Tasks in Informatics with Pre-Existing Algorithms

Pavel S. PANKOV[1], Kirill A. BARYSHNIKOV[2]
[1] *International University of Kyrgyzstan*
[2] *LLC "Finance Soft", Bishkek, Kyrgyzstan*
e-mail: pps50@rambler.ru, baryshnikov.kirill@gmail.com

Abstract. Almost all tasks at informatics olympiads demand developing an algorithm. In this paper, we draw attention to tasks where some algorithm already exists. We propose classification of types of such tasks and give corresponding examples; survey both known and falling under our classification, and new (as we hope) types of tasks; consider the crucial case of a task with a known but too slow algorithm (formally presenting various known types of tasks).

We hope that some classes of such tasks would enlarge scope of tasks for use in olympiads.

Keywords: olympiads in informatics, tasks, existing algorithms, black box

1. Survey of Types of Tasks and the Aim of Paper

There is an up-to-date tendency to design various operations in informatics olympiad tasks as separate algorithms (procedures). For instance, the contestant's program must begin with the function

```
function Start (k) : Integer;
```

instead of the common

```
readln (k);
```

continue with functions of type

```
function Arr (i, k: integer): integer
```

for various integer *i* instead of the common

```
for i:=1 to k do
begin
  readln (arr[i]);
end;
```

and end with the procedure

```
procedure End (f: integer);
```

instead of producing the answer of task with

```
writeln (f);
```

This tendency can be explained as arising from numerous mistakes in input and output formats in preceding informatics olympiads of all levels.

In this paper we consider pre-existing algorithms (procedures) involved in the task due to its essence.

Such tasks can be classified firstly as follows. The algorithm itself is either unknown or is known for the contestant.

For unknown algorithms the contestant‘s task is either to restore the algorithm by its results or to use the algorithm for any purpose.

For known algorithms, the contestant's task might be as following:

- To implement a better equivalent algorithm.
- To implement any other algorithm related to the given one.
- To solve any other task related to the algorithm.

We survey both known and falling under our classification, and new (as we hope) types of tasks.

Section 2 considers various types of tasks with existing but unknown algorithms to the contestant. The contestant is either:

- To restore the algorithm exactly by its results or to implement an effective algorithm yielding same results („Black box“ task).
- To use (or to struggle versus) the algorithm (in the form of an executable file or external library or table of results) for various purposes.

Tasks sufficiently involving an external library are called "reactive" (Opmanis, 2006).

Section 3 deals with the crucial case of acceleration of a given algorithm which is too slow. In our opinion, there arises the principal problem for all informatics similar to Church's thesis for computer science at all: what tasks in informatics can be presented in this form, or in another words: is an algorithmic language sufficient to state tasks in informatics?

Section 4 considers the problem to extract information about a hidden object only by a known algorithm (as a turbid or narrow window).

Section 5 considers tasks on the analysis of known algorithms including ones with arbitrary wandering.

2. Tasks with Unknown Algorithms

2.1. *Tasks of „Black Box“ Type*

Such task may contain the following sentences:

- *Given an unknown algorithm in the form of an executable file, external library or table of results and/or some information about it.*
- *Restore the algorithm by your handle calls*. The number of calls might be bounded.

Or:

- *Write a program what will restore the algorithm by calls within given time"* or *"... and in bounded number of calls"*;
- *Write an effective program yielding same results.*

The simplest example of this type of task with unknown numbers bases requiring knowledge of the bisection method.

Task 1. Given the algorithm with unknown number *K*:
Algorithm 1.

```
writeln ('input integer number X in 0..10^10');
readln (X);
if X > K then writeln ('Greater') else writeln ('Leq') end;
```

and information *"K is an integer in 0..10^10."*

The task might be as follows:

- *Find K.*
- *Find K in less than 20 calls.*

By our experience, children of 5–7 years old solve this task in the range 0..1000 with great pleasure.

The programming version of this task:

There is the following function with unknown integer *K*:

```
function A (X: integer): boolean
begin
  if (X > K) return true;
  return false;
end;
```

Write a program that calls the function *A* to find the number *K*, calling the function *A* as few times as possible.

The following task is slightly more complex, and is also of interest for children.

