Лямов Андрей Геннадьевич,
Поздняков Сергей Николаевич,
Прокопенко Никита Юрьевич

КОМПЬЮТЕРНЫЙ ЭКСПЕРИМЕНТ В ОБУЧЕНИИ МАТЕМАТИКЕ

В конце февраля прошел очередной Всероссийский дистанционный конкурс КИО–2006. Особенностью этого конкурса является его экспериментальный характер. Участникам на три дня предлагается три задачи на исследование физических моделей или математических манипуляторов. Задачи эти связаны с важными математическими идеями, часть из них сформулирована на основе еще не решенных математических проблем. Поэтому участие в конкурсе можно рассматривать как первый шаг к исследовательской деятельности. Каждая предложенная задача допускает различные решения, которые можно сравнивать между собой по некоторому параметру. Побеждают участники, добившиеся по этому признаку лучших результатов. Эти результаты можно считать первыми, экспериментально полученными результатами в решении трудной математической проблемы. Далее их можно попытаться осмыслить теоретически, обобщить и доказать.

В этом году было предложено два уровня конкурса: для младших и старших школьников.

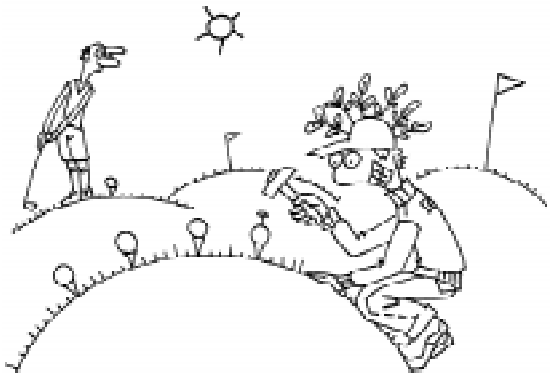
Рассмотрим предложенные задачи с двух точек зрения:

1) с точки зрения математических задач, которые использованы в предложенных лабораториях для проведения экспериментов;

2) с точки зрения технической поддержки экспериментов.

Первая точка зрения будет любопытна участникам олимпиады и учителям. Вторая точка зрения представляет интерес для авторов компьютерных программ учебного назначения.

ЗАДАЧА 1. ЗАДАЧА ТОМСОНА
(для старшего возраста)



«Даны 10 точечных зарядов одного знака: два заряда по 5 кулонов, два – по 4, два – по 3, два – по 2 и два – по 1. Вы можете расположить их на сфере, жестко их зафиксировав («прибить гвоздиками»), чтобы они не перемещались.

Заметим, что если бы заряды смогли перемещаться по сфере свободно, то, действуя друг на друга, они расположились бы так, чтобы их потенциальная энергия достигла определенного минимума. При другом начальном положении они могли бы прийти в равновесное состояние с другой энергией.

Ваша задача – найти состояние с минимальной при данных условиях задачи энергией.

Для решения задачи Вы можете перемещать шарики по сфере (левой кнопкой мышки), поворачивать сферу (правой кнопкой мышки или с помощью стрелок), строить выпуклый многогранник с вершинами в зарядах и прочее.

Вашим рекордом считается наименьшее полученное Вами значение потенциаль-

ной энергии. Улучшение рекорда – уменьшение потенциальной энергии».

Замечание.

При выходе из задачи автоматически записывается текущее решение. Оно же будет открыто, когда Вы снова запустите задачу. Программа, с которой Вы работаете, автоматически сохраняет лучшее решение» (рисунок 1).

