DOI: 10.32517/2221-1993-2022-21-2-58-64

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

РЕКУРСИВНОЕ РЕШЕНИЕ ЗАДАЧ С ПОМОЩЬЮ МЕТОДА «РАЗДЕЛЯЙ И ВЛАСТВУЙ»*

Аннотация

Методика изучения темы «Рекурсивное решение задач с помощью метода “разделяй и властвуй”» при подготовке школьников к олимпиадам по информатике иллюстрируется в статье решением задачи «Голова на плечах». Решение данной задачи включает в себя несколько алгоритмов (подзадач), рассматриваемых последовательно, а именно: сортировка слиянием, подсчет количества инверсий в перестановке, количество пар пересекающихся отрезков. Изучение метода основано на последовательном решении этих подзадач. Для каждой подзадачи приводятся: формулировка подзадачи, 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 неправильного решения 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 отправить решение повторно.

Ключевые слова: рекурсия, «разделяй и властвуй», обучение программированию, олимпиады по информатике, инструментальная система дистанционного обучения.

1. Введение

Подготовка школьников к олимпиадам по информатике и программированию требует значительного объема теоретических знаний и практических навыков решения задач. Поэтому в мире компьютерной книжной и журнальной литературы регулярно появляются книги и статьи как по общей теории алгоритмов [5, 10, 11, 14, 16–18, 21], так и на отдельные актуальные темы: динамическое программирование [12, 23, 24], графы [13, 20–22, 25, 26], хеши [27], рекурсия [1, 2, 4, 15]. Автор вносит свою лепту в разработку темы «Рекурсия» [5–9] специальным подходом, связанным с ранним началом обучения. Как следствие, уже в пятом-шестом классе появляются ученики, которым приходится объяснять такие темы, как «Рекурсия». Поэтому приходится модифицировать изложение в сторону более простого и наглядного объяснения и более медленного продвижения по учебному материалу, явно выделяя и обозначая все этапы этого продвижения. Введение в решение задач с помощью рекурсии изложено в работе [5]. Решение задач с помощью рекурсивной генерации таких комби-

* Материалы к статье можно скачать на сайте ИНФО: http://infojournal.ru/journals/school/school_02-2022/

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

RECURSIVE SOLVING OF THE PROBLEMS USING THE "DIVIDE & CONQUER" METHOD

Abstract

The methodology for studying the theme "Recursive solving of the problem using the "divide & conquer" method" in preparing schoolchildren for Olympiads in informatics is illustrated in the article by solving the "Head on shoulders" problem. The solution of this problem includes several algorithms (subtasks) considered sequentially, namely: merge sort, counting the number of inversions in a permutation, the number of pairs of intersecting segments. The study of the method is based on the sequential solution of these subtasks. For each subtask, the following materials are given: the formulation of the subtask, 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 DL.GSU.BY is the effective technical base for teaching. The system allows to offer for a student a formulation of the task; 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.

Keywords: recursion, "divide & conquer", teaching for programming, Olympiads in informatics, distance learning tools.

наторных объектов, как множество всех подмножеств, сочетания, перестановки, перестановки с повторениями, размещения, описано в работе [6]. В работе [9] описано решение рекурсивных задач «по определению». В работе [7] описано решение задач рекурсивной генерацией чисел. Решение игровых задач с помощью рекурсии описывается в работе [8].

В настоящей статье предлагается материал по решению задачи с помощью основанного на рекурсии метода «разделяй и властвуй».

Проверка решений осуществляется автоматически на сайте DL.GSU.BY. Вдумчивым читателям предлагается после чтения условия задачи/подзадачи попытаться решить ее самостоятельно.

2. Условие задачи «Голова на плечах» (Десятая интернет-олимпиада для школьников, 2 марта 2007 года, Санкт-Петербург)

Изготовитель всемирно известного шампуня компания «Голова на плечах» всерьез заботится о качестве своей продукции. В частности, она постоянно улучшает различные показатели своего шампуня, для чего постоянно проводит исследования.

Но, согласно законодательству, на человеке проводить исследования нельзя, а проводить исследования на животных компания считает ниже своего достоинства.

