DOI: 10.32517/2221-1993-2023-22-3-81-85

М. С. Долинский
Гомельский государственный университет имени Франциска Скорины, г. Гомель, Беларусь

ЗАДАЧИ НА РЕКУРРЕНТНОЕ СООТНОШЕНИЕ «ВСЕ СУММЫ»*

Аннотация

В данной статье на примере решения нескольких задач проиллюстрирована методика изучения темы «Рекуррентное соотношение “все суммы”» при подготовке школьников к олимпиадам по информатике. Изучение основано на последовательном решении усложняющихся задач. Для каждой задачи приводятся следующие материалы: условие задачи, идея решения с предложением придумать самостоятельно реализацию, решение на языке программирования Pascal. Серьезной технической основой является разработанная под управлением автора инструментальная система дистанционного обучения (http://dl.gsu.by), которая позволяет: предложить ученику условие задачи; отправить решение на проверку; получить от системы вердикт — правильное или неправильное решение; для неправильных решений указывается номер теста, на котором решение не прошло. Ученик может взять тест (входные и выходные данные), на котором не прошло его решение, разобраться, в чем ошибка в его программе, исправить и послать решение повторно. Кроме того, для каждой задачи есть ссылка по ней на тему в форуме, где можно задать вопрос по решению этой задачи и/или почитать ответ, если вопросы уже задавались ранее.

Ключевые слова: рекуррентное соотношение, все суммы, олимпиады по информатике, инструментальная система дистанционного обучения.

1. Введение

Рекуррентные соотношения и динамическое программирование — темы, которые не только являются важными для подготовки школьников к олимпиадам по программированию, но и являются востребованными в среде профессиональных программистов. Поэтому данная тема широко освещается в технической литературе, в том числе:

- в книгах по общей теории алгоритмов, включающих разделы о рекуррентных соотношениях и динамическом программировании [2, 6, 7, 10, 14, 15, 17–19, 21–23];
- книгах и статьях, посвященных исключительно рекуррентным соотношениям и динамическому программированию [1, 13, 16, 22, 24, 25];
- сборниках задач олимпиадной тематики, включающих задачи на рекуррентные соотношения и динамическое программирование [4, 5, 8, 9];
- сборниках тестовых заданий при собеседованиях перед поступлением на работу, включающих задачи на рекуррентные соотношения и динамическое программирование [11, 12];
- книгах, содержащих рекомендации для практикующих программистов и включающих рекуррентные соотношения и динамическое программирование [3, 20].

Автор много лет занимается подготовкой к олимпиадам по информатике школьников, начиная с самого раннего возраста. Как следствие, уже в V—VIII классах появляются дети, которым требуется объяснять темы

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_03-2023/

Контактная информация

Долинский Михаил Семенович, канд. тех. наук, доцент, доцент кафедры математических проблем управления и информатики, Гомельский государственный университет имени Франциска Скорины, г. Гомель, Беларусь; *адрес:* 246000, Республика Беларусь, г. Гомель, ул. Советская, д. 104; *e-mail:* dolinsky@gsu.by

M. S. Dolinsky
Francisk Skorina Gomel State University, Belarus

TASKS ON THE RECURRENCE RELATION "ALL SUMS"

Abstract

In the article, using the example of solving few problems, the methodology for studying the theme "Problems on the recurrence relation "All sums" is illustrated in preparing schoolchildren for Olympiads in informatics. The study is based on the sequential solution of increasingly complex tasks. For each problem, the following materials are given: the formulation of the problem, the idea of a solution with a proposal to come up with an implementation on their own, the solution in the Pascal programming language. Distance learning system (http://dl.gsu.by) is the effective technical base for teaching. The system allows to offer for a student a formulation of the problem; to submit the solution for review; to get a verdict from the system — a correct or incorrect solution; for incorrect solution, the number of the test on which the solution did not pass is indicated. A student can take a test (input and output data), on which his solution did not pass, figure out what the error is in his program, correct and send the solution again. In addition, for each task there is a link on it to the topic in the forum at site, where you can ask a question on solving this problem and / or read the answer if the questions have already been asked before.

Keywords: recurrence relation, all sums, Olympiads in informatics, distance learning tools.

«Рекуррентные соотношения» и «Динамическое программирование». Автор выбрал подход «от простого к сложному», когда школьнику предлагаются подтемы в порядке возрастания сложности. А внутри каждой темы — задачи, также в порядке возрастания сложности. Кроме того, сначала предлагается изучение известных стандартных алгоритмов. С одной стороны, такой подход расширяет круг задач, которые смогут решать школьники, а с другой — проводит пропедевтическую работу по развитию способностей придумывать решения нестандартных задач. Первой такой стандартной подтемой является рекуррентное соотношение «все суммы», которой и посвящена данная статья.