Данная задача была сформулирована по мотивам статьи Н.Н. Андреева и М.А. Калиниченко «Задача Томсона», опубликованной в журнале «Компьютерные инструменты в образовании», № 1 за 2005 год. Классическая постановка задачи такая «Поместим на сферу *N* одинаковых зарядов. К каким расположениям будут стремиться заряды, пытаясь минимизировать потенциальную энергию системы?». На сегодня только некоторые частные случаи этой задачи решены математически строго. Для конкурса КИО–2006 задача была сформулирована иначе, без ограничения на одинаковую величину зарядов. По условию были даны 10 точечных зарядов одного знака: два заряда по 5 кулонов, два – по 4, два – по 3, два – по 2 и два – по 1. Их можно было жестко зафиксировать на сфере, чтобы они не перемещались. Требовалось найти состояние с минимальной при данных условиях задачи энергией.

Формально говоря, участникам конкурса нужно было найти минимум функции от 20 переменных (10 зарядов, каждый из которых имеет две координаты на сфере). Для этого участники должны были выдвинуть гипотезу относительно взаимного расположения зарядов (топологическая структура решения), а затем «шевелением» отдельных зарядов найти локальный минимум. Исследование решений победителей показало, что все они получили топологически разные конфигурации (считая заряды вершинами графа). Приводим два лучших решения.

Решение Михайлова Александра (I место, энергия 561.129 Дж) (рисунок 2).

Решение Суворова Александра (II место, энергия 561.148 Дж) (рисунок 3).

Замечания по реализации

Для обеспечения программ удобным интерфейсом и красочным визуальным оформлением были использованы стандартные методы построения трехмерных изображений с использованием библиотеки OpenGL (Open graphic library). Данная библиотека предоставляет достаточно широкие возможности для динамического построения трехмерных сцен, в том числе такие, как: поворот и перенос системы координат в пространстве, наложение текстур, использование прозрачности объектов, создание эффекта освещения и т. д. Главное удобство использования OpenGL заключается в том, что для построения объектов нам предоставляется пространственная система координат и все вершины объектов задаются в виде трех координат. Для примера рассмотрим построение сферы в задаче Томсона. Чтобы построить реалистичную сферу, необходимо сначала включить использование освещения и задать положение источника света, после этого все объекты будут казаться более реалистичными. В программе предусматривается вращение сферы для удобства установки зарядов. Для этого в OpenGL перед рисованием нужно сначала повернуть систему координат на необходимый угол и только потом

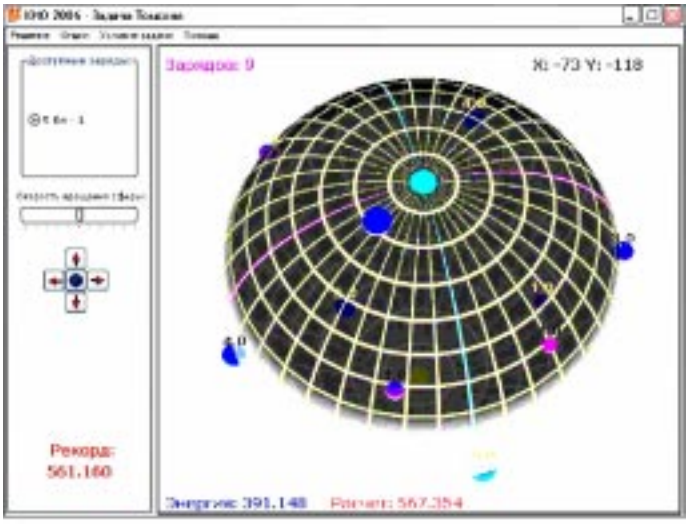


Рисунок 1.

