ICPC SouthWestern Europe Regional Contest 2019–2020

Paris, 26 January 2020

Judges and Problem Setters

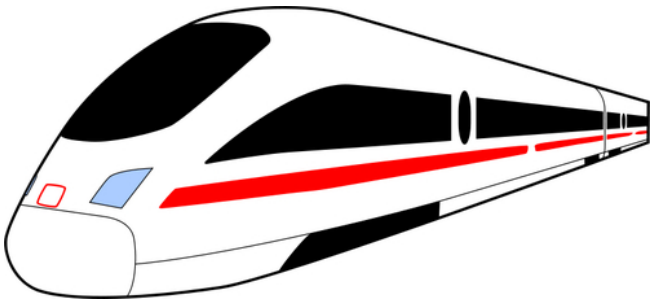
- Mehdi Bouaziz (Nomadic Labs)
- Thomas Deniau (Apple)
- Jean-Christophe Filliâtre (CNRS, LRI)
- Louis Jachiet (Télécom Paris), *deputy chief judge*
- Jacques-Henri Jourdan (CNRS, LRI)
- Vincent Jugé (Université Gustave Eiffel)
- Silviu Maniu (Université Paris-Saclay)
- Raphaël Marinier (Google)
- Carine Pivoteau (Université Gustave Eiffel)
- Pierre Senellart (École normale supérieure), *chief judge*
- Caterina Urban (Inria)

This problem set consists of 12 problems, on 28 pages.

Page intentionally left blank

A: Environment-Friendly Travel

Time limit: 3 seconds



Taking holidays unfortunately has a carbon footprint. The CO_2-cost of a kilometer traveled depends on the mode of transportation. For instance, train travel is less costly than bus travel, which is itself less costly than car travel.

Your objective is to plan a holiday trip from your house, given by 2D coordinates (x_s, y_s), to your travel destination, given by (x_d, y_d). Being aware of the environmental impact of travel, you want to minimize the CO_2-cost of your holiday, but you still want to keep the total number of kilometers traveled within a given budget B.

To aid you in planning, you have access to a map of N stations, possibly linked by T different modes of transportation (airplane, train, etc.) numbered $1, \ldots, T$. Each mode has a CO_2-cost per distance unit $C_1, \ldots, C_T$. You can travel by car from your home to the destination, from your home to any station, and from any station to the destination point, at a cost C_0 per distance unit. C_0 is always greater than any of $C_1, \ldots, C_T$.

Each of the N stations has coordinates (x_i, y_i) for $i = 0, \ldots, N-1$. Each station may be connected to some other stations via one or several of the T modes. Each connection works both ways, so only one direction has to be listed. There can be multiple modes of transportation available between two stations. You can only travel between two stations via their connections using the available transportation modes (car travel is not allowed between stations).

The distance between two points a and b is the 2D distance between (x_a, y_a) and (x_b, y_b), rounded to the nearest integer above:

$$\text{dist}(a, b) = \left\lceil \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2} \right\rceil,$$

and the CO_2-cost of travel between a and b, using transport mode i is:

$$\text{cost}(a, b, i) = C_i \times \text{dist}(a, b).$$

Given two source–destination coordinates, a budget B expressed in distance units, a list of transportation modes and their respective CO_2-costs, and the station network, your task is to compute the minimal CO_2-cost possible while traveling at most B kilometers.

Input

The input consists of the following lines:

- Line 1 contains two space-separated integers, x_s and y_s, the coordinates of your house.
- Line 2 contains two space-separated integers, x_d and y_d, the coordinates of your destination.
- Line 3 contains an integer, B, the maximal distance (in kilometers) that you accept to travel.
- Line 4 contains an integer, C_0, the CO_2-cost per distance unit of traveling by car.

- Line 5 contains an integer, T, the number of modes of transportation (beside cars).
- Line $i+5$, for $1 \leqslant i \leqslant T$, contains the integer C_i, the CO_2-cost per distance unit of transportation mode i.
- Line $T+6$ contains the integer N, the number of stations.
- Line $i+T+7$ describes station i ($0 \leqslant i < N$) and contains $3+2 \times l_i$ space-separated integers. The first three integers are x_i, y_i and l_i. The pair (x_i, y_i) represents the coordinates of station i, while l_i is the number of links between station i and other stations. The l_i remaining pairs of integers (j, m_j) each describe a link to station j ($0 \leqslant j < N$) using transportation mode m_j ($1 \leqslant m_j \leqslant T$). All links are bidirectional.

Output

The output should contain a single line with a single integer representing the minimal feasible CO_2-cost or -1 if no feasible cost is found within the kilometer budget.

Limits

All inputs are integers. All coordinates are in $[0, 100] \times [0, 100]$.

- $1 \leqslant T \leqslant 100$;
- $1 \leqslant N \leqslant 1\,000$;
- $0 \leqslant B \leqslant 100$;
- $1 \leqslant C_1, \ldots, C_T < C_0 \leqslant 100$;
- for each station, there are at most 100 links to other stations.

Sample Input

```
1 1
10 2
12
100
2
10
50
3
2 3 2 1 1 2 2
5 5 1 2 1
9 3 0
```

Sample Output

```
850
```

Sample Explanation

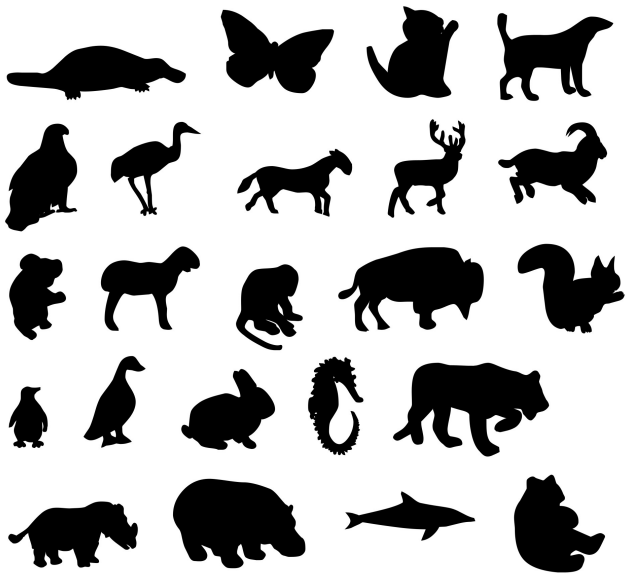
The results corresponds to the CO_2-cost of the following route:

1. from home at (1,1) to station 0 at (2,3) by car for a cost of $100 \times \lceil \sqrt{1^2+2^2} \rceil = 300$,
2. to station 2 at (9,3) via transportation mode 2 for a cost of $50 \times \lceil \sqrt{7^2+0^2} \rceil = 350$,
3. to the destination at (10,2) by car for a cost of $100 \times \lceil \sqrt{1^2+1^2} \rceil = 200$ for a total of 850.

This route is valid because the total distance traveled is $3+7+2 = 12$, within the allowed budget of 12.

B: Biodiversity

Time limit: 3 seconds



Alicia has a huge garden which is the habitat of many animals that she really cares about. After listening to a podcast about biodiversity, she becomes very concerned about the balance between species in her garden. She wants to know if there is a species that could overtake the others. In order to do so, she decides to carry out a census of all animals in the garden, writing down the species of each of them. Can you help her checking if there is strictly more animals from one species than the animals from all others species together?

Input

The input consists of the following lines:

- on the first line: an integer N;
- on each of the next N lines: the species of an animal as a string of length at most 20, containing only ASCII alphanumeric characters.

Limits

- $1 \leqslant N \leqslant 2 \times 10^5$.

Output

A string that appears a number of times that is greater than the sum of the others, if there is any, or the string "NONE" otherwise.

Sample Input 1

```
3
frog
fish
frog
```

Sample Output 1

```
frog
```

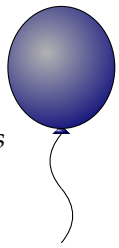
Sample Input 2

```
4
cat
mouse
mouse
cat
```

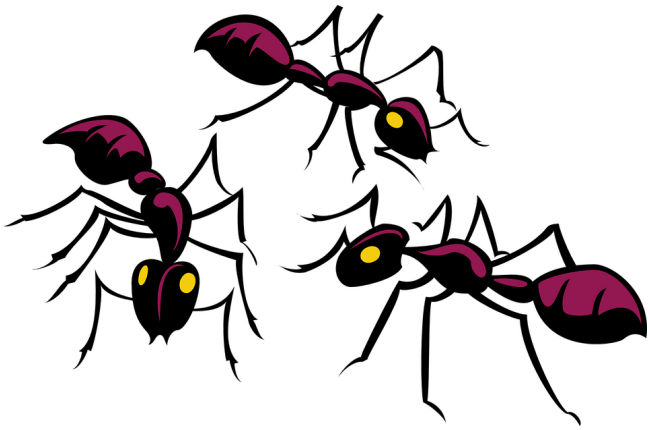
Sample Output 2

```
NONE
```

C: Ants



Time limit: 2 seconds



Charles is fascinated by ants. In order to observe a colony of ants over a long period, Charles managed to build a program that uniquely identifies each ant, using image recognition. (Yes, every ant is unique.) Inside the program, each ant is tagged with a unique nonnegative integer. Whenever there is a birth in the colony, the new ant is given a new tag, different from all tags already assigned. Whenever some ant disappears, its tag falls back into the pool of available tags.

Charles's program works as follows. It first scans the whole colony, building the list of tags of the ants that are recognized. Then it assigns fresh tags to the new ants. To do so, the program simply picks the first natural number (i.e., nonnegative integer) that is not currently assigned to any ant, and so on.

Due to some glitches in the image recognition device and in the program, there are sometimes negative or very large numbers that appear in the input list. These are simply ignored by Charles's program.

Your job is to reimplement the part of Charles's program that finds a fresh tag to assign to a new ant.

Input

The input consists of the following lines:

- on the first line: an integer N;
- on the next N lines: some integers $X_1, \ldots, X_N$, one per line.

Limits

The input satisfies $0 \leqslant N \leqslant 10^6$. Each integer X_i has less than 100 digits.

Output

The smallest natural number that does not belong to the set $\{X_1, \ldots, X_N\}$.

Sample Input

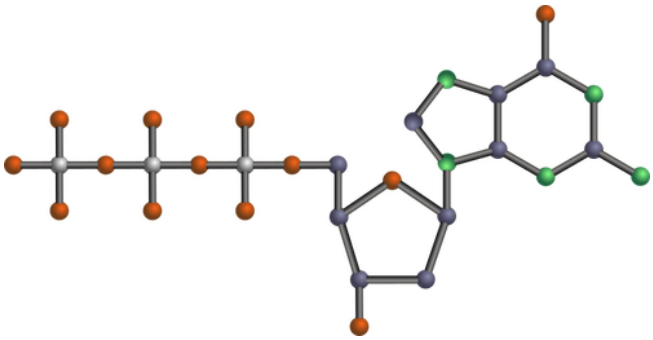
```
5
1
-1
0
3
10
```

Sample Output

```
2
```

D: Gnalcats

Time limit: 0.3 second



Researchers have discovered a new form of life they have named Gnalcats. Gnalcats have a very strange form of DNA and proteins but the researchers have understood how they work. They are now trying to classify species of Gnalcats by comparing their DNA.

A gene of their DNA is a sequence of bases. These genes operate on proteins, which are extremely long chains of amino-acids ($a - b - c - \ldots$). Amino-acids are either simple or complex (composed of two other amino-acids). Proteins always contain several billions of amino-acids.

Genes operate on proteins in the following way: the seven different bases (**C**, **D**, **L**, **P**, **R**, **S**, **U**) correspond to different transformations on the protein. The result of the operation of a gene on a protein is obtained as the combination of the individual transformations by each base of the gene: the first base of the gene transforms the input protein, the resulting protein is then transformed according to the second base of the gene, and so on. Life is not perfect and thus one of these transformations may fail, in which case the overall transformation fails. If, at any point in the transformation, the protein is reduced to a chain of three or fewer amino-acids (simple or complex) then the transformation fails.

The effect of each base is described in the following table where X denotes the tail of the protein, while a, b, and c are amino-acids (either simple or complex):

base	input protein	output protein
C	$a - X$	$a - a - X$
D	$a - X$	X
L	$a - X$	$\begin{cases} b - X & \text{if } a \text{ is the complex amino-acid } \langle b, c \rangle \\ \text{fail} & \text{if } a \text{ is a simple amino-acid} \end{cases}$
P	$a - b - X$	$c - X$ where c is the complex amino-acid $\langle a, b \rangle$
R	$a - X$	$\begin{cases} c - X & \text{if } a \text{ is the complex amino-acid } \langle b, c \rangle \\ \text{fail} & \text{if } a \text{ is a simple amino-acid} \end{cases}$
S	$a - b - X$	$b - a - X$
U	$a - X$	$\begin{cases} b - c - X & \text{if } a \text{ is the complex amino-acid } \langle b, c \rangle \\ \text{fail} & \text{if } a \text{ is a simple amino-acid} \end{cases}$

For example, the gene **PSDSPCRPSDUL** transforms a protein like this:

- the input protein is $a - b - c - d - e - f - \dots$
- then applying the rule for the first **P** produces: $\langle a, b\rangle - c - d - e - f - \dots$
- then applying the rule for **S** produces: $c - \langle a, b\rangle - d - e - f - \dots$
- then **D** gives: $\langle a, b\rangle - d - e - f - \dots$
- then **S** gives: $d - \langle a, b\rangle - e - f - \dots$
- then **P** gives: $\langle d, \langle a, b\rangle\rangle - e - f - \dots$
- then **C** gives: $\langle d, \langle a, b\rangle\rangle - \langle d, \langle a, b\rangle\rangle - e - f - \dots$
- then **R** gives: $\langle a, b\rangle - \langle d, \langle a, b\rangle\rangle - e - f - \dots$
- then **P** gives: $\langle\langle a, b\rangle, \langle d, \langle a, b\rangle\rangle\rangle - e - f - \dots$
- then **S** gives: $e - \langle\langle a, b\rangle, \langle d, \langle a, b\rangle\rangle\rangle - f - \dots$
- then **D** gives: $\langle\langle a, b\rangle, \langle d, \langle a, b\rangle\rangle\rangle - f - \dots$
- then **U** gives: $\langle a, b\rangle - \langle d, \langle a, b\rangle\rangle - f - \dots$
- and finally **L** gives: $a - \langle d, \langle a, b\rangle\rangle - f - \dots$

You are given two genes, and you must decide whether they are equivalent. Two genes are equivalent if for every input protein composed of **at least one billion of simple amino-acids**, either the application of both genes produces the same output protein, or both applications fail.

Input

The input consists of two lines, each representing a Gnalcats gene.

Limits

Each gene contains at least one base. The sum of the length of input genes is not greater than 10^4.

Output

The output consists of a single word: "True" if the genes are equivalent, "False" otherwise.

Sample Input 1

```
PU
SS
```

Sample Output 1

```
True
```

Sample Explanation 1

These genes do nothing: they always transforms a protein into the exact same protein.

Sample Input 2

```
L
R
```

Sample Output 2

```
True
```

Sample Explanation 2

These genes always fail because both `L` and `R` bases fail on simple amino acids.

Sample Input 3

```
PSPCRSL
PS
```

Sample Output 3

```
True
```

Sample Input 4

```
U
C
```

Sample Output 4

```
False
```

Sample Input 5

```
PL
PR
```

Sample Output 5

```
False
```

Page intentionally left blank

E: Pixels

Time limit: 3 seconds

There are various ways to protect wildlife. One is to make posters that will inspire people, and that is the job of artists. Some of those people that need to be inspired are geeks, and that is why some artists focus on pixel art.

An artist prepared a pixel art drawing for depicting animals in black and white. These drawings should be displayed on a gigantic pixelated, rectangular screen, with K rows (ordered from 0 to $K-1$, from top to bottom) and L columns (ordered from 0 to $L-1$, from left to right): displaying drawings on such screens is your job.

When you initialize the screen, it displays only white pixels. Then, its pixels should be controlled individually by switches, and pressing the switch of a pixel P ought to change the color of P, either from black to white or from white to black. However, the screen has a defect: pressing the switch of the pixel P also changes the color of those pixels that have an edge in common with P, i.e., those pixels that are adjacent horizontally or vertically (but not diagonally). Yet, it might still be possible that, by adequately choosing which switches you press, even this defective screen can display the drawing you were handed.

Can you manage to display that drawing? If yes, which set of switches should you press?

Note

Note that the order in which switches are pressed is not important, and that it is never necessary to press twice the same switch. Thus, a solution can be fully described by the set of switches pressed. There might be several solutions, i.e., several adequate sets of switches. In that case, your output should just represent one such solution.

Input

- Line 1 contains two integers K and L, separated by spaces.
- For all i such that $0 \leqslant i < K$, line $i+2$ contains L symbols "B" or "W", separated by spaces: if the j^{th} symbol (with $0 \leqslant j < L$) is a "B", then the pixel that will be displayed on row i and column j should be black; otherwise, it should be a white pixel.

Limits

- $1 \leqslant K \times L \leqslant 100\,000$.

Output

If there is no way to display the desired drawing by pressing switches, your output should contain one line with the word "IMPOSSIBLE" and nothing else.

Otherwise, your output should contain K lines, each one containing L symbols "A" or "P" and separated by spaces:

- For all i and j such that $0 \leqslant i < K$ and $0 \leqslant j < L$, if the j^{th} symbol on line $i+1$ is a "P", then you should press the switch associated with the pixel on row i and column j; otherwise, you should avoid that switch.