Task 2. Given the algorithm with unknown numbers *U, V.* The user's aim is declared in itself:
Algorithm 2.

```
writeln ('Input zero and zero');
readln (X, Y);
D:=U^2+V^2;
writeln ('You are to receive HOT!');
repeat
  writeln ('Input integer X and Y');
  readln (X, Y);
  D1:=(X–U)^2+(Y–V)^2;
  if D1=0 then writeln ('HOT');
  if D1<D then writeln ('WARMER');
  if D1=D then writeln ('EQUAL');
  if D1>D then writeln ('COOLER');
  D:=D1;
until (D1 = 0);
end;
```

and information *"U and V are non-zero integers in – 10000..10000."*

An example with unknown variables:

Task 3. Given the algorithm *B* with unknown variables:
Algorithm 3.

```
writeln ('detect values of U, V, W in the listing');
For M:=1 to 100 do
begin
writeln ('input integer numbers X, Y, Z in 0..100');
readln (X, Y, Z);
  if X > U then Y:=U;
  if Y > V then Z:=V;
  if Z > W then X:=W;
writeln (X, Y, Z);
end;
writeln ('You have exceeded limit of requests');
end;
```

and information "*U, V, W are letters of the set {X,Y,Z}*."

The user is to guess one of 27 arrangements with repetitions of three by three, by means of fewer than 100 calls of the algorithm.

Transformations of such tasks into programming versions are obvious.

Since algorithms within announced classes can be written in different equivalent forms ("`if X>Y` ..." or "`if Y<X` ..."), the tasks in situations mentioned in 2.1 can be different:

"After experimenting with the unknown algorithm write a program giving the same results" and a set of tests covering all branches of the algorithm.

2.2. *Games Versus Unknown Algorithms*

Such a type of tasks is well-known and we do not give references.

General task 4. Rules of a bilateral game are described and

(a) the jury's exe-file exists

or

(b) other contestants' exe-files will be built.

"Write a program which will play

(a) *versus the jury's program*

or

(b) *versus other participants' programs.*

In (a) case the aims may vary: to win; to withstand no fewer than a given number of moves; to obtain as many points as possible etc.

In (b) case to equalize all participants the program is to play versus each other one both as the first player ("Whites") and as the second one ("Blacks").

2.3. *Optimal Imitation of Unknown Algorithms*

Task 5 *"Function of functions"* ("Ugale", 2005)

Function f(x) is defined for all integers from 1 to 2000000000 and function values are positive integers. Given the table of function values for 100 hundred argument values: …

Your task is to write ***as short as*** *possible a program, which implements this function for all x values given in table above. Executing time for one particular test case must not exceed one second. Programs with lower* ***amount of source code in bytes*** *will get higher scores*.

(In such tasks pre-existence of a short algorithm yielding given results is meant).

3. Tasks to Accelerate Given Algorithm

These tasks have the following scheme: given an algorithm (mainly, too slow, of type of full sorting). Write a program yielding the same result in appropriate time.

We propose to write given algorithms in non-formal but clear "semi-algorithmic" language, as above. For brevity, simple non-algorithmic operations also may be involved.

There arises the principal problem: what tasks in informatics can be presented in such form? By our opinion, these types are following.

General task 6. (Find anything in a priori bounded set B).

Algorithm 4.

```
Given the set B, the large number M; |B|≤M; the boolean
function A;
foreach X in B do
begin
  if A(X):=true then writeln (X);
end;
```

The following tasks may be presented in such form: tasks on optimization, on solving equations, on finding intervals for solutions of equations (Pankov, 2013).

It's easy to add the following condition:

```
if (A(X):=false) foreach X∈B
then writeln ('No').
```

Remark 1. In the last IOIs the jury used to add conditions of the following type:
In not less than 50% tests |B|≤M*10^(–3).

General task 7. (Count anything in a priori bounded set B).

Algorithm 5.

```
Given the set B, the large number M; |B|≤M; the boolean
function A:
 S:=0;
```

```
foreach X in B do
begin
  if A(X):=true then S:=S+1;
end;
writeln (S);
end;
```

Remark 2. Since IOI'2008 formulations of the following type are proposed:

```
    writeln (S mod 1000).
```

It avoids necessarily using long numbers and leaves treating such numbers or circumvention of them to the contestant.

General task 8. (Find anything in a priori unbounded set *B*).
Algorithm 6.

```
Given the countable set B and the Boolean function A;
foreach X in B do
begin
  if A(X):=true then writeln (X);
end;
```

For example,

Task 9 (confutation of Euler's hypothesis).
Algorithm 7.

```
U:=0;
Repeat
U:=U+1;
for X:=1 to U do for Y:=1 to U do
for Z:=1 to U do for T:=1 to U do
  begin
    if (X^5+Y^5+Z^5+T^5=U^5) then
    begin
      equal:=true; X1:=X; Y1:=Y; Z1:=Z; T1:=T
      goto M1:
    end;
end; end; end; end;
until equal;
M1:
writeln (X1, Y1, Z1, T1, U).
```

Remark 3. The condition

if there is no such numbers X1, Y1, Z1, T1, for any U then output 'No'

also can be added but it demands special proof here. For example:

Task 10. (Pankov, 2013). Find an interval of width 1 containing a solution of the equation *F*(*X*)*:=X^4+A[3]*X^3+A[2]*X^2+A[1]*X+A[0]=0* or output "No", if there is no solution.