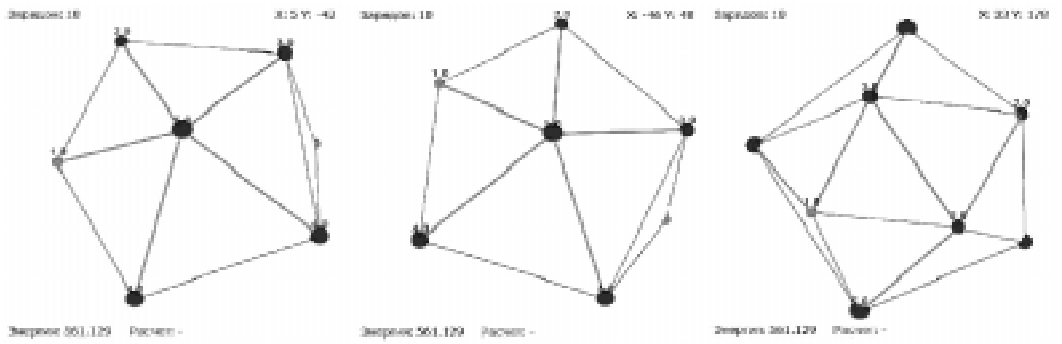


Рисунок 2.

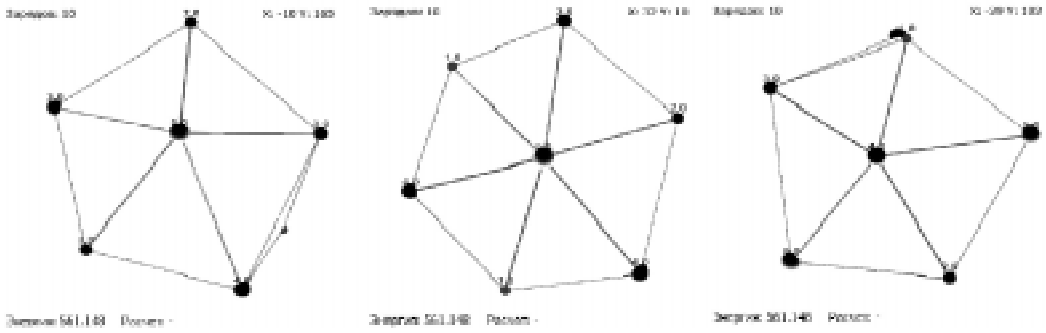


Рисунок 3.

осуществлять рисование. Для создания эффекта сетки на сферу была наложена меридианная текстура, каждой клетке на сфере соответствует угол в 10 градусов. Непосредственно рисование сферы осуществляется после всех манипуляций с системой координат, текстурами и освещением всего в несколько программных строк. Для достижения эффекта плавного вращения сферы в программе происходит непрерывная перерисовка экрана.

Рассмотрим теперь несколько технических проблем, которые нам пришлось решить в ходе написания программы о задаче Томсона.

Для того чтобы при нажатии мышью на сферу заряд установился в нужную точку, необходимо было преобразовать оконные координаты в сферические. Ниже приведем формулы, которые осуществляют данное преобразование:

Px, Py – оконные координаты центра экрана;

X, Y – координаты курсора мыши;

SphereRad – радиус сферы (в оконных координатах);

SAngX, SAngY – искомые сферические углы (в радианах);

$$SAngY := -\arcsin\left(\frac{Y - PY}{SphereRad}\right);$$

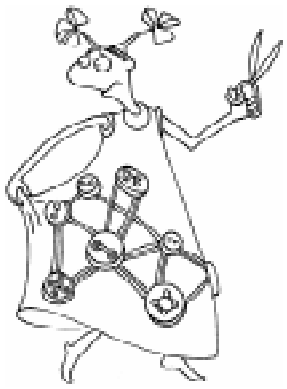
$$SAngX := \arcsin\left[\frac{X - PX}{SphereRad \cdot \sqrt{1 - \left(\frac{|Y - PY|}{SphereRad}\right)^2}}\right]$$

Чтобы получить исходные углы положения заряда необходимо к вычисленным углам добавить текущие углы поворота сферы.

Другой проблемой, с которой пришлось столкнуться, была проблема рисования выпуклого многогранника, вершинами которого являются заряды, находящиеся на сфере. Было использовано самое простое решение, использующее стандартные механизмы проверки видимости элементов конструкции – построено все множество треугольников, соединяющих между собой все заряды методом перебора. Пользователю при этом видны только «внешние» треугольники, которые и образуют выпуклую оболоч-

ку. Недостатком такого подхода является избыточное количество построенных треугольников (для 10 зарядов их общее число составляет 120 – число сочетаний из 10 по 3), часть которых находится внутри выпуклой оболочки и поэтому не видна пользователю.

