DOI: 10.32517/2221-1993-2023-22-4-67-73

З. И. Зиннатуллин
Школа № 1353 имени генерала Д. Ф. Алексеева, г. Москва, Россия

О. М. Корчажкина
Федеральный исследовательский центр «Информатика и управление» Российской академии наук, г. Москва, Россия

ИСПОЛЬЗОВАНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ PYTHON ПРИ РЕШЕНИИ КРИПТОАРИФМЕТИЧЕСКИХ ЗАДАЧ В СРЕДНЕЙ ШКОЛЕ*

Аннотация

В статье приводится способ оптимизации процесса перебора вариантов при решении криптоарифметических (переборных) задач на сложение в столбик двух многоразрядных чисел с помощью языка программирования Python. Отмечается, что решение переборных задач может осуществляться по двум направлениям — комбинаторному (алгебраическому) и программному. Причем выяснилось, что алгебраические способы оптимизации нецелесообразно напрямую выражать в программных кодах, поскольку у языков программирования существуют собственные возможности оптимизации поисковых процедур.

Поэтому обучение способам оптимизации поиска при решении переборных задач следует осуществлять в два этапа. Комбинаторные способы целесообразно рассматривать в курсе информатики на уровне основного общего образования как средство формирования логического мышления и алгоритмических навыков учащихся. Тогда как к программному направлению переходят на более высокой ступени среднего общего образования — при освоении учащимися высокоуровневых языков программирования.

Описанные в статье стратегия оптимизации поиска при решении криптоарифметических задач и ее реализация на языке Python предполагают отказ от использования обычных циклических конструкций, а также известных способов обработки символьных данных. При составлении компьютерной программы «Слова.ру» применялись метод join и функция permutations модуля itertools из стандартной библиотеки языка Python, что значительно упростило запись программных кодов и сократило время перебора при поиске решения.

Ключевые слова: программирование, оптимизация поиска, переборные задачи, криптоарифметические задачи, язык программирования Python.

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_04-2023/

Контактная информация

Зиннатуллин Зуфар Ильдарович, учитель информатики, школа № 1353 имени генерала Д. Ф. Алексеева, г. Москва, Россия; *адрес*: 124498, Россия, г. Москва, г. Зеленоград, Берёзовая ал., д. 8А; *e-mail*: zinnatullin.z.i@mail.ru

Корчажкина Ольга Максимовна, канд. тех. наук, ст. научный сотрудник, Институт кибернетики и образовательной информатики им. А. И. Берга, Федеральный исследовательский центр «Информатика и управление» Российской академии наук. г. Москва, Россия; *адрес*: 119333, Россия, г. Москва, ул. Вавилова, д. 44, к. 2; *e-mail*: olgakomax@gmail.com

Z. I. Zinnatullin
School 1353, Moscow, Russia

O. M. Korchazhkina
Federal Research Centre "Computer Science and Control" of the Russian Academy of Sciences, Moscow, Russia

HOW TO USE THE PYTHON PROGRAMMING LANGUAGE IN FULFILLING CRYPTO-ARITHMETIC TASKS IN SECONDARY SCHOOL

Abstract

The article provides a way to optimize an enumeration process in the Python programming language while solving crypto-arithmetic (enumerative) tasks where two multi-digit numbers add to each other in a column. It is noted that there are two directions in fulfilling enumeration problems which suggest being optimized by combinatorial (algebraic) and programming methods. Moreover, it has been found out that it is impractical to express algebraic optimization methods directly in program codes, since programming languages have their own possibilities for optimizing search procedures.

Therefore, the process of learning how to optimize search while fulfilling enumerative tasks should be carried out in two stages. Combinatorial methods are the best means of forming students' logical mindsets and algorithmic skills. That is the reason why these methods should necessarily be included in the informatics course at the level of general secondary education. Then, at the advanced stage of training, or at the level of secondary general education, it is useful for students to move to the programming direction mastering high-level programming languages.

The optimizing strategy of solving crypto-arithmetic problems described in the article, and its implementation in the Python language involves omitting conventional cyclic structures, as well as widely known methods of processing character data. When compiling a computer program "Слова.ру", the authors applied the join method and the permutations function of the itertools module from the Python standard library. This choice has greatly simplified the recording of the program codes and reduced the time it took to search for the solution.

