



香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Constructive Algorithms & Special Tasks (I)

Daniel Hsieh {QwertyPi}

2026-02-28

Outline

1. Constructive Algorithms
2. Interactive Tasks
3. Minicomp on these two types of tasks

Constructive Algorithm

What is it?

Constructive Algorithms

Groups ▾

Tags ▾

Constructive Algorithms

15

★ ID	Name	# Solved
☆ I0212	Utopia Divided	13
☆ J144	Fair Santa Claus	90
☆ J151	Inverse Problem	277
☆ J161	Model Answer	129
☆ J172	Card Game	163
☆ J182	Rope	125
☆ J193	Hyper Knight II	75
☆ J213	Paint the Wall	26

☆ M1522	Gyeolhap	22
☆ M1532	Inverse Problem 10	36
☆ M1623	Bishop Puzzle	12
☆ S132	Safe Storage	51
☆ S134	Unfair Santa Claus	139
☆ S163	Arithmetic Sequence	109
☆ S213	Chinese Checkers	10

Constructive Problems?

Inverse Problem ☆ ✓

J151	Time Limit: 1.000 s	Memory Limit: 256 MB	▼
------	---------------------	----------------------	---

If there are more than one valid sets, you can output any of them.

Model Answer ☆ ✓

J161	Time Limit: 1.000 s	Memory Limit: 256 MB	▼
------	---------------------	----------------------	---

If there are multiple answers, you may output any one of them.

Card Game ☆ ✓

J172	Time Limit: 1.000 s	Memory Limit: 256 MB	▼
------	---------------------	----------------------	---

If there are several solutions, output any.

Rope ☆ ✓

J182	Time Limit: 1.000 s	Memory Limit: 256 MB	▼
------	---------------------	----------------------	---

Please help Alice to find any possible way to achieve so.

Bishop Puzzle ☆ ✓

M1623	Time Limit: 1.000 s	Memory Limit: 256 MB	▼
-------	---------------------	----------------------	---

If there are more than one solutions to the puzzle, output any.

Arithmetic Sequence ☆ ✓

S163	Time Limit: 1.000 s	Memory Limit: 256 MB	▼
------	---------------------	----------------------	---

If there are more than one arrangement, output any one of them.

Constructive Problems...

- Usually give some requirements / constraints to be fulfilled
- You should construct any arrangement that satisfies the given rules
 - Permutations
 - Sequences
 - Matrices
 - Placements
 - ...
- Usually interesting
- Often require more thinking than coding / knowledge of standard algorithms
- May have various correct solutions and “seemingly correct solutions”

An Example - CF1828A

Codeforces Round #873 (Div. 2) Problem A - Divisible Array

<https://codeforces.com/contest/1828/problem/A>

Given an integer N (≤ 200), find an integer array A of N elements such that:

- $1 \leq A[i] \leq 1000$
- $A[i]$ is divisible by i
- $A[1] + A[2] + \dots + A[N]$ is divisible by N

Sample Input N	Sample Output A
2	[2, 4]
3	[1, 2, 3]

An Example - CF1828A

Let's start with cases we know the answer immediately.

When N is odd, a trivial integer array is

$$[1, 2, 3, 4, \dots, N - 3, N - 2, N - 1, N]$$

Sum of $A[i]$ in this construction = $N(N + 1) / 2$, which is divisible by N as $(N + 1)$ is even.

Sample Input N	Sample Output A
2	[2, 4]
3	[1, 2, 3]

An Example - CF1828A

How about when N is even?

Sadly, the simplest construction $[1, 2, \dots, N - 1, N]$ does not work as the sum $= N(N + 1) / 2$ is not divisible by N when N even.

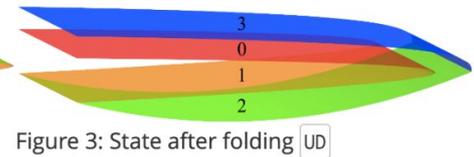
However, as observed from the sample, we can simply multiply every element by 2 to bring the sum up to $N(N + 1)$, which is divisible by N .

Sample Input N	Sample Output A
2	$[2, 4]$
3	$[1, 2, 3]$

Let's solve T222 together :)

The problem: given a paper strip consists of N ($\leq 10^{18}$) cells: $1, 2, \dots, N$, each step you can fold the paper strip into half. Construct a folding sequence of U and D such that two numbers A and B are **closest to each other** vertically.

Examples for $N = 4$, $A = 1$ and $B = 2$:



Answer = UD with minimum distance 1

T222 Wormhole - Simulation

We treat the strips as a two-dimensional arrays, and simulate the folding process

Then, we can make a 'simple' brute-force solution by trying all the possibilities

```
typedef vector<vector<int>> strip;
strip fold_U(strip a){
    assert(a[0].size() % 2 == 0);
    int h = a.size(), w = a[0].size() / 2;
    vector b(h * 2, vector<int>(w));
    for(int i = 0; i < h; i++){
        for(int j = 0; j < w; j++){
            b[i][j] = a[h - 1 - i][w * 2 - 1 - j];
            b[h + i][j] = a[i][j];
        }
    }
    return b;
}
// similar for type D fold
```

T222 Wormhole - Simulation

```
int min_dist(int N, int A, int B, string S){
    strip s; s.push_back(vector<int>(N));
    iota(s[0].begin(), s[0].end(), 0);
    for (auto c : S) {
        if (c == 'D') s = fold_D(s);
        else s = fold_U(s);
    }
    int pos_a, pos_b;
    for (int i = 0; i < s.size(); i++) {
        if (s[i][0] == A) pos_a = i;
        if (s[i][1] == B) pos_b = i;
    }
    return abs(pos_a - pos_b);
}
```

```
pair<int, string> brute_force(int N, int A, int B){
    int k = 0; while (N % 2 == 0) ++k, N /= 2;
    int min_d = 1LL << 60; string folding;
    for (int msk = 0; msk < (1 << k); msk++) {
        string S = "";
        for (int j = 0; j < k; j++) {
            S.push_back(msk & (1 << j) ? 'U' : 'D');
        }
        if (min_dist(N, A, B, S) < min_d) {
            min_d = min_dist(N, A, B, S);
            folding = S;
        }
    }
    return {min_d, folding};
}
```

T222 Wormhole - Checking your solution

You can write a simple checker by simply comparing the answer given by the brute force solution and your solution