Поэтому ученые, работающие в компании, разработали математическую модель человека и проводят исследования на ней. Человек, согласно этой модели, состоит из головы и плеч. Голова представляет собой окружность с центром в точке (0, 0) и радиусом R, а плечи — бесконечную прямую $y = -K$, где $R < K$.

Объектом изучения исследователей являются волосы. Каждый волос в данной модели представлен отрезком, начинающимся на голове (строго на окружности) и заканчивающимся на плечах (строго на прямой). При этом ни один волос не имеет с окружностью головы более одной общей точки.

В данный момент ученые озабочены проблемой секущихся волос. Пара волос называется секущейся, если соответствующие этим волосам отрезки имеют общую точку.

Дана математическая модель человека. Найдите количество секущихся пар волос.

Формат ввода:

В первой строке даны два целых числа R, K ($1 \le R < K \le 1000$).

Во второй строке дано целое число N ($0 \le N \le 100\,000$) — количество волос в модели человека.

В следующих N строках находятся по четыре вещественных числа Xh, Yh, Xs, Ys — координаты начала и конца очередного волоса. Первая пара чисел соответствует концу, лежащему на окружности головы, вторая пара соответствует концу, лежащему на плечах.

Гарантируется, что никакой волос не имеет с окружностью головы более одной общей точки.

Также гарантируется, что среди начальных и конечных точек нет одинаковых.

Формат вывода:

В выходной файл выведите число секущихся пар волос.

Пример ввода	*Пример вывода*
1 2 3 0 -1 -2 -2 1 0 2 -2 -1 0 -1 -2	1

Пример ввода	*Пример вывода*
1 10 3 -1 0 -1 -10 0 -1 0 -10 1 0 1 -10	0

3. Алгоритм «разделяй и властвуй» ("divide & conquer")

Идея решения задачи.

Основной идеей решения задачи является алгоритм «разделяй и властвуй» (divide & conquer), который заключается в том, что ответ для задачи находится рекурсивно делением пополам исходного отрезка $[L, R]$ на подотрезки $[L, M]$ и $[M + 1, R]$ (где $M = (L + R) / 2$) и собиранием ответа из трех: $Ans[L, M]$ и $Ans[M + 1, R]$, которые вычисляются рекурсивно, и $Ans[L, R]$ — ответа на отрезке $[L, R]$, который должен вычисляться линейно (за $O(R - L)$ операций), тогда общая сложность будет $O(N \cdot \mathrm{Log}\ N)$.

Для решения этой задачи нужно понимать сортировку слиянием.

4. Сортировка слиянием

Здесь подотрезки $[L, M]$ и $[M + 1, R]$ все время отсортированы, а процедура *Merge* с помощью двух указателей $i1$ и $i2$ и вспомогательного массива b обеспечивает слияние двух отсортированных массивов.

```
const
  MaxN=100000;
var
  a,b: array [1..MaxN] of longint;
  N,i: longint;

procedure Merge(L1,R1,L2,R2: longint);
var
  i1,i2,ib,i: longint;
begin
  i1:=L1; i2:=L2; ib:=1;
  while (i1<=R1) and (i2<=R2) do
    if a[i1]<a[i2]
      then
        begin
          b[ib]:=a[i1];
          inc(i1); inc(ib);
        end
```

```
      else
        begin
          b[ib]:=a[i2];
          inc(i2);
          inc(ib);
        end;
    if i1>R1
      then
        for i:=i2 to R2 do
          begin
            b[ib]:=a[i];
            inc(ib)
          end
      else
        for i:=i1 to R1 do
          begin
            b[ib]:=a[i];
            inc(ib)
          end;
    ib:=1;
    for i:=L1 to R2 do
      begin
        a[i]:=b[ib];
        inc(ib);
      end;
end;

procedure MergeSort(L,R: longint);
var
  M: longint;
begin
  if L>=R then exit;
  M:=(L+R) div 2 ;
  MergeSort(L,M);
  MergeSort(M+1,R);
  Merge(L,M,M+1,R);
end;

begin
  assign(input,'input.txt');
  reset(input);
  assign(output,'output.txt');
  rewrite(output);
  readln(N);
  for i:=1 to N do
    read(a[i]);
  MergeSort(1,N);
  for i:=1 to N do
    write(a[i],' ');
  writeln;
  close(input); close(output);
end.
```