Keywords: programming, search optimization, enumerative problems, iterative problems, enumerative tasks, iterative tasks, cryptographic problems, crypto-arithmetic problems, cryptographic tasks, crypto-arithmetic tasks, Python programming language.

1. Введение

Решение криптоарифметических задач как подкласса переборных задач, доступных для освоения учащимися средней школы, укладывается в русло одного из направлений, развивающихся в рамках интегративного подхода к организации углубленного изучения информатики [9]. От учащихся, занимающихся по программам естественно-научного, технологического, социально-экономического или универсального профилей на ступенях основного общего и среднего общего образования, требуется повышенный уровень математической подготовки, который не только способствует развитию мышления через овладение способами логических рассуждений для построения алгоритмов решения задач из различных предметных областей, но и мотивирует к изучению современных языков программирования.

Владение языками программирования входит в набор компетенций, необходимых молодому человеку для социализации в современной цифровой среде, выполнения учебных и будущих профессиональных обязанностей, развития качеств *цифровой личности*, основанных на когнитивных способностях «к осуществлению декомпозиции, абстрагированию, распознаванию паттернов, алгоритмизации, моделированию, оценке» [1, с. 4]. Одним из таких языков является высокоуровневый скриптовый язык программирования Python, адаптированный к решению разнообразных задач и реализуемый на многих цифровых платформах [7, 11]. Универсальность и относительная простота этого языка позволяют включить его в программы по информатике на уровне основной и средней школы [3].

2. Криптоарифметические задачи и комбинаторные способы их решения

Криптоарифметические задачи составляют класс переборных задач, в которых обычно заданы два коротких (в пределах семи-восьми букв) слова, записанные в столбик через знак суммы. Как результат «сложения» этих слов приводится третье слово, состоящее из того же количества букв, что и наибольшее из первоначальных слов, или на одну букву (разряд) больше. Во всех трех словах буквы нужно заменить цифрами от 0 до 9 так, чтобы «буквенный» пример трансформировался в «цифровой» с соблюдением правил сложения столбиком. Как правило, такие задачи, поражая своей примитивной формой, на деле оказываются довольно сложными и требуют от исследователя развитого логического мышления и высокоуровневых алгоритмических навыков.

Например, **три классические криптоарифметические задачи**, предложенные выдающимся американским математиком, экономистом, психологом и кибернетиком Гербертом Саймоном (1916–2001), выглядят следующим образом [8, с. 37–41]:

Задача 1.

```
  SEND
+ MORE
-------
 MONEY
```

Задача 2.

```
  DONALD
+ GERALD
--------
 ROBERT
```

Задача 3.

```
  CROSS
+ ROADS
-------
 DANGER
```

Для решения задачи 2* путем «умственных усилий» ученый использовал четыре стратегии (см. подробнее [6, с. 147–148; 8, с. 39–42]):

- ***Стратегия 1 («лобовой» поиск).*** Если механически перебирать все возможные варианты замены 10 букв 10 цифрами, то таких переборов, основанных на методе «проб и ошибок» (назовем их «прямыми»), окажется 10! = 3 628 800.
- ***Стратегия 2 (ограниченный «лобовой» поиск).*** Если начать справа искать подстановки для всех букв в естественном порядке от 0 до 9, то число «прямых» переборов уменьшится до (10! – 8!) = = 3 628 800 – 40 320 = 3 588 480.
- ***Стратегия 3 (поиск и рассуждение).*** Если составить *дерево поиска* и из него путем рассуждений исключить тупиковые ветви, то число «прямых» переборов сократится до 68.
- ***Стратегия 4 (развитие стратегии поиска и рассуждения).*** Если обнаружить те столбцы (ветви), относительно которых уже имеется достаточная определенность, позволяющая производить новые подстановки, то число «прямых» переборов сократится до 3! = 6.

В статье [6] предложен визуальный способ оптимизации поиска, отличный от стратегий Г. Саймона и основанный на исключении поисковых цепочек в соответствии с правилами сложения столбиком, выраженными аналитически. Этот способ позволяет значительно сократить число «прямых» переборов в сравнении со стратегиями Г. Саймона.