You can also check whether or not the folding sequence indeed is valid for that minimum distance

```
void stress_test(){
    int N = 8;
    for (int A = 0; A < N; A++) {
        for (int B = 0; B < N; B++) {
            if (brute_force(N, A, B).first
                != wormhole(N, A, B).first) {
                cout << "WA\n" << N << ' ' << A << ' ' << B;
                break;
            }
        }
    }
}

void correct_fold(int N, int A, int B, int D, string S){
    assert(D == min_dist(N, A, B, S));
}
```

T222 Wormhole - Observation from small cases

We can output the answer of all the pairs (A, B) for a specific N (say 8)

If we try to look at the output, we can observe that it seems to follow some kind of pattern

Seems that we can cut it into 4 4 x 4 squares such that they are most likely the same
=> We can recursively solve the problem

You can get 80% by getting this correct :)

0	7	4	3	2	5	6	1
7	0	3	4	5	2	1	6
4	3	0	7	6	1	2	5
3	4	7	0	1	6	5	2
2	5	6	1	0	7	4	3
5	2	1	6	7	0	3	4
6	1	2	5	4	3	0	7
1	6	5	2	3	4	7	0

Binary vs Ternary

Given two binary strings A and B (Length of $A, B \leq 64$). Your target is to convert A into B using the following operations not more than **512** times:

- Choose a non-empty substring of $A[l, r] = A_l A_{l+1} \dots A_r$.
- Consider it as a ternary (base-3) number and convert it back to binary.
- Example: $(101)_3 = (1010)_2$, so you can transform **110110** to **1101010**.
- A and B starts with 1.
- If impossible, report it.

Binary vs Ternary

Very often it's hard to tackle the problem by solving small cases with programs...

- It's hard to code a general exhaustion program
- It's hard to observe patterns among all possibilities

We may try to work on special / small cases first and 'build' the solutions from them

Binary vs Ternary - Impossible Case

What if $A = 1$? Clearly we cannot change anything.

Therefore, the answer is YES if $B = 1$ also, and NO otherwise. Similar for $B = 1$.

So let's assume A and B both have at least two characters.

Binary vs Ternary - from 10 to anything

What if $A = 10$? Notice that

$$\underline{10} \rightarrow \underline{11} \rightarrow 100$$

So by applying the operation twice, we produce a zero at the end.

We can convert 10 to 11 with cost 1, and 10 to 100 with cost 2.

What if $A = 11$? Notice that

$$\underline{11} \rightarrow 1\underline{00} \rightarrow 10$$

So we can convert 11 to 10 with cost 2.

Binary vs Ternary - from 10 to anything

We now have three moves:

- Convert 10 to 11 with cost 1
- Convert 10 to 100 with cost 2
- Convert 11 to 10 with cost 2

Now, how can we solve the case when $A = 10$ and B is arbitrary?

Binary vs Ternary - from 10 to anything

We now have three moves:

- Convert 10 to 11 with cost 1
- Convert 10 to 100 with cost 2
- Convert 11 to 10 with cost 2

Now, how can we solve the case when $A = 10$ and B is arbitrary?

$$10 \rightarrow 100 \rightarrow 100..00 \rightarrow 110..00 \rightarrow 111..111 \rightarrow 110..0101$$

Binary vs Ternary - from anything to 10

So - how about converting *arbitrary* A to $B = 10$?

We now have *four* moves:

- Convert 10 to 11 with cost 1
- Convert 10 to 100 with cost 2
- Convert 11 to 10 with cost 2
- Convert 00 to 0 with cost 1

Binary vs Ternary - from anything to anything

So - how about converting *arbitrary* A to $B = 10$?

We now have *four* moves:

- Convert 10 to 11 with cost 1
- Convert 10 to 100 with cost 2
- Convert 11 to 10 with cost 2
- Convert 00 to 0 with cost 1

$110..0101 \rightarrow 111..11 \rightarrow 100..00 \rightarrow 10$

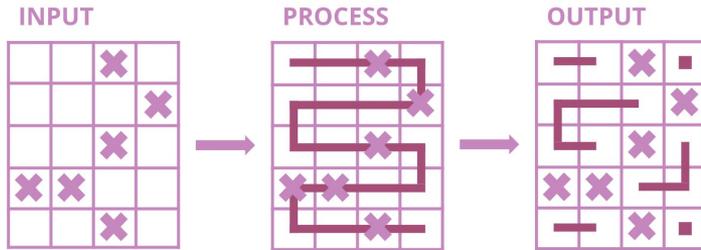
The problem is solved by simply combining two 'special' cases!

Techniques we learnt

- Generalise the problem from special case
- Observe patterns from small cases

J182 - Rope

Generalise the problem from special case



Task: <https://judge.hkoi.org/task/J182>
Tutorial: <https://assets.hkoi.org/training2018/J182.pdf>

S163 - Arithmetic Sequence

Observe patterns by brute-forcing small cases

8	9
1 5 3 2 7 6 4 8	1 9 5 3 2 7 6 4 8
1 5 3 2 7 6 8 4	1 9 5 3 2 7 6 8 4
1 5 3 7 2 6 4 8	1 9 5 3 7 2 6 4 8
1 5 3 7 2 6 8 4	1 9 5 3 7 2 6 8 4
1 5 3 7 4 2 8 6	1 9 5 3 7 4 2 8 6
1 5 3 7 4 8 2 6	1 9 5 3 7 4 8 2 6
1 5 3 7 4 8 6 2	1 9 5 3 7 4 8 6 2
1 5 3 7 6 2 4 8	1 9 5 3 7 6 2 4 8
1 5 3 7 6 2 8 4	1 9 5 3 7 6 2 8 4
1 5 3 7 6 8 2 4	1 9 5 3 7 6 8 2 4
...	...

