Приведённый выше материал был размещён в информационной образовательной среде кафедры общенаучных дисциплин АМТИ как результат студенческой научной работы. Вызвал большой интерес, много вопросов и предложений к авторам. Работа с этим документом сокурсников, студентов других направлений и курсов способствует развитию их научного творчества, самостоятельности в учебной и научной работе.

Список литературы

1. Смольняков И.М., Часов К.В. Формирование НИР студентов посредством информационной образовательной среды // Международный журнал экспериментального образования. – 2014. – № 7 – С. 105-106 URL:www.rae.ru/meo/?section=content&op=show_article&article_id= 5514 (дата обращения: 21.12.2014).

2. Смольняков И.М., Часов К.В. Некоторые свойства прогрессирующих последовательностей // Международный журнал экспериментального образования. – 2014. – № 7 – С. 106-107 URL: www.rae.ru/meo/?section=content&op=show_article&article_id=5515 (дата обращения: 21.12.2014).

3. Смольняков И.М., Часов К.В. Исследование различных последовательностей // Материалы VI Международной студенческой электронной научной конференции «Студенческий научный форум» URL: www.scienceforum.ru/2014/729/6698 (дата обращения: 21.12.2014).

4. Википедия https://ru.wikipedia.org/wiki/%D7%E8%F1%EB%E0_%D4%E8%E1%EE%ED%E0%F7%F7%E8 (дата обращения: 21.12.2014).

5. Википедия https://ru.wikipedia.org/wiki/%D0%97%D0%BE%D0%BB%D0%BE%D1%82%D0%BE%D0%B5_%D1%81%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5 (дата обращения: 21.12.2014).

ТЕХНОЛОГИЯ ПРИМЕНЕНИЯ МЕТОДОВ КОМБИНАТОРНОГО АНАЛИЗА В ГОЛОВОЛОМКАХ С ОПРЕДЕЛЕНИЕМ СОСТОЯНИЯ КЛЕТОЧНОГО ПОЛЯ

Сова С.А., Горовенко Л.А.

Армавирский механико-технологический институт (филиал) ФГБОУ ВПО «Кубанский государственный технологический университет», Армавир, Россия, sova.semyon@yandex.ru

Наука дает нам мощные инструменты, в виде различных технологий и методик, для решения тех или иных задач. Общество находится в постоянном развитии, и ему требуются всё новые решения для новых задач. Прогресс также не стоит на месте – старые методы подвергаются улучшениям, либо полностью заменяются новыми, более эффективными. Эти изменения можно хорошо проследить в головоломках. Основные математические методы решения всех головоломок являются методы комбинаторного анализа. В данной работе, приводится методика решения головоломки с определением состояния клеточного поля, а именно – японский кроссворд.

Правила игры

Японский кроссворд (или, иначе, японский рисунок) – это особый вид головоломки, в котором нужно, базируясь на кодовые числа-подсказки, разгадать зашифрованную картинку. Кодовые числа показывают, сколько слитных клеток данного цвета находятся в соответствующей колонке или строке.

Например, набор чисел 4, 1, и 3 в сетке японского кроссворда (рисунок 1) означает, что в этом ряду есть три группы: первая – из четырех, вторая – из одной, третья – из трех черных клеток.

Такие группы клеток обязательно должны разделяться как минимум одной пустой (белой) клеткой. Вся задача, в основном, и сводится к тому, чтобы узнать, сколько именно пустых клеток разделяют группы.

Существующие на настоящее время методы решения задачи

Существует множество методов решения задачи, но большинство из них связанны с конкретным случаем, то есть являются не чисто математическими, а скорее логическими.