ЗАДАЧА 2. «ПОЧИНКА СЕТИ» (для всех возрастов)



«Десять пуговиц как-то соединены резинками. Как – неизвестно. Из этой паутины поочередно удаляется каждая пуговица вместе с резинками, соединяющими ее с остальной сетью. Полученные десять сеток с девятью пуговицами показаны в верхней части экрана. Ваша задача – реконструировать исходную сеть.

В математике подобные конструкции называются графами, пуговицы – вершинами, резинки – ребрами, а подграфы, полученные отбрасыванием вершин и смежных им ребер, в совокупности – колодой графа.

Используя инструменты построения вершин и ребер, Вы можете построить произвольный граф с десятью вершинами, а потом построить его колоду. Если в полученной колоде окажутся подграфы, которые были в колоде в условии задачи, они будут подсвечены зеленым цветом.

Ваша задача – добиться совпадения колоды, соответствующей построенному Вами графу, с колодой заданного графа.

Вашим рекордом считается наибольшее число совпавших элементов колоды. Улучшение рекорда – увеличение числа совпадений».

Математическое название этой задачи – «Восстановление графа по его колоде».

Исходный граф раскладывается на несколько графов – из исходного графа по очереди удаляется только одна вершина вместе с примыкающими к ней ребрами. Полученные подграфы образуют так называемую колоду. Число элементов в колоде равно числу вершин в исходном графе. До сих пор не доказана в общем виде гипотеза о том, что любой граф однозначно восстанавливается по его колоде.

Вот как формулируется эта гипотеза более формально.

Гипотеза Келли–Улама *(Conjecture of Kelly and Ulam)*

Все графы порядка $n > 2$ реконструируемы. Иначе говоря, любой граф $G = (V, E)$ с $|V| > 2$ однозначно восстанавливается по набору подграфов вида $G\backslash v$, $v \in V$.

Справедливость гипотезы, известной с 1945 г., подтверждена для графов с $3 \le |V| \le 10$.

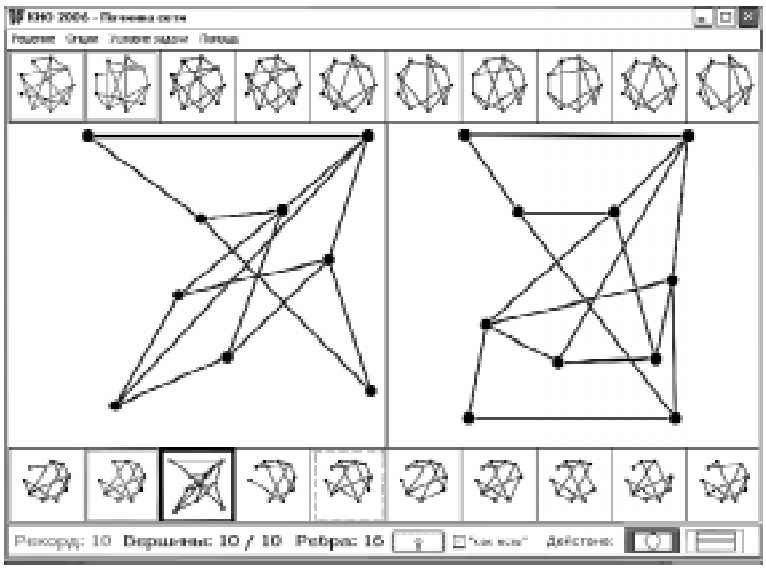


Рисунок 4.

