Interactive, Output-only & Communication tasks

Percy Wong {percywtc}
Tasks Categorization

- Batch task
- Interactive task
- Output-only task
- Communication task (a.k.a. Two-steps task)
How Important?

- [IOI2016] Unscrambling a Messy Bug   [IOI2017] The Big Prize
- [IOI2011] Parrots

- [TFT2017] Constellation
- [TFT2014] Lost Sequence

- Interactive   - Output-only   - Two-steps
What's the difficulty?

- Unfamiliar style
- You may not be able to understand these problems during the contests, if you are the first time facing new types of tasks

- Feedback from inexperienced contestants after TFTs
  - 「唔知條題目講乜」
  - 「睇唔明題目」
Interactive task

- Your program will interact with the judging program
- You can consider it as: (suitable for most interactive tasks)
  - Your program asks some questions
  - The judging program answers your questions
  - Repeat the above until you can solve “something”
  - (Just like playing MASTERMIND / Guess the Number)
- Usually, there will be limits on number of questions asked
- Or, your score is determined by questions asked
M1431 Comparing Game

- N distinct cards not revealed to you
- Your goal: find where are the maximum and the minimum cards

- Question you may ask:
  - “Is card X larger than card Y?”

- Ask no more than ⌊1.5N⌋ questions

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| 3     | Q 1 2  | n = 3
| 0     |        | Is card 1 larger than card 2? No. Card 2 is larger.
| 1     | Q 3 1  | Is card 3 larger than card 1? Yes.
| 1     | Q 2 3  | Is card 2 larger than card 3? Yes.
| 1     | A 2 1  | Max card: 2, Min card: 1.
Interactive, Output-only & Communication tasks

M1431 Comparing Game

- How can our program ask question?
  - using standard I/O

Pascal version

```
 writeln('Q ', x, ' ', y);
 flush(output); // IMPORTANT
 readln(result);
```

C/C++ version

```
 printf("Q %d %d\n", x, y);
 fflush(stdout); // IMPORTANT
 scanf("%d", &result);
```
M1431 Comparing Game

Pascal version (sample partial solution)

```
for i := 1 to N do
  for j := 1 to N do
  begin
    counter := 0;
    if (i <> j) then
    begin
      writeln('Q ', i, ' ', j);
      flush(output);
      readln(result);
      if (result = 1) then
        counter := counter + 1;
    end;
  if (counter = N - 1) then
    bigIndex := i;
  if (counter = 0) then
    smallIndex := i;
  end;
```

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| 3     |        | $n = 3$
| 0     |        | Is card 1 larger than card 2? No. Card 2 is larger.
| 1     |        | Is card 3 larger than card 1? Yes.
| 1     |        | Is card 2 larger than card 3? Yes.
|       | 1 2    | Max card: 2, Min card: 1.
M1431 Comparing Game

C/C++ version (sample partial solution)

```c
for (int i = 1; i <= N; i++)
    for (int j = 1; j <= N; j++) {
        counter = 0;
        if (i != j) {
            printf("Q %d %d\n", i, j);
            fflush(stdout);
            scanf("%d", &result);
            if (result == 1)
                counter++;
        }
        if (counter == N - 1)
            bigIndex = i;
        if (counter == 0)
            smallIndex = i;
    }
```

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| 3     | Q 1 2  | n = 3
|       |        | Is card 1 larger than card 2? |
| 0     | Q 3 1  | No. Card 2 is larger. |
| 1     | Q 2 3  | Is card 3 larger than card 1? |
| 1     | A 2 1  | Yes. |

Max card: 2, Min card: 1.
Interactive task

- The example just now performs interaction through standard I/O
  - writeln / printf
  - readln / scanf
- Some interactive tasks are using another way
  - through the grader program
  - you will be given a template code
  - you will ask questions / get feedback by calling some given functions
I0501 Divisor Game

- An unknown integer $K$ within the range $[1, N]$
- Your goal: find the value of $K$

- Question you may ask:
  - “Is the number $K$ divisible by some integer $x$?”

- Ask minimal questions

Assume that the grader calls your function `play(1000)`.