Sample Input 1

```
1 2
B W
```

Sample Output 1

```
IMPOSSIBLE
```

Sample Explanation 1

Whenever you press a switch, the colors of both pixels changes. Hence, they always have the same color, and the drawing cannot be displayed.

Sample Input 2

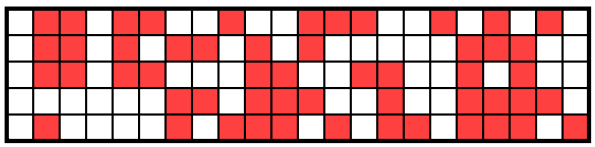
```
5 22
B B B W B W W W B W B B B W B B B B W B B B
B W W W B W W W B W B W W W B W W B W B W W
B B B W B W B W B W B B B W B B B B W B W W
W W B W B W B W B W B W W W B W B W W B W W
B B B W W B W B W W B B B W B W W B W B B B
```

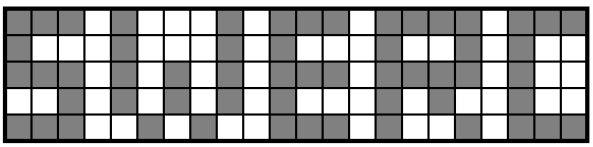
Sample Output 2

```
A P P A P P A A P A A P P P A A P A P A P A
A P P A P A P P A P A P A A A A A P P P A A
A P P A P P A A A P P A A P P A A P A P A A
A A A A A A P P A P P P A A P A A P P P P A
A P A A A A P A P P P A P A P P A P P P A P
```

Sample Explanation 2

You can indeed check that, if you press the switches associated to the red pixels of the left picture, you will obtain the picture depicted on the right.





F: Icebergs

Time limit: 3 seconds

CC BY-SA 3.0 by Serge Ouachée

Tania is a marine biologist. Her goal is to measure the impact of climate change on the population of Macaroni penguins. As most species of penguins, Macaroni penguins live in the southern hemisphere, near Antarctica. Tania is primarily focused on the population of Macaroni penguins near the "Îles Nuageuses" (in English, "Cloudy Islands").

During summer, the ice around the islands melt and the islands becomes too small to host all the birds. Some penguins live on the icebergs floating around. For her study, Tania needs to measure the area of those icebergs.

Using satellite imagery and image recognition, Tania has obtained a map of the icebergs and your goal is to measure their area. The island studied by Tania is quite small and the Earth can locally be approximated as a flat surface. Tania's map thus uses the usual 2D Cartesian coordinate system, and areas are computed in the usual manner. For instance, a rectangle parallel to the axes defined by the equations $x_1 \leqslant x \leqslant x_2$ and $y_1 \leqslant y \leqslant y_2$ has an area of $(x_2 - x_1) \times (y_2 - y_1)$.

In Tania's representation, an iceberg is a polygon represented by its boundary. For each iceberg Tania has noted the sequence of points $p_1, \ldots, p_k$ defining the border of the iceberg. The various icebergs never touch each other and they never overlap. Furthermore the boundary $p_1, \ldots, p_k$ of an iceberg is always a "simple" polygon, i.e. no two segments in $[p_1; p_2], \ldots, [p_k; p_1]$ cross each other.

Input

The input consists of the following lines:

- on the first line, an integer N, describing the number of polygons;
- then N blocks of lines follow, each describing a polygon and composed of:
  - on the first line, an integer P, the number of points defining the polygon border,
  - on the next P lines, two space-separated integers x and y, the coordinates of each border point.

Limits

- The number N of polygons is such that $1 \leqslant N \leqslant 1\,000$.
- Each polygon is described by P points with $3 \leqslant P \leqslant 50$.
- All coordinates are such that $0 \leqslant x, y \leqslant 10^6$.

Output

The output should contain a single integer: the total area rounded to the nearest integer below. In other words, the output should be a single line containing a single integer I such that the total area A of the polygons described in the input is comprised between I included and $I+1$ excluded ($I \leqslant A < I+1$).

Sample Input 1

```
1
4
0 0
1 0
1 1
0 1
```

Sample Output 1

```
1
```

Sample Explanation 1

This sample has a unique iceberg, which is a square of side 1.

Sample Input 2

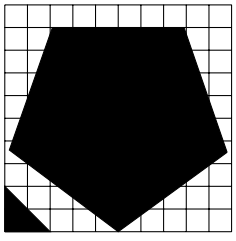
```
2
5
98 35
79 90
21 90
2 36
50 0
3
0 0
20 0
0 20
```

Sample Output 2

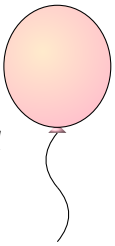
```
6100
```

Sample Explanation 2

In this sample (depicted on the right) there are two icebergs, a triangle of area 200 and a pentagon of area 5900.5.



G: Swapping Places



Time limit: 1 second

CC BY 2.0 by zadam@flickr

Animals are waiting in a line, in a quarantine zone, before they can enter a hunting-free area, where they will find an easier life.

When entering the quarantine zone, animals have to check in with a guard. The guard writes down the animal species, and then the animal is allowed to join the end of the line, in last position. At the other end of the line, animals have to check out: when the animal in first position in the line is finally allowed to enter the hunting-free area, another guard writes down the animal species. Thus, each guard maintains a list of animal species, written down in the chronological order in which the animals have checked in or out. A total of N animals, representing S species, have checked in (and, therefore, checked out).

However, animals may enter the waiting line and leave it in different orders. Indeed, some animal species are friends with each other, and thus two animals from such species, if they occupy adjacent places in the line, may accept to switch their places.

You have a list of those pairs of animal species that may accept to switch their places when being in adjacent positions in the line: this list contains L pairs. You were handed out the check-in list maintained by the first guard. Depending on which animals decided to switch places, several check-out lists might be possible. Among all those possible lists, which one comes first in alphabetical order?

Input

The input consists of the following lines:

- Line 1 contains three space-separated integers S, L and N. S is the number of animal species, L is the number of pairs of species that are friends with each other, and N is the number of animals that entered the waiting line.
- Line $i + 2$, for $0 \leqslant i < S$, contains the name of one of the represented species: this name is made of a single word, with uppercase letters between "A" and "Z", and contains between 1 and 20 letters.
- Line $i + S + 2$, for $0 \leqslant i < L$, contains two space-separated species names A and B describing that A and B are friends with each other.

- Line $S + L + 2$ represents the check-in list, and it contains N space-separated species names: for all $1 \leqslant k \leqslant N$, the kth word is the name of the species of the animal that entered the line in kth position.

Limits

- $1 \leqslant S \leqslant 200$;
- $0 \leqslant L \leqslant 10\,000$;
- $1 \leqslant N \leqslant 100\,000$.

Output

The output should contain a single line containing N words $w_0, \ldots, w_{N-1}$, separated by spaces: the list $w_0, \ldots, w_{N-1}$ must be, among all the possible check-out lists, the one that comes first in alphabetical order.

Sample Input

```
3 2 6
ANTILOPE
CAT
ANT
CAT ANTILOPE
ANTILOPE ANT
ANT CAT CAT ANTILOPE CAT ANT
```

Sample Output

```
ANT ANTILOPE CAT CAT CAT ANT
```

Sample Explanation

The six possible orderings at check-out, sorted in (increasing) alphabetical order, are:

1. `ANT ANTILOPE CAT CAT CAT ANT`
2. `ANT CAT ANTILOPE CAT CAT ANT`
3. `ANT CAT CAT ANTILOPE CAT ANT`
4. `ANT CAT CAT CAT ANT ANTILOPE`
5. `ANT CAT CAT CAT ANTILOPE ANT`
6. `ANTILOPE ANT CAT CAT CAT ANT`

H: Pseudo-Random Number Generator

Time limit: 0.3 second

Donald loves nature. Being a programmer, Donald writes programs to simulate the growth of trees or to build realistic 3D landscapes. For this purpose, Donald needs a good pseudo-random number generator. He devises the following method to produce an infinite sequence of 40-bit unsigned integers (the lines in green are comments).

$$\begin{array}{rcll} M & \coloneqq & 1 << 40 & // = 2^{40} = 1\,099\,511\,627\,776 \\ S(0) & \coloneqq & \texttt{0x600DCAFE} & // = 1\,611\,516\,670 \\ S(n+1) & \coloneqq & (S(n) + (S(n) >> 20) + 12345) \ \% \ M & \end{array}$$

On the last line, $x >> 20$ denotes the quotient of the Euclidean division of x by 2^{20} and $x \ \% \ M$ denotes the remainder of the Euclidean division of x by M.

As a very first test to decide if this is indeed a good pseudo-random number generator, Donald wishes to count the number of even values produced by this sequence, in order to check whether this is close enough to 50%. Your help will be welcome.

Input

The input consists of a single line, containing an integer N.

Limits

The input satisfies $0 \leqslant N < 2^{63}$.

Output

The output should contain a single line with a single integer corresponding to the number of even values in the sequence $S(0), S(1), \ldots, S(N-1)$.

Sample Input 1

```
3
```

Sample Output 1

```
2
```

Sample Input 2

```
500000000
```

Sample Output 2

```
250065867
```

I: Rats

Time limit: 1 second

To celebrate the Lunar New Year of the Rat, Douglas decides to count the number of rats living in his area. It is impossible for him to find all rats, as they tend to be well hidden. However, on the first day of the new year, Douglas manages to capture n_1 rats, and marks each of them with an ear tag before releasing them. On the second day of the new year, Douglas captures n_2 rats, and observes that n_{12} of them had been marked during the first day.

Douglas is asking for your help to estimate the total number of rats in his area. Looking up in your statistics textbook, you propose using the Chapman estimator $\hat{N}$, given by:

$$\hat{N} := \left\lfloor \frac{(n_1+1)(n_2+1)}{n_{12}+1} - 1 \right\rfloor$$

where $\lfloor x \rfloor$ is the *floor* of a real number x, i.e., the closest integer less than or equal to x.

Input

The input consists of a single line, with three space-separated integers: n_1, n_2, n_{12}, in that order.

Output

The output should contain a single line with the single integer $\hat{N}$.

Limits

- $0 \leqslant n_1, n_2 \leqslant 10\,000$;
- $0 \leqslant n_{12} \leqslant \min(n_1, n_2)$.

Sample Input

```
15 18 11
```

Sample Output

```
24
```

Page intentionally left blank

J: Counting Trees

Time limit: 2 seconds

Your friend Darwin is a very famous naturalist, and tells you that he is currently trying to count the number of varieties of a very particular species of trees he discovered two years ago.

He tells you that these trees are very peculiar. First, they are flat: one of their sides is facing south and the other is facing north. Second, when one of their branches splits, it splits in exactly two sub-branches, which go in two opposite directions: one goes towards the west and the other goes towards the east. All in all, such a tree can be very easily represented as a binary tree such as computer scientists are used to. (Mathematically, a binary tree is either the empty tree or an internal node with exactly two children, which are themselves binary trees.)

As a great programmer, you get excited as soon as you hear about binary trees. Darwin continues his explanations: in such a tree, branches are either horizontal or go up. Still obsessed by binary trees, you figure out that this corresponds to saying that, in the binary tree representation, if internal nodes are labelled by the elevation (distance from the ground) of the corresponding branch split, then the label of a non-root node is always larger than or equal to the label of its parent. Empty trees are not labelled.

While you find this property rather boring and not very interesting, Darwin goes on and tells you a really astonishing and subtle fact he recently discovered about his favorite species of trees. After a lengthy discussion with Darwin, you finally understand that this property can be simply expressed in the binary tree representation. Namely, if one does an inorder traversal of the labelled binary tree of any tree of that species, the resulting sequence of elevations is always the same! (The inorder traversal of the empty tree is the empty sequence and, if a tree is not empty, its inorder traversal is the concatenation of the inorder traversal of the left sub-tree, followed by the label of the top node, followed by the inorder traversal of the right sub-tree.)

After having heard the impressive research results of your friend, you urge him to elaborate on the methodology he uses to count the varieties of this particularly surprising species. This leads him to reveal the hypothesis he is currently working on: every tree of a given variety is represented by the same binary tree, which uniquely identifies that variety. Moreover, every labeled binary tree verifying the conditions above indeed corresponds to an existing variety. Darwin believes that some kind of

mathematical tool could then be used to count the varieties, but he has not enough mathematical background to do so.

Now that you are really impressed by Darwin's work, it is your turn: impress him by writing a program which counts the number of varieties of that species.

Input

The input consists of the following lines:

- on the first line: the number N of elements of the sequence of elevations produced by the inorder traversal of any tree of the species;
- the N following lines represent the sequence of elevations, in millimeters, with one integer per line.

Limits

Both N and the elevations are greater than or equal to 0 and less than or equal to 1 000 000.

Output

A single line with a single integer, the number of varieties of trees in the species, modulo 1 000 000 007.

Sample Input 1

```
6
3
1
6
2
4
5
```

Sample Output 1

```
1
```

Sample Input 2

```
6
1
1
1
1
1
1
```

Sample Output 2

```
132
```

K: Bird Watching

Time limit: 3 seconds

CC BY-SA 3.0 by Dick Daniels

Kiara studies an odd species of birds which travel in a very peculiar way. Their movements are best explained using the language of graphs: there exists a directed graph $\mathcal{G}$ where the nodes are trees, and a bird can only fly from a tree T_a to T_b when (T_a, T_b) is an edge of $\mathcal{G}$.

Kiara does not know the real graph $\mathcal{G}$ governing the flight of these birds but, in her previous field study, Kiara has collected data from the journey of many birds. Using this, she has devised a graph $\mathcal{P}$ explaining how they move. Kiara has spent so much time watching them that she is confident that if a bird can fly directly from a to b, then she has witnessed at least one such occurrence. However, it is possible that a bird flew from a to b to c but she only witnessed the stops a and c and then added (a, c) to $\mathcal{P}$. It is also possible that a bird flew from a to b to c to d and she only witnessed a and d, and added (a, d) to $\mathcal{P}$, etc. To sum up, she knows that $\mathcal{P}$ contains all the edges of $\mathcal{G}$ and that $\mathcal{P}$ might contain some other edges (a, b) for which there is a path from a to b in $\mathcal{G}$ (note that $\mathcal{P}$ might not contain all such edges).

For her next field study, Kiara has decided to install her base next to a given tree T. To be warned of the arrival of birds on T, she would also like to install detectors on the trees where the birds can come from (i.e. the trees T' such that there is an edge (T', T) in $\mathcal{G}$). As detectors are not cheap, she only wants to install detectors on the trees T' for which she is sure that (T', T) belongs to $\mathcal{G}$.

Kiara is sure that an edge (a, b) belongs to $\mathcal{G}$ when (a, b) is an edge of $\mathcal{P}$ and all the paths in $\mathcal{P}$ starting from a and ending in b use the edge (a, b). Kiara asks you to compute the set $\mathcal{S}(T)$ of trees T' for which she is sure that (T', T) is an edge of $\mathcal{G}$.

Input

The input describes the graph $\mathcal{P}$. The first line contains three space-separated integers N, M, and T: N is the number of nodes of $\mathcal{P}$, M is the number of edges of $\mathcal{P}$ and T is the node corresponding to the tree on which Kiara will install her base.

The next M lines describe the edges of the graph $\mathcal{P}$. Each contains two space-separated integers a and b ($0 \leqslant a, b < N$ and $a \neq b$) stating that $(a, b) \in \mathcal{P}$. It is guaranteed that the same pair (a, b) will not appear twice.

Limits

- $1 \leqslant N, M \leqslant 100\,000$;
- $0 \leqslant T < N$.

Output

Your output should describe the set $\mathcal{S}(T)$. The first line should contain an integer L, which is the number of nodes in $\mathcal{S}(T)$, followed by L lines, each containing a different element of $\mathcal{S}(T)$. The elements of $\mathcal{S}(T)$ should be printed in increasing order, with one element per line.

Sample Input 1

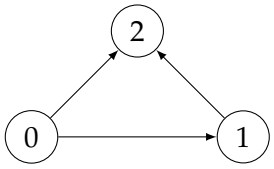
```
3 3 2
0 1
0 2
1 2
```

Sample Output 1

```
1
1
```

Sample Explanation 1

The graph corresponding to this example is depicted on the right. The node 1 belongs to $\mathcal{S}(2)$ because the (only) path from 1 to 2 uses $(1,2)$. The node 0 does not belong to $\mathcal{S}(2)$ because the path $0 \to 1 \to 2$ does not use the edge $(0,2)$.



Sample Input 2

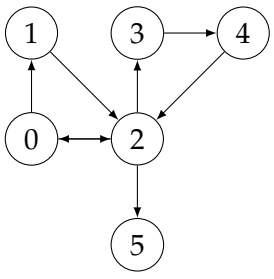
```
6 8 2
0 1
0 2
1 2
2 0
2 3
3 4
4 2
2 5
```

Sample Output 2

```
2
1
4
```

Sample Explanation 2

The graph corresponding to this example is depicted on the right. For the same reason as in Sample 1, the node 0 does not belong to $\mathcal{S}(2)$ while 1 does. The nodes 3 and 5 do not belong to $\mathcal{S}(2)$ because we do not have edges $(3,2)$ or $(5,2)$. Finally 4 belongs to $\mathcal{S}(2)$ because all paths from 4 to 2 use the edge $(4,2)$.