Интерфейс программы (рисунок 4) позволяет не только наглядно демонстрировать решение, но и исследовать колоду, полученную из собственного графа вместе с исходной колодой, отслеживать совпавшие подграфы колод, располагать элементы колоды в удобном порядке. В общем, он обеспечивает осмысленный процесс экспериментов, не сводящийся к простому угадыванию.

Стратегия решения определяется сложностью исходного графа. Некоторые частные случаи при задании условия позволяют решить задачу моментально.

Например, если в исходном графе есть изолированная вершина (оторванная от сети пуговица), то, взяв элемент колоды, соответствующий этой вершине, и добавив к нему одиночную вершину, получим исходный граф.

Или, например, если в исходном графе есть вершина, к которой примыкает только одно ребро, то, взяв элемент колоды, в котором этой вершины нет, можно получить решение перебором вариантов поочередного соединения новой вершины с остальными вершинами элемента колоды. Именно так, например, можно было решить задачу I уровня – достаточно было взять первый граф колоды, добавить к нему вершину и, по очереди соединяя ее с остальными, получить совпадение всех десяти подграфов. Причем для заданного графа 4 из 9 вариантов оказались бы успешными (рисунок 5).

Наличие нескольких компонент связности также позволяет без труда восстановить исходный граф, так как в колоде обязательно будут элементы, содержащие компоненты связности нетронутыми.

«Распутывание» графа может привести к тому, что никакие ребра его не будут пересекаться (такие графы называют планарными). «Распутав» планарный граф, поиск решения можно заметно упростить.

Замечания по реализации

После ввода участником графа–решения получаемая из него колода должна пройти сравнение с исходной, а именно, каждый граф полученной колоды нужно сравнить на совпадение (изоморфность) с графами другой колоды. Понятие изоморфности на интуитивном уровне означает, что «распутыванием и запутыванием» одного графа можно получить другой. Формально это означает возможность так пронумеровать вершины обоих графов, что для каждого ребра, соединяющего вершины с номерами A и B одного графа, у другого графа также есть ребро, соединяющее вершины с этими номерами.

Полный перебор всех вариантов нумерации вершин с последующим сравнением – операция достаточно трудоемкая, поэтому для сравнения двух графов на изоморфность была разработана функция, не использующая полный перебор (при полном переборе для графа на девяти вершинах необходимо было бы рассмотреть 9! = 362880 вариантов).

Граф в программе задается количеством вершин и массивом ребер (граф неориентированный, ребро задается парой вершин). Работа функции содержит несколько стадий проверки, позволяющих быстро отбросить явно неизоморфные графы.

На первой стадии проверки функция производит сравнение по числу вершин и количеству ребер.

Предусловием для второй стадии является одинаковая нумерация вершин каждого графа (то есть, например, вершины обоих графов нумеруются от 1 до числа вершин), так как в основе функции лежит поиск одинаковых ребер после перенумерации вершин одного из графов.

На второй стадии вершины разбиваются на классы – сортируются по степеням (количеству ребер, выходящих из данной вершины). Затем сравнивается количество вершин каждой степени в одном графе с аналогичным количеством в другом графе.

Перед третьей стадией отсортированные по классам вершины перенумеровываются одинаковым образом (например, от 1 до числа вершин). На третьей стадии рассматриваются все возможные варианты пе-

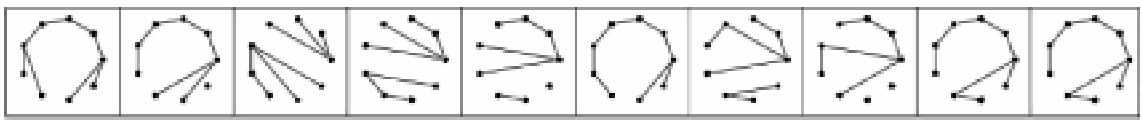


Рисунок 5.