По аналогии с задачами Саймона можно придумать множество задач, записанных не только на латинице, но и на кириллице, например:

Задача 4.

```
  ГОЛОС
+ СЛОВА
-------
 ПЕСНЯ
```

Задача 5.

```
  САДИК
+ ШКОЛА
-------
 МОСКВА
```

Задача 6.

```
  ВЕТКА
+ ПАЛКА
-------
 СТВОЛ
```

Тем не менее практика показала, что комбинаторные (алгебраические) *методы решения криптоарифметических задач являются трудозатратными и в плане времени, и в плане умственных усилий*. С одной стороны, их применение на уроках информатики в средней школе при освоении навыков алгоритмизации и оптимизации найденных способов решения приводит к активизации логического мышления учащихся. С другой стороны, представляется совершенно естественным переход к следующему после алгоритмизации этапу решения переборных задач — программированию на одном из высокоуровневых языков, изучаемых в средней школе, например на языке Python.

---

* Г. Саймон формулирует дополнительное начальное условие в задаче с мужскими именами: D = 5. В то же время для машинного решения задачи задавать такое условие не потребовалось (см. раздел 3 данной статьи «Программа для решения криптоарифметических задач на языке Python» и рисунок 2, *а*). Кроме того, отметим, что снижение числа переборов для каждой стратегии Г. Саймона обусловлено возрастающим объемом «умственных усилий».

3. Программа для решения криптоарифметических задач на языке Python

Для решения криптоарифметических задач по типу Г. Саймона была составлена **программа «Слова.ру»**, скрипт которой на языке программирования Python представлен на рисунке 1.

Опишем основные командные модули программы.

`import itertools` — подключение модулей `itertools` из стандартной библиотеки Python, которые позволяют оптимизировать решение переборных задач и тем самым упростить алгоритм поиска.

Стандартный ввод данных:

```
s1=input('Введите первое слово: ')
s2=input('Введите второе слово: ')
s3=input('Введите слово-результат: ');
```

Слова могут вводиться как прописными, так и строчными буквами — на латинице, кириллице, в виде иероглифов или арабской вязи, со смыслом или в хаотичном порядке, а также с помощью любых других символов, знаков, операторов или фигур.

`s4=s1+s2+s3` — конвертация (объединение) слов в строку.

`s5` — переменная для последующего вывода ответа задачи (всех возможных комбинаций) в цифрах.

`n=0` — начальное число комбинаций.

`count=0` — обнуление счетчика.

Печать слов в строковой записи:

```
print(s1,'+',s2,'=',s3);
```

Устранение дублирующих букв (символов) в строке `s4` для замены их цифрами:

```
s=''.join(dict.fromkeys(s4));
```

Оператор `join` позволяет избавиться от повторяющихся букв (символов) и конвертировать список букв в словах для сохранения и дальнейшего преобразования их в цифры*.

Первичная проверка соотношения длины слов (количества букв в словах):

```
if len(s3)<len(s1)
  or len(s3)<len(s2)
  or len(s)>10;
```

При выполнении хотя бы одного из перечисленных выше условий решение задачи прекращается и на экран выводится результат:

* С другими способами обработки символьных данных можно познакомиться в статье [2].

Слова.py - C:\Users\olgak\OneDrive\Рабочий стол\Слова.py (3.11.2)

File Edit Format Run Options Window Help

```
import itertools
s1=input('Введите первое слово: ')
s2=input('Введите второе слово: ')
s3=input('Введите слово-результат: ')
s4=s1+s2+s3
s5=''
n=0
count=0
print(s1,'+',s2,'=',s3)
s= ''.join(dict.fromkeys(s4)) #убрали дублирующие символы
if len(s3)<len(s1) or len(s3)<len(s2) or len(s)>10:
    print('Невозможно составить комбинации из этого набора букв,т.к.', len(s),'>10')
else:
    cifra=[0,1,2,3,4,5,6,7,8,9]
    perebor= itertools.permutations(cifra,len(s))#перебор цифр по количеству символов
    for i in perebor:
        rez = ''.join(str(x) for x in i)
        for j in s4:
            for k in range(len(s)):
                count+=1
                if j==s[k] :
                    s5+=rez[k]
        if s5[0]!='0' and s5[len(s1)]!='0' and s5[len(s1)+len(s2)]!='0'\
            and int(s5[:len(s1)])+int(s5[len(s1):len(s1)+len(s2)])==int(s5[len(s1)+len(s2):]):
                print (s5[:len(s1)],'+',s5[len(s1):len(s1)+len(s2)],'=',s5[len(s1)+len(s2):])
                n+=1
        s5=''
    if n==0:
        print('Перебор не дал результата')
    else:
        print ('Количество найденных комбинаций -', n)
print('Количество перебранных комбинаций -',count)
```