Проверка решений осуществляется автоматически на сайте DL.GSU.BY. Вдумчивым читателям предлагается после чтения условия задачи/подзадачи попытаться решить ее самостоятельно. Все задачи находятся в учебном курсе «Олимпиады по информатике».

Прежде чем приступить к рассмотрению задач, напомним, что такое рекуррентное соотношение: в математике ***рекуррентное соотношение*** *— это уравнение, согласно которому n-й член последовательности чисел равен некоторой комбинации предыдущих членов.*

2. Задача «Собиратель» (Гомельская городская олимпиада для I—IX классов, 2010 год)

Вовочка собирает деньги разных стран. Сейчас в его коллекции насчитывается N купюр. Мальчик часто оценивает свою коллекцию. Он считает, что чем больше различных сумм можно составить, используя купюры, тем ценнее коллекция.

Вам известны достоинства каждой из купюр. Помогите Вовочке определить ценность его коллекции.

Формат ввода.
N — количество купюр
$a[1]$
...
$a[N]$
Здесь $a[i]$ — достоинство i-й купюры.

Ограничения.
$1 \le N \le 100$;
$1 \le a[i] \le 1000$.

Формат вывода.
Одно число — количество различных сумм, которые можно составить.

Пример ввода	*Пример вывода*
5 2 2 1 4 5	14

Идея решения задачи.
Это стандартная задача на рекуррентное соотношение «все суммы». Заводим массив s, в котором на i-й позиции стоит 0, если сумма i еще не собрана, и 1, если сумма i уже собиралась. Сколько элементов в этом массиве (константа *MaxS*)? У нас 100 целых чисел, каждое из которых имеет значение от 1 до 1000, т. е. максимальное число, которое можно получить в виде суммы исходных чисел, — это $100 \cdot 1000 = 100\,000$.

Далее в 0-й элемент массива заносим 1 (сумму 0 собираем, если не взяли ни одного числа). Затем для каждого считанного числа a пробегаем массив s с конца к началу и, если $s[i] = 1$, заносим 1 в элемент $s[i + a]$. То есть, если раньше мы смогли собрать сумму i, то с новым числом a мы можем собрать и сумму $i + a$.

Полный текст решения приведен в листинге 1.

Почему мы идем от конца к началу, а не от начала к концу?

```
const
  MaxS = 100000;
var
  s         : array [0..MaxS] of longint;
  N,i,j,k,a : longint;
begin
  readln(N);
  s[0]:=1; for i:=1 to MaxS do s[i]:=0;
  for i:=1 to N do
    begin
      readln(a);
      for j:=MaxS-a downto 0 do
        if s[j]=1 then s[j+a]:=1;
    end;
  k:=0;
  for i:=1 to MaxS do
    if s[i]=1 then inc(k);
  writeln(k);
end.
```

Листинг 1

```
const
  MaxS = 100000;
var
  s         : array [0..MaxS+1000] of
longint;
  N,i,j,k,a : longint;
begin
  readln(N);
  s[0]:=1; for i:=1 to MaxS do s[i]:=0;
  for i:=1 to N do
    begin
      readln(a);
      for j:=0 to MaxS-a do
        if (s[j]>0) and (s[j]<=i)
                    and (s[j+a]=0)
          then s[j+a]:=i+1;
    end;
  k:=0;
  for i:=1 to MaxS do
    if s[i]>1 then inc(k);
  writeln(k);
end.
```

Листинг 2

Потому что, если мы будем идти от начала, нужно как-то *отличать* суммы, полученные с текущим числом, от сумм, полученных без него, чтобы *не прибавлять* числа повторно к суммам, которые получены уже с его прибавлением.

Если требуется все-таки идти вперед (а это вскоре потребуется), то можно использовать *массив с версиями*. То есть, обрабатывая входное число с номером i, мы заносим в нужную позицию массива s не 1, как раньше, а $i + 1$. Тогда получается, что все суммы, полученные *раньше*, будут помечены числами от 1 до i включительно, а все новые суммы (полученные с использованием текущего числа $a[i]$) помечены числом $i + 1$ — это и позволит нам их различать.

Тогда программа будет выглядеть так, как представлено в листинге 2.

3. Задача «Натуральные числа» (Гомельская областная олимпиада для I—IX классов, 2012 год)

Дано N целых чисел. Необходимо определить наименьшее натуральное число, которое невозможно представить в виде суммы данных чисел (сумма может состоять и из одного слагаемого).

Формат ввода.
N — количество чисел. Во второй строке через пробел следуют N чисел $a[i]$.

Ограничения.
$1 \le N \le 100$;
$1 \le a[i] \le 1000$.

Формат вывода.
Ans — наименьшее натуральное число, которое не представимо суммой заданных. чисел.