5. Подсчет количества инверсий в перестановке

Следующий шаг — **подсчет количества инверсий в перестановке** (заданной в одномерном массиве), пользуясь сортировкой слиянием и считая количество инверсий как сумму инверсий на отрезках $[L, M]$ и $[M+1, R]$ рекурсивно и на отрезке $[L, R]$ в момент слияния двух отсортированных массивов. Если меньшим оказывается элемент из второго массива, то к ответу на отрезке $[L, R]$ нужно прибавить количество уже обработанных элементов первого массива `R1-i1+1`.

```
const
  MaxN=100{000};
var
  a,b    : array[1..MaxN] of longint;
  N,i,Ans: longint;

function Kinv_Merge(L1,R1,L2,R2: longint):
  longint;
var
  i1,i2,ib,i,kinv: longint;
begin
  i1:=L1; i2:=L2; ib:=1; kinv:=0;
  while (i1<=R1) and (i2<=R2) do
    if a[i1]<a[i2]
      then
        begin
          b[ib]:=a[i1];
          inc(i1);
          inc(ib);
        end
      else
        begin
          b[ib]:=a[i2];
          inc(i2);
          inc(ib);
          inc(kinv,R1-i1+1);
        end;
  if i1>R1
    then
      for i:=i2 to R2 do
        begin
          b[ib]:=a[i];
          inc(ib)
        end
    else
      for i:=i1 to R1 do
        begin
          b[ib]:=a[i];
          inc(ib)
        end;
  ib:=1;
  for i:=L1 to R2 do
    begin
      a[i]:=b[ib];
      inc(ib);
    end;
  Kinv_Merge:=kinv;
end;

function Kinv_MergeSort(L,R: longint): longint;
var
  M,Kinv,i: longint;
begin
  if L>=R then begin Kinv_MergeSort:=0; exit; end;
  M:=(L+R) div 2;
  Kinv:=Kinv_MergeSort(L,M)+Kinv_MergeSort(M+1,R);
  inc(Kinv,Kinv_Merge(L,M,M+1,R));
```

```
  Kinv_MergeSort:=Kinv;
end;

begin
  assign(input,'input.txt');
  reset(input);
  assign(output,'output.txt');
  rewrite(output);
  readln(N);
  for i:=1 to N do read(a[i]);
  Ans:=Kinv_MergeSort(1,N);
  writeln(Ans);
  close(input); close(output);
end.
```

6. Использование отладочной печати

Для упрощения понимания процесса подсчета количества инверсий во время рекурсивной сортировки слиянием можно воспользоваться отладочной печатью, представленной ниже.

```
const
  MaxN=100{000};
var
  a,b     : array[1..MaxN] of longint;
  N,i,Ans : longint;

function Kinv_Merge(L1,R1,L2,R2: longint):
  longint;
var
  i1,i2,ib,i,kinv: longint;
begin
  i1:=L1; i2:=L2; ib:=1; kinv:=0;
  while (i1<=R1) and (i2<=R2) do
    if a[i1]<a[i2]
      then
        begin
          b[ib]:=a[i1];
          inc(i1);
          inc(ib);
        end
      else
        begin
          b[ib]:=a[i2];
          inc(i2);
          inc(ib);
          inc(kinv,R1-i1+1);
        end;
  if i1>R1
    then
      for i:=i2 to R2 do
        begin
          b[ib]:=a[i];
          inc(ib)
        end
    else
      for i:=i1 to R1 do
        begin
          b[ib]:=a[i];
          inc(ib)
        end;
  ib:=1;
  for i:=L1 to R2 do
    begin
      a[i]:=b[ib];
      inc(ib);
    end;
  Kinv_Merge:=kinv;
end;

function Kinv_MergeSort(L,R: longint): longint;
var
  M,Kinv,i: longint;
begin
  if L>=R then begin Kinv_MergeSort:=0; exit; end;
  M:=(L+R) div 2;
  Kinv:=Kinv_MergeSort(L,M)+Kinv_MergeSort(M+1,R);
  for i:=L to M do write(a[i],' ');
write('---');
  for i:=M+1 to R do write(a[i],' '); write;
  inc(Kinv,Kinv_Merge(L,M,M+1,R));
  writeln({L,' ',M,',',M+1,' ',R,}'=',KInv,'   ');
  Kinv_MergeSort:=Kinv;
end;

begin
  assign(input,'input.txt');
  reset(input);
  assign(output,'output.txt');
  rewrite(output);
  readln(N);
  for i:=1 to N do read(a[i]);
  Ans:=Kinv_MergeSort(1,N);
  for i:=1 to N do write(a[i],' '); writeln;
  writeln(Ans);
  close(input); close(output);
end.
```

