SOME ALGORITHMS AND PROGRAMS FOR EASY PROBLEMS

Authors: Nicolai FALICO, assoc. prof., PhD; Mihail KULEV, assoc. prof., PhD

Technical University of Moldova

Abstract: *In the paper four simple problems have been discussed. The straightforward solutions of these problems may be very complicated. Below easy algorithms for solving such kind of problems have been proposed.*

Keywords: *Easy algorithm, problem and solution, code in C language.*

1. Introduction

Among the programming problems the following one can be often obtained: for a sequence of similar elements (usually numbers) it is required to be found some characteristic (standard deviation, minimum value, continuous subsequence with the largest sum etc.) by only one traversal of the sequence [1]. Additional restriction – the sequence may be very long and can be not fit in the memory. Other restrictions on the sequence are not normally applied**.** With these tasks all is more or less clear. But there are also other problems. They have additional restrictions on sequence elements in whole, and these restrictions have to be used essentially to solve the problem. Also, sequence often may be very long.

2. Problems and Solutions

The simplest one from such problems is as follows:

Problem 1. The sequence contains the integers from 1 to N in a random order, but one of the numbers is missing (the others are found in the sequence exactly once). N is not known beforehand. Determine the missing number.

Solution.

The solution is obvious. We input these numbers and find the sum S of all numbers. By assumption the sum of all numbers from 1 to N will be equal to (N + 1) * (N) / 2, and the missing number equals (N + 1) * (N) / 2-S. If for some reason you are afraid of overflow, then work with unsigned numbers (be careful when calculating the (N + 1) * (N) / 2).

Code in C language:

```
S = 0;
for(i=0;i<N-1;i++){
        scanf("%d",&tmp);
        S += tmp;
}
printf("%d", (N+1)*N/2-S);
```

Problem 2. The sequence contains integers. Total amount of numbers may be very large. One of the numbers occurs an odd number of times, the other - an even number of times. Find the number that occurs an odd number of times.

Solution.

The following identities are valid for every integers A, B, C and follow from definition XOR:
A XOR A = 0, A XOR B = B XOR A, (A XOR B) XOR C = A XOR (B XOR C). Therefore, everything is simple: find the XOR of all the numbers and this will be the answer. More simply, the same elements in the summation mutually destroyed.

Code in C language:

```
s=0;
for(i=0;i<N;i++){
        scanf("%d",&tmp); s^=tmp; }
printf("%d",s);
```

Problem 3. The sequence contains integers, more than half of which are equal to the same number X. Find this number. The algorithm must be liniar.

Solution.

Note that if we delete from sequence two different numbers, the condition of the problem remains true.

Therefore, we can delete pairs of different numbers until all elements are not equal to the same number. This number will be X.
To implement this method, make the cell in which we will store any integers from the sequence, and the counter - how many copies of this element we have seen and has not yet deleted. When we read the next element, we have three options:
- Counter is zero. Put a read element in a cell, increasing the count to 1
- The next element is the value in the cell. Increase the counter by 1
- The next element is not equal to the value in the cell. Reduce the count to 1.
Once we will review all sequence number would be required in the cell.
Code in C language:

```
int cell,counter=0;
for(i=0;i<N;i++){
        scanf("%d",&tmp);
        if (counter==0) cell=tmp;
        else if(tmp==cell) counter++;
        else   counter--;
}
printf("%d",cell);
```

Problem 4. The sequence contains integers. Total amount of numbers may be very large. Find the maximum product of three elements of the sequence [2].
Solution.
If all integers are positive then the answer is the product of three maximum numbers. But in the sequence may be negative numbers. Therefore on every step we must find not only maximum product of three numbers but also, maximum and minimum products of two numbers and maximum and minimum numbers. Details are clear from the code of program.
Code in C language:

```
#define max(a,b) ((a)>(b) ? (a) : (b))
#define min(a,b) ((a)<(b) ? (a) : (b))
highest = max(list_of_ints[0], list_of_ints[1])
lowest =  min(list_of_ints[0], list_of_ints[1])
highest_product_of_2 = list_of_ints[0] * list_of_ints[1]
lowest_product_of_2  = list_of_ints[0] * list_of_ints[1]
highest_product_of_three = list_of_ints[0] * list_of_ints[1] * list_of_ints[2]
for (i=2;i<n;i++){
      highest_product_of_three =max( highest_product_of_three,
                  max(current * highest_product_of_2,      current * lowest_product_of_2) );
      highest_product_of_2 = max( highest_product_of_2,
                                  max(current * highest, current * lowest));
      lowest_product_of_2 = min( lowest_product_of_2,
                           min( current * highest, current * lowest) );
}
 highest = max(highest, current);
 lowest = min(lowest, current);
```

Bibliography

1. Д.Э. Кнут ., Искусство программирования . т.2. Третье издание . Москва, 2000.
2. С. Окулов., Программирование в алгоритмах.,изд. Бином.Лаборатория знаний, 2002.
3. George E. Collins, David R. Musser: Analysis of the Pope-Stein Division Algorithm. Information Processing Letters, 151-155, V.6, N.5, October 1977.
4. E.V. Krishnamurthy, Salil K. Nandi: On the normalization requirement of divisor in divide-and-correct methods. Communications of the ACM (CACM), 809-813. V. 10, N. 12, December 1967.