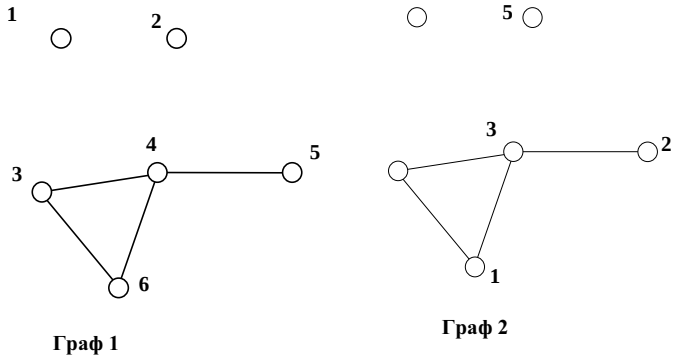


Рисунок 6.

рестановок в каждом классе вершин (вершин, имеющих одинаковые степени). При этом после генерации очередной перестановки происходит сравнение графов на совпадение. Для ускорения работы функции при генерации меняются местами только два элемента, для перенумерации используется матрица соответствия между текущей и новой перестановками – при такой организации удается уменьшить даже количество операций присваивания при перенумерации. Вершины нулевой степени можно не рассматривать, так как их номера не используются в массиве ребер.

Рассмотрим работу функции на примере двух изоморфных графов (см. рисунок 6).

Очевидно, первая стадия проверки выполняется (число вершин и ребер одинаково).

Вторая стадия также выполняется (графы имеют одинаковое количество вершин с одинаковыми степенями) (таблица 1).

На третьей стадии работа функции выглядит следующим образом (таблица 2).

После того как вершины были разбиты на классы и перенумерованы, осталось рассмотреть всего две перестановки – перестановки вершин со степенью 2.

Таблица 1.

Граф 1		**Граф 2**	
Степень вершины	Количество вершин с такой степенью	Степень вершины	Количество вершин с такой степенью
0	2	0	2
1	1	1	1
2	2	2	2
3	1	3	1
4	0	4	0
5	0	5	0

Таблица 2.

	Граф 1								**Граф 2**							
Степень вершины, i	0		1	2		3	4	5	0		1	2		3	4	5
Количество вершин с такой степенью, $m(i)$	2 (не рассматриваются)		1	2		1	0	0	2 (не рассматриваются)		1	2		1	0	0
Номер вершины	1	2	5	3	6	4	–	–	6	5	2	1	4	3	–	–
Новый номер вершины	–	–	1	2	3	4	–	–	–	–	1	2	3	4	–	–

В общем случае максимальное число вариантов N, которые необходимо рассмотреть, вычисляется по формуле:

$$N = \prod_{i=1}^{n} (m(i))!,$$

где n – число вершин графа, $m(i)$ – количество вершин степени i.

В рассмотренном примере при полном переборе требовалось бы рассмотреть не более 6! = 720 вариантов, при работе функции – гарантировано не более 2 вариантов. Работа функции тем эффективнее, чем разнообразнее степенной состав вершин и чем больше вершин нулевой степени.

ЗАДАЧА 3. «ВОЗВЕДЕНИЕ В СТЕПЕНЬ» (для младшего возраста)

«Имеется конструктор с двумя видами автоматов.

Автоматы первого вида умножают входное выражение на A, а автоматы второго вида – возводят его в квадрат. Число автоматов каждого вида не ограничено. Их можно перетягивать мышкой на рабочее поле и расставлять в удобном порядке. У каждого автомата слева имеется вход, а справа – выход. Перетягиванием мышкой можно соединять вход одного автомата с выходом другого.

Ваша задача – создать из автоматов цепочку, которая возводит A в 89 степень.

Не забудьте соединить начало цепочки со входом схемы на левой границе рабочего поля, а конец – с ее выходом на правой границе.

Проверить результат можно, нажав соответствующую кнопку.

Вашим рекордом считается наименьшее число автоматов, с помощью которых Вы получили заданную степень. Улучшение рекорда – уменьшение числа автоматов в цепи».