Ln: 23 Col: 0

Рис. 1. Скрипт программы «Слова.ру» на языке Python

```
print('Невозможно составить комбинации из
  этого набора букв, т.к.', len(s), '> 10');
```

Значение `len(s)` указывает на число неповторяющихся букв во всех трех словах, которых должно быть не более 10.

В противном случае формируется массив цифр, участвующих в дальнейшей замене букв:

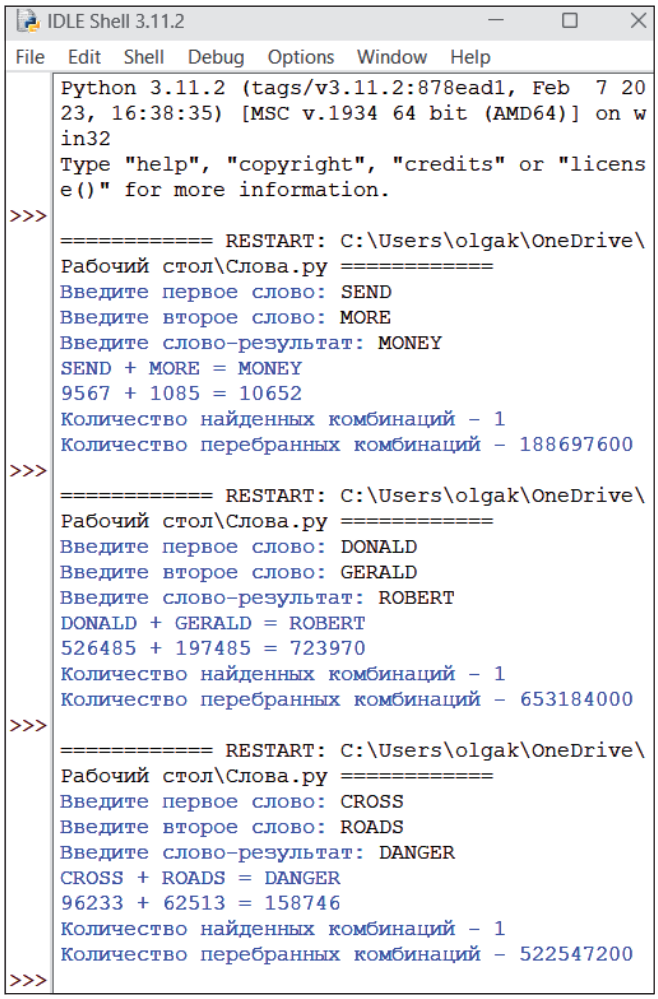
```
cifra=[0,1,2,3,4,5,6,7,8,9];
```

Функция `permutations` модуля `itertools` определяет, какой массив цифр использовать и какое количество цифр оставить в объекте `perebor`:

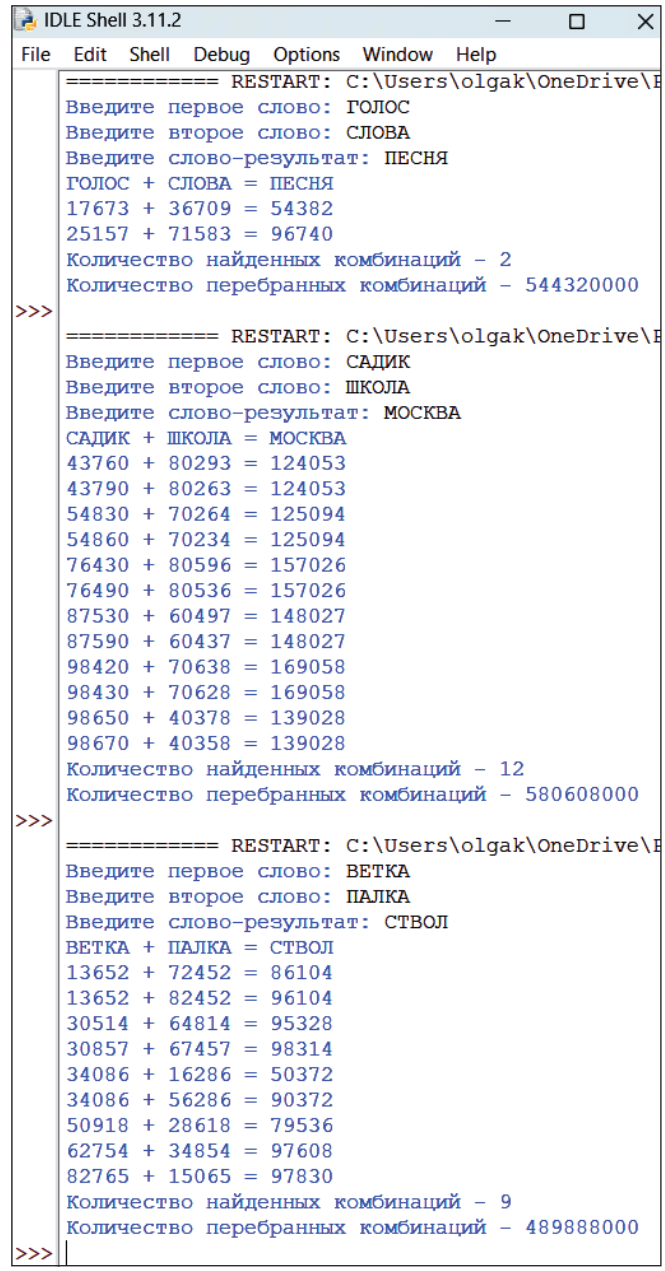
```
perebor=itertools.permutations(cifra,len(s));
```

То есть с помощью функции `permutations` осуществляется перебор цифр по количеству символов (длине слов) и создаются все возможные комбинации, возникающие при замене букв цифрами; это позволяет многократно уменьшить время для перебора вариантов по сравнению с обычным использованием циклов и условия.

Каждая комбинация цифр (вариант перебора) разбивается на числа в соответствии с длиной слов, и формируется значение результата `s5`; при этом цикл `for i in perebor` создает новые массивы `i` с различными



а



б

Рис. 2. Результаты решения криптоарифметических задач 1–3 (а) и 4–6 (б)
(число комбинаций для разных задач может быть весьма значительным, например:
***ах** + **ох** = **ура** — 20 комбинаций;*
***утро** + **кофе** = **лень** — 68 комбинаций;*
***кофе** + **лень** = **утро** — 72 комбинации;*
***Обь** + **Лена** = **реки** — 144 комбинации;*
***два** + **три** = **пять** — 180 комбинаций)*

комбинациями, а результирующим является массив значений [k]:

```
for i in perebor:
  rez = ''.join(str(x) for x in i)
  for j in s4:
    for k in range(len(s)):
count+=1
      if j==s[k] :
        s5+=rez[k];
```

В условии проверяется справедливость математического выражения — равна ли сумма числовых выражений двух первых слов числовому выражению третьего слова:

```
if s5[0]!='0' and s5[len(s1)]!='0' and
s5[len(s1)+len(s2)]!='0'\
    and int(s5[:len(s1)])+int(s5[len(s1):
      len(s1)+len(s2)])==int(s5[len(s1)+
      len(s2):]):
    print (s5[:len(s1)],'+',s5[len(s1):
      len(s1)+len(s2)],'=',s5[len(s1)+
      len(s2):])
s5=''
```

Вывод результата на экран (наличие/отсутствие результата, число комбинаций, число выполненных переборов):

```
if n==0:
  print ('Перебор не дал результата')
else:
  print ('Количество найденных комбинаций -
      ', n)
print('Количество перебранных комбинаций
  -',count)
```