Task: <https://judge.hkoi.org/task/S163>
Tutorial: <https://assets.hkoi.org/training2016/2016-sen3.pdf>

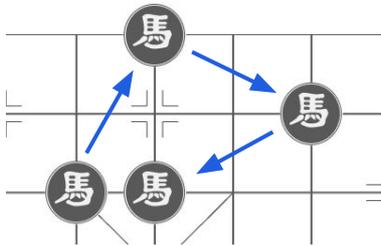
More Information: <https://assets.hkoi.org/training2023/cast-i.pdf>

Techniques we learnt

- Reducing the problem into smaller cases / lower dimensions
- Divide and Conquer

J193 - Hyper Knight II

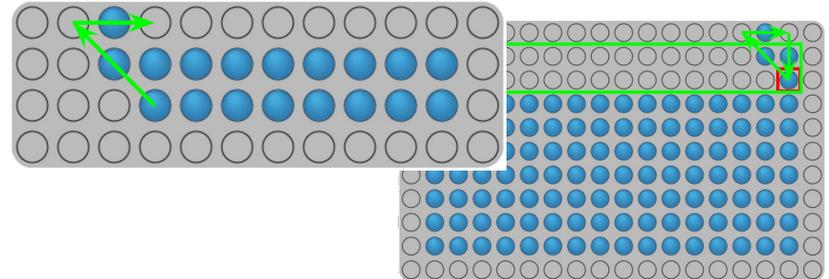
Repeating small cases of moving just a few steps



Task: <https://judge.hkoi.org/task/J193>
Tutorial: <https://assets.hkoi.org/training2019/J193.pdf>

S213 - Chinese Checkers

Reducing the problem to smaller problems



Task: <https://judge.hkoi.org/task/S213>
Tutorial: <https://assets.hkoi.org/training2021/S213.pdf>

Machine Building Tasks

- You are asked to generate a program in some artificial language that fulfills certain requirements
- You can only use certain set of instructions to build your program
- This type of constructive task has recently become more common
- Examples:
 - [\[IOI2012\] Pebbling Odometer](#)
 - [\[IOI2019\] Vision Program](#)
 - [\[IOI2021\] Bit Shift Registers](#)
 - [\[TFT2023\] Complete the Sequence](#)
 - [\[APIO2023\] Alice, Bob and Circuit](#)

Machine Building Tasks

Some tips:

- The set of instructions given may be quite unfamiliar
 - Try solving the problem using familiar operations
 - See if you can construct the operations you want (eg addition) with the instructions they give you (eg a bunch of logic gates)
 - Even if you cannot construct that operation, your “solution” might provide some insight to a solution that uses the given instructions
- My code looks so confusing I don't know what is going on anymore
 - It is very easy to get lost in your own code in these types of problems
 - A **very good idea** is to make sure that your code is well commented
 - Also make good use of things like functions and loops
 - Remember you are submitting a C++ file not a txt file

M23A1 The Great Wilderness Calculation

- Source: NOI 2016 曠野大計算
- Build a computer using some given operations, that is able to solve some computational tasks.

Name	Operator (type)	Operand	Result
Input Node	I	None	Reads a real number from the terminal as x_t
Output Node	O	i	$x_t = x_i$ and outputs x_i to the terminal
Addition Node	$+$	i, j	$x_t = x_i + x_j$
Offset Node	C	i, c	$x_t = x_i + c$
Invert Node	$-$	i	$x_t = -x_i$
Left-shift Node	$<$	i, k	$x_t = x_i \cdot 2^k$
Right-shift Node	$>$	i, k	$x_t = x_i \cdot 2^{-k}$
S-type Node	S	i	$x_t = s(x_i)$
Compare Node	P	i, j	$x_t = \begin{cases} -1 & x_i < x_j \\ 0 & x_i = x_j \\ 1 & x_i > x_j \end{cases}$
Max Node	M	i, j	$x_t = \begin{cases} x_i & x_i > x_j \\ x_j & x_i \leq x_j \end{cases}$
Multiply Node	$*$	i, j	$x_t = x_i \cdot x_j$

No.	Input	Input Constraints	Output
1	a, b	$ a , b \leq 10^9$ No more than 9 decimal places	$-2a - 2b$
2	a	$ a \leq 10^9$ No more than 9 decimal places	$\frac{1}{1 + e^{17a}}$
3	a	$ a \leq 10^9$ No more than 9 decimal places	$\begin{cases} -1 & a < 0 \\ 0 & a = 0 \\ 1 & a > 0 \end{cases}$
4	a	$ a \leq 10^9$ No more than 9 decimal places	$ a $, i.e. absolute value of a
5	a_1, \dots, a_{32}	$a_1, \dots, a_{32} \in \{0, 1\}$	Consider a_1, \dots, a_{32} as a binary integer from left to right, with the high bit on the left and the low bit on the right, and output the value of that integer
6	a	$0 \leq a < 2^{32}$ a is an integer	Outputs 32 integers, from high to low, as a binary representation of a (0's in high places if less than 32 bits)
7	a, b	$0 \leq a, b < 2^{32}$ a, b are integers	Result of bitwise xor of a, b
8	a	$ a \leq 10^9$ No more than 9 decimal places	$\frac{a}{10}$
9	a_1, \dots, a_{16}	$ a_1 , \dots, a_{16} \leq 10^9$ No more than 9 decimal places	Output 16 real numbers, representing the result of sorting a_1, \dots, a_{16} from smallest to largest
10	a, b, m	$0 \leq a, b < 2^{32}$ $1 \leq m < 2^{29}$ a, b, m are integers	Remainder of $a \cdot b$ divided by m

M23A1 The Great Wilderness Calculation

- This problem is very complicated so we won't cover it here
 - If you are interested, you can have a look at <https://assets.hkoi.org/training2023/cast-i.pdf>
- Main objective now isn't how to solve this problem - but how can we simplify the implementation?
- Every computation node depends on previous ones
- We need to know the IDs of the nodes of our interest
- *How?* How can we not do this manually?

Input	Output
1	I
	+ 1 1
	- 2
	I
	+ 4 4
	- 5
	+ 3 6
	- 7
	- 8
	0 9

M23A1 The Great Wilderness Calculation - Utility Functions

Instead of manually deducing the IDs, we can write utility functions to make our code looks tidier

The functions do the followings:

- Takes several (possibly none) node IDs as input
- Output the node information
- Return the ID of the newly created node

```
int called_cnt = 0;
int I(){
    cout << "I" << endl;
    return ++called_cnt;
}
int O(int i){
    cout << "O " << i << endl;
    return ++called_cnt;
}
... omitted ...
int MAX(int i, int j){
    cout << "M " << i << ' ' << j << endl;
    return ++called_cnt;
}
int MUL(int i, int j){
    cout << "*" << i << ' ' << j << endl;
    return ++called_cnt;
}
```

M23A1 The Great Wilderness Calculation - Utility Functions

Then, using those utility functions, we can simply use the result from previous function calls as input

Let's use the sample as sample:

1

Input	Output
1	I
	+ 1 1
	- 2
	I
	+ 4 4
	- 5
	+ 3 6
	- 7
	- 8
	0 9

```
void subtask_1(){
    int a = I();
    int c = INV(ADD(a, a)); // = -2a
    int b = I();
    int d = INV(ADD(b, b)); // = -2b
    int res = ADD(c, d); // = -2a - 2b
    O(INV(INV(res)));
}
```

M23A1 The Great Wilderness Calculation - Overloading Operators

Can we abuse the power of C++?