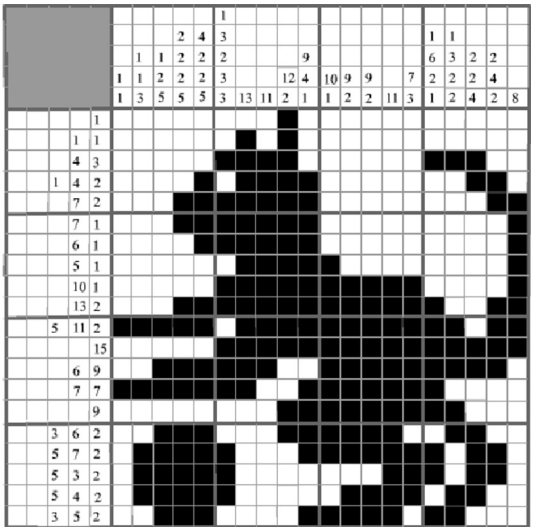

Рис. 1. Японский кроссворд

Рассмотрим один из подходов к математическому решению задачи.

Оценим стоимость полного перебора всех возможных размещений зарисованных клеточек с дальнейшей проверкой соответствия заданному на входе описанию. Например, для простенького кроссворда из 80 клеточек, из них 49 закрашенные, количество размещений $C_{80}^{49} = 1.5 * 10^{23}$, что для современных ПЭВМ означает время выполнения, приблизительно равное геологическому возрасту Земли. (http://www.thalion.kiev.ua/idx.php/128/665/article/)

Безусловно, значительную пользу может принести использование принципа «отбрасывать не отдельные размещения, а большие совокупности размещений». В частности, можно строить картинку последовательно по строкам: пробуя разместить зарисованные клеточки в первой строке, сразу проверять их на соответствие описанию первой строки, и переходить к попыткам размещение клеточек второй строки лишь тогда, когда найден один из удовлетворительных способов размещения в первой и т.д. Удовлетворительных размещений зарисованных клеточек в отдельной строке может быть много, и (как правило) лишь одно из них приводит к правильно разгаданному кроссворду. То есть, найдя допустимое размещение клеточек во всех строчках, нужно возвратиться к первой строке, попробовать найти другое допустимое размещение клеточек и снова рассмотреть все возможные размещения в следующих строках. То есть, получается алгоритм с возвращениями.

Для каждого из этих вариантов нужно проверить, не противоречит ли он уже известным окончательным состояниям клеточек. Говоря, что вариант размещения противоречит известным окончательным состояниям, мы имеем в виду примерно следующее: если из предшествующего анализа мы знаем, что 5-ая клеточка линии имеет окончательное состояние «не зарисована», то размещать блок длиной 4 в клеточках с 3-ей по 6-ую нельзя, так как это будет противоречить ранее найденному состоянию 5-ой клеточки. То есть, противоречивость варианта размещения блоков с окончательными состояниями появляется или когда пробуем разместить блок поверх гарантированно не зарисованной клеточки, или «симметричным» образом, когда какая-то клеточка имеет окончательное состояние «зарисованная» и оказывается в промежутке между блоками.

Реализация рекурсионной функции на языке Pascal (ссылка на статью выше):

```
function TryBlock(theblock,thestart:shortint):boolean;
var i,startnext:shortint;
   res:boolean;
Begin
  for i:=thestart to thestart+bl_len[theblock]-1 do
    if cells[i]=0 then begin TryBlock:=false; exit end;
  if theblock&lt;N then begin
    res:=false;
    for startnext:=thestart+bl_len[theblock]+1 to
                L-bl_len[theblock+1]+1 do begin
      if cells[startnext-1]=1 then break;
      if TryBlock(theblock+1,startnext) then begin
        res:=true;
(*какое-то непротиворечивое размещение дальнейших блоков существует*)
      end;
    end;
    TryBlock:=res
  end else begin (* theblock = N *)
    for i:=thestart+bl_len[theblock] to L do
      if cells[i]=1 then begin TryBlock:=false; exit end;
(*данное размещение последнего блока непротиворечиво*)
    TryBlock:=true
  end
End;
```

Таким образом, получаем функцию рекурсивного просмотра непротиворечивых размещений блоков в линии. Теперь уже можно включать в нее заполнение элементов can_one и can_zero для части линии, соответствующей положению текущего блока. Эти заполнения нужно включить в места, обозначенные в приведенном фрагменте комментариями.