7. Количество пар пересекающихся отрезков

Для лучшего понимания дальнейшего текста полезно нарисовать оба теста из условия. Получаем следующую **переформулированную задачу:**

Дано множество отрезков, начала которых ($XH[i]$, $YH[i]$) лежат на окружности радиуса R с центром в точке (0, 0), а концы ($Xs[i]$, $Ys[i]$) — на прямой $Y = -K$ ($R < K$).

Требуется найти **количество пар пересекающихся отрезков.**

По факту эта задача эквивалентна подсчету количества инверсий перестановки, составленной по данным отрезкам. Отрезки пересекаются, если конец отрезка, начало которого раньше, находится правее конца отрезка, начало которого позже.

Первый шаг — найти полярные углы всех точек начала, ведя отсчет от оси Y.

```
for i:=1 to N do
  begin
    if xh[i]<>0
      then mat:=ArcTan(abs(Yh[i]/Xh[i]));
```

```
    if (Xh[i]=0) and (Yh[i]>0)
      then Angle[i]:=0 else
    if (Xh[i]=0) and (Yh[i]<0)
      then Angle[i]:=Pi else
    if (Yh[i]=0) and (Xh[i]<0)
      then Angle[i]:=Pi/2 else
    if (Yh[i]=0) and (Xh[i]>0)
      then Angle[i]:=3*Pi/2 else
    if (Yh[i]>0) and (Xh[i]<0)
      then Angle[i]:=Pi/2-mat else
    if (Yh[i]<0) and (Xh[i]<0)
      then Angle[i]:=Pi/2+mat else
    if (Yh[i]<0) and (Xh[i]>0)
      then Angle[i]:=3*Pi/2-mat else
    if (Yh[i]>0) and (Xh[i]>0)
      then Angle[i]:=3*Pi/2+mat;
  end;
```

Здесь *ArcTan* — стандартная функция Pascal, вычисляющая арктангенс угла (в интервале от –Pi/2 до +Pi/2), *mat* — модуль этого арктангенса, с помощью которого в зависимости от квадранта плоскости, в которой лежит точка, мы и вычисляем угол *Angle*[*i*].

Затем для решения задачи используем следующий код:

```
QSortAngleWithXs(1,N);
for i:=1 to N do Num[i]:=i;
QSortXsWithNum(1,N);
for i:=1 to N do a[i]:=Num[i];
Ans:=Kinv_MergeSort(1,N);
writeln(Ans);
```

Сортируем точки по возрастанию этих углов вместе с их *Xs*-координатами с помощью процедуры *QSortAngleWithXs* и перенумеровываем точки в этом порядке.

Далее с помощью процедуры *QSortXsWithNum*(1, *N*) сортируем с номерами точки по возрастанию *Xs*.

В результате по отсортированным номерам и получаем перестановку *Num*.

Копируем эту перестановку в массив *a*, для которого и вызываем ранее написанный код подсчета количества инверсий в перестановке.

Количество инверсий может превысить диапазон longint, поэтому заменяем тип соответствующих переменных на int64.

