Analysis: GLO
Gloves

HISTORY:
• v. 1.00: 2008.03.13, MKed, elaboration of the problem

dokument systemu SINOL 1.6

1 Introduction

In the problem we are given two n terms-long progressions: (a_i) and (b_i) of non-negative integers and we have to produce two numbers $0 \le A \le \sum_i a_i$ and $0 \le B \le \sum_i b_i$, such that for each two progressions a'_i and b'_i satisfying the following conditions:

$$\forall_i 0 \le a'_i \le a_i \tag{1}$$

$$\forall_i 0 \le b'_i \le b_i \tag{2}$$

$$\sum_i a'_i = A \tag{3}$$

$$\sum_i b'_i = B \tag{4}$$

it is true that:

$$\exists_{i \in \{1,2,\ldots,n\}} a'_i > 0 \wedge b'_i > 0 \tag{5}$$

We will call such pairs (A,B) *acceptable*. Among all acceptable pairs, we are asked to produce one, which minimizes the sum $A+B$. Obviously, there may be more than one correct solution.

2 Notation

- Let us denote $Z_n = \{1,2,\ldots n\}$.
- For any $X \subset Z_n$ let us denote $A_X = \sum_{i \in X} a_i$ and $B_X = \sum_{i \in X} b_i$.

3 Model solution

3.1 Acceptable pairs

First of all, we should find a way to determine, whether a pair (A,B) is acceptable or not.

Theorem 1
A pair (A,B), for which $0 \le A \le \sum_i a_i$ and $0 \le B \le \sum_i b_i$ is acceptable if and only if $\forall_{X \subset Z_n} A > A_X \vee B > B_{Z_n \setminus X}$.

Proof
Let us assume, that a pair (A,B) is not acceptable. It means, that there exists a pair of progressions $((a'_i),(b'_i))$, whose sums are A and B respectively, and:

$$\forall_i a'_i = 0 \vee b'_i = 0 \tag{6}$$

If we denote $X = \{i : a'_i > 0\}$ and $Y = \{i : b'_i > 0\}$, then assumption (6) says, that $X \cap Y = \emptyset$.

Moreover $A \le A_X \wedge B \le B_Y$. Because X and Y are disjoint, $Y \subset Z_n \setminus X$, so due to terms of (b_i) being non-negative: $B \le B_Y \le B_{Z_n \setminus X}$.

Thus we have proved that if a pair (A,B) is not acceptable, then $\exists_{X\subset Z_n} A \le A_X \wedge B \le B_{Z_n\setminus X}$.

Now let us take a pair, which is acceptable. Then if we take any progressions $(a'_i),(b'_i)$ satisfying (1) – (4), then sets X and Y constructed as above are not disjoint.

It means, that for any disjoint sets $X,Y \subset Z_n$ there exists no pair of progressions satisfying conditions (1) – (5) and such, that

$$X = \{i : a'_i > 0\} \wedge Y = \{i : b'_i > 0\} \tag{7}$$

If for any pair of progressions $(a'_i),(b'_i)$ satisfying (1) – (2) and (7) it was true, that $\sum_i a'_i \ge A \wedge \sum_i b'_i \ge B$, then decreasing some terms we would find a pair of progressions satisfying (1) – (4) and (7). A contradiction. This leads to a conclusion, that for every pair of progressions satisfying (1) – (2) and (7) we have $\sum_i a'_i < A \vee \sum_i b'_i < B$. In particular, for

$$a'_i = \begin{cases} 0 & i \notin X \\ a_i & i \in X \end{cases} \qquad b'_i = \begin{cases} 0 & i \notin Y \\ b_i & i \in Y \end{cases}$$

we get $A_X < A \vee B_Y < B$. Taking $Y = Z_n \setminus X$ we get, that $\forall_{X\subset Z_n} A_X < A \vee B_{Z_n\setminus X} < B$, Quod Erat Demonstradum.