Основой конструктора было наглядное представление цепочки автоматов с возможностью пошагового контроля решения. На рабочем поле можно было «собирать» несколько цепочек и поочередно тестировать их, сравнивать результаты. Например, на рисунке 7 выделен третий автомат, после его подсоединения при пошаговом тестировании получен результат A^5.

Задача давалась участникам КИО – 2 006 I уровня, требовалось возвести входное выражение A в 89 степень. Многие ребята справились с этой задачей, решив ее с использованием 9 автоматов. Рассуждения для оптимального решения могут быть такими.

Возводить в квадрат «выгоднее», поэтому надо стараться всегда, когда можно возводить результат в квадрат (единственное исключение составляет первый шаг: возведение A в квадрат или умножение A на A дает одинаковый результат). Поэтому решение задачи дает обратный анализ:

– последний шаг не мог быть возведением в квадрат, так как степень 89 нечет-

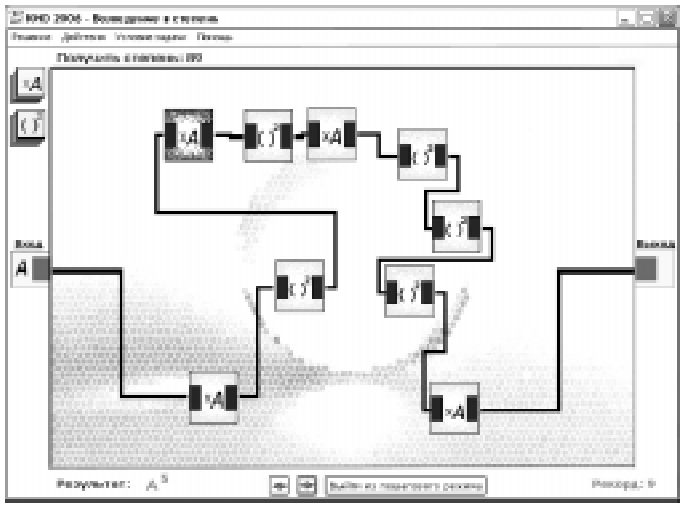


Рисунок 7.

ная, значит это умножение на A, тогда на входе автомата умножения на A было выражение A^{88};

– степень 88 четная, поэтому выражение A^{88} может быть получено возведением в квадрат выражения A^{44};

– степень 44 четная, поэтому выражение A^{44} может быть получено возведением в квадрат выражения A^{22} и т. д.

Получается последовательность автоматов $ABBBABAB$(A или B) или в прямом порядке: (B или A)$BABABBBA$. Это и есть ответ (B обозначает автомат возведения в квадрат). Заметим, что два автомата A никогда не идут подряд в этой цепочке, так как применение автомата A меняет четность степени, а для четной степени выгоднее использование автомата B. Если произвести перекодировку обратной последовательности, двигаясь слева направо и заменяя AB единицей, или, если следующий автомат B, то заменяя его нулем, получим

$$100110 = (AB)(B)(B)(AB)(AB)(B).$$

Если перевернуть полученную двоичную запись и добавить впереди 1, то получим двоичную запись степени 89 :

$$(89)_{10} = (1011001)_2.$$

Объяснить это легко. Сформулированный выше алгоритм обратного анализа задачи в точности соответствует алгоритму перевода степени из десятичной записи в двоичную. При этом для нечетной степени деление с остатком на 2 соответствует последовательному действию двух автоматов:

$$N = 2*M + 1 \leftrightarrow A^N = (A^M)^2*A.$$

Добавление первой единицы соответствует получению A из 1 по описанной формуле:

$$A = (1)^2*A.$$

ЗАДАЧА 4. «ПЕРЕЛИВАЙКА» (для всех возрастов)

«Имеются три ведра заданного объема, вмещающая «сколько угодно» воды бочка и море. Вы можете зачерпывать воду ведрами из моря и выливать ее в бочку, выливать всю воду из ведра или бочки в море, наливать воду из бочки в ведра.