Предлагаемый алгоритм

Представленный выше алгоритм, при всей своей продуманности и эффективности, имеет несколько недостатков. Во-первых, он довольно сложный для понимания и реализации, а во вторых не предусматривает вариант головоломки, каждая строка и каждый столбец которого состоит из 1 блока единичной длины.

Поэтому, мной разработан более простой алгоритм, основанный на переборе групп закрашенных клеток каждой строки.

Суть алгоритма состоит в том, что компьютер, начиная с первой строки, анализирует подсказки. Сначала, он определяет число групп в данной строке, и, на основе расположения чисел относительно друг друга, составляет группы клеток, находящиеся в данной строке.

К примеру, возьмем одну строку, ключ которой (здесь и далее, ключ – подсказки для данной строки (или столбца)) равен: 132. Количество столбцов равно девяти.

132 □□□□□□□□□

Соответственно, в данной строке находятся три группы клеток, состоящие из одной, трех и двух клеток, последовательно соответственно расположенные. Расположение клеток относительно друг друга фиксировано ключом, а так как между группами должна быть как минимум одна незакрашенная клетка, то каждую группу можно представить:

1: ■□

3: ■■■□

2: ■■

Общая сумма клеток, занимаемая всеми тремя группами, в соответствии с ключом равна: 2+4+2=8

А так как столбцов девять, то остается одна не заполненная клетка. Исходя из всего этого, можно понять, что всего комбинируемых групп 4 (3 группы известных клеток, и одна неизвестная), но при этом, первые три группы должны всегда оставаться в одинаковом положении относительно другу друга, чтобы не противоречить ключу.

Общее число перестановок 4 по 4 (т.е. полный перебор) равен 4! = 24.

Сразу откинем все, что противоречат взаимному расположению известных групп, и количество возможных расположений групп клеток, в соответствии с ключом, будет равно 4 (1320, 1032, 0132, 1302).

Дальше, мы выбираем одно из расположений и переходим к другой строке, где проделываем аналогичные действия, и так до самой последней строки.

Далее, когда клетки расставлены по местам в соответствии с их левыми ключами, начинаем проверять их на соответствие верхним ключам.

В случае несовпадения, начинаем процесс с самого начала, но уже с другим расположением групп. В конечном итоге, клетки будут расставлены в соответствии со всеми ключами, и кроссворд будет решен.

УЧАСТИЕ СТУДЕНТОВ В СОЗДАНИИ СОВРЕМЕННОЙ ОБРАЗОВАТЕЛЬНОЙ СРЕДЫ

Сова С.А., Шарнова В.А., Дедикова Т.Г.

ФГБОУ ВПО «Кубанский государственный технологический университет», Армавир, Россия, scharnova.veronika@yandex.ru

Участие студентов в создании современной образовательной среды [1-3] – это один из этапов, который позволяет формировать навыки программирования будущего бакалавра.

В работе [1] показано использование языка C# для нахождения корней химического уравнения. Например, для реакции окисления железа (II):

$$X_1FeSO_4 \mid X_2K_2Cr_2O_7 \mid X_3H_2SO_4 = X_4Fe_2(SO_4)_3 + X_5Cr_2(SO_4)_3 + X_6K_2SO_4 + X_7H_2O$$

Составляется система линейных уравнений. Число уравнений равно числу элементов, участвующих в реакции. Например, для железа:

$$X_1 = 2X_4$$

Программа вычисляет значения корней-коэффициентов, пользователь расставляет значения коэффициентов:

$$6FeSO_4 + K_2Cr_2O_7 + 7H_2SO_4 = 3Fe_2(SO_4)_3 + Cr_2(SO_4)_3 + K_2SO_4 + 7H_2O$$

Использование такого подхода позволяет учащемуся понимать связь между математическими выражениями и химическими символами.

Для решения экологических проблем города создаётся отдельная база данных, которую можно использовать в реальном проектировании. Например, для расчёта площади полигона для твёрдых бытовых отходов (рисунок). Из расчётов видна величина площади, а следовательно, можно реально указать на необходимость постройки завода по утилизации различных отходов.