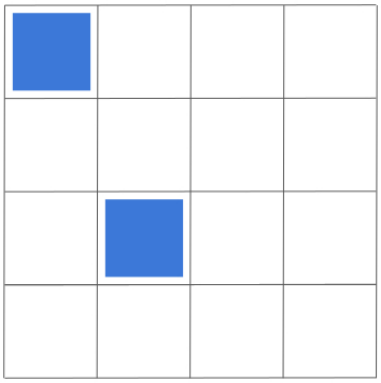
L: River Game

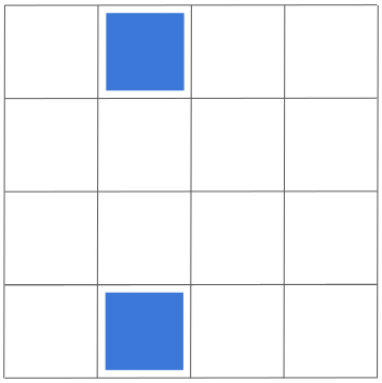
Time limit: 0.5 second

Together with an ornithologist friend, you are spending hours bird-watching in wetlands, waiting for a rare bird that you want to take a picture of.

In order to kill time, you devised a little game. The wetland is represented by an $N \times N$ grid made of 3 types of squares: firm ground, wet zone, and protected zone.

Connected wet zone squares form wet areas, which are maximal sets of wet zone squares such that we can go from any square of the area to any other square of the area by a path of connected wet zone squares. Two squares are considered connected when they share one edge, so that squares not on the border of the grid are connected to 4 other squares. Each wet area must be connected to both the left and right sides of the grid, and does not contain more than $2N$ wet zone squares. Two wet zone squares that belong to two different wet areas will always be at distance at least 3 from each other, the distance being counted by moving horizontally or vertically on the grid. For example, in the two examples below, the blue squares are at distance 3 from one another.





One after the other, you and your friend will each take turns placing a camera on the wetland with the following constraints:

- The camera must be on a firm ground square.
- The camera must be adjacent to a wet zone square, so that you can take bird photos.
- The camera must not be on a protected zone square.
- There cannot be two cameras on the same square.
- No two cameras adjacent to the same wet area can be adjacent (again, the notion of adjacency means sharing an edge).

The first player who cannot place a camera anymore loses the game.

Assuming both players play optimally, will the first player win or lose?

Input

The input consists of the following lines:

- on the first line, the integer N;
- on the next N lines, a string representing a row of the grid: "*" represents a wet zone square, "." represents a firm ground square and "x" represents a protected square.

Limits

The input satisfies $1 \leqslant N \leqslant 10$.

Output

The four words "First player will win" if the first player wins the game provided both players play optimally, or "Second player will win" otherwise.

Sample Input 1

```
3
...
...
***
```

Sample Output 1

```
First player will win
```

Sample Explanation 1

The first player can place a camera in three possible places: if he places a camera in the left or right position, then the second player can place one in the other position, and wins the game. On the other hand, if the first player places a camera in the middle, the second player cannot place another camera and loses.

Sample Input 2

```
10
**********
*.x.*.....
*...*.....
*****.....
..........
..........
xx*******..
***x...*..
..*...x*.x
..********
```

Sample Output 2

```
Second player will win
```

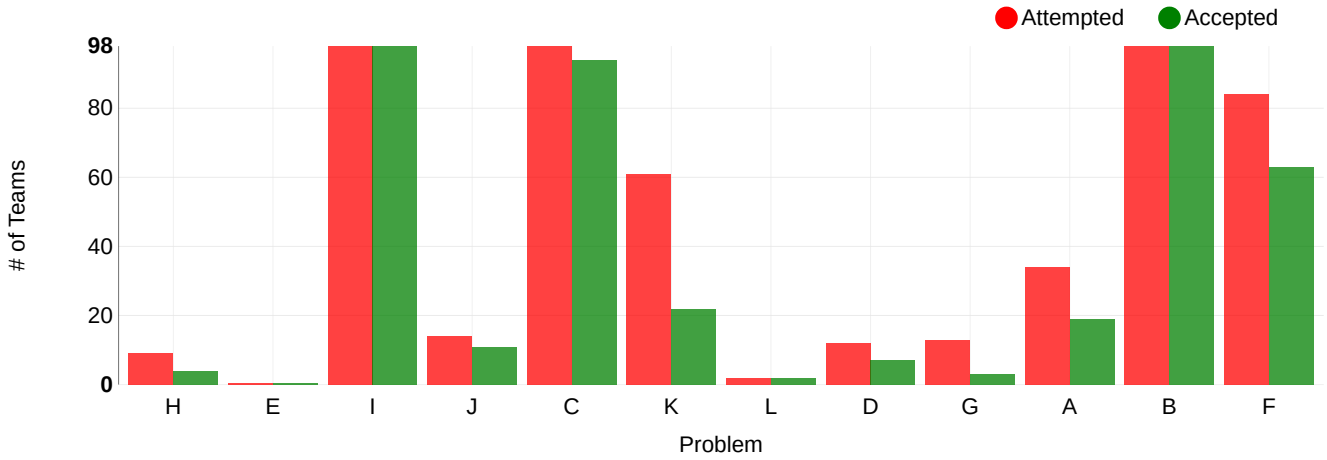
Problem Analysis Session

SWERC judges

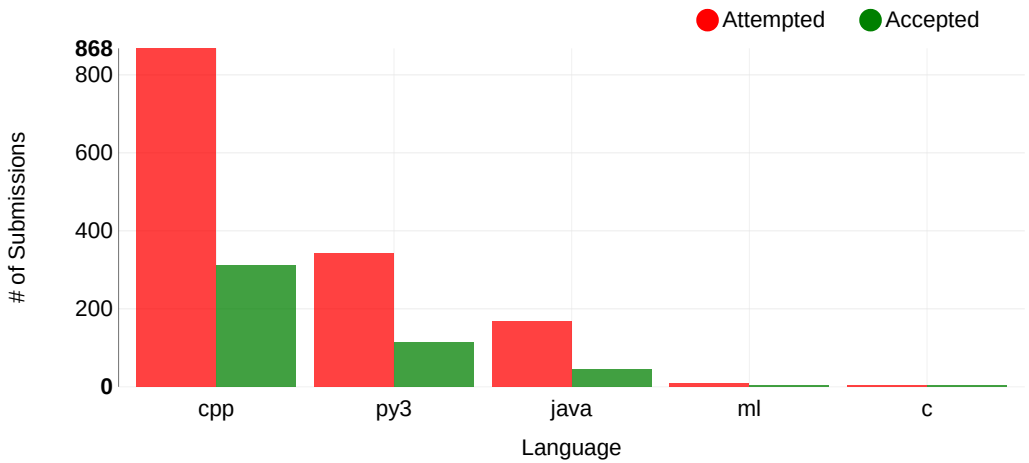
January 26, 2020

Statistics

Number of submissions: about 1400

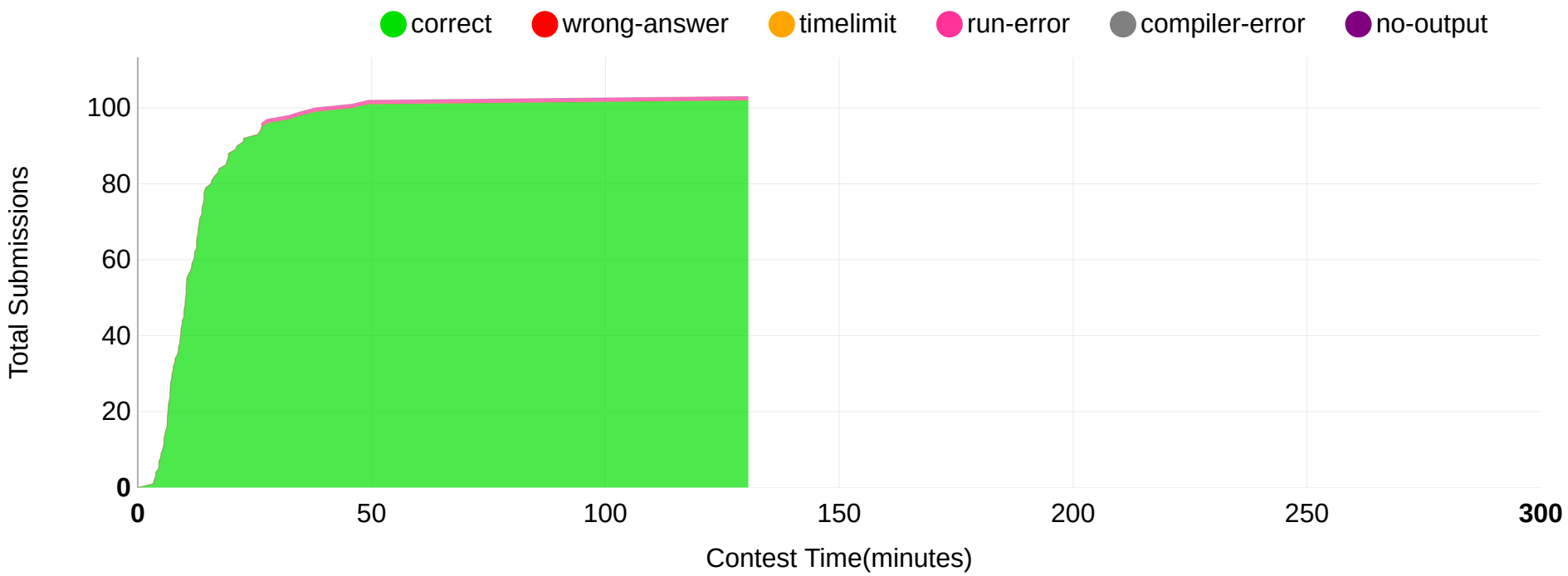


Number of clarification requests: 35 (28 answered "No comment.")



I – Rats

Solved by all the teams before freeze.
First solved after 3 min by
UnivRennes ISTIC.



I – Rats

This was the easiest problem of the contest.

Problem

Use mark-recapture to estimate the size of an animal population.

Solution

Given

- n_1: number of animals captured and marked on the first day
- n_2: number of animals captured on the second day
- n_{12}: number of animals recaptured (and thus already marked)

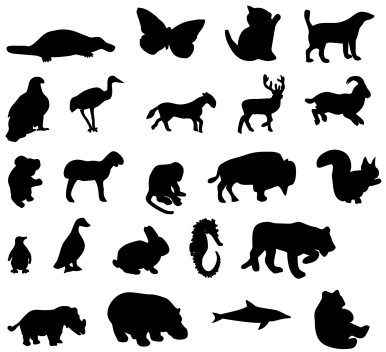
Use the Chapman estimator formula directly:

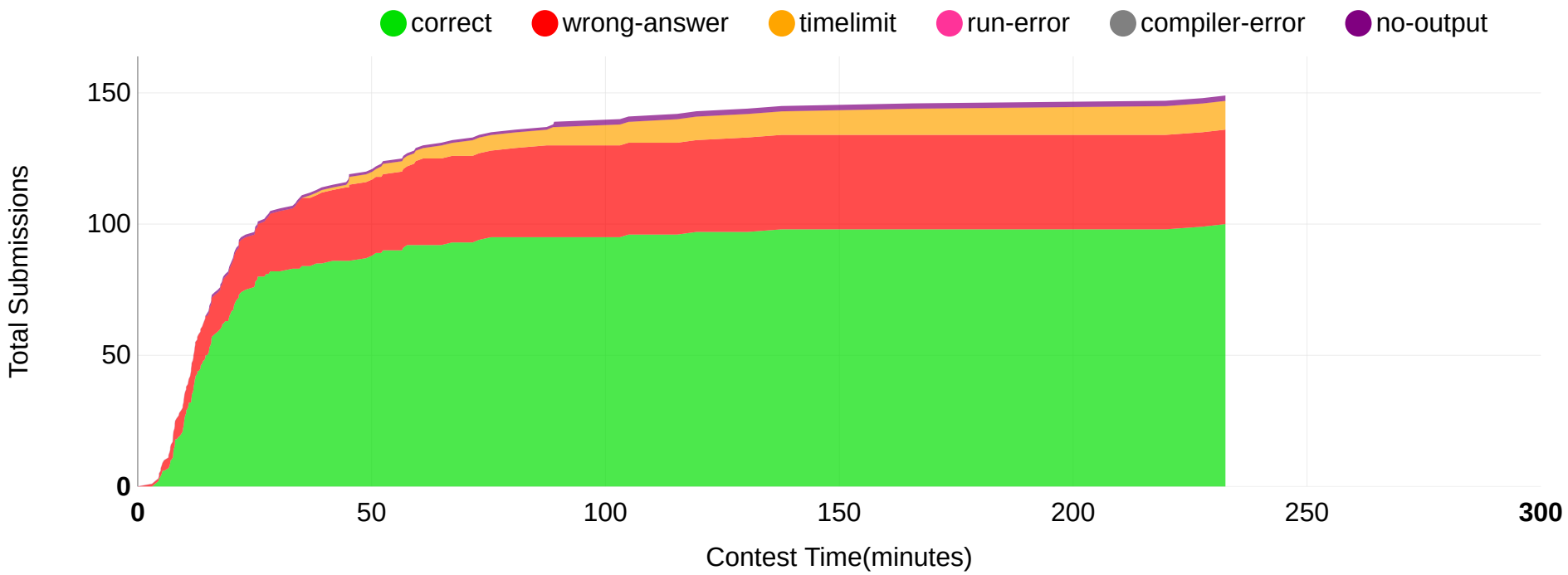
$$\text{population size} = \left\lfloor \frac{(n_1 + 1)(n_2 + 1)}{n_{12} + 1} - 1 \right\rfloor$$

No need for floats.

B – Biodiversity

Solved by all the teams before freeze.
First solved after 3 min by **Télécommander**.





B – Biodiversity

Problem (easy)

Computing the majority (more than half of the total) of N strings.

Solution (linear time and space)

Use a standard **hash table** to compute the number of occurences of each string and look for one that appears more than $N/2$ times.

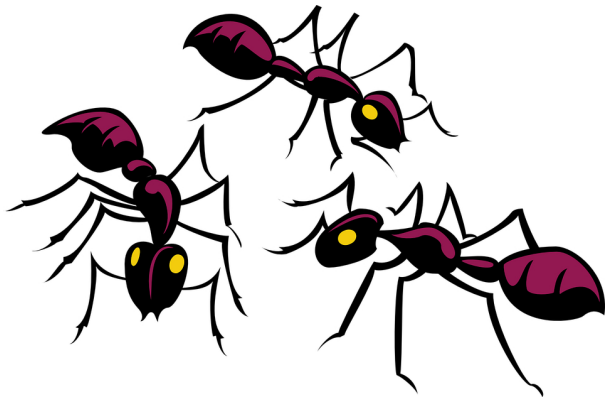
Alternate Solution (linear time and space)

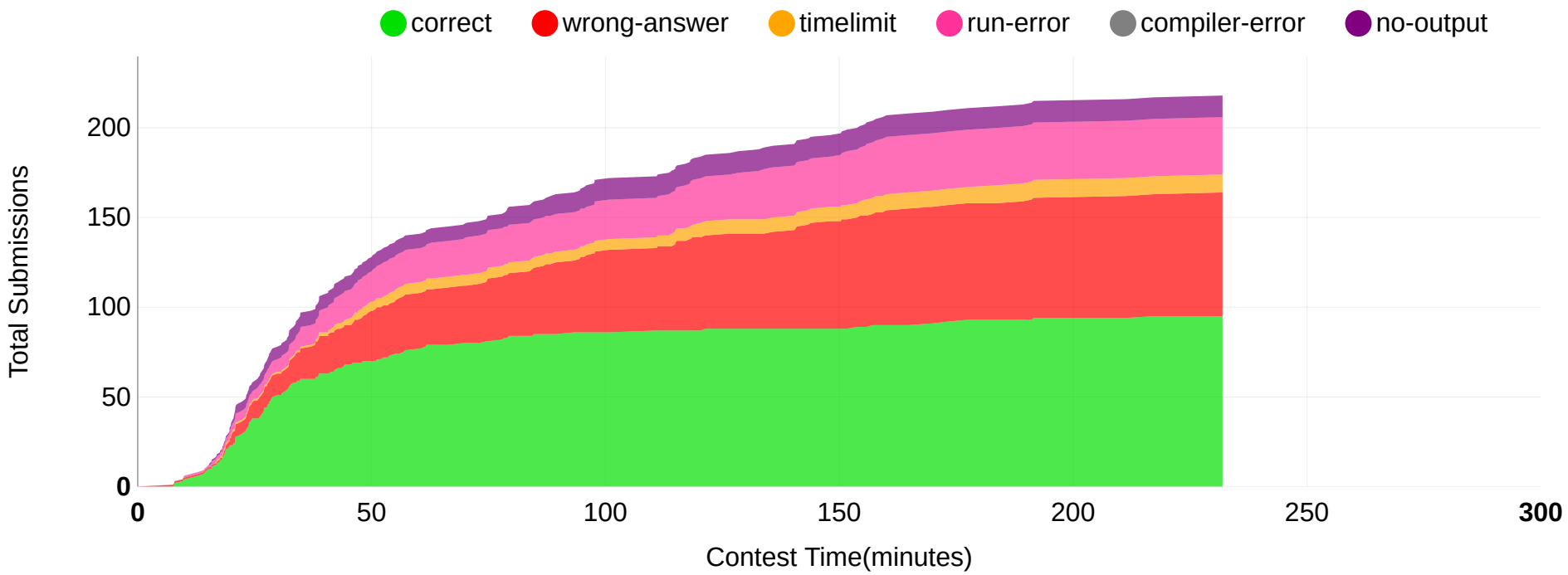
Use Boyer-Moore algorithm to find a candidate and check if the candidate actually appears more than $N/2$ times:

$count \leftarrow 0$
for each string S **do**
    if $count = 0$ **then** $candidate \leftarrow S$
    if $S = candidate$ **then** $count \leftarrow count + 1$
    else $count \leftarrow count - 1$

C – Ants

Solved by 94 teams before freeze.
First solved after 7 min by
Rubber Duck Forces.





