DOI: 10.32517/2221-1993-2023-22-5-82-86

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

ЗАДАЧИ НА ТЕМУ «МАКСИМАЛЬНАЯ НЕУБЫВАЮЩАЯ ПОДПОСЛЕДОВАТЕЛЬНОСТЬ»

Аннотация
В данной статье на примере решения нескольких задач проиллюстрирована методика изучения темы «Максимальная неубывающая подпоследовательность» при подготовке школьников к олимпиадам по информатике. Изучение основано на последовательном решении усложняющихся задач. Для каждой задачи приводятся следующие материалы: условие задачи, идея решения, решение на языке программирования Pascal. Технической основой для изучения темы и решения задач 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 ее и послать решение повторно. Кроме того, для каждой задачи в системе 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, максимальная неубывающая подпоследовательность, олимпиады по информатике, инструментальная система дистанционного обучения.

1. Введение

В данной статье мы продолжаем начатое ранее в наших публикациях в журнале «Информатика в школе» рассмотрение методики преподавания темы «Рекуррентные соотношения» на основе решения набора постепенно усложняющихся задач. Эта тема важна как для подготовки школьников к олимпиадам по программированию, так и в сфере профессиональных программистов. Поэтому она широко освещается в технической литературе, в том числе можно рекомендовать для знакомства с ней:

- книги по общей теории алгоритмов, включающие разделы о рекуррентных соотношениях и динамическом программировании [2, 6, 7, 10, 14, 15, 17–19, 21–23];
- книги и статьи, посвященные исключительно рекуррентным соотношениям и динамическому программированию [1, 13, 16, 22, 24, 25];
- сборники задач олимпиадной тематики, включающие задачи на рекуррентные соотношения и динамическое программирование [5, 8, 9];
- сборники тестовых заданий для собеседования перед поступлением на работу, включающие задачи на рекуррентные соотношения и динамическое программирование [11, 12];
- книги, содержащие рекомендации для практикующих программистов относительно рекуррентных соотношений и динамического программирования [3, 20]

Наш многолетний опыт подготовки школьников к олимпиадам показывает, что уже в V—VIII классах

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_05-2023/

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

PROBLEMS ON THE THEME "MAXIMUM NON-DECREASING SUBSEQUENCE"

Abstract
In the article, using the example of solving few problems, the methodology for studying the theme "Maximum non-decreasing subsequence" is illustrated in preparing schoolchildren for Olympiads in informatics. The study is based on the sequential solution of increasingly complex problems. For each problem, the following materials are given: the formulation of the problem, the idea of a solution, the solution in the Pascal programming language. Distance learning system (http://dl.gsu.by) is the 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 it and send the solution again. In addition, for each problem 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, maximum non-decreasing subsequence, Olympiads in informatics, distance learning tools.

есть дети, которым интересна тема рекуррентных соотношений. Для их обучения нами был выбран подход «от простого к сложному», когда школьнику предлагаются подтемы в порядке возрастания сложности, а внутри каждой темы — задачи, также в порядке возрастания их сложности. Причем сначала ребятам предлагается изучение известных стандартных алгоритмов. Такой подход и расширяет круг задач, которые смогут решать школьники, и позволяет проводить пропедевтическую работу по развитию у учащихся способностей придумывать решения нестандартных задач. Первая подтема, которая предлагается учащимся в теме «Рекуррентные соотношения», — «Рекуррентное соотношение "Все суммы"», которой была посвящена наша статья [4]. В текущей статье мы рассмотрим следующую подтему — «Максимальная неубывающая подпоследовательность».

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».

2. Стандартная задача «Неубывающая подпоследовательность максимальной длины»

Пусть задана последовательность. Требуется найти длину ее неубывающей подпоследовательности максимальной длины.

Например, для исходной последовательности:

1 2 3 3 4 2 10 15 10 100

ответ будет 8, а требуемая подпоследовательность такова:

1 2 3 3 4 10 15 100.

Решение.

Стандартный рекуррентный алгоритм, который решает данную задачу, приведен в листинге 1.

```
for i:=1 to N do L[i]:=1;
for i:=1 to N do P[i]:=0;
for i:= 1 to N-1 do
  for j:= i+1 to N do
    if (a[j]>=a[i]) and (L[i]+1>L[j])
      then begin
                L[j]:=L[i]+1;
                P[j]:=i;
           end;
MaxL:=L[1];
for i:=2 to N do
  if L[i]>MaxL then MaxL:=L[i];
```

Листинг 1

Здесь:

$L[i]$ — длина наибольшей неубывающей подпоследовательности, которую можно построить на элементах от 1 до i, причем i-й элемент — максимальный в этой подпоследовательности.

$P[i]$ — номер предыдущего элемента этой последовательности максимальной длины.

То есть на самом деле этот алгоритм строит неубывающие подпоследовательности, i-я из которых заканчивается на элементе $a[i]$.

