что лучше начинать с Паскаля, а курсе на третьем «ломать привычку» и переходить к другому языку (у нас на кафедре для этого используют C). Такая замена необходима для того, чтобы человек не относился к Паскалю как к чему-то единственно возможному, он должен сравнивать. И это дает хорошие плоды.

В заключение приведу условия нескольких задач по программированию, которые в прежние годы предлагались школьникам или студентам.

1. Множества на прямой заданы последовательностями отрезков (монотонно возрастающими). Выразите таким способом объединение и пересечение двух множеств. (Это один из примеров работы со строками, которые можно рассматривать не только как фрагмент текста, но и как последовательность некоторой длины.)

2. Есть три множества: A, B и C. Требуется за один просмотр файлов сосчитать $(A \cap B) \cup (A \cap C) \cup (B \cap C)$



Мы создаем массив достаточно большого размера...

3. При составлении словаря используют сжатие слова (формула). Дальше пишут, сколько позаимствовано букв из предыдущего слова, затем оставшиеся буквы. Это очень эффективный прием для специального текста, в котором слова длинные и различаются лишь окончаниями. А вопрос такой: задана верхняя граница длины слова и упорядоченный словарь. Найдите длины групп.

Многие мои коллеги считают, что лучше начинать с Паскаля...

Мы можем быстро объяснить такое условие. Вся необходимая вычислительная сторона описана словами, и ребенок при общем лозунге «Экономия места» должен все понять. Такого рода задачи я бы рекомендовал. При этом я знаю, что в информатике, в отличие от многих областей математики, решение часто неоднозначно. Но значения все же одинаковы.

4. На Всероссийской олимпиаде прошлого года мы предлагали следующую задачу. Плоскость разбита на квадратные плитки. Задана точка. Расстоянием от точки до плитки назовем расстояние до ближайшей точки плитки. Требуется упорядочить плитки в порядке увеличения расстояния. Эта задача, несмотря на кажущуюся простоту, оказалась довольно сложной для участников олимпиады.

Я постарался свои воззрения на темы дискретного анализа, заложенные в курс, который я читаю на мат-мехе, изложить в книге «Название книги?». Мне говорили, что эта книга содержит больше материала для школы, чем для мат-меха. Но это вводный курс. Я не хочу уверять, что это все, чему хотелось бы научить в дискретном анализе.

***

Вопрос: *Для какой школы рекомендуется то, о чем Вы рассказывали? Для общеобразовательной или для школы с углубленным изучением математики?*

У меня была шкала, предполагающая несколько уровней. Вопросы кодирования, того, что может встретиться в E-mail, я бы рассказывал всем. А дальше, в зависимости от «продвинутости» школы, я бы добавлял материал. Я считаю, что следует объяснять вещи, которые помогают решить те или иные проблемы. Например, в Excel: необходимо для сравнения слов и записей объяснить, что «нерпа» может стоять перед «нахалом», если у «нерпы» случайно написано «е» латинское. Это же полезно, особенно для тех, кто не очень разбирается.

Вопрос: *Почему в информатике до сих пор не установилась единая терминология – например, строковые переменные называют то литерными, то символьными?*

Выработка единой системы – это процесс очень долгий, может быть, к счастью, поскольку выработать единую терминологию безумно трудно. Иногда выбранные термины получаются по недоразумению, например, слово «синус» в переводе означает «залив» (ошибся переводчик, перепутав слова «хорда» и «залив»).

Способствует разногласиям в определениях и стремление многих непременно присваивать новым изобретениям иностранные названия.

НАШИ АВТОРЫ

Романовский Иосиф Владимирович,
доктор физ.-мат. наук,
профессор СПбГУ.