C – Ants

This was an easy problem.

Problem: Minimum Excluded a.k.a. mex

Find the smallest *natural* number out of $\{X_1, X_2, \ldots, X_N\}$.

Straightforward solution

Store all the X_i in a set (e.g. a hash table), then linearly check for 0,1,...

A better solution

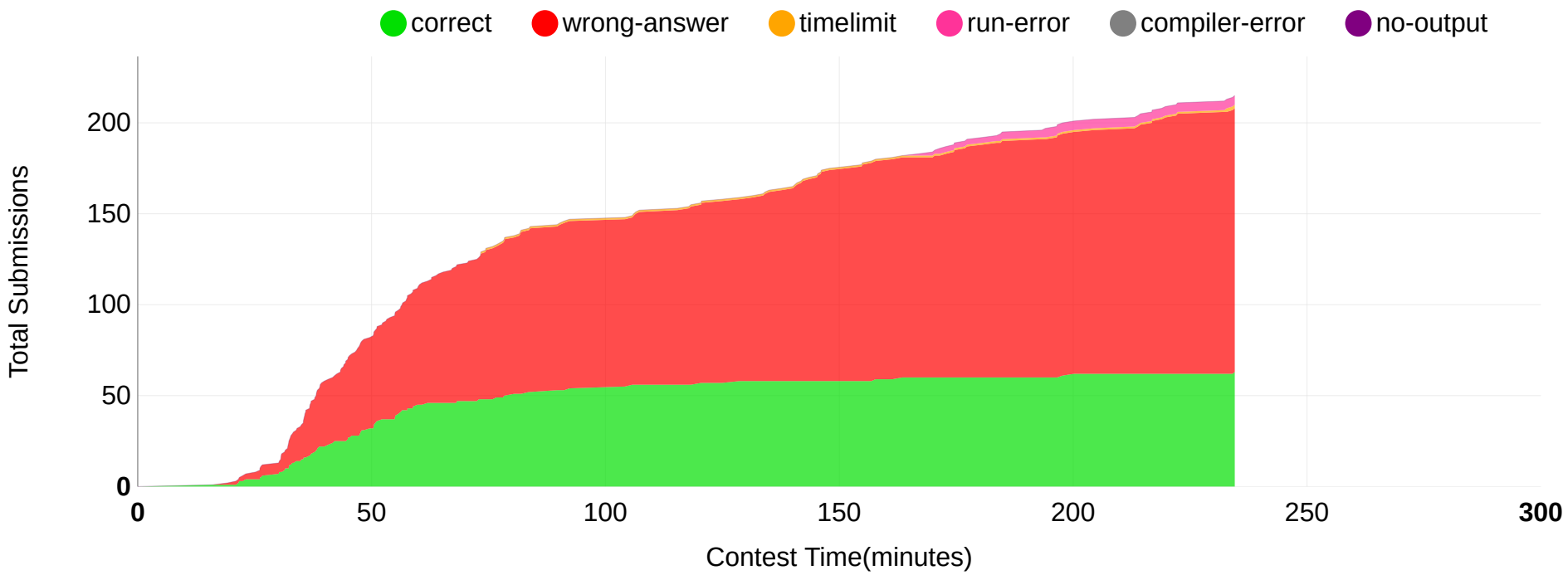
- The answer necessarily belongs to $\{0, 1, \ldots, N\}$.
- So we can use an array and ignore values out of that interval.

A more challenging variant

Do it in place.

F – Icebergs

Solved by 63 teams before freeze.
First solved after 15 min by **UPC-1**.



F – Icebergs

Problem

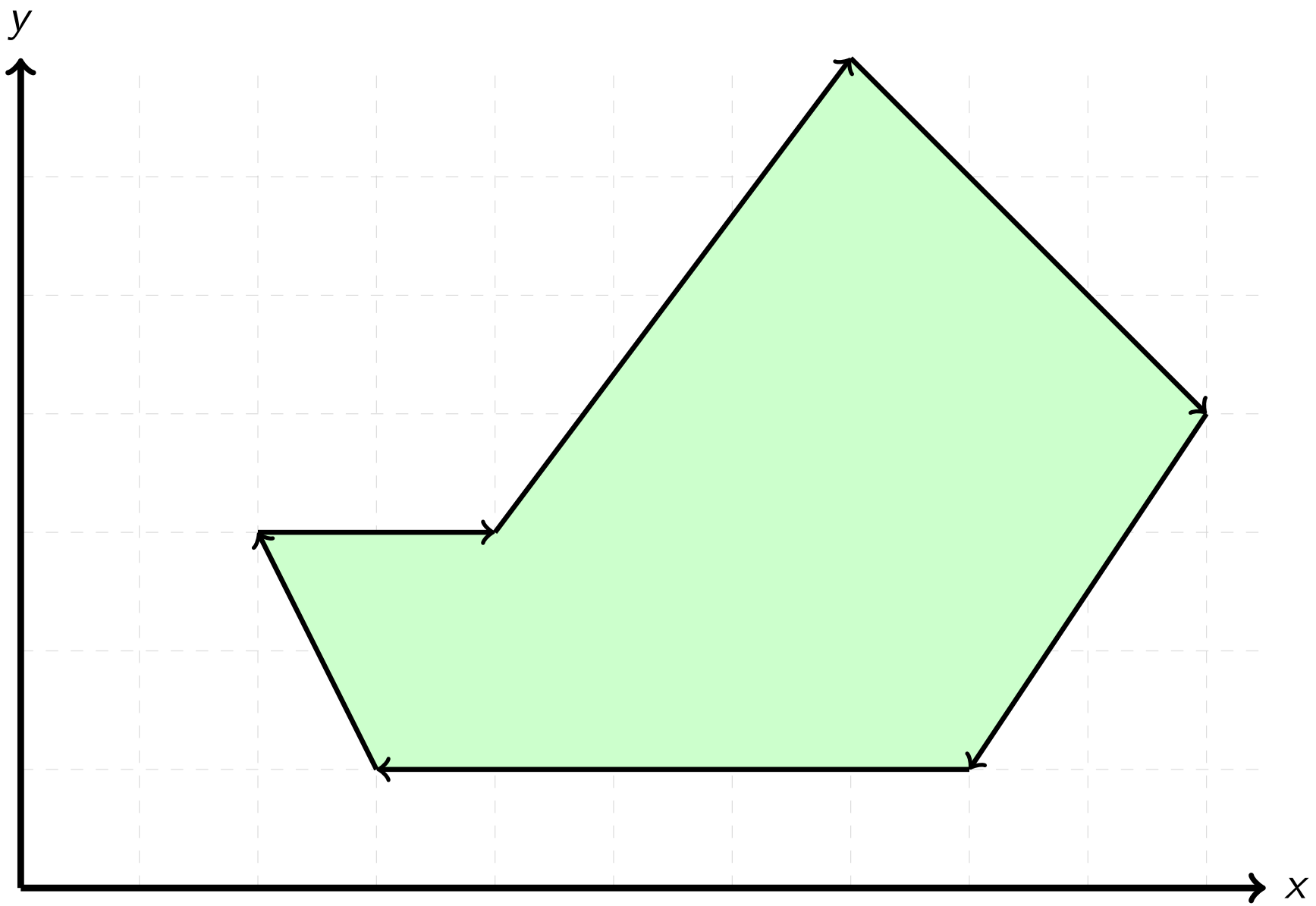
Given N polygons, compute their total area.

Remark

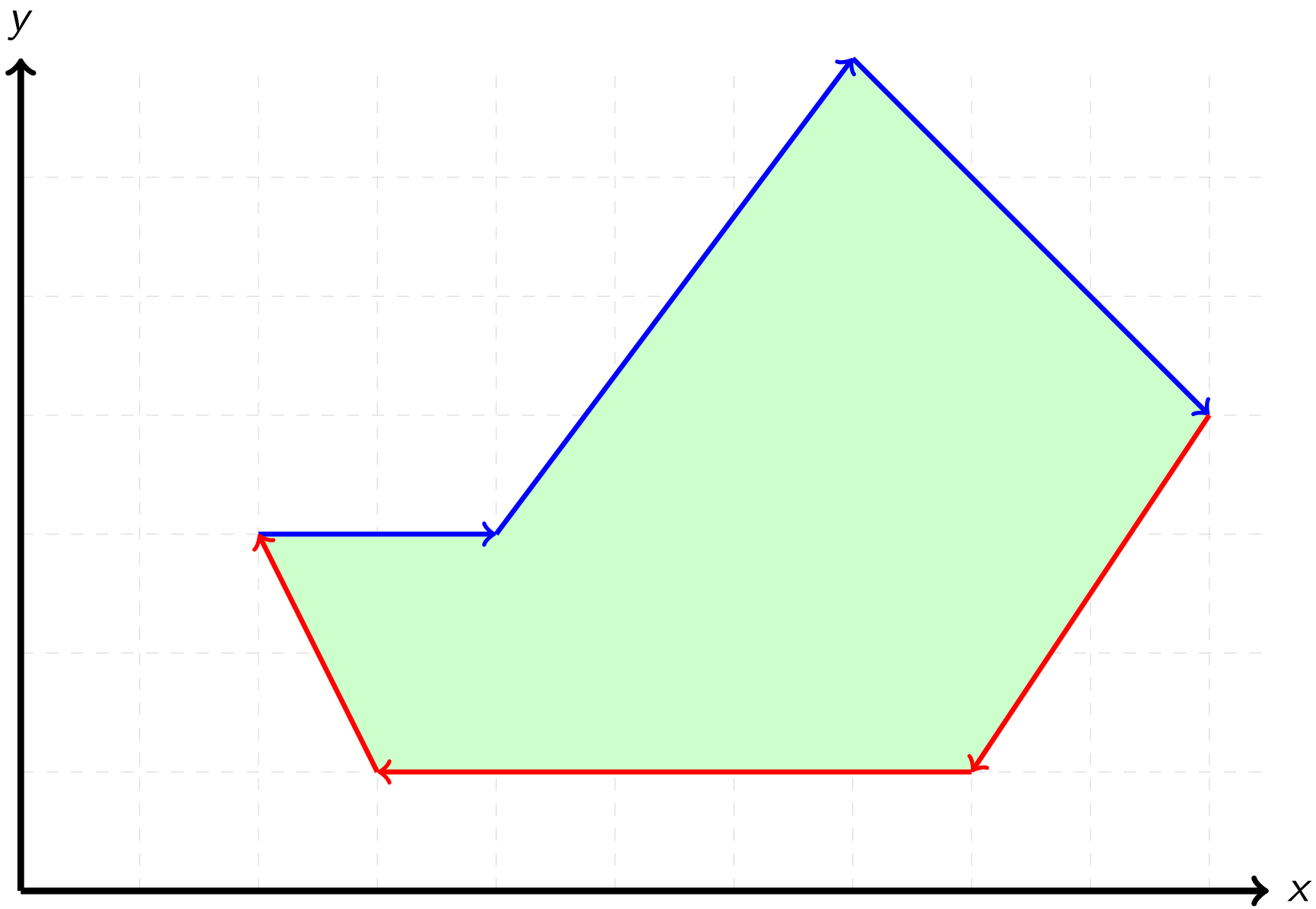
Polygons can be treated separately.

$\Rightarrow$ How to compute the area of a polygon?