Ваша задача – с помощью ведер получить в бочке необходимый объем морской воды.

Вашим рекордом считается наименьшее число ходов, за которое будет получен необходимый объем воды в бочке. Улучшение рекорда – уменьшение числа ходов».

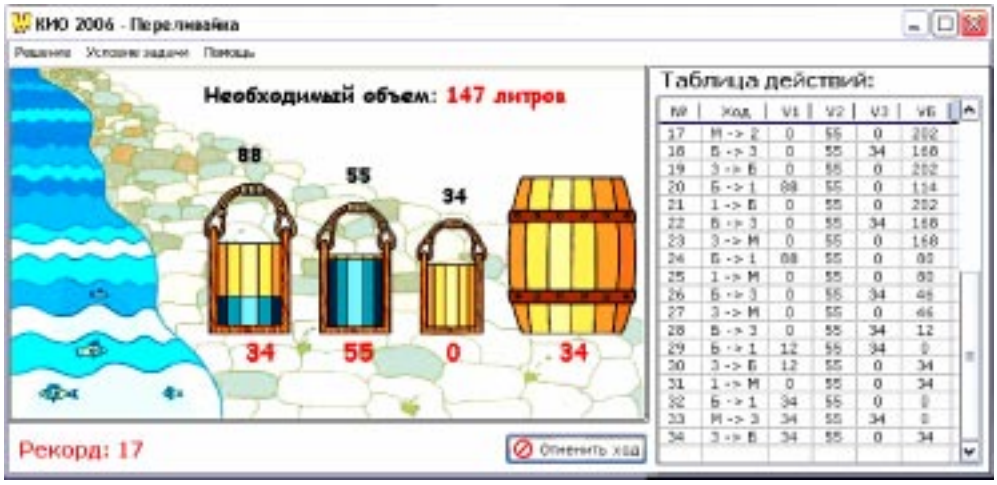


Рисунок 8.

Решение этой задачи связано с алгоритмом Евклида (расширенным алгоритмом Евклида), который обсуждался в рубрике «Заочная школа современного программирования» в журнале «Компьютерные инструменты в образовании», № 1 за 1999 г.) в статье «Алгоритмы над целыми числами». Задачи подобного вида относятся к классу диофантовых уравнений. Задача «Переливайка» подробно обсуждается в статье А.В. Паньгина в этом номере журнала.

ТЕХНИЧЕСКИЕ ЗАМЕЧАНИЯ О ПРОВЕРКЕ РЕШЕНИЙ

Для обеспечения удобной проверки полученных от участников конкурса решений была написана отдельная программа, осуществляющая проверку решения по каждой задаче и составляющая отчет. Все решения помещались в одну папку, из которой программа последовательно просматривала каждое из них. Главной ее особенностью было то, что из файлов с решениями она считывала только конфигурацию, то есть для задачи Томсона – это расположение зарядов, для задачи о графах – информация о ребрах, для «переливайки» – последовательность выполненных участником переливаний. На основе полученной конфигурации программа проверяла ее правильность, например, для задачи Томсона проверялось наличие всех зарядов, для «переливайки» – возможность осуществления заданных действий. Таким образом, если файл с решениями был изменен или поврежден, то программа проверки обязательно это обнаружит. По каждому присланному решению составлялся отчет в виде XML файла, содержащего анкетные данные участника и его результат по каждой задаче. Те решения, которые оказались испорченными, помещались в отдельную папку. Так удалось быстро проверить все присланные участниками решения.

Лямов Андрей Геннадьевич,
студент СПбГТЭУ (ЛЭТИ),
Поздняков Сергей Николаевич,
профессор кафедры ВМ-2
СПбГТЭУ (ЛЭТИ),
Прокопенко Никита Юрьевич,
студент СПбГЭТУ (ЛЭТИ).