Teaching Schoolchildren to Solve Problems on the Topic "Working with Bits"

MICHAEL DOLINSKY
Faculty of Mathematics and Programming Technologies,
Francisk Skorina Gomel State University,
Sovetskaya Str.104, 246019, Gomel,
BELARUS

Abstract: - The author has taught programming to schoolchildren of different ages for many years. All this time, the author has been creating literature for independent study by schoolchildren, trying to present the material in the simplest, most visual, and understandable form possible. This article provides an example of such material for teaching problem-solving in computer science on the topic "Working with bits". Such material may be of interest to teachers both as an illustration of the teaching methodology and in terms of content. At the same time, the author believes that this material may be very useful and interesting for schoolchildren engaged in self-study. All interested parties are offered the following work order: put the article aside and try to independently complete the proposed task the first time - after reading the problem statement, the second time - after reading the instructions for the solution.

Key-Words: - computer science olympiads, instrumental system of distance learning, programming, DL.GSU.BY, secondary school, high school, working with bits, problem-solving.

Received: March 14, 2024. Revised: December 14, 2024. Accepted: February 11, 2025. Published: April 1, 2025.

1 Introduction

Increasing the effectiveness of programming classes largely depends on the availability of literature corresponding to the level of training of students. Therefore, they are actively published books, [1], [2], [3], [4], [5] with system explanations of set of algorithms as well as articles devoted to concrete themes such as loops [6], numeral systems [7], Chinese remainder theorem [8], sorting [9], recursion [10], [11], dynamic programming [12], [13], graphs [14], [15], maximum flow [16], binary indexed tree [17], wavelet tree [18], string hashing [19].

2 Problem Formulation

It has been very accurately noted that the study of programming automatically leads to an increase in motivation for doing mathematics, as well as to an improvement in the quality of assimilation of mathematical knowledge, [20]. This is especially true when the topics of study in programming are closely related to mathematical concepts. One of these topics is the topic "Working with Bits". In parallel with studying the theoretical foundations of the topic, it is also important to develop computational thinking [21] and assess its quality [22]. The study of programming in Gomel begins for those interested in the first grade, so in grades 5-8 there are children who have approached the topic “Working with bits” according to the curriculum, but, of course, books intended for university and high school students are not suitable for their learning.

3 Problem Solution

The author has been teaching programming to schoolchildren of different ages for many years [23] based on the distance learning instrumental system DL.GSU.BY [24], developed under the supervision of the author. All this time, the author has been creating literature for self-study by schoolchildren, trying to present the material in the simplest, clearest, and most understandable form possible. This article provides an example of such material for learning to solve problems in computer science on the topic of "Working with bits". Such material may be of interest to teachers both as an illustration of the teaching methodology and in terms of content. At the

same time, the author believes that this material can be very useful and interesting for both schoolchildren involved in self-study. All those interested are invited to the following order of work: put the article aside and try to independently complete the proposed task after reading the conditions of the task.

3.1 Enumeration of the Set of All Subsets

Often, we have to solve problems in which, in order to obtain answers, we need to go through and analyze all possible combinations of a certain set of elements. We discuss the theory in the process of solving the following Olympiad tasks.

3.2 Going to the Movies (USA Computing Olympiad, 2008, April)

Farmer John is taking some of his cows to the movies! While his truck has a limited capacity of C (100 <= C <= 5000) kilograms, he wants to take the cows that, in the aggregate, weigh as much as possible without exceeding the limit C.

Given N (1 <= N <= 16) cows and their respective weights W_i, determine the weight of the heaviest group of cows that FJ can take to the movies.