Полный текст решения представлен ниже:

```
const
  MaxN=100000;
var
  Num              : array[1..MaxN] of longint;
  Xh,Yh,Xs,Ys,Angle: array[1..MaxN] of real;
  Rad,N,K,i        : longint;
  mat              : real;

procedure QSortAngleWithXs(p,r:longint);
var
  i,j,t: longint;
  tA,x : real;
begin
  if p>=r then exit;
  x:=Angle[p]; i:=p-1; j:=r+1;
  while i<j do
    begin
      repeat j:=j-1 until Angle[j]<=x;
      repeat i:=i+1 until Angle[i]>=x;
      if i<j
        then
          begin
            tA:=Angle[i];
            Angle[i]:=Angle[j];
            Angle[j]:=tA;
            tA:=Xs[i];
            Xs[i]:=Xs[j];
            Xs[j]:=tA;
          end;
    end;
  QSortAngleWithXs(p,j);
  QSortAngleWithXs(j+1,r);
end;

procedure QSortXsWithNum(p,r:longint);
var
  i,j,t: longint;
  tA,x : real;
begin
  if p>=r then exit;
  x:= Xs[p]; i:=p-1; j:=r+1;
  while i<j do
    begin
      repeat j:=j-1 until Xs[j]<=x;
      repeat i:=i+1 until Xs[i]>=x;
      if i<j
        then
          begin
            tA:= Xs[i];
            Xs[i]:=Xs[j];
            Xs[j]:=tA;
            t:=Num[i];
            Num[i]:=Num[j];
            Num[j]:=t;
          end;
    end;
  QSortXsWithNum(p,j);
  QSortXsWithNum(j+1,r);
end;

{ ----  MergeSort of array a ---- }
var
  a,b     : array[1..MaxN] of longint;
  {N,i,}Ans: int64;

function Kinv_Merge(L1,R1,L2,R2: longint): int64;
var
  i1,i2,ib,i: longint;
  kinv      : int64;
begin
  i1:=L1; i2:=L2; ib:=1; kinv:=0;
  while (i1<=R1) and (i2<=R2) do
    if a[i1]<a[i2]
      then
        begin
          b[ib]:=a[i1];
          inc(i1);
          inc(ib);
        end
```

```
      else
        begin
          b[ib]:=a[i2];
          inc(i2);
          inc(ib);
          inc(kinv,R1-i1+1);
        end;
    if i1>R1
      then
        for i:=i2 to R2 do
          begin
            b[ib]:=a[i];
            inc(ib)
          end
      else
        for i:=i1 to R1 do
          begin
            b[ib]:=a[i];
            inc(ib)
          end;
  ib:=1;
  for i:=L1 to R2 do
    begin
      a[i]:=b[ib];
      inc(ib);
    end;
  Kinv_Merge:=kinv;
end;

function Kinv_MergeSort(L,R: longint): longint;
var
  M,Kinv,i: longint;
begin
  if L>=R then begin Kinv_MergeSort:=0; exit;
end;
  M:=(L+R) div 2;
  Kinv:=Kinv_MergeSort(L,M)+Kinv_MergeSort(M+1,R);
  inc(Kinv,Kinv_Merge(L,M,M+1,R));
  Kinv_MergeSort:=Kinv;
end;

begin
  assign(input,'headand.in');
  reset(input);
  assign(output,'headand.out');
  rewrite(output);
  readln(Rad,K);
  readln(N);
  for i:=1 to N do
    readln(Xh[i],Yh[i],Xs[i],Ys[i]);
  for i:=1 to N do
    begin
      if xh[i]<>0
        then mat:=ArcTan(abs(Yh[i]/Xh[i]));
      if (Xh[i]=0) and (Yh[i]>0)
        then Angle[i]:=0 else
      if (Xh[i]=0) and (Yh[i]<0)
        then Angle[i]:=Pi else
      if (Yh[i]=0) and (Xh[i]<0)
        then Angle[i]:=Pi/2 else
      if (Yh[i]=0) and (Xh[i]>0)
        then Angle[i]:=3*Pi/2 else
      if (Yh[i]>0) and (Xh[i]<0)
        then Angle[i]:=Pi/2-mat else
      if (Yh[i]<0) and (Xh[i]<0)
        then Angle[i]:=Pi/2+mat else
      if (Yh[i]<0) and (Xh[i]>0)
        then Angle[i]:=3*Pi/2-mat else
      if (Yh[i]>0) and (Xh[i]>0)
        then Angle[i]:=3*Pi/2+mat;
    end;
  QSortAngleWithXs(1,N);
  for i:=1 to N do Num[i]:=i;
  QSortXsWithNum(1,N);
  for i:=1 to N do a[i]:=Num[i];
  Ans:=Kinv_MergeSort(1,N);
  writeln(Ans);
  close(input); close(output);
end.
```