Пример ввода	*Пример вывода*
5 1 2 4 2 5	15

```
const
  MaxS = 100001;
var
  s       : array [0..MaxS] of longint;
  N,i,j,a : longint;
begin
  readln(N);
  s[0]:=1; for i:=1 to MaxS do s[i]:=0;
  for i:=1 to N do
    begin
      read(a);
      for j:=MaxS-a downto 0 do
        if s[j]=1 then s[j+a]:=1;
    end;
  i:=1;
  while (i<=MaxS) and (s[i]=1) do inc(i);
  writeln(i);
end.
```

Листинг 3

Идея решения задачи.
Заполним массив «все суммы» (подробное описание алгоритма дано в предыдущей задаче) и найдем первый с начала элемент, равный 0. Заметим, что на всякий случай в массиве s заводим на один элемент больше (с конца). Если все числа от 1 до 100 000 могут быть составлены суммами исходных $a[i]$, то тогда ответ будет 100 001.

Полный текст решения приведен в листинге 3.

4. Задача «Собиратель-2» (Гомельская областная олимпиада для I—IX классов, 2010 год)

Вовочка собирает деньги разных стран. Сейчас в его коллекции насчитывается N купюр. Мальчик часто оценивает свою коллекцию. Вовочка постоянно совершенствует способ оценивания. Сейчас он считает, что, чем больше различных сумм-разностей можно составить, используя купюры, тем ценнее коллекция.

Под суммой-разностью понимается то, что достоинства купюр можно брать со знаком минус. Например, из купюр достоинством 2 и 3 можно составить следующие суммы-разности: 2, 3, 5 (2 + 3) и 1 (3 – 2). Вовочке интересны только положительные суммы-разности, поэтому в приведенном примере –1 (2 – 3) не следует учитывать при подсчете ценности.

Вам известны достоинства каждой из купюр. Помогите Вовочке оценить ценность его коллекции.

Формат ввода.
N — количество купюр.
$a[1]$
...
$a[N]$
Здесь $a[i]$ — достоинство i-й купюры.

Ограничения.
$1 \le N \le 100$;
$1 \le a[i] \le 1000$.

Формат вывода.
Одно число — количество различных сумм-разностей, которые можно составить.

Пример ввода	*Пример вывода*
4 2 2 3 5	11

Идея решения задачи.
Поскольку числа можно вычитать, то:
1) могут получаться нужные числа из ранее отрицательных сумм; потому нужно изменить диапазон массива на: от $-MaxS$ до $MaxS$;
2) нам не поможет движение от конца к началу для различения новых и старых сумм; поэтому требуется использование массива с версиями (более подробное описание представлено в решении первой задачи).

На этапе обработки числа с номером i мы будем заносить новые пометки числом $i + 1$. Тогда все $s[i]$ со

значениями от 1 до *i* включительно — это старые суммы, а со значением *i* + 1 — новые суммы.

Полный текст решения приведен в листинге 4.

```
const
  MaxS = 100000;
var
  s       : array [-MaxS..MaxS] of longint;
  N,i,j,k,a : longint;

begin
  readln(N);
  for i:=-MaxS to MaxS do s[i]:=0; s[0]:=1;
  for i:=1 to N do
    begin
      readln(a);
      for j:=-MaxS to MaxS do
        if (s[j]>0) and (s[j]<=i)
          then begin
                 if (j+a<= MaxS) and
                    (s[j+a]=0)
                      then s[j+a]:=i+1;
                 if (j-a>=-MaxS) and
                    (s[j-a]=0)
                      then s[j-a]:=i+1;
               end;
    end;
  k:=0;
  for i:=1 to MaxS do
    if s[i]>0 then inc(k);
  writeln(k);
end.
```

Листинг 4

5. Задача «Поиск числа» (Гомельская городская олимпиада для X—XI классов, 2012 год)

Задано *N* целых чисел. Необходимо найти такое наименьшее положительное число, большее 1, у которого нет общих делителей (кроме 1) ни с одним из заданных чисел.

Формат ввода.
В первой строке задано число *N*. Во второй строке находятся сами *N* чисел.

Ограничения.
$1 \le N \le 1000$;
все заданные числа — больше 1 и меньше 10^6.

Формат вывода.
Одно число — наименьшее число, удовлетворяющее описанным требованиям.