INPUT FORMAT:
* Line 1: Two space-separated integers: C and N
* Lines 2..N+1: Line i+1 contains a single integer: W_i

SAMPLE INPUT (file cowflix.in):
259 5
81
58
42
33
61
OUTPUT FORMAT:
* Line 1: A single integer that is the weight of the heaviest group of cows that can go to the movies
SAMPLE OUTPUT (file cowflix.out):
242
OUTPUT DETAILS:
81+58+42+61 = 242; this is the best possible sum
Directions for solution:

To get the answer, you need to go through all possible combinations of one, two, three, ... fourteen, fifteen, and sixteen cows. Each time add up their weights, compare them with the carrying capacity, and remember the largest of the total weights that does not exceed the carrying capacity.

In such problems, you can use the binary representation of numbers.

Recall that a binary number is a sequence of zeros and ones, such as the binary counting sequence from 0 to 15, shown below (Table 1).

Table 1. Decimal and binary counting from 0 to 15

0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

It turns out that this sequence provides us with an enumeration of all possible combinations of four elements. Let the first column be responsible for the presence/absence of the first element, the second column be responsible for the presence/absence of the second element, the third column be responsible for the presence/absence of the third element, and the fourth column be responsible for the presence/absence of the fourth element (for example, 0 means that the element is absent, and 1 that the element is present). Then, line 0000 means that all elements are absent (empty set), line 1111 means that all elements are present, and line 1100 means that the first and second elements are present, and the third and fourth are absent.

Thus, if we create an integer variable i and count in it (using the operator i:=i+1) from 0 to 15, we will get the set of all combinations of 4 elements. And if we have 6 elements, then what is the maximum number we need to count to? Up to 63 (2 to the 6th power minus 1). In general, if we have K elements, then we need to count from 0 to 2 to the power of K minus 1.

During the calculation process, it is necessary to convert the value of the variable I into a binary

representation and check whether this combination of elements fits the problem condition.

Below is a code in the Pascal programming language that solves this problem.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
   ifstream cin("cowflix.in");
   ofstream cout("cowflix.out");
   int c,n,a[1000],max,i,x,sum,k;
   cin >> c >> n;
   for (i=1; i<=n; i++) cin >> a[i-1];
   max=0;
   for (i=1; i<32*1000; i++)
   {
      x=i;
      sum=0; k=0;
      while (x!=0)
      {
         if (x%2!=0) sum=sum+a[k];
         k++;
         x=x >> 1;
      }
      if ((sum<=c) and (sum>max)) max=sum;
   }
   cout << max;
}
```

Explanations:

The weights of the cows are read into the array a[i] from 0 to 15. Accordingly, the cows are numbered from 0 to 15. The variable k indicates the number of the cow and, at the same time, the number of the bit (digit in binary representation) of the number i. The bits are also numbered from 0 to 15 from right to left.

3.3 Cow Pizza (USA Computing Olympiad 2006 April Bronze)

Oh, how the cows love their pizza! Even though they are picky, they do love variety, too. They order from the local pizza parlor that features T (1 <= T <= 20) different toppings in addition to a complete set of soft drinks and healthy salads.

The toppings are conveniently numbered 1..T so the cows can order by number. Your job is to calculate how many possible pizzas can be created given that some cows will not tolerate various combinations of toppings (e.g., some cows simply will not eat Anchovies or the combination of Mushrooms and Asparagus). Given a set of N (1 <= N <= 52) constraints, figure out how many pizzas can be made using all possible combinations of the ingredients (which, of course, includes no ingredients at all). Each constraint is a set of numbers of size 1..T that lists the ingredients that disqualify a pizza from being considered. A constraint like "5 3" means that no pizza can contain ingredient #5 and also ingredient #3. This means pizzas with ingredients 3, 5, and 6 will not be counted as acceptable.

INPUT FORMAT:
* Line 1: Two space-separated integers: T and N
*Lines 2..N+1: Each line describes a constraint using space-separated integers. The first integer is the number of ingredients in the constraint, Z (1 <= Z <= T). The subsequent Z integers (which are unique) list the ingredient(s) whose combination disqualifies a pizza from consideration for the cows.
SAMPLE INPUT (file pizza.in):
6 5
1 1
2 4 2
3 3 2 6
1 5
3 3 4 6
INPUT DETAILS:
Six ingredients numbered 1..6. Five constraints: No pizza can contain ingredient 1, no pizza can contain ingredients numbered 2 and 4, etc.
OUTPUT FORMAT:
* Line 1: A single integer that is the total number of pizzas that can be created using the number of Ingredients and constraints.
SAMPLE OUTPUT (file pizza.out):
10
OUTPUT DETAILS:

This includes pizzas with: no incredients; ingredient 1; ingr's 2, 3; ingr's 2, 6; ingr 2; ingr's 3, 4; ingr's 3, 6; ingr 3; ingr's 4, 6; and ingr 4.

Instructions for solving the problem:
First of all, let's pay attention to the error in the explanations of the answer (OUTPUT DETAILS in the original). The answer 10 itself is correct, but the given list of permitted combinations of seasonings is not. For example, pizza with seasoning 1 is

prohibited by the first restriction. And this is not a translation error. Those who wish can check that the English (original) version of the conditions contains the same numbers in OUTPUT DETAILS. The error was deliberately not corrected to emphasize once again that this is possible in real Olympiads. If the participant has the opportunity, he should check with the jury, and if not, MAKE A DECISION HIMSELF. Compared to the previous task (enumerating all possible options), a complication is introduced here - it is necessary to exclude some options from consideration, and generalized lists of these prohibited options are presented: for example, exclude all options containing element 1 or all options containing elements 4 and 2. And here the binary system comes to our aid. You can create an array of numbers p2, which are powers of 2 from 0 to 20 (there are a maximum of 20 spices). And directly at the input, encode the forbidden combination of options into the number e, and their sequence into the array z[i]. See the fragment of code below:

```
Int p2[21] = {1,2,4,8,16,32,64, 128,256,512,1024,
2048,4096,8192,16384,32768,64*1024,128*1024,25
6*1024,512*1024,1024*1024};
…
   cin >> t >> n;
   for (i=1; i<=n; i++)  {
      cin >> k; e=0;
      for (j=1; j<=k; j++)  {
         cin >> x;
         e+=p2[x-1];
      }
       z[i]=e;
  }
```

And the check whether the combination z[i] prohibits the current version of x is performed as follows:

1) j=x & z[i];

As a result, the variable j will receive the value 0 at all positions where z[j] contains 0, and at all other positions (where z[j]=1) the value of the corresponding bit from the variable x will be copied there.

Let us recall the truth table for the and operation.

x	z[i]	and
0	0	0
0	1	0
1	0	0
1	1	1

2) j=j ^ z[i];

Now, if j and z[i] have the same bits in one position (0 in both variables, or 1 in both variables), then the corresponding bit of variable j will receive the value 0. But if j and z[i] have different values (0 in one and 1 in the other, or 1 in one and 0 in the other), then the corresponding bit of variable j will receive the value 1.

return j!=0;

This statement assigns the function the value true if the variable j is not equal to 0 and the constraint z[i] does not prohibit the combination x.
The full text of the decision is provided below.

```
#include <bits/stdc++.h>
using namespace std;
int p2[21] = {1,2,4,8,16,32,64,128,256,512,1024,
2048,4096,8192,16384,32768,64*1024,128*1024,25
6*1024,512*1024,1024*1024};
int t,n,i,j,k,z[53],x,e;

bool good2 (int i,int x) {
   int j;
   j=x & z[i];
   j=j ^ z[i];
   return j!=0;
}