8. Заключение

В данной статье представлена методика изучения темы «Рекурсивное решение задач с помощью метода “разделяй и властвуй”» с учащимися V—VIII классов, предлагающая, по мнению автора, наиболее простой вариант постепенного осознания механизма рекурсии и способа решения задач с ее помощью. Методика включает в себя последовательность задач в порядке возрастания сложности, снабженных, где необходимо, предварительными общими пояснениями и последующими полными решениями предлагаемых задач. В данной статье представлены пояснения к первой из серии задач по теме «Рекурсивное решение задач с помощью метода “разделяй и властвуй”». Серьезной технической основой является разработанная под управлением автора инструментальная система дистанционного обучения (DL.GSU.BY), которая позволяет максимально автоматизировать процесс предъявления условий задач и проверки их решений.

Список источников

1. *Баррон Д.* Рекурсивные методы в программировании. М.: Мир, 1974. 79 с.
2. *Бердж В.* Методы рекурсивного программирования. М.: Машиностроение, 1983. 256 с.
3. *Головешкин В. А., Ульянов В. В.* Теория рекурсии для программистов. М.: Физматлит, 2006. 296 с.
4. *Дасгупта С., Пападимитриу Х., Вазирани У.* Алгоритмы. М.: МЦНМО, 2019. 320 с.
5. *Долинский М. С.* Введение в решение задач с помощью рекурсивных процедур и функций // Информатика в школе. 2016. № 9. С. 49–56.
6. *Долинский М. С.* Генерация комбинаторных объектов с помощью рекурсивных процедур и функций // Информатика в школе. 2019. № 4. С. 59–63.
7. *Долинский М. С.* Решение задач рекурсивной генерацией чисел // Информатика в школе. 2021. № 1. С. 46–51.
8. *Долинский М. С.* Решение игровых задач с помощью рекурсии // Информатика в школе. 2021. № 9. С. 43–50.
9. *Долинский М. С.* Решение рекурсивных задач по определению // Информатика в школе. 2020. № 2. С. 60–66.
10. *Златопольский Д. М.* 1400 задач по программированию. М.: ДМК, 2019. 192 с.
11. *Луридас П.* Алгоритмы для начинающих. Теория и практика для разработчика. М.: ЭКСМО, 2018. 608 с.

12. *Рафгарден Т.* Совершенный алгоритм. Графовые алгоритмы и структуры данных. СПб.: Питер, 2020. 256 с.

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

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

15. *Рубио-Санчес М.* Введение в рекурсивное программирование. М.: ДМК Пресс, 2019. 436 с.

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

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

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

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

20. *Castro R., Lehmann N., Pérez J., Subercaseaux B.* Wavelet trees for competitive programming // Olympiad in Informatics. 2016. Vol. 10. P. 19–37.

21. *Do P. T., Pham B. T., Than V. C.* Latest algorithms on particular graph classes // Olympiad in Informatics. 2020. Vol. 14. P. 21–35.

22. *Erdősné Németh A.* Teaching graphs for contestants in lower-secondary-school-age // Olympiad in Informatics. 2017. Vol. 11. P. 41–53.

23. *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.

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

25. *Manev K.* Tasks on graphs // Olympiad in Informatics. 2008. Vol. 2. P. 90–104.

26. *Manev K., Nikolov N., Markov M.* Reconstruction of trees using metric properties // Olympiad in Informatics. 2011. Vol. 5. P. 82–91.

27. *Pachocki J., Radoszewskij J.* Where to use and how not to use polynomial string hashing // Olympiad in Informatics. 2013. Vol. 7. P. 90–100.