In C++, you can overload operators for self-defined struct - That means you can use operators like `+`, `*` for some data type you make

For example, we can declare a struct called `number`, treating it as a computational node

=> Our Target

```
void subtask_1(){
    number a, b; in >> a >> b;
    out << -((a + b) << 1);
}
```

M23A1 The Great Wilderness Calculation - Overloading Operators

In the struct `number`, we simply store the ID of that computation node

Then we can overload the operators in correspondence with the given functions

- for example, here we defined `!x` as `S(x)`, a function given, for simplicity

We can then do computations as normal like in C++!

```
struct number{
    int id;
    number operator+ (number y) {
        number z {ADD(id, y.id)};
        return z;
    }
    number operator- () {
        number z {INV(id)};
        return z;
    }
    number operator! () {
        number z {S(id)};
        return z;
    }
};
```

M23A1 The Great Wilderness Calculation - Overloading Operators

How about input and output? We cannot simply read from cin and write to cout

To resolve this, simply declare two more structs `input_device` and `output_device` and overload them also with `number`

```
struct input_device{
    input_device& operator>> (number& x) {
        x.id = I();
        return *this;
    }
} in;

struct output_device{
    output_device& operator<< (const number& x) {
        O(x.id);
        return *this;
    }
} out;
```

M23A1 The Great Wilderness Calculation - Overloading Operators

=> Final Product

Of course, it will already likely to be enough with utility functions

- You can strike a balance between time and simplicity
- However, always remember to think about *how* to implement before implementing something really complicated

```
void subtask_1(){  
    number a, b; in >> a >> b;  
    out << -((a + b) << 1);  
}
```

Building common operations from boolean operators

Addition of 2 binary numbers:

- Adding the digits (and the carry) starting from the least significant bit

Assume A, B, C, D are N-bit integers

- Input: A, B
- Carry: C
- Output: D (= A + B)

```
for (int i = 0; i < N; i++) {  
    bool b1 = A[i] && B[i];  
    bool b2 = C[i] && (A[i] || B[i]);  
    D[i] = b1 || b2;  
    C[i + 1] = A[i] ^ B[i] ^ C[i];  
}
```

Building common operations from boolean operators

A way to represent a negative N-bit integer: two's complement

- Flipping the sign of x : $-x = \sim x + 1$
- You can think of it as
$$2^N - x = ((2^N - 1) - x) + 1$$
- Note that most significant bit is 1 if and only if it is negative

Assume A, B are N-bit integers

- Input: A
- Output: $B (= -A)$

```
for (int i = 0; i < N; i++) {  
    A[i] = !A[i];  
}  
  
bool c = true;  
for (int i = 0; i < N; i++) {  
    B[i] = A[i] ^ c;  
    c = A[i] && c;  
}
```

Building common operations from bitwise operators

Subtraction:

- Addition but with the sign of the second number flipped

Assume A, B, C, D are N -bit integers

- Input: A, B
- Output: $C (= A - B)$

See if you can think of more things to do :)

```
D := 1
B := ~B
B := B + D
C := A + B
```

Some more tips...

- Use some random ideas and carefully analyze why are they incorrect
- Be careful on small / special cases
- Be aware of some special constraints set in the task (if any)
- Double-check the cases you solve manually / the exhaustion program
- Don't think too much :)

Some more problems...

- Codeforces
 - [Fraction](#)
 - [Lesha and array splitting](#)
 - [Dasha and Puzzle](#)
 - [Puzzling Language](#) (April Fools Contest!!!)
 - [Minimum Diameter Tree](#)
 - [Seating of Students](#)
 - [Construct a tree](#)
- AtCoder
 - [Four Coloring](#)
- LS-PC Programming Challenge
 - [Annoying Mathematics](#) (2016)
 - [Labyrinth](#) (2018)
 - [Monorail](#) (2016)
 - [Bob the Builder](#) (2018)
 - [Gravitational Tetris](#) (2017)
 - [Go](#) (2018)
 - [How to Get Rice](#) (2021)
 - [Lockout](#) (2021)
 - [Carpark](#) (2022)
 - [Delivery](#) (2022)
 - [Handful of Balls](#) (2023)
 - [Lift Problem](#) (2023)

During the break...

A Practice Problem on Machine Tasks :)

- Given a **32-bit** machine supporting operations (\ll , \gg , $|$, $\&$, \wedge , \sim):
- There is a mysterious 32-bit integer X
- You are only given the value of $Y = X \wedge (X \gg 1)$
 - $X = 5$ then $Y = 7$, $X = 11$ then $Y = 14$
- Create a sequence of instructions such that it can decode the value of X

- *Hint: It is always possible*

Interactive Tasks

How Important?

- [IOI2013] Cave [IOI2014] Game [IOI2015] Scales
 - [IOI2016] Unscrambling a Messy Bug [IOI2017] The Big Prize
 - [IOI2018] Highway Tolls [IOI2018] Combo
 - [IOI2019] Cycle* [IOI2019] Vision
 - [IOI2020] Routers* [IOI2020] Mushrooms
 - [IOI2022] Connected Towns* [IOI2022] Rarest Insects
-
- [NOI2019] I君的探險 [NOI2022] 樹上鄰域數點
-
- [TFT2012] Debug! [TFT2013] The Forgotten Triangle
 - [TFT2016] Model Answer II [TFT2018] Cave Exploration
 - [TFT2019] Liquid Layers [TFT2021] Re:Zero

*appeared in practice section

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」 *“Don't know what it's talking about”*
 - 「睇唔明題目」 *“Don't understand the task”*
 - 「唔識用 grader」 *“Don't know how to use grader”*

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 Wordle)
- Usually, there will be limits on number of questions asked
- Or, your score is determined by questions asked

A	R	I	S	E
R	O	U	T	E
R	U	L	E	S
R	E	B	U	S

Types of Interactive tasks

Interaction method

- Standard I/O
- Grader
 - Without sample grader
 - With sample grader

Interactor behavior

- Pre-defined cases
 - E.g., Wordle
- Adaptive interactor
 - **ABSURDLE** by [qntm](https://qntm.org)



S	T	E	A	M
B	O	U	G	H
W	I	P	E	R
R	I	V	E	R
L	I	N	E	R
C	I	D	E	R
F	I	X	E	R

<https://qntm.org/absurdle>

Using standard I/O: 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 $\lfloor 1.5N \rfloor$ questions

Input	Output	Explanation
3		$n = 3$
	Q 1 2	Is card 1 larger than card 2?
0		No. Card 2 is larger.
	Q 3 1	Is card 3 larger than card 1?
1		Yes.
	Q 2 3	Is card 2 larger than card 3?
1		Yes.
	A 2 1	Max card: 2, Min card: 1.