Полный текст решения с вводом данных и выводом всех массивов (которые можно использовать для анализа результатов тестового примера для упрощения понимания его работы) приведен в листинге 2.

```
var
  a,L,P       : array [1..500] of longint;
  i,j,N, MaxL : longint;
begin
  assign(input,'input.txt');
  reset(input);
  assign(output,'output.txt');
  rewrite(output);
  readln(N);
  for i:=1 to N do readln(a[i]);
  for i:=1 to N do L[i]:=1;
  for i:=1 to N do P[i]:=0;
  for i:=1 to N-1 do
    for j:=i+1 to N do
      if (a[j]>=a[i]) and (L[i]+1>L[j])
        then begin
                  L[j]:=L[i]+1;
                  P[j]:=i;
             end;
  MaxL:=L[1];
  for i:=2 to N do
    if L[i]>MaxL then MaxL:=L[i];
  writeln(MaxL);
  for i:=1 to N do write(a[i]:4); writeln;
  for i:=1 to N do write(L[i]:4); writeln;
  for i:=1 to N do write(P[i]:4); writeln;
  close(input);
  close(output);
end.
```

Листинг 2

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

В заданной последовательности целых чисел длины N найти максимально длинную подпоследовательность чисел (необязательно подряд идущих), такую, что каждый последующий элемент подпоследовательности делился нацело на предыдущий.

Формат ввода.

N
$a[1]$
$a[2]$
...
$a[N]$

Здесь:

N — количество чисел в последовательности;
$a[i]$ — i-е число последовательности.

Ограничения.

$1 \le N \le 500$;
$1 \le a[i] \le 1000$.

Формат вывода.

Одно целое число — максимальная длина подпоследовательности чисел (необязательно подряд идущих), такая, что каждый последующий элемент подпоследовательности делился нацело на предыдущий.

Пример ввода	*Пример вывода*
10 1 2 3 3 4 2 10 15 10 100	6

Пояснение к примеру.

Искомой подпоследовательностью приведенной в примере последовательности будет подпоследовательность: 1, 2, 2, 10, 10, 100. Ее длина равна шести, соответственно, ответ будет 6.

Идея решения задачи.

Отличие этой задачи от стандартной, рассмотренной в разделе 2 данной статьи, заключается в основном *свойстве* элементов подпоследовательности: каждый последующий элемент подпоследовательности должен делиться нацело на предыдущий. То есть вместо `a[j]>=a[i]` нужно проверять, что `(a[j] mod a[i])=0`.

Полный текст решения представлен в листинге 3.

```
var
  a,L,P      : array [1..500] of longint;
  i,j,N,MaxL : longint;
begin
  readln(N);
  for i:=1 to N do read(a[i]);
  for i:=1 to N do L[i]:=1;
  for i:=1 to N do P[i]:=0;
  for i:=1 to N-1 do
    for j:=i+1 to N do
      if ((a[j] mod a[i])=0)
          and (L[i]+1>L[j])
        then begin
                  L[j]:=L[i]+1;
                  P[j]:=i;
             end;
  MaxL:=L[1];
  for i:=2 to N do
    if L[i]>MaxL then MaxL:=L[i];
  writeln(MaxL);
end.
```

Листинг 3

4. Задача «Взаимно простые соседи» (Гомельская городская олимпиада для I—IX классов, 2012 год)

Дана последовательность из N чисел. Ваша задача — определить длину максимальной подпоследовательности чисел (необязательно подряд идущих), в которой два любых соседних числа взаимно простые.

Формат ввода.

N — количество чисел.

Далее следует последовательность из N чисел $a[i]$, разделенных пробелами.

Ограничения.

$2 \le N \le 1000$;

$2 \le a[i] \le 2000$.

Формат вывода.

Одно целое число — длина искомой подпоследовательности.

Пример ввода	*Пример вывода*
5 2 4 7 14 9	3

Пояснение к примеру.

Искомой подпоследовательностью приведенной в примере последовательности будет последовательность: 2, 7, 9. Ее длина равна трем, соответственно, ответ будет 3.

Идея решения задачи.

От стандартной задачи о максимальной подпоследовательности (см. раздел 2 данной статьи) эта задача отличается свойством, которое нужно проверять: соседние два числа должны быть взаимно просты, т. е. их наибольший общий делитель равен 1.

Соответственно, проверка должна выглядеть так:

```
if (NOD(a[j],a[i])=1) and (L[i]+1>L[j])
```

Здесь NOD($a[i]$, $a[j]$) — функция вычисления наибольшего общего делителя чисел $a[i]$ и $a[j]$.

Полный текст программы приведен в листинге 4.

```
var
  a,L,P      : array [1..1000] of longint;
  i,j,N,MaxL : longint;

function NOD(a,b:longint):longint;
  begin
    if a=0 then NOD:=b else
      if b=0 then NOD:=a else
        if a>b
          then NOD:=NOD(a mod b, b)
          else NOD:=NOD(b mod a, a)
  end;

begin
  readln(N);
  for i:=1 to N do read(a[i]);
  for i:=1 to N do L[i]:=1;
  for i:=1 to N do P[i]:=0;
  for i:=1 to N-1 do
    for j:=i+1 to N do
      if (NOD(a[j],a[i])=1)
          and (L[i]+1>L[j])
        then begin
                  L[j]:=L[i]+1;
                  P[j]:=i;
             end;
  MaxL:=L[1];
  for i:=2 to N do
    if L[i]>MaxL then MaxL:=L[i];
  writeln(MaxL);
end.
```

Листинг 4

5. Задача «Максимальная подпоследовательность брусков» (Гомельская городская олимпиада для I—IX классов, 2015 год)

Имеется N ($3 \le N \le 20$) брусков высотой 1. Каждый брусок имеет уникальную ширину и уникальную длину.