bool good (int x) {
   int i=1;
   while ((i<=n) && (good2(i,x))) i++;
   return i>n;
}
int main() {
   ifstream cin("pizza.in");
   ofstream cout("pizza.out");
   cin >> t >> n;
   for (i=1; i<=n; i++)  {
      cin >> k; e=0;
      for (j=1; j<=k; j++) {
         cin >> x;
         e+=p2[x-1];
       }
      z[i]=e;
   }
   j=1;
   for (i=1; i<p2[t]-1; i++)
      if (good(i))  j++;
    cout << j;
}
```

3.4 Problem the Water Bowls (USA Computing Olympiad 2006 January Bronze)

The cows have a line of 20 water bowls from which they drink. The bowls can be either right-side-up (properly oriented to serve refreshing cool water) or upside-down (a position which holds no water). They want all 20 water bowls to be right-side-up and thus use their wide snouts to flip bowls. Their snouts, though, are so wide that they flip not only one bowl but also the bowls on either side of that bowl (a total of three or -- in the case of either end bowl -- two bowls).

Given the initial state of the bowls (1=undrinkable, 0=drinkable -- it even looks like a bowl), what is the minimum number of bowl flips necessary to turn all the bowls right-side-up?

INPUT FORMAT:

* Line 1: A single line with 20 space-separated integers

SAMPLE INPUT (file bowls.in):

0 0 1 1 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0

OUTPUT FORMAT:

* Line 1: The minimum number of bowl flips necessary to flip all the bowls right-side-up (i.e., to 0). For the inputs given, it will always be possible to find some combination of flips that will manipulate the bowls to 20 0's.

SAMPLE OUTPUT (file bowls.out):

3

OUTPUT DETAILS:

Flip bowls 4, 9, and 11 to make them all drinkable:

0 0 1 1 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 [initial state]

0 0 0 0 0 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 [after flipping bowl 4]

0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 [after flipping bowl 9]

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 [after flipping bowl 11]

Directions for solution:

First, when entering, it is necessary to convert the entered sequence of 0 and 1 into a number represented by the following binary sequence:

```
b=0;
for (i=1; i<=20; i++) {
   cin >> k;
   if (k==1) b=b+p2[20-i];
}
```

Here P2 is again an array of powers of two.

B is the number corresponding to the entered binary sequence. The main iteration loop might look like this:

```
mn=1e9;
for (i=0; i<=p2[20]-1; i++) {
      k=numbit(i);
      if ((k<mn) && (zero(i,b))) mn=k;
}
```

Here the units in the number I will indicate which cylinders need to be turned over as central ones. The numbit(i) function calculates the number of ones in the number i. The function zero(i,b) checks whether turning over the initial state of cylinders b according to the law i will lead to the required state 0.

If there are fewer units than the current minimum, and flipping it turns it into 0, then we remember the new number of units (the cylinders that need to be flipped as central ones).

The number of ones in a number x is calculated using the AND operation. Recall that the numbers in the p2 array (powers of two) are characterized by having a 1 in one bit and zeros in all the others.

```
k=0;
for (i=0; i<=19; i++)
   if ((x & p2[i])!=0) k++;
```

Finally, the test to see if combination i takes the original sequence b to the zero state is as follows:

b – original cylinders
i – defines the cylinders to be turned over

```
if ((i & p2[0])  !=0) b=b^(p2[0]+p2[1]);
if ((i & p2[19])!=0) b=b^(p2[19]+p2[18]);
for (j=1; j<=18; j++)
  if ((i & p2[j])!=0) b=b^(p2[j-1]+p2[j]+p2[j+1])
return b==0;
```

Let us also recall that, according to the problem statement, in the case of extreme cylinders (0 and 19), only one neighboring one is flipped. And in the case of "middle cylinders" (1 through 18), it and both its neighbors - to the left and to the right - are flipped. The flipping is performed using the xor operation. If the result is that the variable b gets the value 0, then the combination I flips all the cylinders back to their original state. The full text of the decision is provided below.