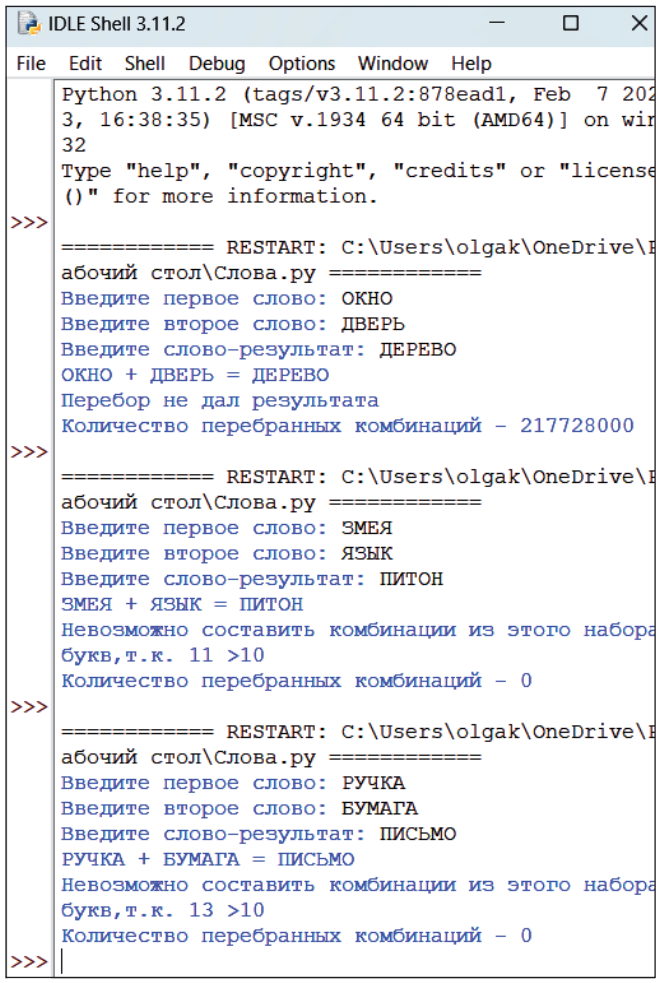
На рисунке 2, *а* приведены результаты решения трех задач Г. Саймона (1–3), на рисунке 2, *б* — трех задач на кириллице (4–6) из раздела 2 данной статьи.

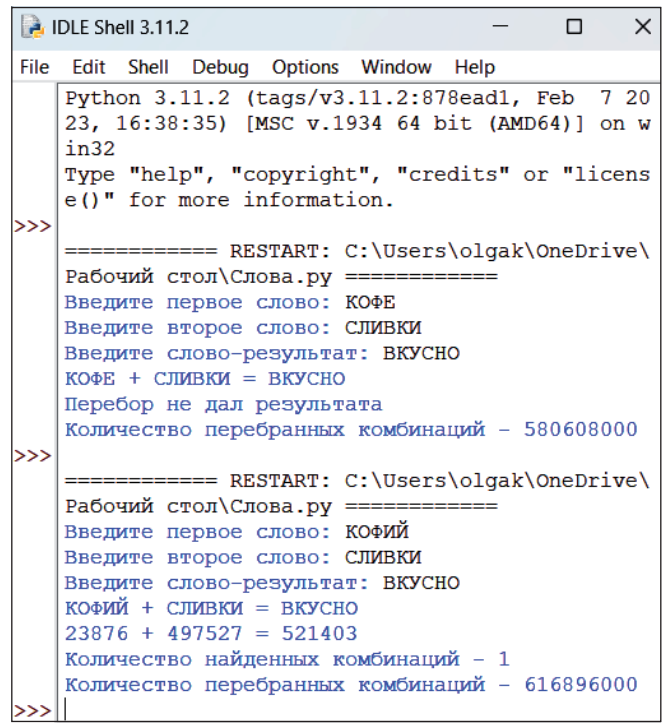
Отметим два важных момента, на которые следует обратить внимание при использовании программы «Слова.ру».