3.2 Minimization

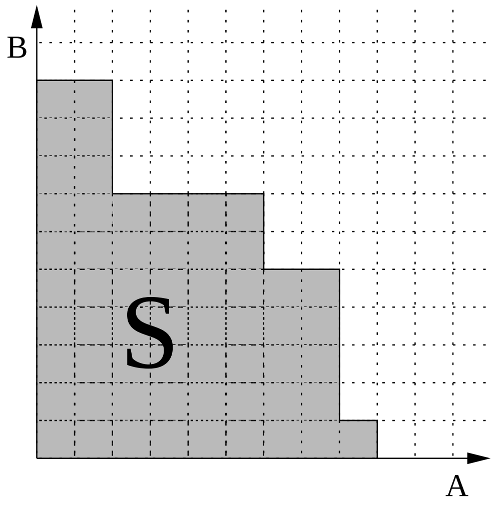
Theorem 1 simplifies the problem significantly. Now, we can check all 2^n subsets $X \subset Z_n$, and for each one compute the pair $(A_j,B_j) = (A_X, B_{Z_n\setminus X})$. For each of these pairs (A_j,B_j) we can draw a rectangle $[0,A_j] \times [0,B_j]$ on the plane $\Re^2$. A pair (A,B) will be acceptable if and only if the point (A,B) is outside all of the drawn rectangles and at the same time inside the rectangle $R = [0,A_{Z_n}] \times [0,B_{Z_n}]$.

Finally, we have to minimize the sum $A+B$ among all acceptable points, i.e. among all points outside all of the drawn rectangles and inside the rectangle R. In order to do this, let us notice, that the sum of all the rectangles:

$$S = \sum_{1\le j\le 2^n} [0,A_j] \times [0,B_j]$$

is a very remarkable figure. Namely, it is a kind of a 'staircase shaped' poly-line with interior. I will simply call such a figure *a staircase*.

An example of a staircase is shown below:



The acceptable points are all the points outside set S and within the rectangle R. We are to find the one with a minimal sum of coordinates $A+B$. Obviously the sought point must be a neighbour of some point $p \in S$, because otherwise, it would not be minimal. Moreover, we only have to consider neighbours of the vertices of the drawn staircase.

3.3 Algorithm

Our remarks let us construct the following algorithm:

```
1. for each subset $X \subset Z_n$:
    1a. compute $A = A_X$ and $B = B_{Z_n \setminus X}$
    1b. put the pair $(A,B)$ to the sequence $T$
2. sort the sequence $T$ lexicographically
3. iterate through sequence $T$ building the staircase on a stack $st$
4. for each concave vertex $(A,B)$ of the staircase $st$ consider the point $(A+1,B+1)$ as a candidate for the solution
5. Among all considered candidates find the point $(A,B)$, which minimizes the sum $A+B$ and print it
```

This algorithms needs yet some clarification.

Having sorted the sequence T of pairs representing rectangles, we do not have to worry for each rectangle, where to put it in the constructed staircase, because it will always fit 'at the end'. There is however a need to check, whether this new rectangle does not include any other rectangle as a subset. Fortunately, it is enough to check (and delete if needed) the most recently added rectangles. This makes the problem of building the staircase easier — we can store it on a stack st and for each consequent rectangle r first pop the stack as long as its top is a subset of r and then put r on the stack st.

In the previous subsection we have concluded that it is enough to check only the neighbours of the vertices of the staircase and find the minimal point among them. However, we can put the simplification further. Namely, it is sufficient to check only the 'north-east' neighbour of each *concave* vertex, i.e. a vertex of the staircase, which is not a vertex of any rectangle, but a point at which borders of two rectangles intersect. On the image depicting a sample staircase there are three concave vertices.

This solution has time complexity $O(k \cdot \log k) = O(n \cdot 2^n)$, where $k = 2^n$ and it is due to the performed sorting. All other operations together have complexity $O(k)$.

The model solution has been implemented in `prog/glo.cpp` and `prog/glo2.pas`.

4 Other solutions

4.1 Slower solutions

`prog/glos1.cpp`, `prog/glos11.pas` This is the most brute-force solution I have prepared. It iterates through all consequent sums of gloves (left and right) possible to take. For every such sum it iterates through all partitions of the sum into left and right gloves and for each pair (L,R) such that $L+R = sum$, checks recursively all possible choices of L coloured left gloves and R coloured right gloves from respective drawers in order to find out, whether the pair (L,R) is acceptable.

It prints the first acceptable pair found and quits.

The accurate time complexity of this solution is difficult to state.

prog/glos2.cpp, prog/glos12.pas The same solution as `prog/glos1.cpp`, but taking advantage of the fact, that when we check whether a pair (L,R) is acceptable, we do not need to try all possible choices of coloured gloves, but only try to construct a choice without an equally-coloured pair.

The accurate time complexity of this solution is difficult to state.

prog/glos3.cpp, prog/glos13.pas The same solution as `prog/glos2.cpp`, but taking at once all the left gloves of one colour or all the right gloves of this colour, without trying only some of them.