```
#include <bits/stdc++.h>
using namespace std;
int p2[21],i,k,j,b,mn;

int numbit(int x) {
   int k,i;
   k=0;
   for (i=0; i<=19; i++)
      if ((x & p2[i])!=0) k++;
   return k;
}

bool zero(int i,int b){
   int j;
   if ((i & p2[0])!=0) b=b^(p2[0]+p2[1]);
   if ((i & p2[19])!=0) b=b^(p2[19]+p2[18]);
for (j=1; j<=18; j++)
    if ((i & p2[j])!=0) b=b^(p2[j-1]+p2[j]+p2[j+1]);
return b==0;
}

int main() {
   ifstream cin("bowls.in");
   ofstream cout("bowls.out");
   p2[0]=1;
   for (i=1; i<=20; i++) p2[i]=2*p2[i-1];
   b=0;
   for (i=1; i<=20; i++) {
      cin >> k;
      if (k==1) b=b+p2[20-i];
   }
   mn=1e9;
   for (i=0; i<=p2[20]-1; i++) {
      k=numbit(i);
      if ((k<mn) && (zero(i,b))) mn=k;
   }
   cout << mn;
}
```

3.5 Training Methodology

The described problems are part of a package of problems on the topic “Working with bits”. The student is asked to solve problems in order of increasing difficulty. The DL system provides automatic verification of solutions in the programming languages Pascal, C++, Python, Java, C#, and the original author’s set of tests. If a student’s solution on some test produces an answer that is not equal to the author’s, the student can take the input data and the author’s answer on this test from the DL system, find an error in his program, correct it and send it for testing again. If a student cannot develop a complete solution to a problem, he receives a teacher's description of the solution to the problem, including the source text of the correct solution in Pascal, parses it, and, if necessary, can consult with other students or the teacher. Then he marks what is needed in his notebook. And he always rewrites the solution without using the notes. More precisely, he can use notes, but in this case, he must rewrite the solution from a blank screen, and so on until he writes the solution completely himself, from the first to the last letter.

4 Conclusion

This article provides material for teaching how to solve computer science problems on the topic "Working with bits". The technical basis of the methodology is the developed instrumental system of distance learning (Distance Learning Belarus - http://dl.gsu.by). All the tasks given in the article can be passed in the course "Methods of Algorithmization".

References:

[1] Dasgupta, S., Papadimitriou, C. H., & Vazirani, U. V. (2006). Algorithms. NewYork: McGraw-Hill, ISBN: 978-0073523408.

[2] Kleinberg, J., Tardos E. (2006). Algorithm Design Pearson/Addison-Wesley, ISBN: 978-0321295354.

[3] Cormen, T.H., Leiserson, C.E, Rivest, R.L., Stein C. (2009). Introduction to algorithms. MIT Press, ISBN: ISBN: 9780262533058.

[4] Shen, A. (2010). Algorithms and Programming: Problems and solutions. New York: Springer. https://doi.org/10.1007/978-1-4419-1748-5.

[5] Skiena, S. (2017). Analysis of algorithms: Backtracing. Stony Brook University New York, [Online]. https://www3.cs.stonybrook.edu/~skiena/373/newlectures/lecture15.pdf (Accessed Date: December 16, 2024).

[6] Vaníček, J., Dobiáš, V., Šimandl, V. (2023). Understanding loops: What are the misconceptions of lower-secondary pupils? *Informatics in Education,* 22(3), 525-554. https://doi.org/10.15388/infedu.2023.20.

[7] Matos, Z., Kónya, E. (2021) Teaching numeral systems based on history in high school. *Annales Mathematicae et Informaticae* 54. pp. 195-204 https://doi.org/10.33039/ami.2021.08.003.

[8] Ďuriš, V., Bojdová, V., Šumný, T. (2022) Solving selected problems on the Chinese remainder theorem. *Annales Mathematicae et Informaticae.* 55. pp. 196-207. https://doi.org/10.33039/ami.2022.02.002.