Первый момент связан с выводом на экран неблагоприятного результата. В программе предусмотрены две формулировки: `Невозможно составить комбинации из этого набора букв, т.к.', len(s), '> 10'` и `Перебор не дал результата` (рис. 3, *а*).

Первая формулировка выдается программой, когда при проверке количества неповторяющихся букв во всех трех словах выясняется, что это число превышает 10 — набор цифр от 0 до 9. В этом случае программа не приступает к поиску вариантов.

Вторая формулировка показывает, что число неповторяющихся букв менее 10 и программа провела пол-





а *б*

Рис. 3. Варианты демонстрации неблагоприятного исхода при переборе вариантов (а) и корректировка результата путем изменения условий поиска (б)

ный перебор вариантов, однако он также не увенчался успехом.

В ряде случаев неблагоприятный исход поиска может быть скорректирован изменением начальных условий задачи (рис. 3, *б*).

После каждого полученного решения с помощью функции `print('Количество перебранных комбинаций -',count)` программа выводит на экран количество совершенных переборов, которое чаще всего выражается девятизначным числом. Тем не менее огромное число переборов не является показателем неэффективной работы программы. Например, время решения задачи 2 с числом переборов 653 184 000 составляет 2 мин 37,71 с ≈ 160 с. Если за показатель эффективности работы программы принять скорость обработки данных, то для задачи 2 она равна 4 082 400 операций в секунду. Для задачи «Обь + Лена = реки» с позитивным ответом в 144 комбинации скорость обработки данных равна 3 441 103 операций в секунду, а для задачи «два + три = пять» с позитивным ответом в 180 комбинаций — 3 201 882 операций в секунду. Когда программа к перебору не приступает, на экран выводятся соответствующие формулировки: `Невозможно составить комбинации из этого набора букв, т.к.', len(s), '> 10'` и `Количество перебранных комбинаций - 0` (см. рис. 3, *а*).

Второй момент касается непосредственно оптимизации процедуры поиска (перебора возможных комбинаций). При работе по автоматизации решения криптоарифметических задач на языке Python обнаружилась одна интересная особенность, которая, по-видимому, будет наблюдаться и при использовании любого другого языка программирования.

Как отмечалось выше, применение предложенных в работах [5, 6] «ручных» оптимизационных алгоритмов поиска на примере решения сложных переборных задач, как и алгоритмов самого Г. Саймона [8], оказалось весьма продуктивным в плане формирования у школьников алгоритмических навыков. С этой целью потребовался переход от способа «лобового» поиска, изначально отвергнутого как нерациональный, к комбинаторным процедурам оптимизации, основанным на закономерностях переноса разрядов при суммировании, которые выражались линейными алгебраическими функциями*.

Однако воплощение в программном варианте комбинаторных способов оптимизации при использовании обычных циклических конструкций, например цикла For [4] и условия, оказалось настолько трудоемким, что от них пришлось отказаться и искать более рациональные пути перебора возможных вариантов. Так, для варианта с комбинациями из четырех цифр часть программы, состоящая из циклов и условия, выглядела бы следующим образом:

```
for a1 in cifra:
  for a2 in cifra:
    for a3 in cifra:
      for a4 in cifra:
        if a1!=a2!=a3!=a4:
          a=str(a1)+str(a2)+str(a3)+str(4).
```

Кроме того, при подобном использовании циклов пришлось бы всякий раз задавать определенное количество вложенных циклов в зависимости от числа символов (букв), а время работы программы для одной строки из четырех цифр составило бы 15–16 мс, что при немалом числе переборов значительно повысило бы общее время исполнения программы.

Чтобы избежать непродуктивного использования времени и добиться оптимизации вычислительных процедур, при составлении компьютерной программы «Слова.ру» на языке Python были применены метод `join` и модуль `itertools`.

Метод `join` позволил оперативно исключить дублирующие символы и оставить только набор неповторяющихся букв для дальнейшей замены их цифрами. Кроме того, с помощью метода `join` список символов (букв) был объединен в строку, в которой производилась замена букв на цифры. А выражение приведенной выше циклической конструкции в виде функции `permutations` модуля `itertools`:

```
perebor=itertools.permutations(cifra,4);
  for i in perebor:
```

позволило не привязываться к количеству символов в начальных словах, что значительно сократило время перебора комбинаций (до 1 мс для строки из четырех цифр).