Требуется выбрать такое подмножество брусков, с помощью которого можно построить пирамиду максимальной высоты. При этом один брусок можно класть поверх другого только в том случае, если его ширина и длина меньше, чем ширина и длина бруска под ним. Запрещается поворачивать бруски, чтобы менять длину с шириной.

Формат ввода.

Первая строка содержит одно целое число N.

Каждая из последующих N строк описывает брусок двумя разделенными пробелом целыми числами, соответствующими длине и ширине брусков.

Формат вывода.

Одно целое число — высота самой высокой пирамиды, которую можно построить по заданным правилам из данных брусков.

Пример ввода	*Пример вывода*
6 6 9 10 12 9 11 8 10 7 8 5 3	5

Пояснение к примеру.

В данном примере рассматриваются шесть брусков с разными параметрами длины и ширины.

Самая высокая пирамида, которую можно построить из данного набора брусков, имеет высоту, равную 5. Это, например, пирамида из следующих брусков:

10 12
9 11
8 10
6 9
5 3

(это не единственный вариант построения пирамиды высотой 5).

Идея решения задачи.

Сортируем бруски по длине и выбираем максимальную подпоследовательность брусков, удовлетворяющих условию. Пусть ее длина $L1$.

Сортируем бруски по ширине и выбираем максимальную подпоследовательность брусков, удовлетворяющих условию. Пусть ее длина $L2$.

Выводим максимум из $L1$ и $L2$.

Полный текст решения приведен в листинге 5.

Замечание. В этой задаче можно было попытаться сделать одну процедуру сортировки и передавать массивы в качестве параметров, однако это усложняет понимание решения, поэтому данные действия не были сделаны, чтобы «не усложнять сущности без необходимости».

```
var
  a,b,L,P     : array [1..20] of longint;
  i,j,N,L1,L2 : longint;

procedure SortA;
  var
    i,j,t : longint;
  begin
    for i:=1 to N do
      for j:=1 to N-1 do
        if a[j]<a[j+1]
          then begin
                 t:=a[j];
                 a[j]:=a[j+1];
                 a[j+1]:=t;
                 t:=b[j];
                 b[j]:=b[j+1];
                 b[j+1]:=t;
               end;
  end;

procedure SortB;
  var
    i,j,t : longint;
  begin
    for i:=1 to N do
      for j:=1 to N-1 do
        if b[j]<b[j+1]
          then begin
                 t:=a[j];
                 a[j]:=a[j+1];
                 a[j+1]:=t;
                 t:=b[j];
                 b[j]:=b[j+1];
                 b[j+1]:=t;
               end;
  end;

function LMaxSubSeq:longint;
  var
    i,j,MaxL : longint;
  begin
    for i:=1 to N do L[i]:=1;
    for i:=1 to N do P[i]:=0;
    for i:=1 to N-1 do
      for j:=i+1 to N do
        if (a[j]<=a[i]) and (b[j]<=b[i])
            and (L[i]+1>L[j])
          then begin
                 L[j]:=L[i]+1;
                 P[j]:=i;
               end;
    MaxL:=L[1];
    for i:=2 to N do
      if L[i]>MaxL then MaxL:=L[i];
    LMaxSubSeq:=MaxL;
  end;

function Max(a,b:longint):longint;
  begin
    if a>b then Max:=a else Max:=b;
  end;
```

Листинг 5

```
begin
  assign(input,'btwr.in');
  reset(input);
  assign(output,'btwr.out');
  rewrite(output);
  readln(N);
  for i:=1 to N do readln(a[i],b[i]);
  SortA; L1:=LMaxSubSeq;
  SortB; L2:=LMaxSubSeq;
  writeln(Max(L1,L2));
  close(input); close(output);
end.
```

Листинг 5 (окончание)

6. Заключение

В данной статье на примере решения нескольких задач рассмотрена методика изучения темы «Максимальная неубывающая подпоследовательность», предлагающая наиболее простой, по мнению автора, способ постепенного осознания учащимися механизма соответствующего рекуррентного соотношения и способа решения задач с его помощью. Методика включает в себя решение нескольких задач в порядке возрастания их сложности. Задачи снабжены, где необходимо, предварительными общими пояснениями и последующими полными решениями. Технической основой для решения задач 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 URSS. 2021. 200 с.

4. *Долинский М. С.* Задачи на рекуррентное соотношение «все суммы» // Информатика в школе. 2023. № 3. С. 81–85. EDN: VDWEZF. DOI: 10.32517/2221-1993-2023-22-3-81-85

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, 2103. 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, 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. *Стрченков 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.