Using standard I/O: M1431 Comparing Game

How can our program asks questions?

Input	Output	Explanation
3		$n = 3$
	Q 1 2	Is card 1 larger than card 2?
0		No. Card 2 is larger.
	Q 3 1	Is card 3 larger than card 1?
1		Yes.
	Q 2 3	Is card 2 larger than card 3?
1		Yes.
	A 2 1	Max card: 2, Min card: 1.

```
cout << "Q " << i << " " << j << endl;
fflush(stdout); // IMPORTANT
cin >> result;
```

Actually `std::endl` will flush after putting endline character automatically.

Using standard I/O: M1431 Comparing Game

Recalling that...

- Ask no more than $\lfloor 1.5N \rfloor$ questions

We have asked $N(N-1)$ questions :(

Some hints to the full solution:

- We can use $0.5N$ questions to split the cards into two groups...
- For S numbers, $S-1$ comparison is sufficient to find the max/min number...

Grader

- The example just now performs interaction through standard I/O
 - `cout / printf`
 - `cin / 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

Grader: 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)`.

Call	Returns	Explanation
<code>isDivisibleBy(10)</code>	1	K is divisible by 10.
<code>isDivisibleBy(100)</code>	1	K is divisible by 100.
<code>isDivisibleBy(1000)</code>	0	K is not divisible by 1000.
<code>isDivisibleBy(200)</code>	0	K is not divisible by 200.
<code>isDivisibleBy(300)</code>	0	K is not divisible by 300.
<code>isDivisibleBy(500)</code>	0	K is not divisible by 500.
<code>isDivisibleBy(700)</code>	0	K is not divisible by 700.

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

Grader: I0501 Divisor Game

- What is given?

TEMPLATE

Download [official grader files](#). Please note that you may need to make changes for them to be usable.

```
Pascal C/C++
1 #ifdef __cplusplus
2 extern "C" {
3 #endif
4 int isDivisibleBy(int M);
5 int play(int N);
6 #ifdef __cplusplus
7 }
8 #endif
9
10 // TODO: global variables can be declared here
11
12 int play(int N) {
13     // TODO: implementation
14 }
15
```

Grader: I0501 Divisor Game

How can our program ask question?

- using grader functions

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

Call	Returns	Explanation
<code>isDivisibleBy(10)</code>	1	K is divisible by 10.
<code>isDivisibleBy(100)</code>	1	K is divisible by 100.
<code>isDivisibleBy(1000)</code>	0	K is not divisible by 1000.
<code>isDivisibleBy(200)</code>	0	K is not divisible by 200.
<code>isDivisibleBy(300)</code>	0	K is not divisible by 300.
<code>isDivisibleBy(500)</code>	0	K is not divisible by 500.
<code>isDivisibleBy(700)</code>	0	K is not divisible by 700.

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

```
result = isDivisibleBy(x);
```

Grader: I0501 Divisor Game

- You cannot compile the program even if you have completed `play()`
 - it's because the main program is missing
- You cannot 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?

Grader: I0501 Divisor Game

So what can we do to test our program?

- We can implement the remaining functions
 - `int isDivisibleBy(int M)`
 - `int main()`
 - put them inside the same source code file
- Delete these parts before submitting
- Or you can use
 - `#ifndef ONLINE_JUDGE`
 - `#endif`

```
const int MAX = 1'000'000;
int secret, trials;

int isDivisibleBy(int M) {
    trials++;
    return secret % M == 0;
}

int main() {
    srand(time(0));
    for (int t = 0; t < 10; t++) {
        secret = rand() % MAX + 1;
        trials = 0;
        int guess = play(MAX);
        cout << "secret = " << secret << endl;
        cout << "guess = " << guess << endl;
        cout << "trials = " << trials << endl;
        cout << endl;
    }
}
```

Grader: I0501 Divisor Game

- ~~Delete these parts before submitting~~
- You can use
 - `#ifndef some_flag_here`
 - `#endif`

Programming language specifications

<https://judge.hkoi.org/help>

Language	Compiler	Version	Compilation Flags	Execution Con
C	/usr/bin/gcc-4.9	4.9.4-2	<code>-DONLINE_JUDGE -s -O2 -o program.exe program.c -lm</code>	program.exe
C++	/usr/bin/g++-4.9	4.9.4-2	<code>-DONLINE_JUDGE -lm -s -O2 -o program.exe program.cpp</code>	program.exe
C++11	/usr/bin/g++-4.9	4.9.4-2	<code>-DONLINE_JUDGE -lm -s -O2 -o program.exe program.cpp</code>	program.exe

Compilation Commands

The grading system uses the following commands to compile the contestants' submissions.
system.

<https://ioi2018.jp/competition/competition-environment/>

C++

```
/usr/bin/g++ -DEVAL -std=gnu++14 -O2 -pipe -static -s -o task task.cpp
```

```
#ifndef ONLINE_JUDGE
const int MAX = 1000000;
int secret, trials;

int isDivisibleBy(int M) {
    trials++;
    return secret % M == 0;
}

int main() {
    srand(time(0));
    for (int t = 0; t < 10; t++) {
        secret = rand() % MAX + 1;
        trials = 0;
        int guess = play(MAX);
        cout << "secret = " << secret << endl;
        cout << "guess = " << guess << endl;
        cout << "trials = " << trials << endl;
        cout << endl;
    }
#endif
```

Sample grader: T193 Liquid Layers

Some problems (like [T193](#)) provide sample grader files for your testing

So you don't need to implement other functions by yourselves... hurray!!?