<table>
<thead>
<tr>
<th>Call</th>
<th>Returns</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>isDivisibleBy(10)</code></td>
<td>1</td>
<td>$K$ is divisible by 10.</td>
</tr>
<tr>
<td><code>isDivisibleBy(100)</code></td>
<td>1</td>
<td>$K$ is divisible by 100.</td>
</tr>
<tr>
<td><code>isDivisibleBy(1000)</code></td>
<td>0</td>
<td>$K$ is not divisible by 1000.</td>
</tr>
<tr>
<td><code>isDivisibleBy(200)</code></td>
<td>0</td>
<td>$K$ is not divisible by 200.</td>
</tr>
<tr>
<td><code>isDivisibleBy(300)</code></td>
<td>0</td>
<td>$K$ is not divisible by 300.</td>
</tr>
<tr>
<td><code>isDivisibleBy(500)</code></td>
<td>0</td>
<td>$K$ is not divisible by 500.</td>
</tr>
<tr>
<td><code>isDivisibleBy(700)</code></td>
<td>0</td>
<td>$K$ is not divisible by 700.</td>
</tr>
</tbody>
</table>

Your function `play` should return 100, the number $K$ Alice has in mind.
I0501 Divisor Game

● What is given?

**TEMPLATE**

Download official grader files. Please note that you may need to make

<table>
<thead>
<tr>
<th>Pascal</th>
<th>C/C++</th>
</tr>
</thead>
</table>
| ```pascal```
| ```c++``` |

```pascal```
```
unit submission;

interface
function isDivisibleBy(M: longint): longint; cdecl; external;

var
  // TODO: global variables can be declared here

implementation
function play(N: longint): longint; cdecl; export;

var
  // TODO: implementation

end;
```

```c++```
```
#include "cplusplus"

extern "C" {
    #ifdef __cplusplus
        
        int isDivisibleBy(int M);
        int play(int N);
    #endif
}
```

```pascal```
```
```

```
```
I0501 Divisor Game

- How can our program ask question?
  - using grader functions

Pascal version

result := isDivisibleBy(x);

C/C++ version

result = isDivisibleBy(x);

Assume that the grader calls your function \( \text{play}(1000) \).

<table>
<thead>
<tr>
<th>Call</th>
<th>Returns</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>isDivisibleBy(10)</td>
<td>1</td>
<td>( K ) is divisible by 10.</td>
</tr>
<tr>
<td>isDivisibleBy(100)</td>
<td>1</td>
<td>( K ) is divisible by 100.</td>
</tr>
<tr>
<td>isDivisibleBy(1000)</td>
<td>0</td>
<td>( K ) is not divisible by 1000.</td>
</tr>
<tr>
<td>isDivisibleBy(200)</td>
<td>0</td>
<td>( K ) is not divisible by 200.</td>
</tr>
<tr>
<td>isDivisibleBy(300)</td>
<td>0</td>
<td>( K ) is not divisible by 300.</td>
</tr>
<tr>
<td>isDivisibleBy(500)</td>
<td>0</td>
<td>( K ) is not divisible by 500.</td>
</tr>
<tr>
<td>isDivisibleBy(700)</td>
<td>0</td>
<td>( K ) is not divisible by 700.</td>
</tr>
</tbody>
</table>

Your function \( \text{play} \) should return 100, the number \( K \) Alice has in mind.
I0501 Divisor Game

- You cannot compile the program even if you have completed `play()`
  - it’s because the main program is missing
- You cannot successfully test the program
  - it’s because the function `isDivisibleBy()` is not implemented
  - this function is implemented by the judging program
  - you are only required to implement `play()`

- So what can we do to test our program?
I0501 Divisor Game

- So what can we do to test our program?
  - we can implement the remaining functions to test our part
  - and remember to delete these lines before submit

- Being familiar with problems using grader IS VERY DIFFICULT !!!

- Please make use of the Online Judge, and try these tasks
Output-only task

- **Formal Definition**
  - Input files are given to you
  - You are not required to upload any source codes, just the output files

- **Actual meaning**
  - No need to worry about failing some unknown cases, all cases are revealed :D
  - No time limits / memory limits (actually there are... TL = 5hrs, ML = your machine)
  - You can even solve the cases manually :D :D :D
Output-only task

● Common stuffs?
  ○ not expecting optimal solutions (or not even exist)
  ○ some formulas to determine how good your outputs are (and how much you score)
  ○ good-enough solutions can get good-enough scores

● What you can do?
  ○ Usually there exists some small cases (can be manually solved)
  ○ You can write programs to analyze the cases / solve the cases
  ○ You can even solve cases separately with different approach and codes
T174 Constellation

- Given a set of $N$ points with integral coordinates lying on xy-plane
- Build a polyline consisting of $V$ points, connecting most points
**T174 Constellation**

- 10 cases in total (each 10pts)
- Scoring are based on number of points you connect
  - $10 \times 10^{(P - V) / (T - V)}$

<table>
<thead>
<tr>
<th>Case</th>
<th>Input</th>
<th>Output</th>
<th>$N$</th>
<th>$V$</th>
<th>$T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>stars1.txt</td>
<td>const1.txt</td>
<td>25</td>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>stars2.txt</td>
<td>const2.txt</td>
<td>49</td>
<td>13</td>
<td>49</td>
</tr>
<tr>
<td>3</td>
<td>stars3.txt</td>
<td>const3.txt</td>
<td>12</td>
<td>7</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>stars4.txt</td>
<td>const4.txt</td>
<td>80</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>stars5.txt</td>
<td>const5.txt</td>
<td>200</td>
<td>41</td>
<td>180</td>
</tr>
<tr>
<td>6</td>
<td>stars6.txt</td>
<td>const6.txt</td>
<td>90</td>
<td>20</td>
<td>90</td>
</tr>
<tr>
<td>7</td>
<td>stars7.txt</td>
<td>const7.txt</td>
<td>40</td>
<td>11</td>
<td>28</td>
</tr>
<tr>
<td>8</td>
<td>stars8.txt</td>
<td>const8.txt</td>
<td>120</td>
<td>25</td>
<td>115</td>
</tr>
<tr>
<td>9</td>
<td>stars9.txt</td>
<td>const9.txt</td>
<td>200</td>
<td>35</td>
<td>185</td>
</tr>
<tr>
<td>10</td>
<td>stars10.txt</td>
<td>const10.txt</td>
<td>200</td>
<td>50</td>
<td>200</td>
</tr>
</tbody>
</table>
I1711 Nowruz

- Given an $n \times m$ grid with some obstacle cells
- Build a maze that has as many 「堀頭路」 (dead end) as possible
  - 「堀頭路」 (dead end): cell that has exactly one free neighbour