Time complexity: $O((solA+solB)^2 * 2^n)$, where $(solA, solB)$ is the correct solution.

prog/glos4.cpp, prog/glos14.pas A slightly modified model solution. The only difference is that this one draws all rectangles on a bitmap and then finds the minimal unchecked point on it. In program `prog/glos4.cpp` the bitmap is dynamically allocated, whereas in `prog/glos14.pas` it is a static array.

Time complexity: $O(\sum_{X\subset Z_n} A_X \cdot B_{Z_n\setminus X}) = O(A_{Z_n} \cdot B_{Z_n} \cdot 2^n)$.

prog/glos5.cpp, prog/glos15.pas This solution is based on the problem's author's idea: We prepare sequence T (as in the model solution), then we iterate through all pairs of rectangles, intersect each pair, for each such point of intersection (A, B) check, whether the point (A+1, B+1) is outside every rectangle and if so - we make it a candidate for the minimal point. We still use guardians as in the model solution.

Time complexity: $O(2^{3n})$.

prog/glos6.cpp, prog/glos16.pas This is the model solution without use of a stack. After creating sequence T and sorting it, this solution checks for each rectangle, whether it fits inside any other rectangle and if so — deletes it. Then the sequence with marked deleted rectangles emulates the stack used in the model solution to find the minimal acceptable point.

Time complexity: $O(2^{2n})$.

prog/glos7.cpp, prog/glos17.pas This solution is based on the problem's author's idea: By dynamic programming we compute the sequence $t[]$ (described by the problem's author) and transform it to the sequence $T[]$ (which has the same meaning as in the model solution). Then we sieve the sequence $T[]$ using a stack and find the minimal point just as in the model solution.

Time complexity: $O(n \cdot (max_i\{a_i\} + max_i\{b_i\}))$. This solution would work for much greater n than 20, but only for relatively small numbers of gloves (around 10^6). For big numbers of gloves these programs enter an infinite loop (in order to follow formal requirements, which do not accept incorrect answers from slower solutions).

4.2 Incorrect solutions

prog/glob1.cpp The same as the model solution, but without making sure, that the found pair (A,B) satisfies $A \le \sum_i a_i$ and $B \le \sum_i b_i$, i.e. that there exist enough left or right gloves in respective drawers. In order to make sure, that the minimal point outside the staircase is found (but not necessarily inside the rectangle $[0, A_{Z_n}] \times [0, B_{Z_n}]$), this program adds two 'guardians' to the staircase, which are the rectangles $[0,\infty] \times [0,0]$ and $[0,0] \times [0,\infty]$ and then it considers the neighbour $(x+1, y+1)$ of each concave vertex (x,y) of the built staircase.

A problem with this can occur only in a situation, where one of the rectangles $[0, A_\emptyset] \times [0, B_{Z_n}]$ or $[0, A_{Z_n}] \times [0, B_\emptyset]$ is a subset of another rectangle, because in such a situation it would be deleted in the process of building the staircase.

Time complexity: $O(n \cdot 2^n)$.

`prog/glob2.cpp` The same as the model solution, but using 32-bit integers instead of 64-bit ones.

5 Tests

I have prepared the following tests:

- `glo0.IN` (ε sek.) $n = 4$, sample test case (from the problem statement)
- `glo1ocen.IN` (ε sek.) $n = 4$, a small random test
- `glo2ocen.IN` (ε sek.) $n = 5$, a small random test for correctness, shows problem with `prog/glob1.cpp`
- `glo3ocen.IN` (ε sek.) $n = 7$, a medium random test
- `glo4ocen.IN` (ε sek.) $n = 10$, a medium schematic test: all $a_i = 1024$ and all $b_i = 123$
- `glo5ocen.IN` (1 sek.) $n = 20$, a big schematic test with all numbers but n equal to $1\,000\,000\,000$
- `glo1 -- 9.IN` (ε sek.) $n = 4$, all numbers less or equal to 10; small random tests accepting `prog/glos11.pas` and better solutions
- `glo10.IN` (ε sek.) $n = 5$, numbers ≤ 10; a small random test accepting `prog/glos12.pas` and better solutions
- `glo11.IN` (ε sek.) $n = 15$, numbers ≤ 20; a random test with medium n and small other numbers
- `glo12a.IN` (ε sek.) $n = 14$, numbers ≤ 80; a random test with medium n and small other numbers
- `glo12b.IN` (ε sek.) $n = 13$, numbers ≤ 50; a medium random test with many zeros detecting an error in `prog/glob1.cpp`
- `glo13a.IN` (ε sek.) $n = 8$, numbers $\leq 1\,000$; a random test with small n and medium other numbers
- `glo13b.IN` (ε sek.) $n = 6$, numbers $\leq 2\,000$; a medium random test with many zeros detecting an error in `prog/glob1.cpp`
- `glo14a.IN` (ε sek.) $n = 7$, numbers $\leq 10\,000$; a random test with small n and medium other numbers
- `glo14b.IN` (ε sek.) $n = 7$, numbers $\leq 15\,000$; a medium random test with many zeros detecting an error in `prog/glob1.cpp`
- `glo15a.IN` (ε sek.) $n = 10$, numbers $\leq 500\,000$; a medium random test
- `glo15b.IN` (ε sek.) $n = 10$, numbers $\leq 100\,000$; a test detecting an error in `prog/glob1.cpp`
- `glo16a.IN` (ε sek.) $n = 10$, a test with consequent powers of 2 from 2^{20} to 2^{29}, shuffled
- `glo16b.IN` (ε sek.) $n = 10$, numbers $\leq 1\,000\,000\,000$; a test detecting an error in `prog/glob1.cpp`