- Make sure that you know how to use them :(

SAMPLE GRADER

In order to test your program, you may [download the sample grader files](#). To use the sample the programming language, and follow the instructions below:

Language	Source Code Filename	Compilation Command	Execution Command
Pascal	experiment.pas	./compile_pas.sh	./experiment
C	experiment.c	./compile_c.sh	./experiment
C++11	experiment.cpp	./compile_cpp.sh	./experiment

```
$ tree # showing the unzipped folder
.
├── c
│   ├── compile_c.sh
│   ├── experiment.c
│   └── sample-grader.o
├── cpp
│   ├── compile_cpp.sh
│   ├── experiment.cpp
│   └── sample-grader.o
└── pas
    ├── compile_pas.sh
    ├── experiment.pas
    └── sample-grader.o

3 directories, 9 files

$ cd cpp # moving to the cpp/ directory

$ ./compile_cpp.sh # compile your C++ code

$ ls # see the compiled executable "experiment"
compile_cpp.sh experiment experiment.cpp sample-grader.o
```

Sample grader: T193 Liquid Layers

```
#ifndef __cplusplus
#include <stdbool.h>
#else
extern "C" {
#endif
    void pourLiquid(int index);
    int getReading();
    void answer(int order[]);

    void experiment(int N);
#ifdef __cplusplus
}
#endif

// TODO: global variables can be declared here

void experiment(int N) {
    // TODO: implementation
}
```



```
#include <bits/stdc++.h> // don't forget the headers
using namespace std;

#ifndef __cplusplus
// ...
#endif

int order[105]; // you can declare global variables

void experiment(int N) {
    iota(order, order + N, 1);

    sort(order, order + N, [](int u, int v) {
        pourLiquid(u);
        pourLiquid(v);
        return getReading() == 0;
    });

    answer(order);
}
```

Sample grader: T193 Liquid Layers

Read how to use the sample grader carefully

When testing your programs with the sample grader, your input should match the format and constraints from the task statement. Otherwise, unspecified behaviors may occur. The sample grader reads the input in the following format:

- line 1: N
- line 2: $order_0 \ order_1 \ \dots \ order_{N-1}$, where
 - the liquid with label $order_0$ has the highest density.
 - the liquid with label $order_1$ has the second highest density.
 - ...
 - the liquid with label $order_{N-1}$ has the lowest density.

The labels should be distinct integers between 1 to N (inclusive).

If your program correctly finds out the order of the density, the sample grader outputs `Correct`, followed by the value of C and your score on that case. Otherwise, it outputs `Wrong Answer`, followed by a message suggesting what might have been done incorrectly. The sample grader also prints the function calls in the standard error stream.

```
$ ./compile_cpp.sh && ./experiment
3
2 1 3
pourLiquid(2)
pourLiquid(1)
getReading()
                                returns 0
pourLiquid(3)
pourLiquid(2)
getReading()
                                returns 1
pourLiquid(3)
pourLiquid(1)
getReading()
                                returns 1
answer([2, 1, 3])
Correct.
C = 6, score = 100.000
```

Sample grader: T193 Liquid Layers

Learn how to use files and stream redirections to facilitate your testing

If your program correctly finds out the order of the density, the sample grader outputs `Correct`, followed by the value of C and your score on that case. Otherwise, it outputs `Wrong Answer`, followed by a message suggesting what might have been done incorrectly. The sample grader also prints the function calls in the standard error stream.

To read from file, you may use: `./experiment < input.txt`

To print the function calls to file, you may use: `./experiment 2> calls.txt`

To read from file and print the function calls to file, you may use:

`./experiment < input.txt 2> calls.txt`

<	stdin	cin, scanf
>	stdout	cout, printf
2>	stderr	cerr, fprintf(stderr,

```
$ cat input.txt # assume we've created this file
3
2 1 3

$ ./experiment < input.txt > output.txt 2> calls.txt

$ cat output.txt
Correct.
C = 6, score = 100.000

$ cat calls.txt
pourLiquid(2)
pourLiquid(1)
getReading()
                returns 0
pourLiquid(3)
pourLiquid(2)
getReading()
                returns 1
pourLiquid(3)
pourLiquid(1)
getReading()
                returns 1
answer([2, 1, 3])
```

Adaptive interactor

Some tasks use predefined cases to test your submission

- e.g. M1431, T182, T214
- this is not always the case

Some other tasks use “adaptive grader/interactor” to test your submission

- tries different strategies to exploit the weakness / worst case of your program
- randomized strategies likely won't work

*I0501 also uses adaptive grader to judge without specify explicitly

Adaptive interactor: T193 Liquid Layers

Most of the time, this information is not very helpful in solving the tasks

However, please be reminded that:

- not to expect random solutions for the adaptive grading tasks

SCORING

IMPORTANT: In some test cases the behavior of the grader is adaptive. This means that in these test cases the grader does not have a fixed order of density of the N liquids. Instead, the answers given by the grader may depend on the questions asked by your solution. It is guaranteed that the grader answers in such a way that after each answer there is at least one order of density consistent with all the answers given so far.

<https://judge.hkoi.org/task/T193>

T214 Re: Zero - Modifying Sample Grader

Some problems also provide *modifiable* sample grader

- The given grader file is .cpp instead of .o
- Not the case for T193

You can in fact change the sample grader code to print debug message

For example, if you want to know the query made by the program

T214 Re: Zero - Modifying Sample Grader

```
std::vector<long long> surrender(long long D) {
    queries++;
    assert(0 <= D && D <= 100000000000000LL);

    std::vector<long long> results(N);
    for (int i = 0; i < N; i++) {
        results[i] = Ps[i] + Px[i] * (D / Pt[i]);
    }

    return results;
}
```



```
#include <bits/stdc++.h>
using namespace std;
std::vector<long long> surrender(long long D) {
    queries++;
    cout << "Query " << queries << ": ";
    assert(0 <= D && D <= 100000000000000LL);

    std::vector<long long> results(N);
    for (int i = 0; i < N; i++) {
        results[i] = Ps[i] + Px[i] * (D / Pt[i]);
    }
    for (auto x : results)
        cout << x << ' ';
    cout << endl;
    return results;
}
```

T214 Re: Zero - Caching Query Result

It is a good idea to store down the information you have queried

You may self-define a query function so that you won't query the same thing twice

```
#include <bits/stdc++.h>
using namespace std;

map<long long, vector<long long>> cache;
vector<long long> self_surrender(long long D)
{
    if (cache.count(D))
        return cache[D];
    else
        return cache[D] = surrender(D);
}
```

T214 Re: Zero - Scoring Curve

SCORING

If in any of the test cases, the calls to the procedure `surrender` do not conform to the rules mentioned above, or the return value of `analyze_monsters` is incorrect, the score of your solution will be 0. Otherwise, let W be the maximum number of calls to the procedure `surrender` among all test cases. Then, the score will be calculated according to the following table:

Condition	Score
$130 < W$	0
$8 \leq W \leq 130$	$72 - 6.5 \cdot \sqrt{W - 7.5}$
$4 \leq W \leq 7$	$100 - 6.9 \cdot (W - 3)$
$W \leq 3$	100

Your score on this task is the lowest score you get among all test cases.