```
. . . . #
. # . . #
. . . # .
. . . # .
```

```
. X . X #
. # . . #
. . . # X
X X . . #
```

```
1 X 2 X #
. # . 3 #
. . . # X
X X . 4 #
```
Here are some of the input cases (there are total 10 cases)

01.in
16 × 16
random

02.in
64 × 64
empty

03.in
64 × 64
random
Communication task (Two-steps task)

- You have to write two subprograms (or two modes)
- Judging flow:
  - [source input] → [program mode A] → [output A]
  - [input based on output A] → [program mode B] → [final output]
- Your score usually depends on the length / efficiency of [output A]

- Program mode A
  - build up a strategy that can transfer more information with shorter length
- Program mode B
  - build up a strategy to interpret the [output A] and extract some useful data
I1123 Parrots

- Original message $M$ consists of at most 64 integers within $[0, 255]$
Conclusion

● Just like constructive task, non-batch task is another type of problems
  ○ NOT LIMITED by any algorithms, topics
  ○ therefore, no standard rules to deal with them
  ○ again, “practice makes perfect”
  ○ as long as you solve / take a look at more non-batch tasks,
  ○ more techniques / experiences you can accumulate

● From the history of Team Formation Test,
  ○ non-batch tasks often appear :)
  ○ good luck :)

Interactive, Output-only & Communication tasks
Practice Tasks

- [IOI2010] Cluedo
- [IOI2010] Languages
- [IOI2010] Memory
- [IOI2013] Cave
- [IOI2014] Game
- [IOI2015] Scales
- [IOI2016] Unscrambling a Messy Bug
- [IOI2017] The Big Prize
- [IOI2010] Maze
- [IOI2012] Pebbling Odometer
- [IOI2017] Nowruz
- [IOI2011] Parrots
- [TFT2011] Stones Rearrangement
- [TFT2012] Debug!
- [TFT2013] The Forgotten Triangle
- [TFT2016] Model Answer II
- [TFT2017] Constellation
- [TFT2014] Lost Sequence

Interactive, Output-only & Communication tasks