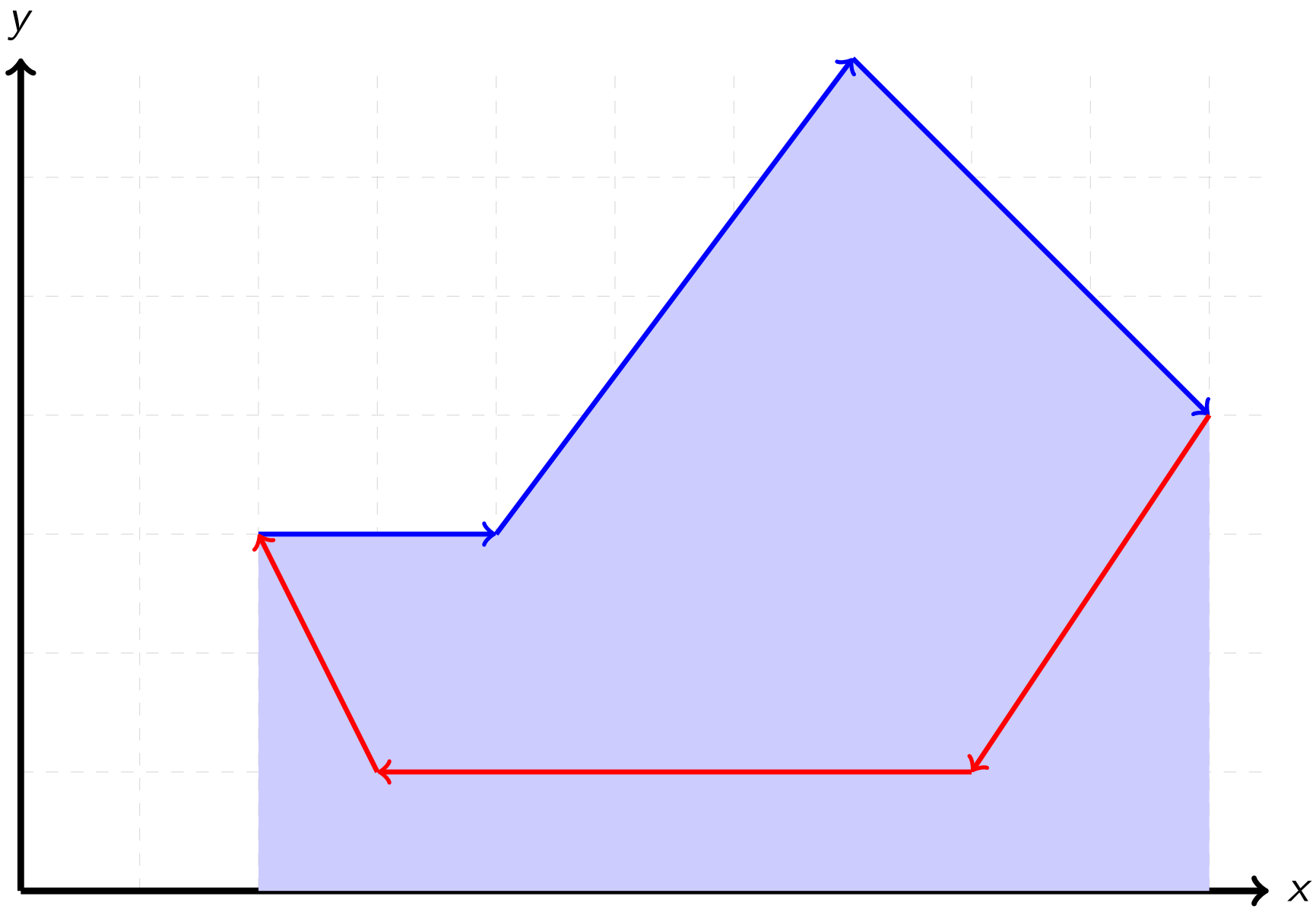
F – Icebergs



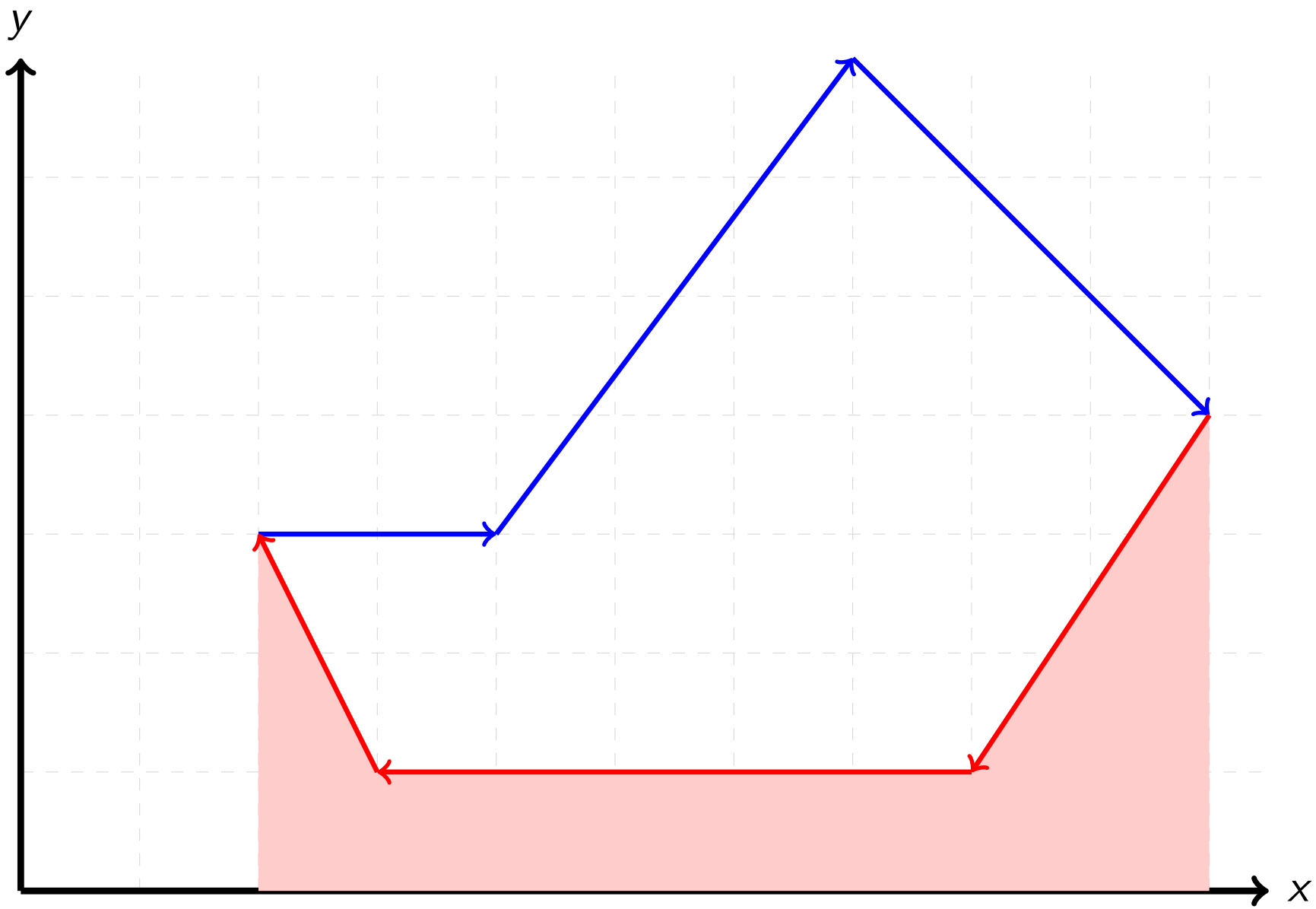
F – Icebergs



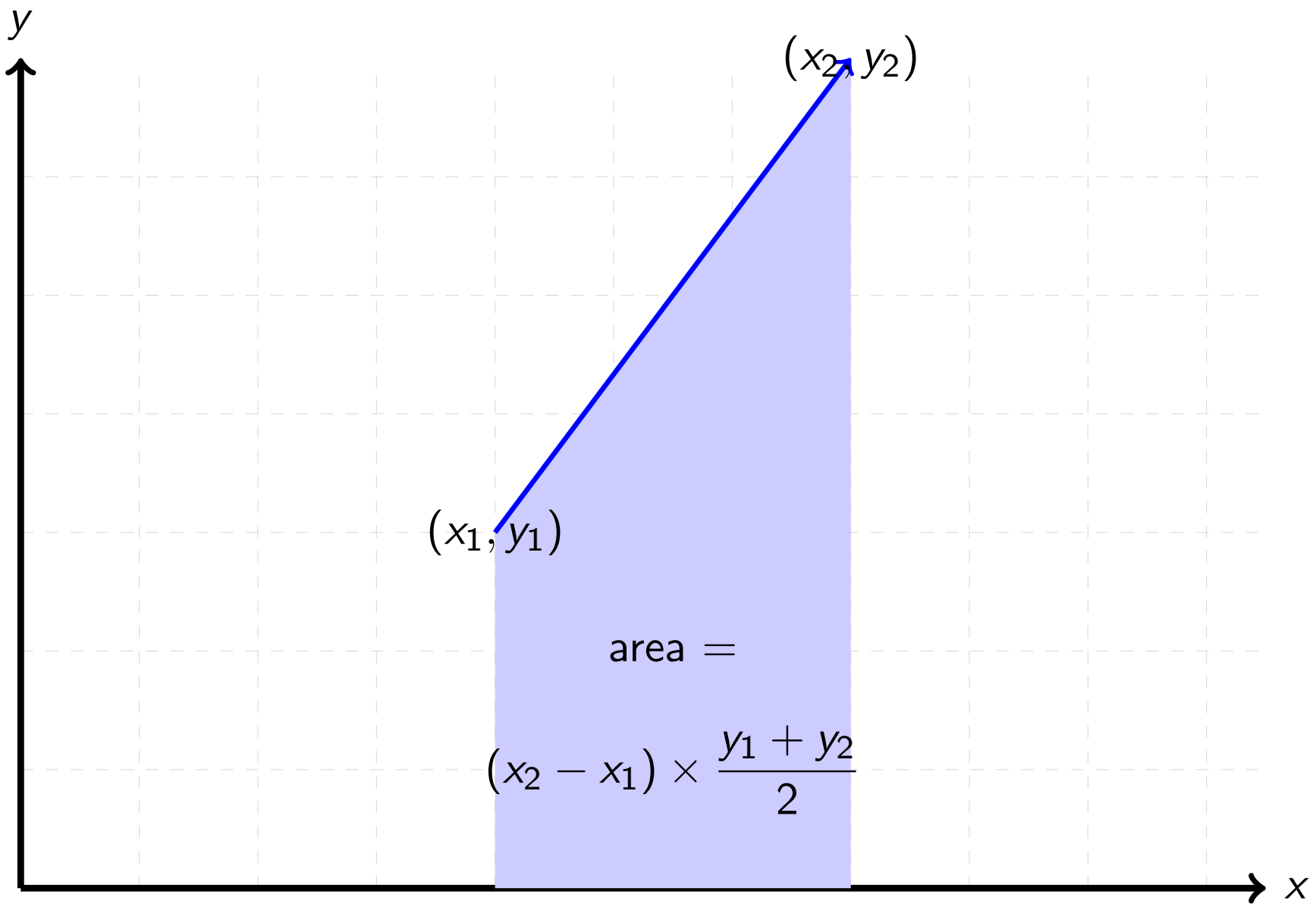
F – Icebergs



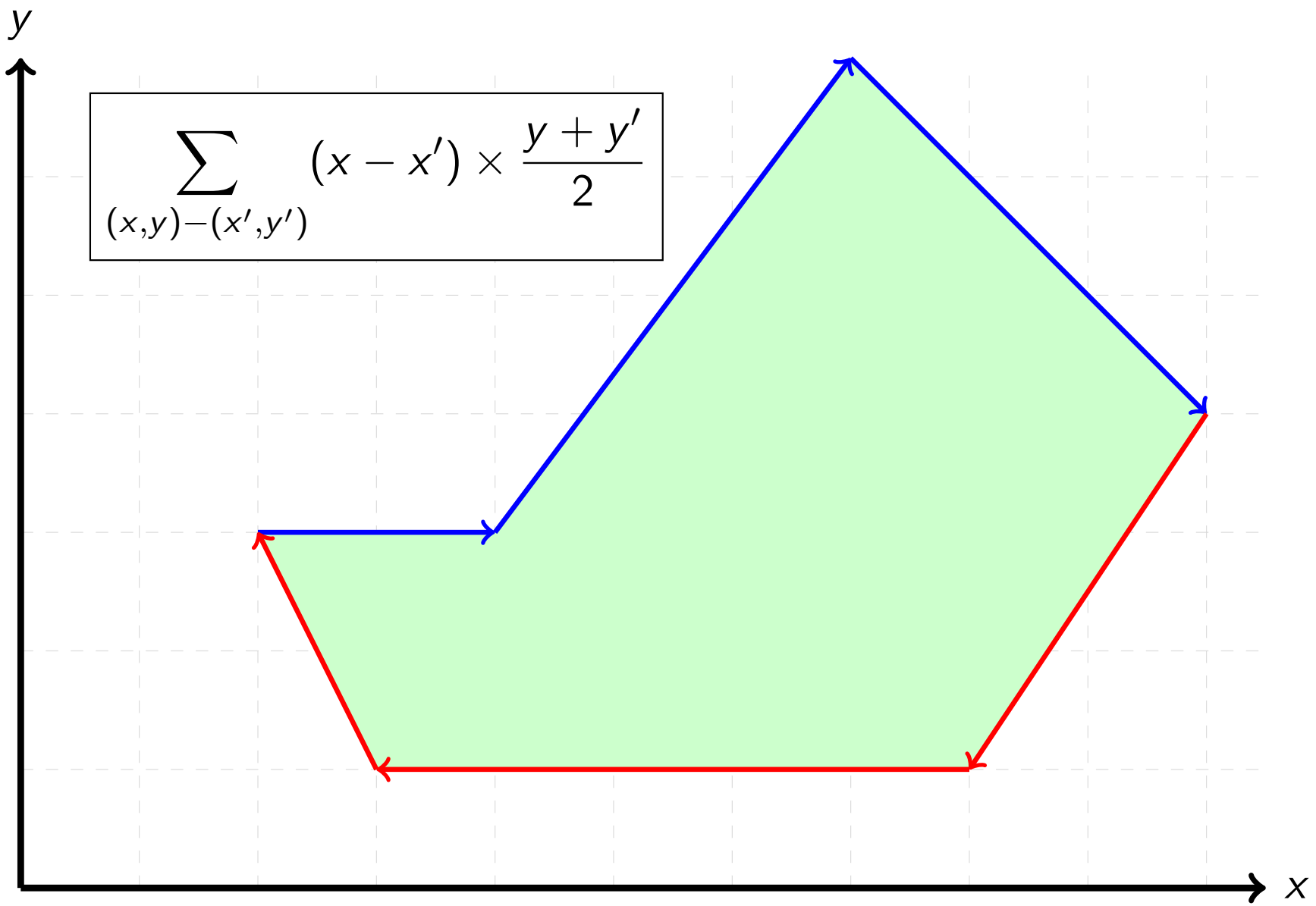
F – Icebergs



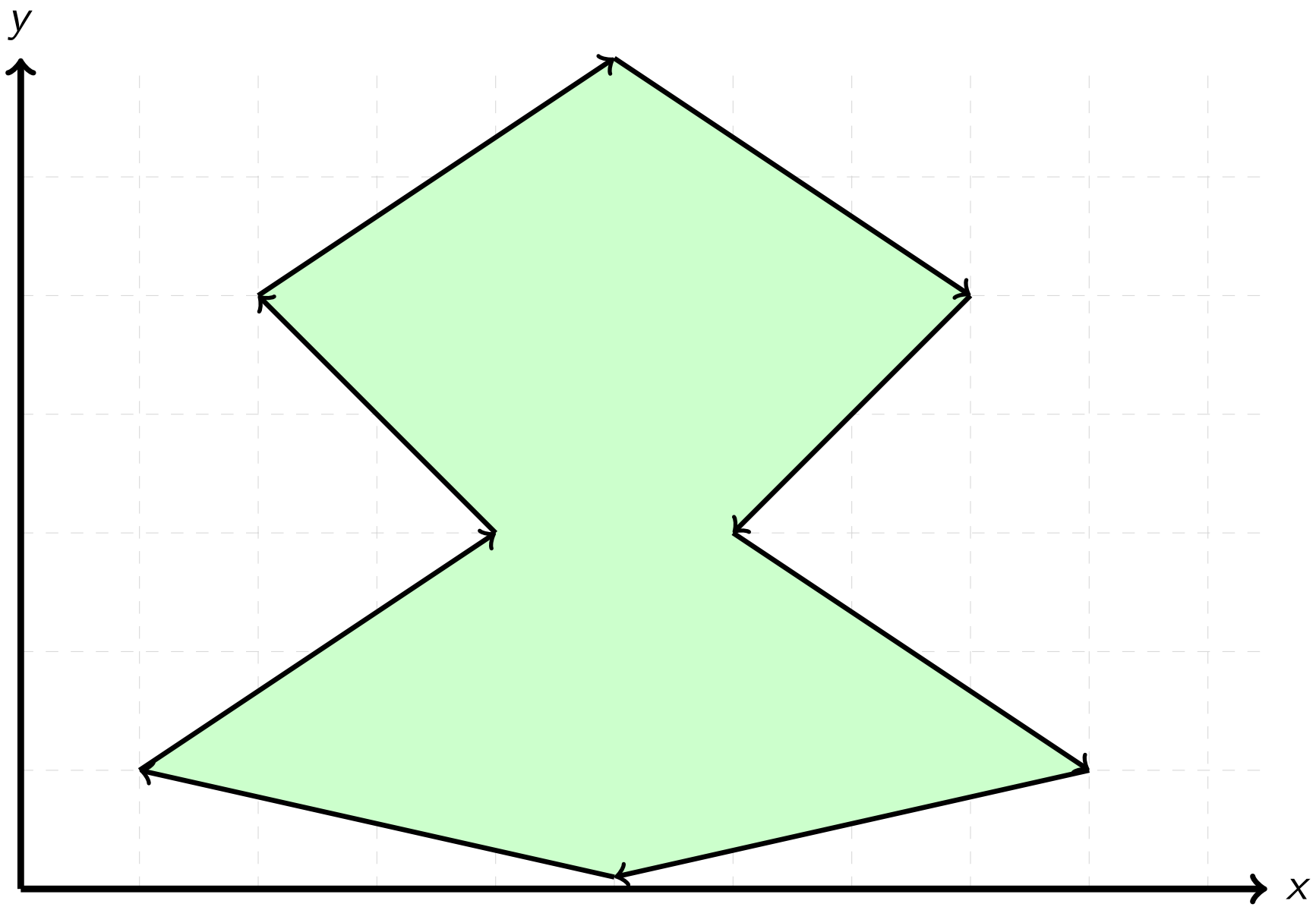
F – Icebergs



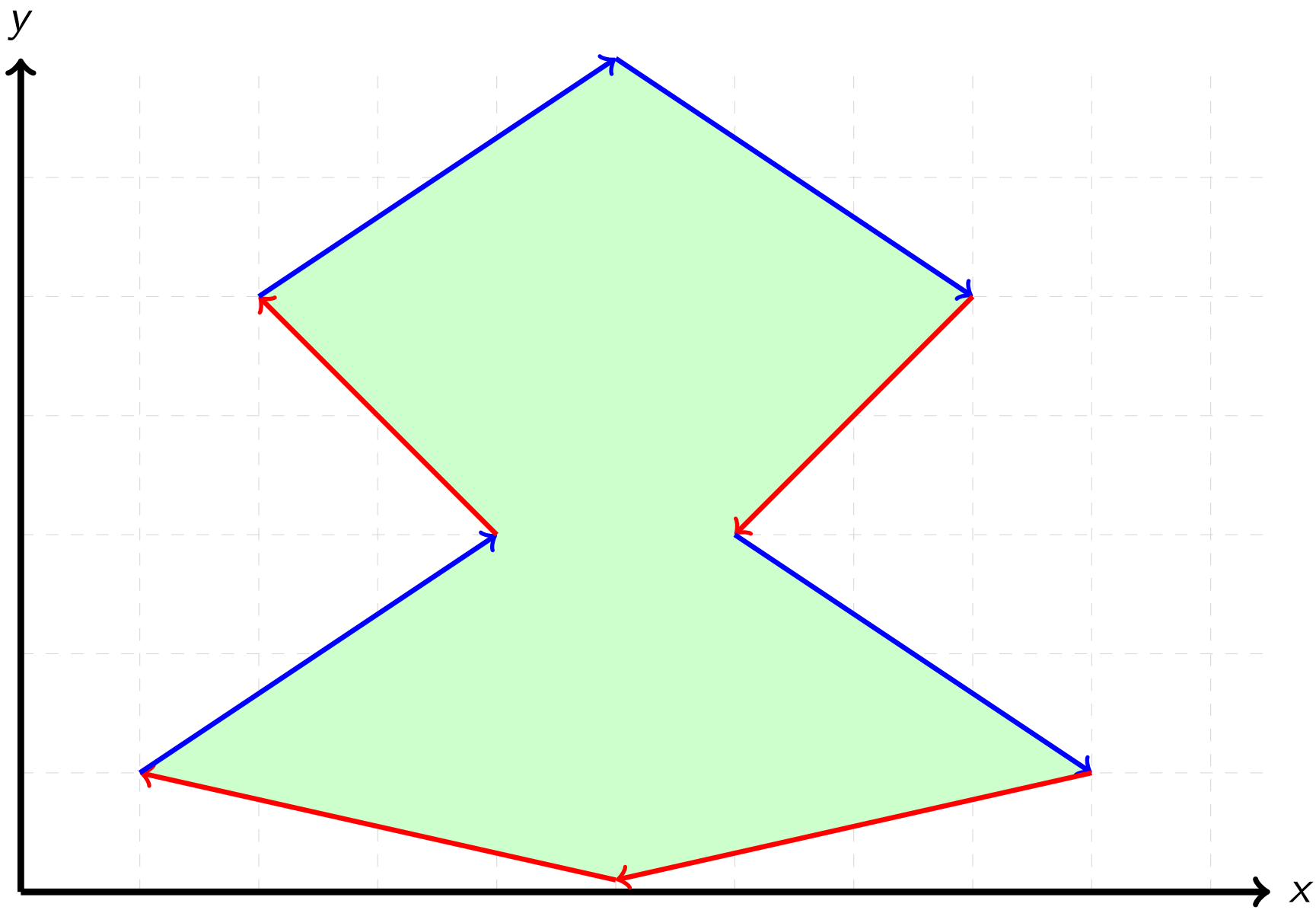
F – Icebergs



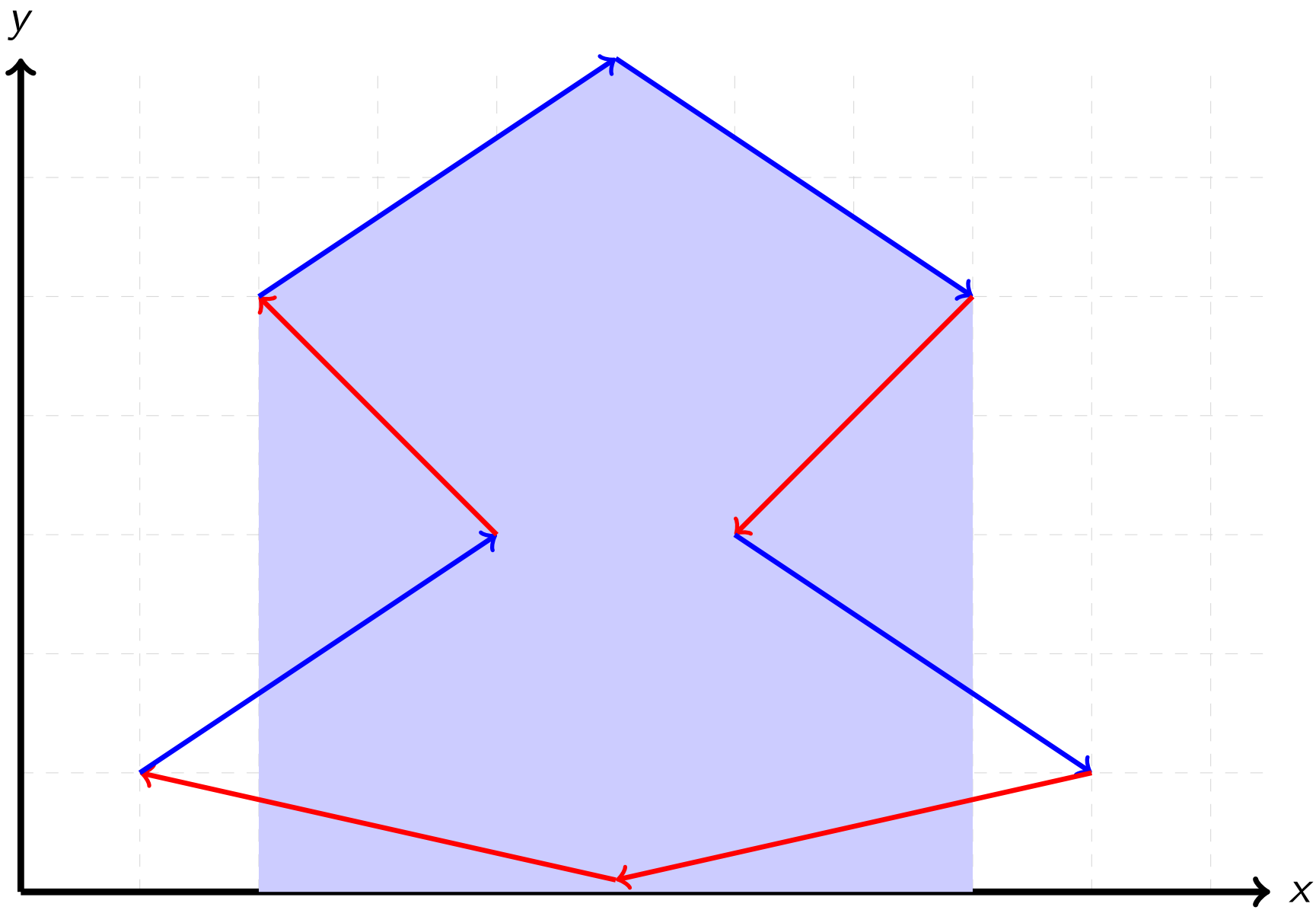
F – Icebergs



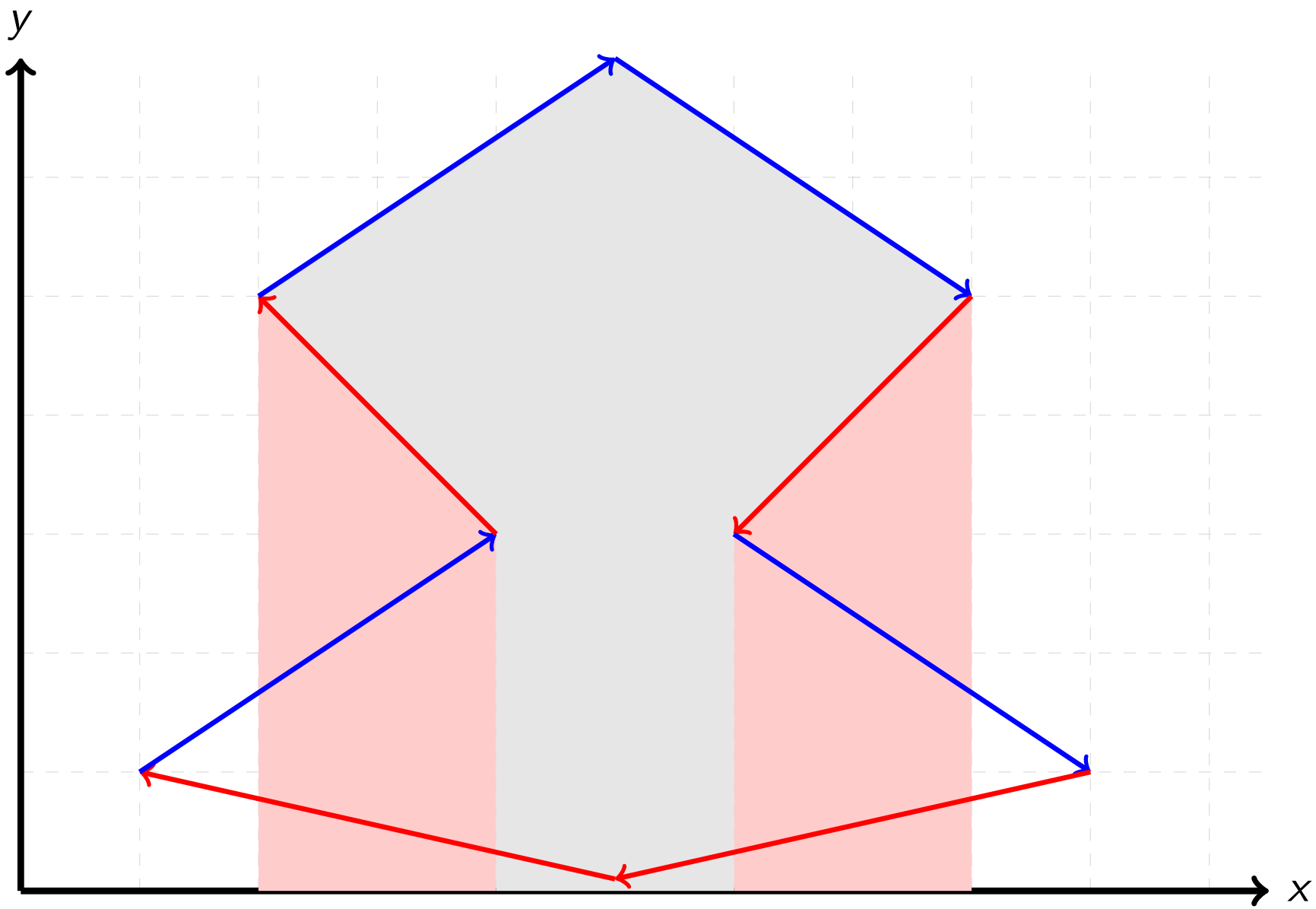
F – Icebergs



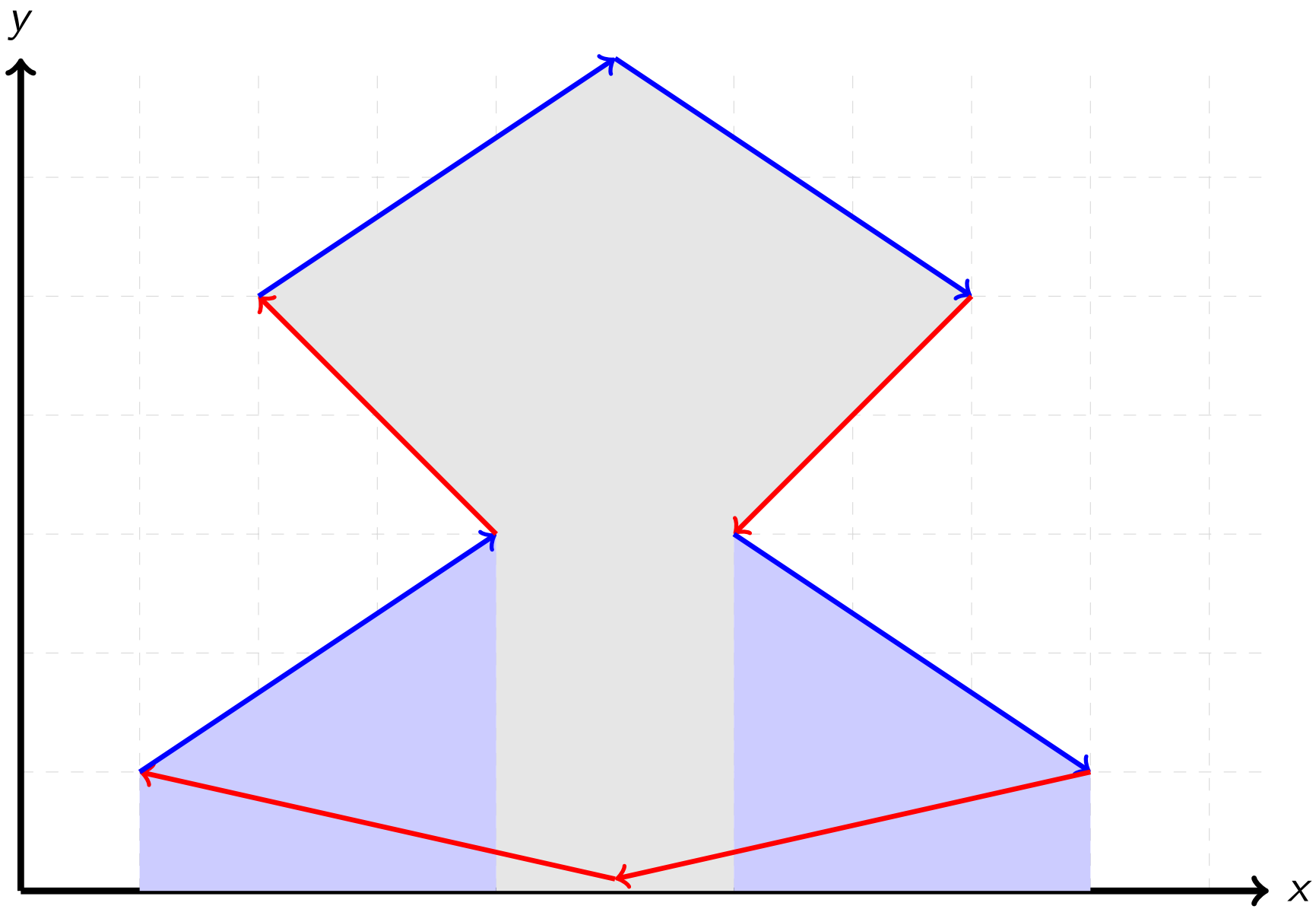
F – Icebergs



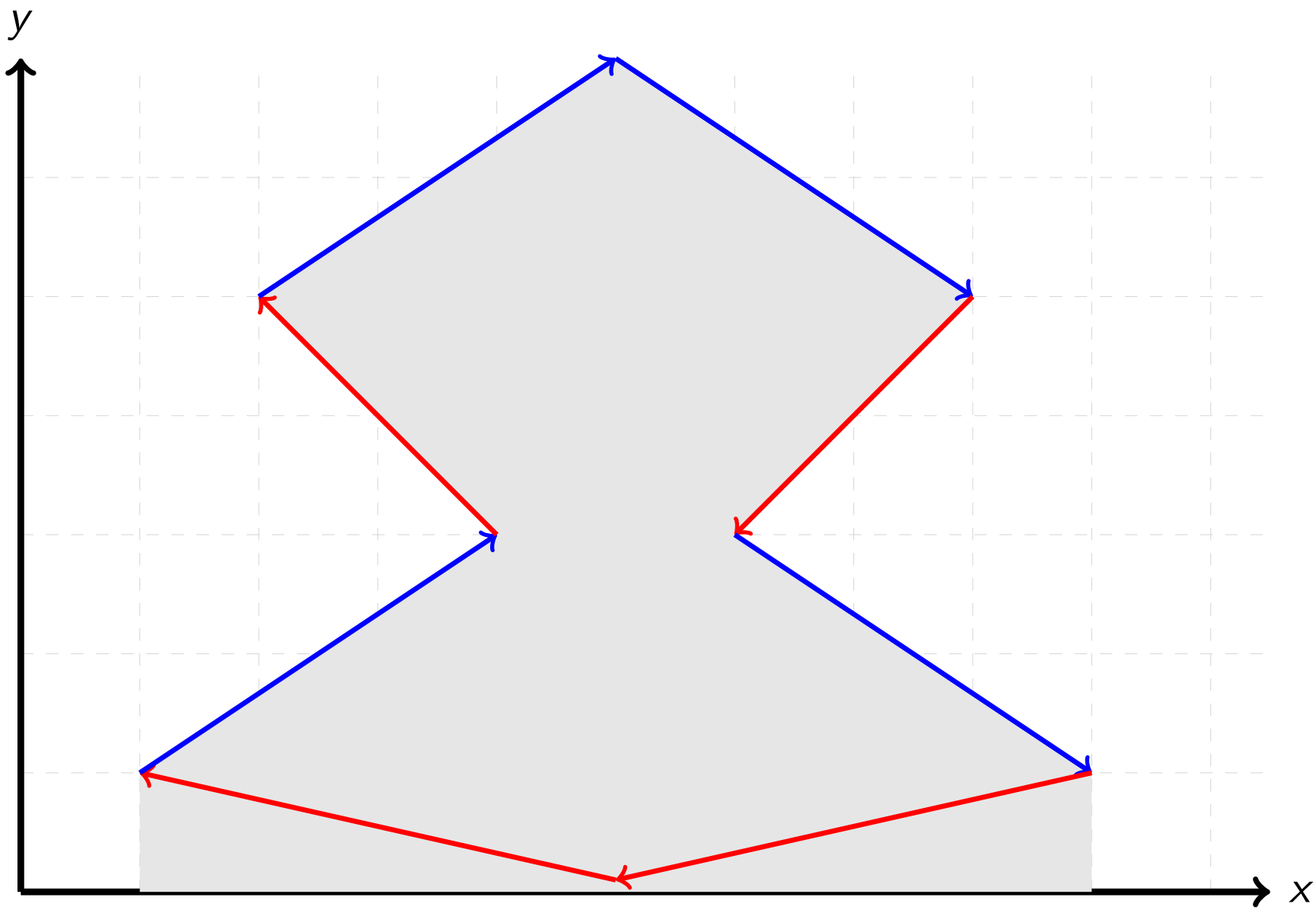
F – Icebergs



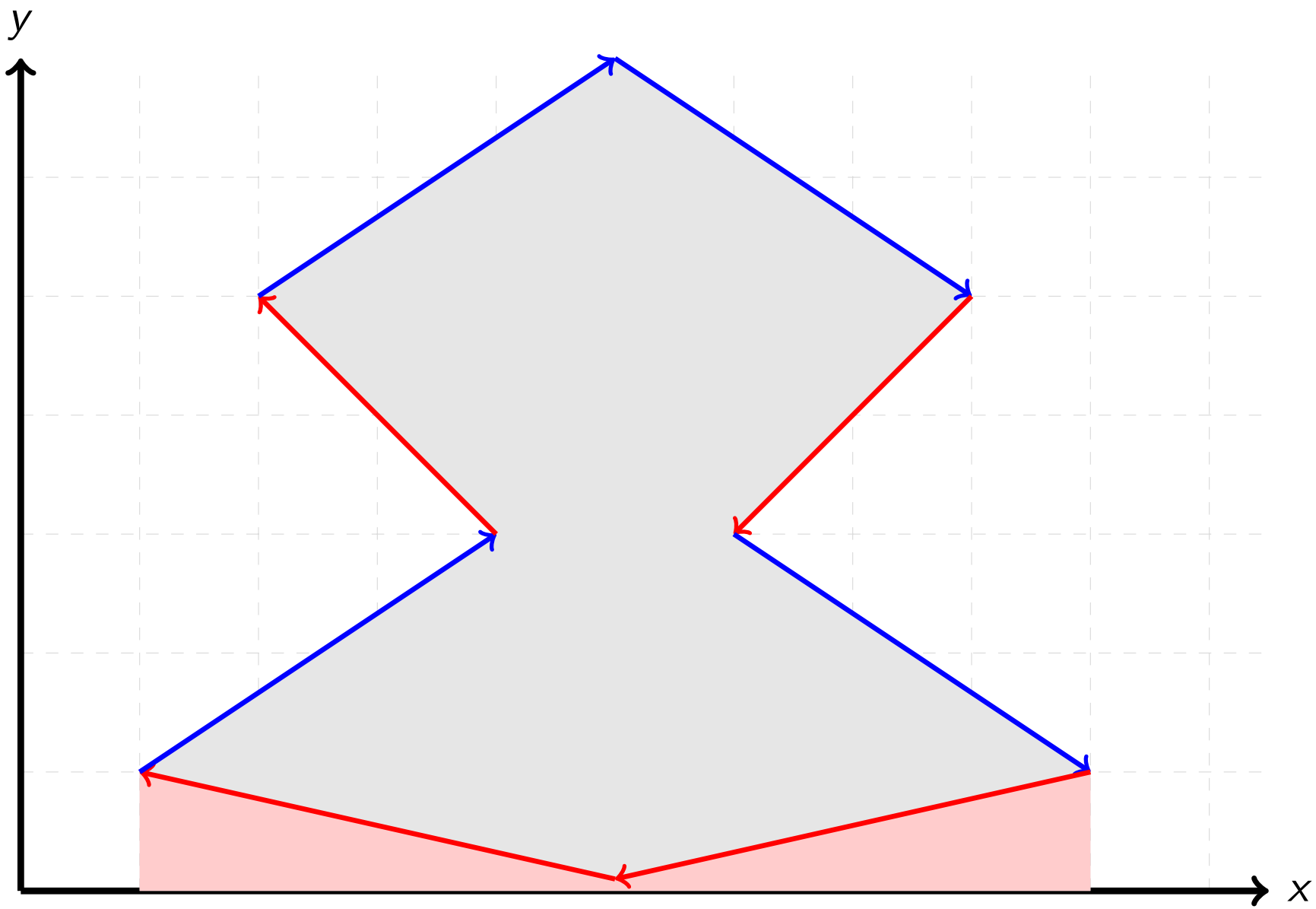