Пример ввода	*Пример вывода*
6 4 12 15 77 65 34	19

Идея решения задачи.
Составим массив *d* всех делителей заданных чисел *a*[*i*]. Поскольку сами числа не превышают 10^6, то все их делители можем искать максимум до:

```
sqrt(10^6)=10^3=1000
```

так как бóльшие делители будем получать как результат деления *a*[*i*] на найденный делитель. А затем, аналогично тому, как это делается в рекуррентном соотношении «все суммы», заполним массив «все произведения»:

```
s[1]:=1;
s[i*j]:=1
```

Полный текст решения приведен в листинге 5.

```
const
  MaxS = 1000000;
var
  s,d       : array [1..MaxS] of longint;
  N,i,j,k,a : longint;
begin
  readln(N);
  for i:=1 to MaxS do d[i]:=0;
  for i:=1 to N do
    begin
      read(a);
      for j:=2 to round(sqrt(a)) do
        if (a mod j)=0
          then begin
                 d[j]:=1; d[a div j]:=1;
               end;
    end;
  s[1]:=1; for i:=2 to MaxS do s[i]:=0;
  for i:=2 to MaxD do
    if d[i]<>0
      then for j:=1 to (MaxS div i) do
             if s[j]=1 then s[j*i]:=1;
  i:=2;
  while (i<=MaxS) and (s[i]=1) do inc(i);
  writeln(i);
end.
```

Листинг 5

6. Заключение

В данной статье на примере решения нескольких задач рассмотрена методика изучения темы «Рекуррентное соотношение “все суммы”», предлагающая, по мнению автора, наиболее простой способ постепенного осознания механизма рекуррентного соотношения «все суммы» и способа решения задач с его помощью. Методика включает в себя последовательность задач в порядке возрастания сложности, снабженных, где необходимо, предварительными общими пояснениями и последующими полными решениями предлагаемых задач. Серьезной технической основой является разработанная под управлением автора инструментальная система дистанционного обучения (http://dl.gsu.by), которая позволяет максимально автоматизировать процесс предъявления условий задач и проверки их решений.

Список источников

1. *Беллман Р., Энджел Э.* Динамическое программирование и уравнения в частных производных. М.: Мир, 1974. 254 с.

2. *Дасгупта С., Пападимитриу Х., Вазирани У.* Алгоритмы. М.: МЦНМО, 2019. 320 с.

3. *Довгалюк П. М.* Динамическое программирование и все-все-все: Как решать олимпиадные и «жизненные» программистские задачи. М.: ЛЕНАРД, 2021. 200 с.

4. *Златопольский Д. М.* 1400 задач по программированию. М.: ДМК Пресс, 2019. 192 с.

5. *Калихман И. Л., Войтенко М. А.* Динамическое программирование в примерах и задачах. М.: Высшая школа, 1978. 125 с.

6. *Клейнберг Дж., Тардос Е.* Алгоритмы: разработка и применение. СПб.: Питер, 2016. 800 с.

7. *Кормен Т.* Алгоритмы: построение и анализ. М.: Вильямс, 2020. 1328 с.

8. *Котов В. М.* Сборник задач по теории алгоритмов. Минск: БГУ, 2017. 183 с.

9. *Лааксонен А.* Олимпиадное программирование / пер. с англ. А. А. Слинкин. М.: ДМК Пресс, 2018. 300 с.

10. *Луридас П.* Алгоритмы для начинающих. Теория и практика для разработчика. М.: ЭКСМО, 2018. 608 с.

11. *Макдауэл Г. Л.* Карьера программиста. СПб.: Питер, 2020. 688 с.

12. *Монган Дж.* Работа мечты для программиста. СПб.: Питер, 2014. 368 с.

13. *Окулов С. М., Пестов О. А.* Динамическое программирование. М.: БИНОМ. Лаборатория знаний, 2015. 299 с.

14. *Пароджанов В. Д.* Дружелюбные алгоритмы, понятные каждому. М.: ДМК Пресс, 2014. 464 с.

15. *Потопахин В. В.* Искусство алгоритмизации. М.: ДМК Пресс, 2013. 320 с.

16. *Рафгарден Т.* Совершенный алгоритм. Жадные алгоритмы и динамическое программирование. СПб.: Питер, 2020. 256 с.

17. *Рафгарден Т.* Совершенный алгоритм. Основы. СПб.: Питер, 2019. 256 с.

18. *Скиена С.* Алгоритмы. Руководство по разработке. СПб.: BHV, 2011. 720 с.

19. *Солтис М.* Введение в анализ алгоритмов. М.: ДМК Пресс, 2019. 278 с.

20. *Спрингер В.* Гид по Computer Science для каждого программиста. СПб.: Питер, 2020. 192 с.

21. *Стивенс Р.* Алгоритмы. Теория и практическое применение. М.: ЭКСМО, 2016. 544 с.

22. *Струченков В. И.* Динамическое программирование в примерах и задачах. М. — Берлин: Директ-Медиа, 2015. 275 с.

23. *Шень А.* Программирование: теоремы и задачи. М.: МЦНМО, 2017. 320 с.

24. *Erdősné Németh A., Zsakó L.* The place of the dynamic programming concept in the progression of contestants' thinking // Olympiad in Informatics. 2016. Vol. 10. P. 61–72.

25. *Forisek M.* Towards a better way to teach dynamic programming // Olympiad in Informatics. 2015. Vol. 9. P. 45–55.