I1722 Simurgh - Extracting Information

This problem does not use a scoring curve

What can we infer from the subtasks?

- Subtask 1: $q \approx 6n!$
- Subtask 2: $q \approx 0.24n^3$ or $12n^2$
- Subtask 3: $q \approx n^2/2 \approx m$
- Subtask 4, 5: ???

Constant matters here - they tell you how many operations can be performed for each node (n) or edge (m)

(Graph Problem)

- $2 \leq n \leq 500$
- $n - 1 \leq m \leq n(n - 1)/2$

Subtasks

1. (13 points) $n \leq 7, q = 30\,000$
2. (17 points) $n \leq 50, q = 30\,000$
3. (21 points) $n \leq 240, q = 30\,000$
4. (19 points) $q = 12\,000$ and there is a road between every pair of cities
5. (30 points) $q = 8000$



<https://codeforces.com/blog/entry/88134?#comment-765758>

Some more...

- [Interactive Problems: Guide for Participants](#) from Codeforces

Practice problems:

- [01084 Celebrity](#) from HKOI Online Judge
- [I1021 Memory](#) available on HKOI Online Judge
- [T054 Guess](#) from HKOI Online Judge
- [T134 The Forgotten Triangle](#) from HKOI Online Judge
- [some other problems...](#) suggested by the Codeforces community
- [Codeforces tasks with "interactive" tag](#)

Minicomp

- Try to work on some special tasks!
 - Teams of 2, work collaboratively!
 - Duration: Around 1 hour (depends on lesson time)
- Create a Maze (50%)
 - Constructive Task
- Square or Rectangle? (50%)
 - Interactive Task
- Team with highest score will get prizes

Minicomp Solution Session

- Create a Maze
 - Source: Codeforces 715D, moderately(?) buffed
 - Constructive Task
- Square or Rectangle?
 - Source: Singapore NOI 2019 Preliminary P4
 - Interactive Task
- We will go over some interesting insights gain from the question, not aiming to explain the whole solutions in details.

M26A1

Create a Maze

The Problem

- Consider a $N * M$ grid. Neighbouring cells are connected by a door. Each door can either be locked or unlocked.
- Given an integer T in input, construct a grid such that there are exactly T ways to go from $(1, 1)$ to (N, M) using unlocked doors and walking rightwards / downwards only.
- The smaller the values of N , M , and K (number of locked doors), the better.
 $N, M \leq 39, K \leq 206$ to get Accepted

Motivation

- Find “building blocks” (in this problem, they are actual blocks)
- Figure out which blocks can duplicate quickly (exponentially)
- Fix / modify its structure so that it works for all input T

- The rest of the editorial to this problem will contain as few words as possible, since the more explanation, the more confusing it will be.

Subtask 1 (25%): T is power of 2

- Our building block is

1	1
1	2

- In general, if we have x , we can generate $2x$ using

x	x
x	$2x$

1	1	0	0	0	0	0	0
1	2	2	0	0	0	0	0
0	2	4	4	0	0	0	0
0	0	4	8	8	0	0	0
0	0	0	8	16	16	0	0
0	0	0	0	16	32	32	0
0	0	0	0	0	32	64	64
0	0	0	0	0	0	64	128

Subtask 2 (75%)

- Other building blocks?

1	1	1
1	2	3
1	3	6

- Perhaps base 6?
Given an x , we can generate $6x$. What about the remainders?

x	x	x
x	$2x$	$3x$
x	$3x$	$6x$

1	1	1	0	0	0	0	0
1	2	3	0	0	0	0	0
1	3	6	6	6	0	0	0
0	0	6	12	18	0	0	0
0	0	6	18	36	36	36	0
0	0	0	0	36	72	108	0
0	0	0	0	36	108	216	216
0	0	0	0	0	0	216	432

Subtask 2 (75%)

- So we can do base 6. Construction is via representing the input T using base 6, then mending the area around the block corresponding to the digit.
- More building blocks? Extend to 4×4 ?

1	1	1	1	0	0	0	0
1	2	3	4	0	0	0	0
1	3	6	10	0	0	0	0
1	4	10	20	20	20	20	0
0	0	6	20	40	60	80	0
0	0	0	20	60	120	200	0
0	0	0	20	80	200	400	400
0	0	0	0	0	0	400	800

Subtask 2 (75%)

- What's the problem with this 4 x 4 block?

1	1	1	1
1	2	3	4
1	3	6	10
1	4	10	20

- Look at the purple blocks on the edges: $\{1, 1, 4, 4, 10, 10\}$
- We cannot form a 3

1	1	1	1	0	0	0	0
1	2		4	0		0	0
1							0
1	4					20	0
0	0					80	0
0							0
0	0		20	80		400	400
0	0	0	0	0	0	400	800

Subtask 2 (75%)

- However we can fix it easily:

1	1	1	0
1	2	3	3
1	3	6	9
1	4	10	19

- The numbers on the edges are $\{1, 1, 3, 4, 9, 10\}$, so we can do base 19 now.

1	1	1	0	0	0	0	0
1	2	3	3	0	0	0	0
1	3	6	9	0	0	0	0
1	4	10	19	19	19	0	0
0	0	6	19	38	57	57	0
0	0	0	19	57	114	171	0
0	0	0	19	76	190	361	361
0	0	0	0	0	0	361	722

Subtask 2 (75%)

- Can we create even larger building blocks?
- Perhaps for $n = 5$? $n = 6$?
... and beyond?
- From our hands-on experience, if we denote S to be the multiset of numbers on the edges (the purple cells), then we want every nonnegative integer x smaller than the base that we are using to be representable by a subset sum of S

1	1	1	0	0	0	0	0
1	2	3	3	0	0	0	0
1	3	6	9	0	0	0	0
1	4	10	19	19	19	0	0
0	0	6	19	38	57	57	0
0	0	0	19	57	114	171	0
0	0	0	19	76	190	361	361
0	0	0	0	0	0	361	722

Subtask 2 (75%)

- A sufficient condition for a representable condition is that if we sort the elements of S in non-decreasing order, then $S_1 = 1$ and $S_1 + S_2 + \dots + S_{i-1} + 1 \geq S_i$ for every $i \leq |S|$.
- For each n , write a brute force (or DP for larger values of n) locally to find the largest possible base representable using a $n \times n$ block.