An a priori boundary is found by

Algorithm 8.

```
for X:=1,2,3… do
begin
  if (X^4–|A[3]|*X^3–|A[2]|*X^2–|A[1]|*X–|A[0]|)>0 then
  begin
    XM:=X;
    break;
  end;
end;
S:=false;
for X:=–XM to XM–1do
begin
  if (F(X)*F(X+1)≤0) then
  begin
    writeln (X, X+1);
    S:=true;
    break;
  end;
end;
if (S:=false) then writeln ('No'); end.
```

4. Tasks on Restricted Access to Object by Known Algorithm

Such mathematical tasks were initiated by (Tikhonov, 1943). There is an unknown object (underground ore body) and we can obtain only sums (integrals) of its components by means of geological methods (further: or by X-rays, i.e. tomography). Detect its shape with some error.

Such tasks in informatics can be described as applying a known algorithm to an unknown object.

We recall two examples.

General Task 11 (by the task *Giza*; see, for example, Burton *et al.*, 2008). There is an unknown ("secret") double array, given cross-section total sums of its elements (or: an algorithm counting such sums). Restore the array (or: some elements of it).

We propose an example of Tikhonov's type:

Task 12. There is a hidden double array *A[0..N, 0..N]* of numbers connected as topological torus (Pankov, 2008) defined as follows.

```
For all Y = 0..N Points (0, Y) and (N, Y) are glued and
For all X = 0..N Points (X, 0) and (X, N) are glued
```

The algorithm *S* yields the sum of each element and its eight (twenty-four …) neighbors. Find *max {A[X,Y]: X = 0..N, Y = 0..N}*.

Task 13 “Mean” (IOI’2000). There is a hidden array *A[0..N]* of different numbers and algorithm *M* which reworks each three indexes into such one that the element with such index is between two other elements. By means of using this algorithm detect indexes of the greatest and the least objects.

5. Tasks on Analysis of Given Algorithms

Such type of tasks was distinguished (Opmanis, 2009).

5.1. *Tasks on Restoration of Input Data*

We cite

Task 14 (Aivars Žogla), see (Opmanis, 2009). Given text of an algorithm that sorts the elements of number array *A[0..n−1]* from position low till position high in non-decreasing order. By taking *low = 0* and *high = n−1*, the entire array will be sorted. Your task is to find an array containing each of the numbers from *1* to *20* exactly once, for which sorting by calling procedure *sort(A,0,19),* uses the maximum number of array element comparison operations.

On the base of this task we propose:

General task 15 “Inverse algorithm”. Given text of an algorithm and output data. Find input data yielding this output data (or: with any optimization, as in Task 6).

5.2. *Tasks on Arbitrary Inputs to Known Algorithm*

General task 16. Given an interactive algorithm containing the initial situation, possible inputs and final situation. Some inputs change the situation. Achieving the final situation ends the algorithm with corresponding results.

The sense of the task is the following: the user is to guess the aim of the algorithm and write an improved algorithm (without inputs!) or calculate the optimal answer.

As well as in section 3, we state that most of tasks in informatics can be reworked into this form.

We offered such tasks on national olympiads but did not meet mention on this genre.

Task 17 (this mathematical task was proposed Pankov, 1970).
Given:
Algorithm 9.

```
P:=1;
readln (N);
Repeat
  writeln ('input natural number');
  readln (X);
  if (X<=N) then
  begin
```

```
    P:=P*X;
    N:=N–X;
  end;
  if (N:=0) then
  begin
    writeln (P);
    end;
  end;
Until (1 = 1); end.
```

Detect the greatest possible number *P* which can be obtained by this algorithm (or, for large *N*): calculate *P mod 1000*.

Remark 4.

- For $N < 100$ this is an easy exercise in dynamical programming.
- For $N < 10000$ there will be problems with too large *P*.
- For $N <$ 10^100 the task is to be solved mathematically (the result is easy to be guessed by the results obtained by dynamical programming for $N = 10, 11, 12,\ldots$).