F – Icebergs



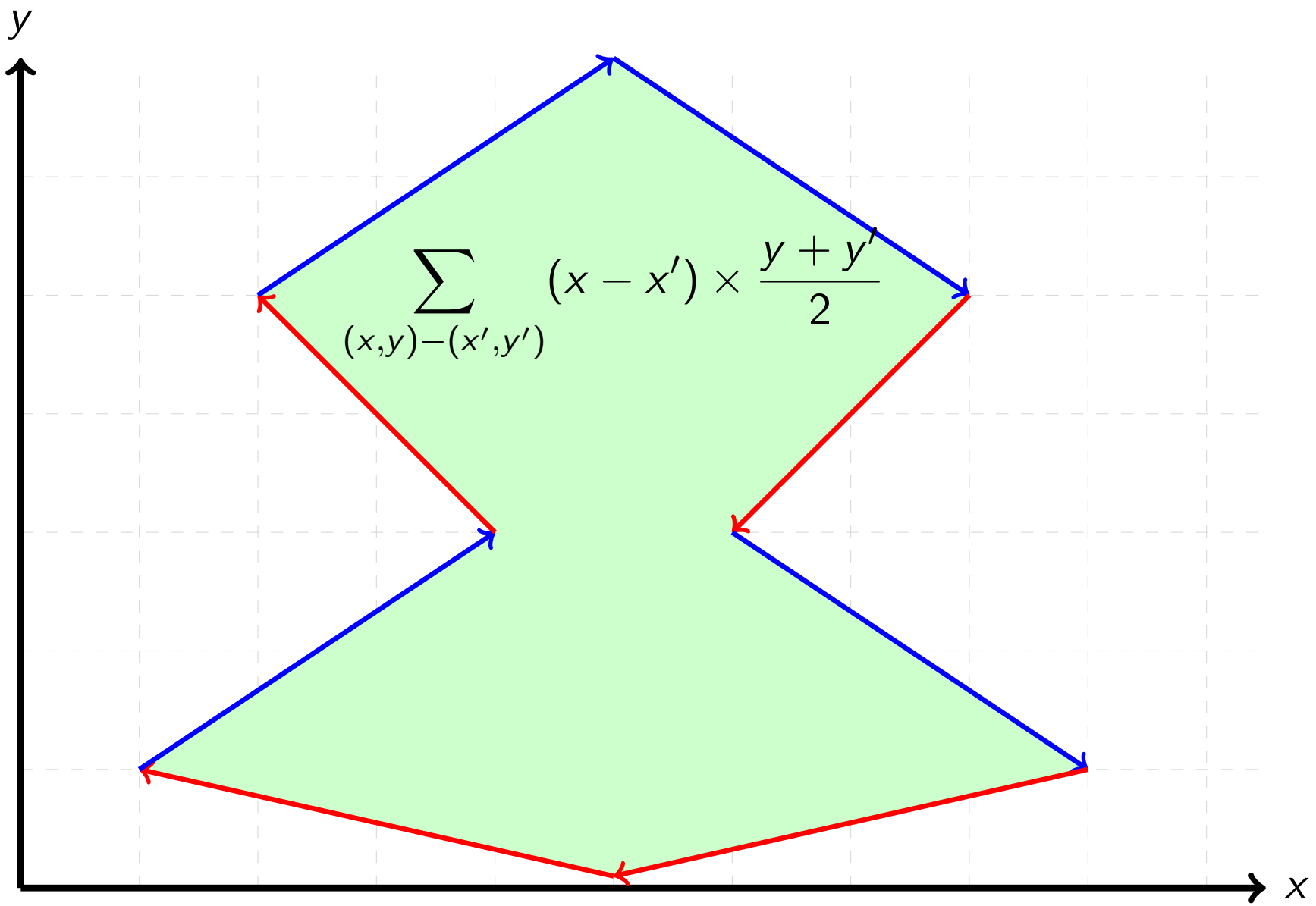
F – Icebergs



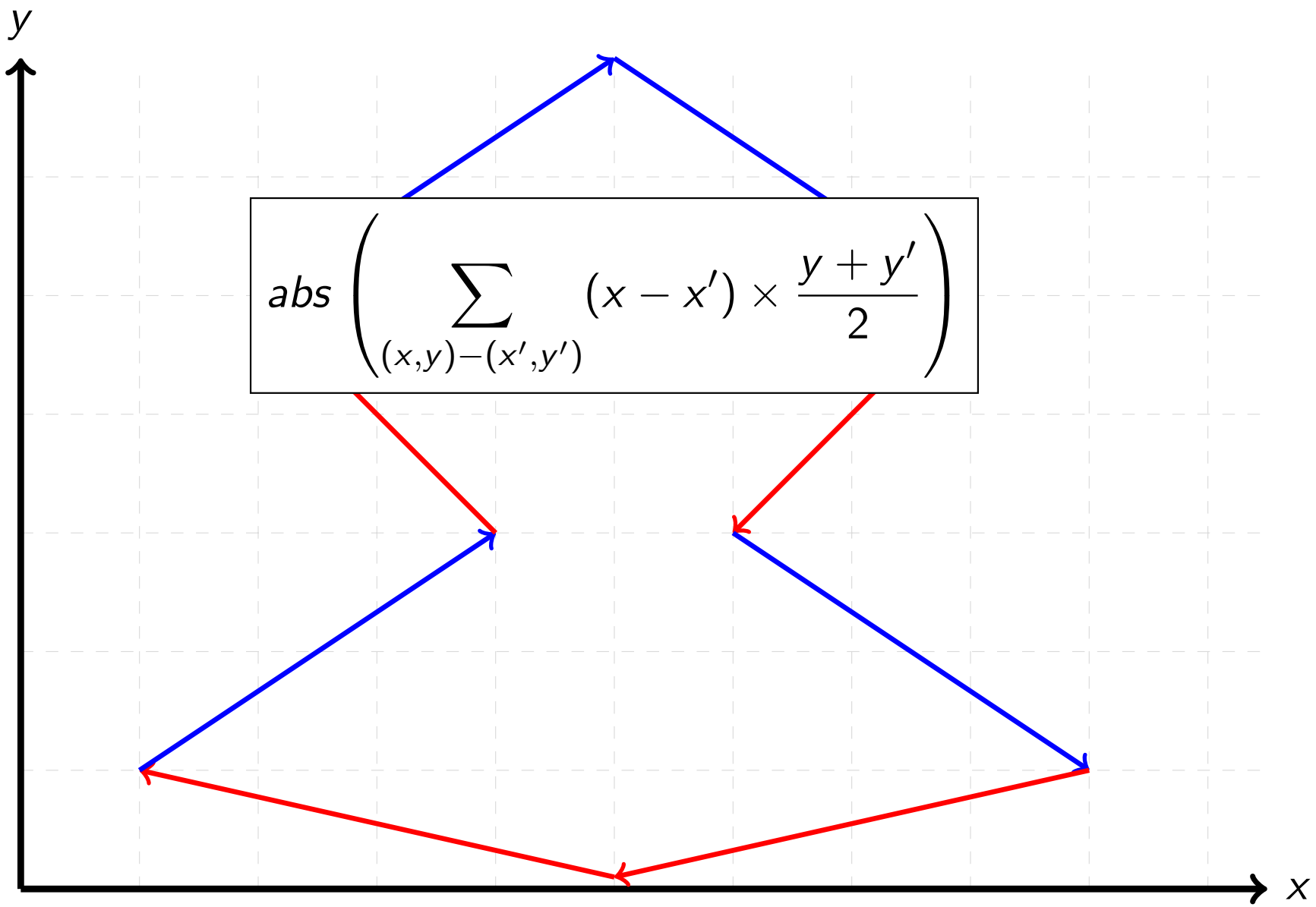
F – Icebergs



F – Icebergs

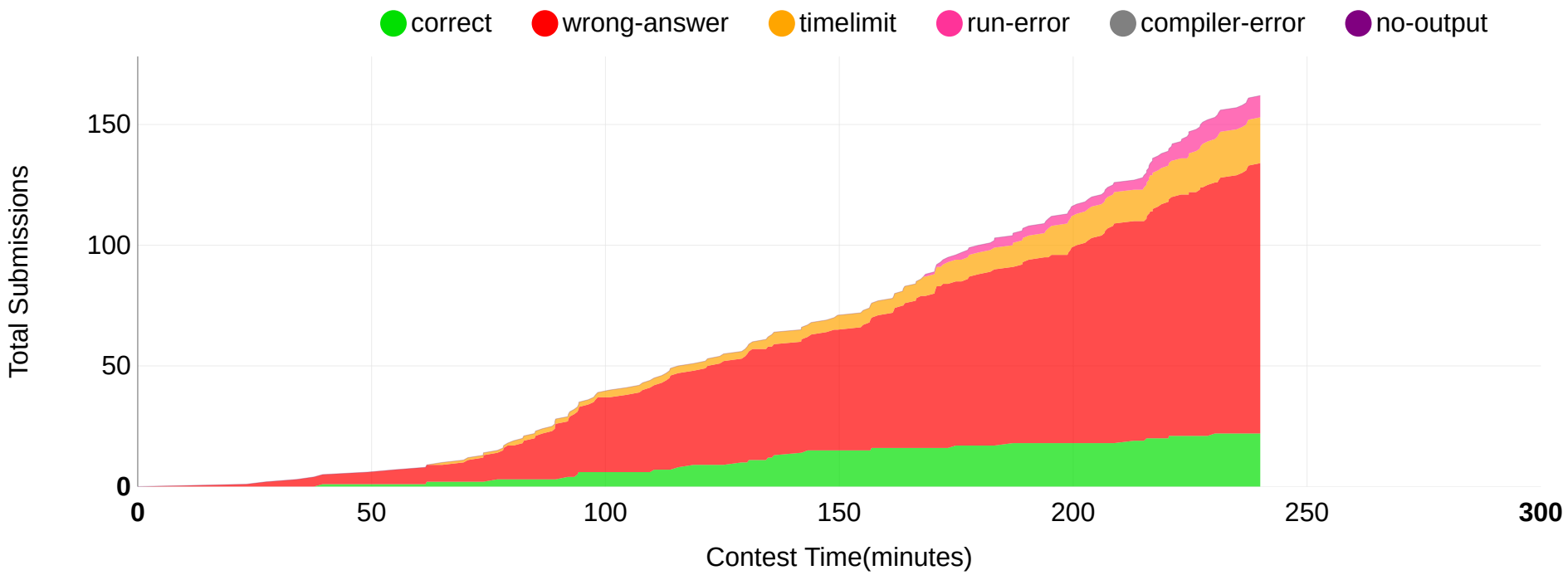


F – Icebergs



K – Bird Watching

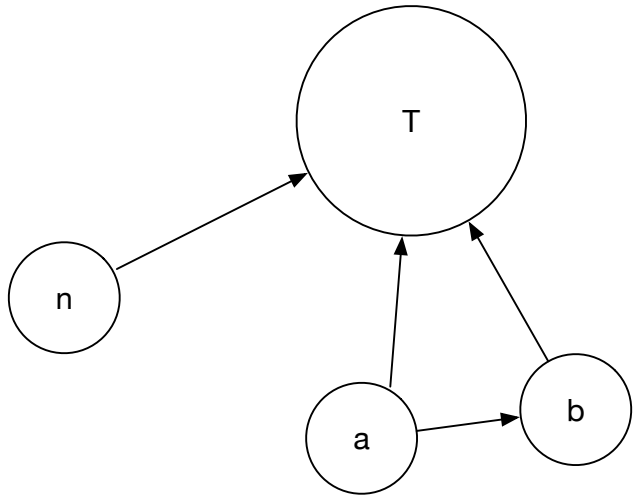
Solved by 22 teams before freeze.
First solved after 39 min by **UPC-1**.



K – Bird Watching

Problem

Given a vertex T in a directed graph $\mathcal{P}$, find all nodes n such that the edge (n, T) is the only path from n to T.



Naive approach

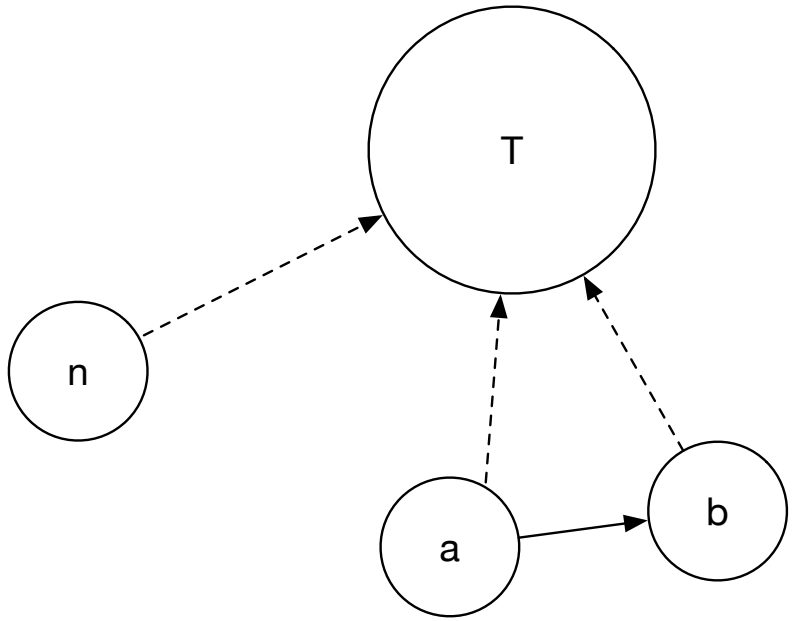
Remove (n, T) and check whether you can still reach T.