1	1	1	0	0	0	0	0
1	2	3	3	0	0	0	0
1	3	6	9	0	0	0	0
1	4	10	19	19	19	0	0
0	0	6	19	38	57	57	0
0	0	0	19	57	114	171	0
0	0	0	19	76	190	361	361
0	0	0	0	0	0	361	722

Subtask 2 (75%)

- The preparer's solution involves using these two fundamental blocks:

1	1	1	0	0	0
1	2	3	3	3	0
1	3	6	9	12	12
1	4	10	19	31	43
1	5	15	34	65	108
1	6	21	55	120	228

- With four $n = 9$ blocks and one $n = 6$ block, we can handle input up to $9994^4 \times 228 = 2.2 \times 10^{18}$.

1	1	1	0	0	0	0	0	0
1	2	3	3	3	0	0	0	0
1	3	6	9	12	12	0	0	0
1	4	10	19	31	43	43	0	0
1	5	15	34	65	108	151	151	0
1	6	21	55	120	228	379	530	530
0	0	21	76	196	424	803	1333	1863
0	0	0	76	272	696	1499	2832	4695
0	0	0	0	272	968	2467	5299	9994

M26A2

Square or Rectangle?

The Problem

- Given a square / rectangle shape on a $N * N$ grid (where $N = 100$), identify whether it is a square or a rectangle.
- You can ask `inside_shape(X, Y)` which tells you whether the cell (x, y) is in the shape or not.
- Partial scoring available, ≤ 33 queries to get full score.

Solution 1

- For every (i, j) where $1 \leq i, j \leq N$, ask `inside_shape(i, j)` to know whether it is part of the shape or not.
- So you can recover the whole grid in N^2 queries and determine easily.
- Expected score: $0 + 14 = 14$

Solution 2

- Once you have found any cell (a, b) which is inside the shape, you can then find each of the four boundaries of the shape (e.g. using binary search).
- Query random cells on the grid until you have found such a cell.
- This uses $(\text{random}) + 4 \log N$ queries.
- Expected score: $18 + 14 = 32$

Solution 2.1

- **Observation 1:** If the shape is a square, then it has side length at least 20. (because the area must be at least $0.04 * N^2 = 400$)
- Instead of querying random cells (a, b) , we now query the cells where a and b are both multiples of 20. There are $5^2 = 25$ such cells.
- If none of those 25 cells are inside the shape, then the shape must be a rectangle. Otherwise, do our previous solution.
- This takes $25 + 4 * \log N$ queries in worst case.
- Expected score: $[0, 18] + [33, 54] = [33, 72]$ depending on implementation

Solution 2.2

- **Observation 2:** Once we have found out where three of the four boundaries (up / down / left / right) are, we can simply use two queries to find out whether the remaining boundary coincides with the boundary if the shape had been a square or not.
- If we know (after three binary searches) the values of U, D, and L, we can just ask `inside_shape(U, L + (D - U))` and `inside_shape(U, L + (D - U + 1))`, and there's no need to binary search for R.
- This takes $25 + 3 * \log N + 2$ queries in worst case.
- Expected score: $[0, 18] + [53, 66] = [53, 84]$

Solution 3

How to further reduce the number of queries used?

- Our previous solution consists of two parts
 1. Sampling cells (a, b) where a and b are multiples of 20
 2. Binary search for the boundaries starting at (a, b)
- Which part looks like it has more potential to be optimised?

Solution 3

- In the first part (sampling multiples of 20), you can sense that the cells on the edge, namely those of the form $(100, b)$ and $(a, 100)$, “give you less information” than those in the middle (with coordinates $20 / 40 / 60 / 80$).
- With this in mind, what is the most intuitive thing to try next?
- Maybe only query the 4^2 cells $(20 / 40 / 60 / 80, 20 / 40 / 60 / 80)$?
- But this means we might have “missed” a square on the L-shaped boundary in the cells with coordinates 81 to 100...

Solution 3.1

Nevertheless, let's continue exploring using this idea:

- **Observation 3:** if we “missed” a square using the algorithm (so the square is entirely within the L-shaped boundary), then the side length of the square is exactly 20.
- If the shape is not “missed”, good, just use our previous algorithm.
- If the shape is “missed” by our initial sweep, we ask the 9 points that we originally omitted asking. But now thanks to **Observation 3**, it suffices to query just one boundary (instead of three).
- This uses $\max(16 + 3 * \log N + 2, 25 + \log N + 2)$ queries.
- Expected Score: $18 + [69, 82] = [87, 100]$

Solution 3.2

Good, now we have successfully optimised the first part. What next?

- Obviously, the most natural thing to do is try optimising the second part.
- **Observation 4:** after querying all 16 points in the first part, we are able to bound the search space boundaries of the shape to a interval of length 20 determined by the result from the first part.

This means it is no longer necessary for our binary searches in the second part to search over the whole interval $[1..N]$.

- So all the $\log N$ before can be replaced by $\log 20 = 5$.

So this uses $\max(16 + 3 * 5 + 2, 25 + 5 + 2) = 33$ queries in the worst case, just enough to get Accepted.

Takeaways from this Minicomp

- If a problem has **partial scoring**, it usually (not always!) means that there are numerous different ways to obtain partial progress.
⇒ you will (normally) be awarded correspondingly depending on how much progress you have made.
- Generate as many brief ideas as possible. Then, think about how these brief ideas can be turned into “**building blocks**” of a potential solution.
Don't get stuck with one approach! We used 4 observations in total just now, none of which is difficult. However, to get Accepted, you have to gather all four of them altogether.

Takeaways from this Minicomp

- It is especially important for constructive and interactive tasks to try to come up with **several brief ideas**, ideally from **different perspectives**.
- Even though there is a easy subtask that will be “overridden” by a subsequent subtask, think of the “intended solution” of the easy subtask first — that might be a crucial building block that opens up more versatile optimisations later. Do not ignore a subtask because you “thought” it is useless!

Takeaways

- Don't be afraid to attempt non-batch tasks
 - Sometimes, a simple solution can get a very high score
 - Usually, non-batch tasks don't require lots of standard tricks
- Generate ideas by analyzing special cases or small cases
 - For example, for tasks involving graphs, think of complete graph, chain, star, etc.
 - List a bunch of small cases and try to solve them manually, maybe this will reveal some insight
- Binary search is a good tool