Examples which also may be called "Arbitrary wandering":

Task 18. Computer models of verbs "pass" and "return" (Pankov *et al.*, 2009).
Comment. "Gate" is the couple of points (20, 39) and (20, 41).
Given:
Algorithm 10.

```
X:=0; Y:=0; X1:=0; Y1:=1; L:=0; P:=false;
Repeat
  writeln ('input one of (R,S,L)');
  readln (M);
  L:=L+1;
  if (M:=R) then
  begin
    X2:=X1+(Y1–Y);
    Y2:=Y1–(X1–X);
  end;
  if (M:=S) then
  begin
    X2:=X1+(X1–X);
    Y2:=Y1+(Y1–Y);
  end;
  if (M:=L) then
  begin
    X2:=X1–(Y1–Y);
    Y2:=Y1+(X1–X);
  end;
  if (X2:=20 and (Y2:=39 or Y2:=41)) then L:=L–1
  else
```

```
    begin
      if (X1:=20 and Y1:=40) then P:=true;
      if (X1=0 and Y1=0 and P:=true) then
      begin
        writeln ('Congratulations! You have PASSED the
        gates and RETURNED');
        writeln (L);
        break;
      end;
    end;
  until (1=1); end.
```

Detect the least possible number *L* which can be obtained by this algorithm.

Task 19 (classical).

Algorithm 11.

```
  Given integer number XM≠0; natural numbers F {jump
  forward}; B {jump backward};
  X:=0;
  Repeat
    writeln ('input one of (–,+)');
    readln (S);
    L:=L+1;
    if (S:='–') then X:=X–B;
    if (S:='+') then X:=X+F;
      if (X:=XM) then
    begin
      writeln (L);
      end;
    end;
  until (1=1); end.
```

Detect the least possible number *L* which can be obtained by this algorithm or "No" if it is not possible.

We did not meet the following type of tasks on games (pendant to 2.2).

General task 20. Rules of a bilateral game are described and the jury's algorithm is given both as listing and as an executable file.

Examine the listing and write a program which will successfully play versus the same algorithm as an executable file.

6. Conclusion

There exist special programs "optimizing" programs (for instance, withdrawing from cycles at programming level, organizing some operations within registers, not with memory at microprogramming level etc.). The last sections of this paper are devoted to "optimizing" algorithms at "intellectual" level.

We hope that types of tasks considered in this paper would both enlarge the scope of tasks in informatics olympiads and contribute to the general problem of computer science in practice: relations between algorithm-statement-of-task and algorithm-solution-of-task.

References

Tikhonov, A.N. (1943). On the stability of inverse problems. *Doklady Akademii Nauk SSSR*, 39 (5), 195–198.

Pankov, P.S. (1970). A problem. *Mathematics in School,* 5. (In Russian)

"Ugale" (2005). *Team Competitions in Mathematics and Informatics "Ugale"*. (In Latvian). `http://www.uvsk.apollo.lv/p113.htm`

Opmanis, M. (2006). Some ways to improve olympiads in informatics. *Informatics in Education*, 5(1), 113–124.

Burton, B.A., Hiron, M. (2008). Creating informatics olympiad tasks: exploring the black art. *Olympiads in Informatics*, 2, 16–36.

Pankov, P.S. (2008). Naturalness in tasks for olympiads in informatics. *Olympiads in Informatics*, 2, 115–121.

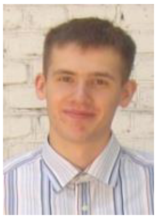
Opmanis, M. (2009). Team competition in mathematics and informatics "Ugale" – finding new task types. *Olympiads in Informatics,* 3, 80–100.

Pankov, P., Dolmatova, P. (2009). Geometrical problems in interactive computer presentation of notions of natural languages. In: *Actual Problems of Control Theory, Topology and Operator Equations. International Jubilee Conference at the Kyrgyz-Russian Slavic University.* Shaker Verlag, Aachen, 85–87.

Pankov, P.S., Baryshnikov, K.A. (2012). Tasks of a priori unbounded complexity. *Olympiads in Informatics,* 6, 110–114.

Pankov, P.S. (2013). Tasks in informatics of continuous content. *Olympiads in Informatics,* 7, 101–112.

P.S. Pankov (1950), doctor of physical-math. sciences, prof., corr. member of Kyrgyzstani National Academy of Sciences (KR NAS), is the chairman of jury of Bishkek City OIs since 1985, of Republican OIs since 1987, the leader of Kyrgyzstani teams at IOIs since 2002. Graduated from the Kyrgyz State University in 1969, is a main research worker of Institute of theoretical and applied mathematics of KR NAS, a professor of the International University of Kyrgyzstan.



K.A. Baryshnikov (1985), LLC "Finance Soft", Bishkek, Kyrgyzstan. Participated in IOI'2002, in training the Kyrgyzstani teams for IOIs in forthcoming years. Graduated from the Kyrgyz-Russian Slavic University in 2007.