This requires $|V|$ DFSs, i.e., $|V| \times |E| \approx 10^{10}$ operations.
$\Rightarrow$ How do we cut the search?

K – Bird Watching

Auxiliary graph

$\mathcal{P}^*$: Remove all edges leading to T

n is a solution when there is no other node n' where the edge $n' \to T$ is in $\mathcal{P}$ and there is a path from n to n' in $\mathcal{P}^*$.

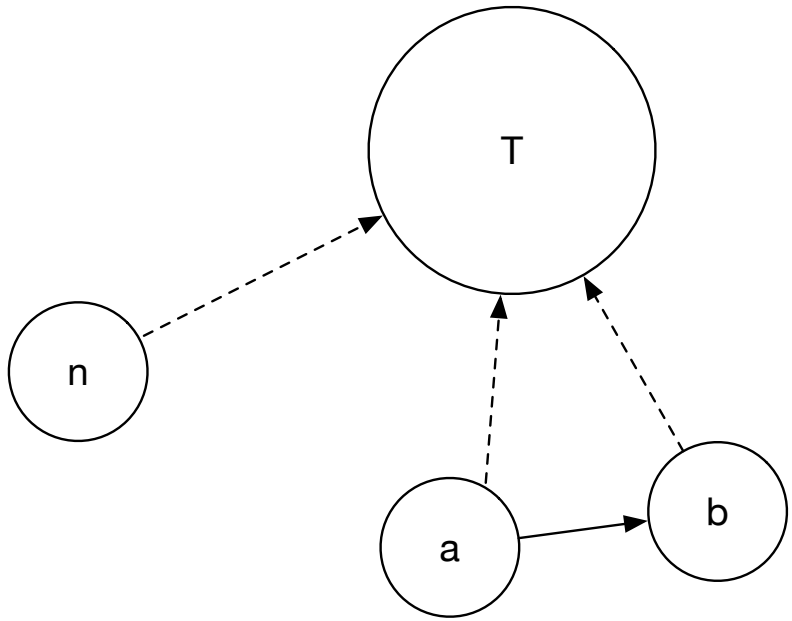


n is a solution, a is not $(b = n')$

K – Bird Watching

Simplified algorithm

For each n, find some n' satisfying the previous requirements and stop the search to cut branches.