[9] Nikházy, L., Nloszaly, Á., Deak, B.(2021) Why You Should Know and Not Only Use Sorting Algorithms: Some Beautiful Problems. *Olympiads in Informatics*, 2021 Vol. 15, pp.53–74, https://doi.org/10.15388/ioi.2021.05.

[10] Menyhárt, L., & Zsakó, L. (2021). Task variations for backtrack. *Teaching Mathematics and Computer Science*, *18*(2), 107–120. https://doi.org/10.5485/TMCS.2020.0511.

[11] Thorgeirsson, S., Lais, L.C., Weidmann, T.B., Su,Z. (2024). Recursion in Secondary Computer Science Education: A Comparative Study of Visual Programming Approaches. *In Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024).* Association for Computing Machinery, New York, NY, USA, 1321–1327. https://doi.org/10.1145/3626252.3630916.

[12] Forisek, M. (2015). Towards a better way to teach dynamic programming. *Olympiad in Informatics,* vol.9, pp.45–55, http://dx.doi.org/10.15388/ioi.2015.05.

[13] Németh, A.E., Zsako L. (2016). The Place of the Dynamic Programming Concept in the Progression of Contestants' Thinking. *Olympiads in Informatics*, 2016, Vol. 10, pp. 61–72, https://doi.org/10.15388/ioi.2016.04.

[14] Nemeth, Á.E. (2017). Teaching graphs for contestants in lower-secondary-school-age *Olympiad in Informatics,* vol.11, pp.41–53, https://doi.org/10.15388/ioi.2017.04.

[15] Do, P.T., Pham B.,T., Than V.,C. (2020). Latest Algorithms on Particular Graph Classes *Olympiads in Informatics*, Vol. 14, pp.21–35, https://doi.org/10.15388/ioi.2020.02.

[16] Fontaine M.C. (2018) Tidal Flow: A Fast and Teachable Maximum Flow Algorithm. *Olympiads in Informatics,* 2018, Vol. 12, pp.25–41, doi: 10.15388/ioi.2018.03.

[17] Dima, M., Ceterchi, R. (2015). Efficient Range Minimum Queries using Binary Indexed Trees.*Olympiads in Informatics*, 2015, Vol. 9, pp.39–44 http://dx.doi.org/10.15388/ioi.2015.04.

[18] Castro, R.,Lehmann, N.,Perez, J.,Subercaseaux, B. (2016). Wavelet Trees for Competitive Programming. *Olympiads in Informatics*, 2016, Vol. 10, pp.19–37, https://doi.org/10.15388/ioi.2016.02.

[19] Pachocki, J., Radoszewsk J. (2013). Where to Use and How not to Use Polynomial String Hashing. *Olympiads in Informatics,* 2013, Vol. 7, pp.90–100, [Online]. https://ioinformatics.org/journal/INFOL119.pdf. (Accessed Date: December 16, 2024).

[20] Fried, K., Fekete, I., & Princz, P. (2020). Better understanding mathematics by algorithmic thinking and computer programming. *Teaching Mathematics and Computer Science*, *18*(4), pp.295–305. https://doi.org/10.5485/TMCS.2020.0486.

[21] Ginat, D. (2022). Abstraction, Declarative Observations and Algorithmic Problem Solving. *Informatics in Education*, 20(4), 567-582. https://doi.org/10.15388/infedu.2021.25.

[22] Bubica, N., Boljat, I. (2022). Assessment of Computational Thinking – A Croatian Evidence-Centered Design model. *Informatics in Education,* 21(3), 425-463. https://doi.org/10.15388/infedu.2022.17.

[23] Dolinsky, M. (2024) Teaching Programming to Schoolchildren in Gomel (Belarus). *WSEAS Transactions on Advances in Engineering Education*, vol.21, pp.11-16, https://doi.org/10.37394/232010.2024.21.2.

[24] Dolinsky, M. (2022). Instrumental System of Distance Learning DL.GSU.BY and Examples of its application. *Global Journal of Computer Science and Technology Interdisciplinary*, 22(1), pp.45-53, [Online]. https://globaljournals.org/GJCST_Volume22/6-Instrumental-System-of-Distance-Learning.pdf (Accessed Date: December 16, 2024).

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The author contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The author has no conflicts of interest to declare.