- `glo17.IN` (0.25 sek.) $n = 18$, a test with consequent powers of 2 from 2^0 to 2^{17}, shuffled
- `glo18.IN` (1.2 sek.) $n = 20$, numbers $\leq 20\,000$; a random test with big n and medium other numbers
- `glo19.IN` (1.2 sek.) $n = 20$, numbers $\leq 50\,000$; a random test with big n and medium other numbers
- `glo20 -- 22.IN` (1.2 sek.) $n = 20$, numbers $\leq 1\,000\,000\,000$; big random tests designed for sieving out the optimal solution

5.1 Statistics

The program `prog/gloinwer.cpp` produces some statistical data concerning the tests. Here is the explanation of respective coefficients:

- n, $max\_a = max_i\{a_i\}$ and $max\_b = max_i\{b_i\}$
- $subsums\_a$ and $subsums\_b$ which describe the number of different numbers A_X and B_X for all possible $X \subset Z_n$.
- $density\_a = subsums\_a \div max\_subsum\_a$, where $max\_subsum\_a = max_{X \subset Z_n}\{\sum_{i \in X} a_i\} = \sum_{i \in Z_n} a_i$
- $density\_b = subsums\_b \div max\_subsum\_b$, where $max\_subsum\_b = max_{X \subset Z_n}\{\sum_{i \in X} b_i\} = \sum_{i \in Z_n} b_i$

5.2 Performance of different solutions

These are different correct solutions along with lists of test, which they pass:

`prog/glos1.cpp`, `prog/glos11.pas` 0, 1ocen, 2ocen and 1 — 9.

`prog/glos2.cpp`, `prog/glos12.pas` 0, 1ocen, 2ocen and 1 — 10.

`prog/glos3.cpp`, `prog/glos13.pas` 0, 1ocen — 4ocen and 1 — 12.

`prog/glos4.cpp` (with dynamically allocated bitmap) 0, 1ocen — 3ocen and 1 — 13

`prog/glos14.pas` (with a static bitmap) 0, 1ocen — 3ocen and 1 — 11.

`prog/glos5.cpp`, `prog/glos15.pas` 0, 1ocen — 3ocen, 1 — 10 and 13 — 14.

`prog/glos6.cpp`, `prog/glos16.pas` 0, 1ocen — 4ocen, 1 — 10 and 13 — 16.

`prog/glos7.cpp`, `prog/glos17.pas` 0, 1ocen — 4ocen, 1 — 15 and 17 — 19.

The incorrect solution `prog/glob1.cpp` does not pass any of the following tests: 2ocen, 1 — 11, 12b, 13b, 14b, 15b, 16b.

The incorrect solution `prog/glob2.cpp` does not pass any of the following tests: 5ocen, 20 — 22.

6 Limits

I have set the numbers range in the problem to 10^9 in order to make use of 64-bit integers necessary, but not obvious.

I have set the memory limit to 64 MB, because the model solution in Pascal `prog/glo2.pas` needs around 50 MB.

I have set the time limit to 4 seconds, which was 3.3 times more than the run time of the model solution on my computer (1.6 GHz).

7 Changes in the problem statement

- Generalization for $n \leq 20$ colours of gloves.
- Changes of limits.

8 Remarks

It is quite easy to devise and implement a correct brute-force solution for this problem. The optimal one is more difficult to find, but if found, it does not involve much trouble in implementation. On the whole, I think, that this would be rather an easy problem for international contestants.