4. Заключение

При создании программы «Слова.ру» для решения криптоарифметических задач использовался ряд модулей, функций и процедур стандартной библиотеки Python, что позволило оптимизировать процесс перебора возможных вариантов и продемонстрировало богатые обучающие возможности языка для формирования и развития как алгоритмических, так и программистских навыков и умений учащихся.

Однако следует признать, что значительная часть стандартной библиотеки Python, охватывающей несколько тысяч компонентов, не может быть по понятным причинам включена в программу по информатике даже на уровне профильной школы. Тем не менее использование при подготовке к ЕГЭ или к олимпиадам по криптографии некоторых функций, методов и модулей стандартной библиотеки Python, выходящих за рамки школьной программы, для решения переборных задач с элементами оптимизации могло бы заинтересовать старшеклассников и стать дополнительным стимулом для овладения ими богатым и содержательным инструментарием языка программирования Python.

Список источников

1. *Босова Л. Л.* Программирование как инструмент формирования вычислительного мышления обучающихся // Информатика в школе. 2020. № 10. С. 4–10. EDN: GURIPH. DOI: 10.32517/2221-1993-2020-19-10-4-10.

* Сходная проблема рассматривается, например, в статье [10], где автор сравнивает два способа выполнения задания № 8 ЕГЭ по информатике и комбинаторным способом, и способом программирования на Python. При этом отмечается, что решение задач методом программирования является более эффективным, поскольку этот метод базируется на легко запоминаемых конструкциях и без особых усилий позволяет учесть ограничения, накладываемые в условиях задачи.

2. *Босова Л. Л.*, *Аквилянов Н. А.* Обработка символьных данных: от простого к сложному // Информатика в школе. 2022. № 6. С. 5–11. EDN: DVGZGT. DOI: 10.32517/2221-1993-2022-21-6-5-11.

3. *Босова Л. Л., Павлов Д. И., Ткач Т. В., Бутарев К. В.* О содержании «Базового курса информатики» в проекте «ИТ-класс в московской школе» // Информатика в школе. 2021. № 10. С. 9–18. EDN: FVCHQS. DOI: 10.32517/2221-1993-2021-20-10-9-18

4. *Заславская О. Ю.*, *Любутов О. Д.* Особенности методики преподавания цикла For в языке Python в средней школе // Информатика в школе. 2022. № 4. С. 70–73. EDN: KGUBSE. DOI: 10.32517/2221-1993-2022-21-4-70-73.

5. *Корчажкина О. М.* Криптоарифметические и другие переборные задачи на уроках информатики // Актуальные проблемы методики обучения информатике и математике в современной школе: Материалы международной научно-практической интернет-конференции (г. Москва, 18–24 апреля 2022 г.). М.: МГПУ, 2022. С. 159–170. EDN: IRPYVZ.

6. *Корчажкина О. М.* Оптимизация поиска при решении переборных задач в углубленном курсе информатики на уровне основного общего образования // Системы и средства информатики. 2022. Т. 32. № 4. С. 145–156. EDN: PFCFPE. DOI: 10.14357/08696527220414.

7. *Пэйн Б.* Python для детей и родителей: пер. с англ. М. А. Райтмана. М.: Издательство «Э», 2017. 352 с.

8. *Саймон Г.* Науки об искусственном: пер. с англ. 2-е изд. М.: Едиториал УРСС, 2004. 144 с.

9. *Самылкина Н. Н.* Организация углубленного обучения информатике на основе интегративного подхода: монография. М.: МПГУ, 2020. 346 с.

10. *Смольняков В. Г.* Методика выполнения задания № 8 ЕГЭ по информатике и ИКТ комбинаторным способом и способом программирования на Python // Информатика в школе. 2021. № 10. С. 54–60. EDN: JIOUDT. DOI: 10.32517/2221-1993-2021-20-10-54-60.

11. *Щерба А. В.* Программирование на Python®: Первые шаги. М.: Лаборатория знаний, 2022. 250 с.