K – Bird Watching

Simplified algorithm

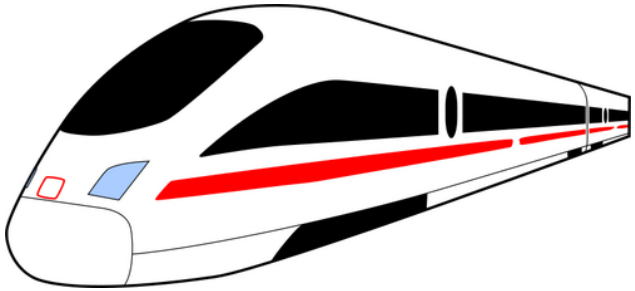
Call annotate(r, r) for each r predecessor of T:

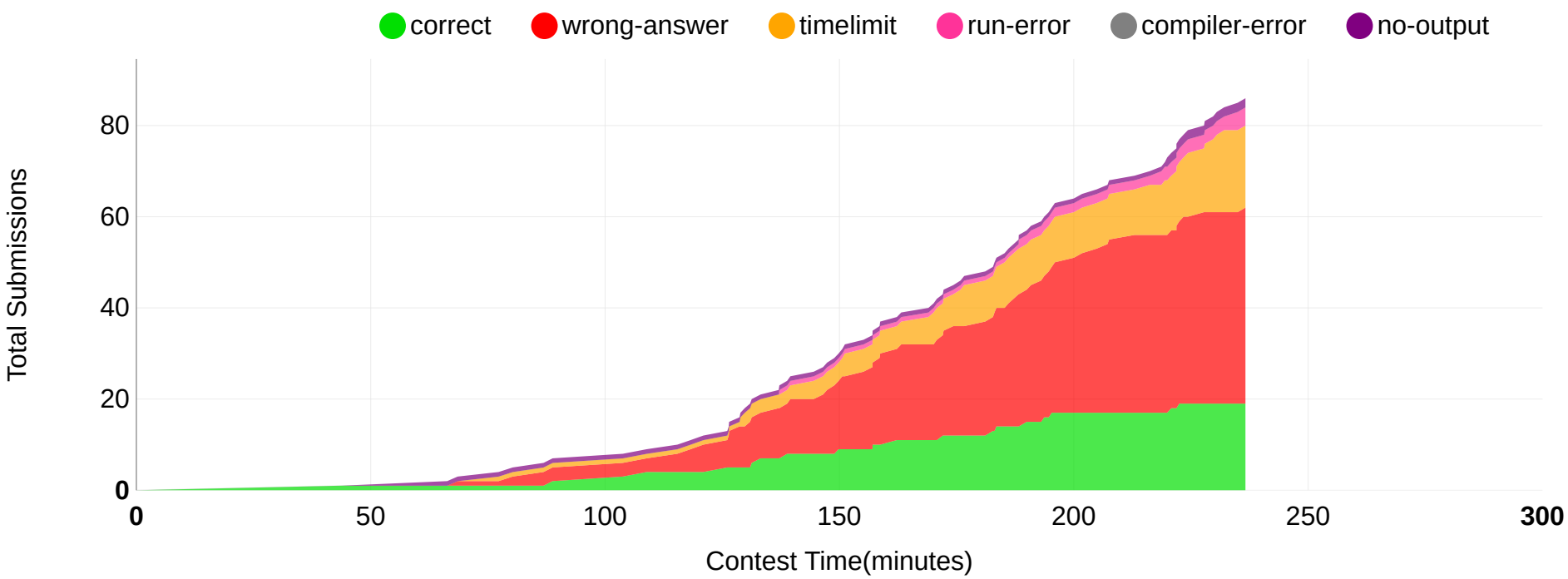
- goal(n) is a set of predecessors of T that are accessible from n in $\mathcal{P}^*$ (with at most 2 elements)
- a predecessor n of T is a solution iff $|\text{goal}(n)| = 1$ (contains only n).

```
annotate(n, r):
  if r ∈ goal(n): stop
  if |goal(n)| ⩾ 2: stop
  goal(n) ← goal(n) ∪ {r}
  for each (u, n) ∈ P*: annotate(u, r)
```

A – Environment-Friendly Travel

Solved by 19 teams before freeze.
First solved after 43 min by **UNIBOis**.

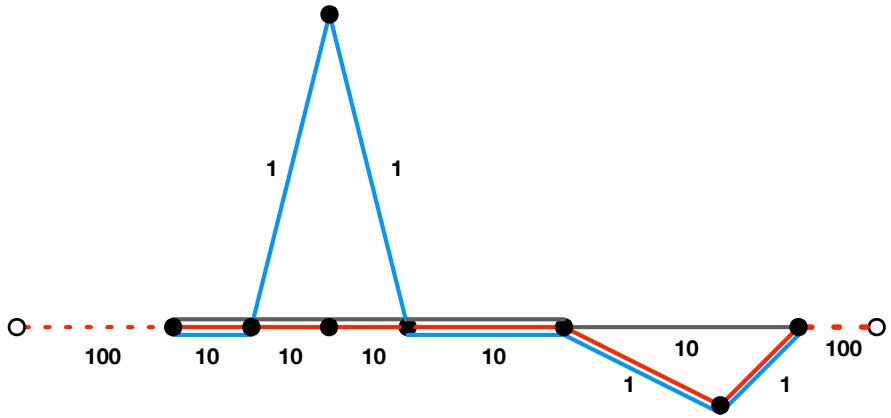




A – Environment-Friendly Travel

Problem

Given two distances d_1 (CO_2-cost), d_2 (Euclidean distance) on a graph G, find the smallest $d_1(s, t)$ such that $d_2(s, t) \leqslant B$.



E.g., for a budget of $B = 15$:

1. The **shortest path** (distance) costs too much CO_2.
2. The **cheapest path** is too long.
3. The best one is the **red** path.

A – Environment-Friendly Travel

Problem

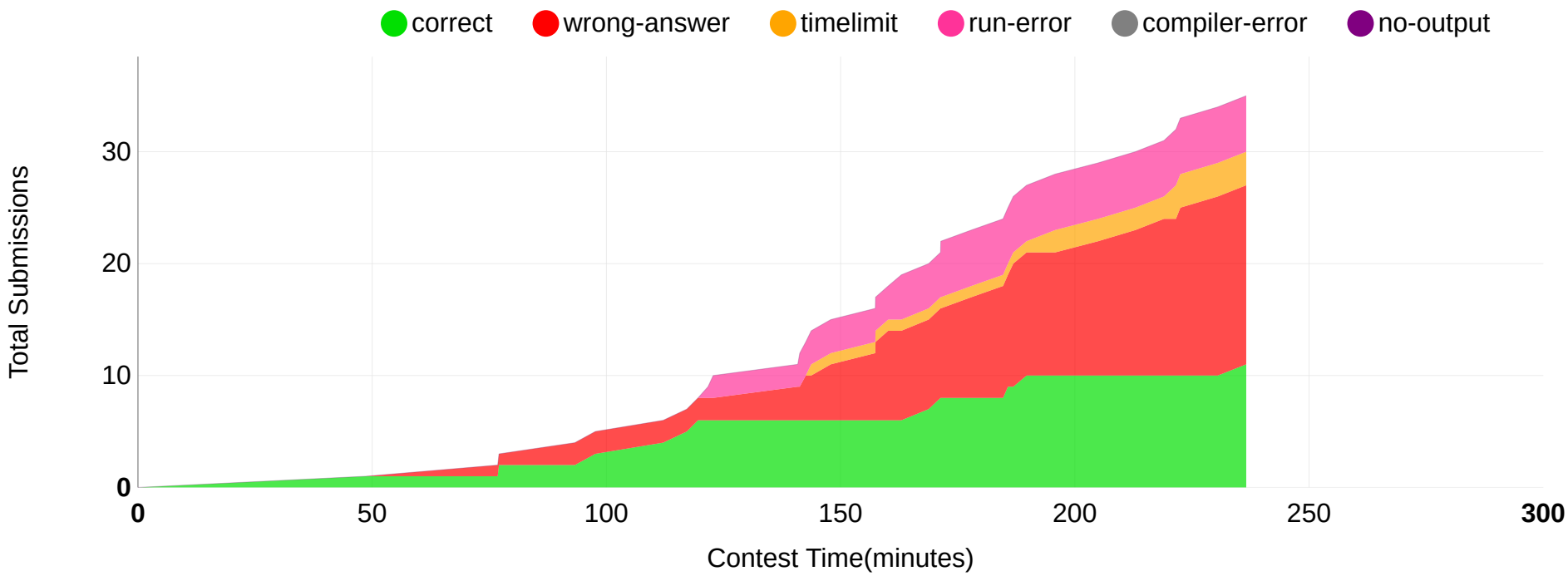
Given two distances d_1 (CO_2), d_2 (Euclidean distance) on a graph G, find the smallest $d_1(s, t)$ such that $d_2(s, t) \leqslant B$.

Solution

Run a shortest-path algorithm (Dijkstra) on the cost graph (d_1), keep only the paths for which $d_1 \leqslant B$.

J – Counting Trees

Solved by 11 teams before freeze.
First solved after 48 min by **ENS Ulm 1**.



J – Counting Trees

Problem: Cartesian trees

Count the number of integer-labelled binary trees which:

- have the min-heap property, and
- have a given integer sequence as their in-order traversal.

Basic DP solution (too slow)

How many trees for a given sub-sequence? Complexity: $\mathcal{O}(n^3)$.

J – Counting Trees

Choosing one of these trees:

$$2, 3, 1, 2, 2, 1, 1, 3, 2, 3$$

J – Counting Trees

Choosing one of these trees:

1. Locate the occurrences of the minimum of the sequence

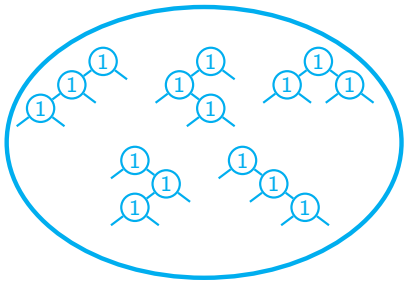
$$2, 3, 1, 2, 2, 1, 1, 3, 2, 3$$

J – Counting Trees

Choosing one of these trees:

1. Locate the occurrences of the minimum of the sequence
2. Choose an arrangement of these nodes at the top of the tree
   - Number of choices: Catalan number $C_n = \frac{1}{n+1}\binom{2n}{n}$

$$2, 3, 1, 2, 2, 1, 1, 3, 2, 3$$

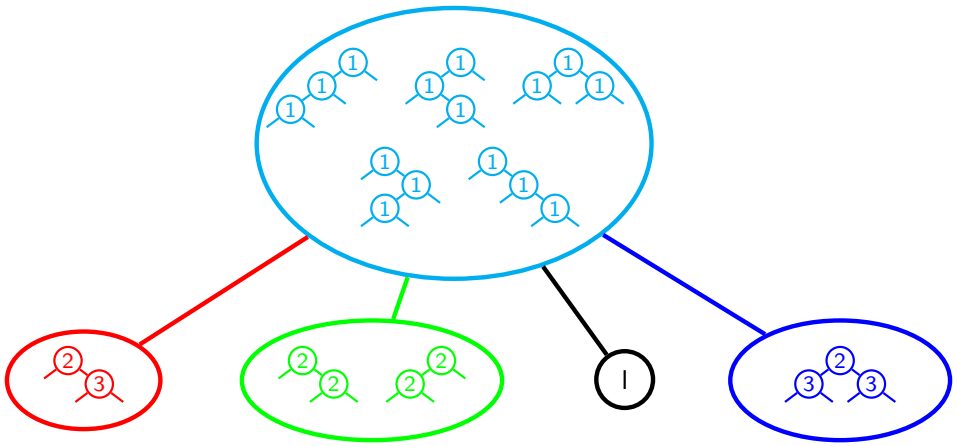


J – Counting Trees

Choosing one of these trees:

1. Locate the occurrences of the minimum of the sequence
2. Choose an arrangement of these nodes at the top of the tree
   - Number of choices: Catalan number $C_n = \frac{1}{n+1}\binom{2n}{n}$
3. Choose the sub-trees recursively, for each of the delimited sub-sequences.

$2, 3, 1, 2, 2, 1, 1, 3, 2, 3$



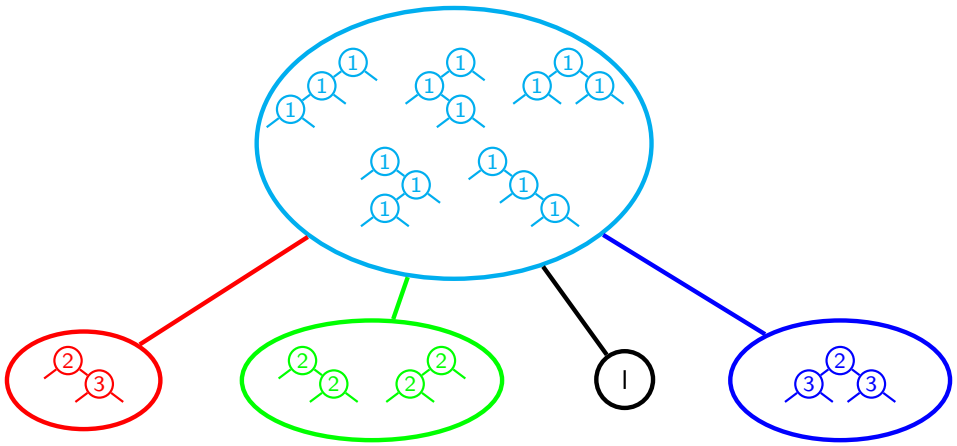
J – Counting Trees

Choosing one of these trees:

1. Locate the occurrences of the minimum of the sequence
2. Choose an arrangement of these nodes at the top of the tree
   - Number of choices: Catalan number $C_n = \frac{1}{n+1}\binom{2n}{n}$
3. Choose the sub-trees recursively, for each of the delimited sub-sequences.

Total complexity: $\mathcal{O}(n^2)$, or $\mathcal{O}(n \log n)$ with a min-range data structure.

$2, 3, 1, 2, 2, 1, 1, 3, 2, 3$



J – Counting Trees

Simpler algorithm

The result is a product of Catalan numbers.

Each factor C_n corresponds to a group of n elements of the sequence which:

- have the same value,
- is not separated by a smaller element.

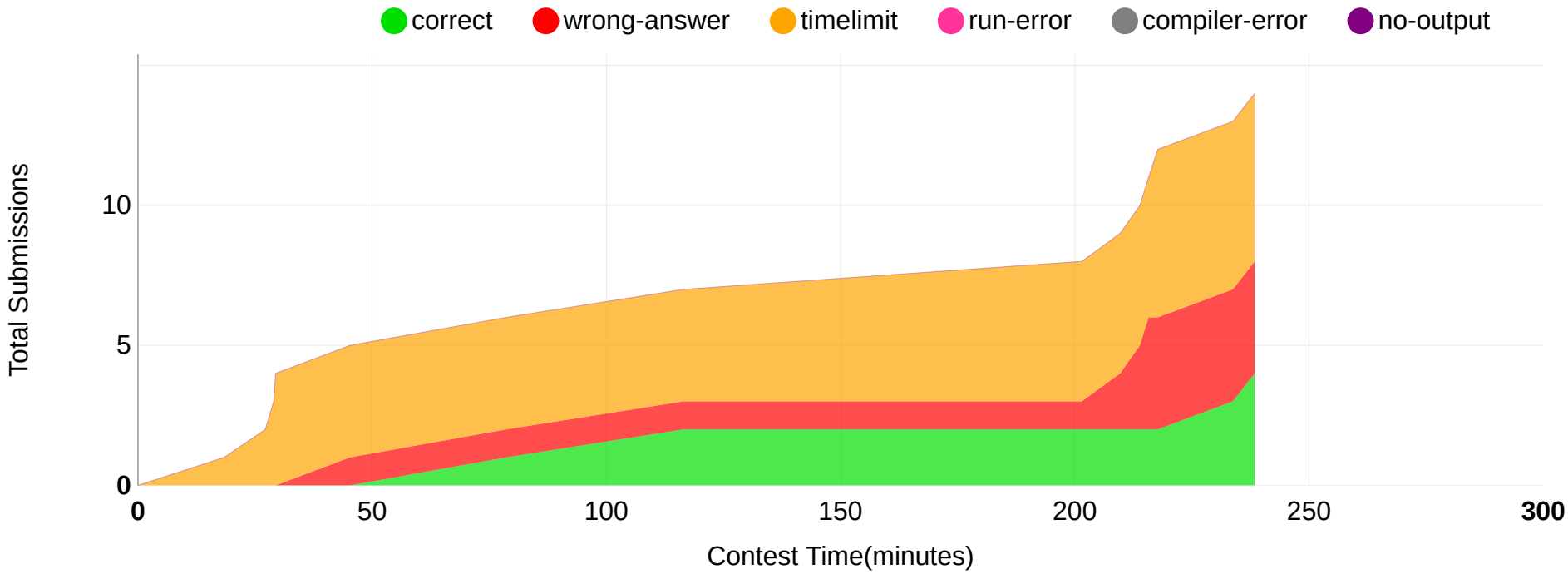
We can compute these groups using a stack in one pass on the sequence.

$\Rightarrow \mathcal{O}(n)$ algorithm

$\Rightarrow$ All included: 15 lines of simple Python code!

H – Pseudo-Random Number Generator

Solved by 4 teams before freeze.
First solved after 78 min by **LaStatale Blue**.



H – Pseudo-Random Number Generator

Problem

A pseudo-random number generator for 40-bit unsigned integers is defined as the iteration of a function f, that is,

$$\begin{aligned} S_0 &= \text{some given value,} \\ S_{i+1} &= f(S_i). \end{aligned}$$

Find the number of even values in the sequence $S_0, S_1, \ldots, S_{N-1}$.

Limits

The number N can be large (up to 2^{63}) so we cannot simply compute all the values.

H – Pseudo-Random Number Generator

Analysis

Since there are finitely-many values, the sequence S eventually cycles after a certain point: there exist $period \geq 1$ and $start \geq 0$ such that

$$S_{i+period} = S_i \quad \text{for } i \geq start.$$

Idea

Before submission,

- find *period* and *start*;
- pre-compute the number of even values for
  - the whole initial sequence $S_0, S_1, \ldots, S_{start-1}$,
  - the whole cycle $S_{start}, S_{start+1}, \ldots, S_{start+period-1}$,
  - blocks of consecutive S_i (e.g. 1000 blocks in total).

Submit a code that tests whether $N < start$ or $N = start + q \cdot period + r$ with $0 \leq r < period$ and uses the pre-computed values.

H – Pseudo-Random Number Generator

Cycle detection

We are left with the problem of finding *period* and *start*.
Storing all S_i until we find the cycle requires too much memory.

Solution

Use Floyd's *tortoise and hare* algorithm:

$t,\ h \leftarrow 0, 1$
while $S_t \neq S_h$ **do** $t,\ h \leftarrow t+1,\ h+2$
$i \leftarrow 0$
while $S_i \neq S_{t+i}$ **do** $i \leftarrow i+1$

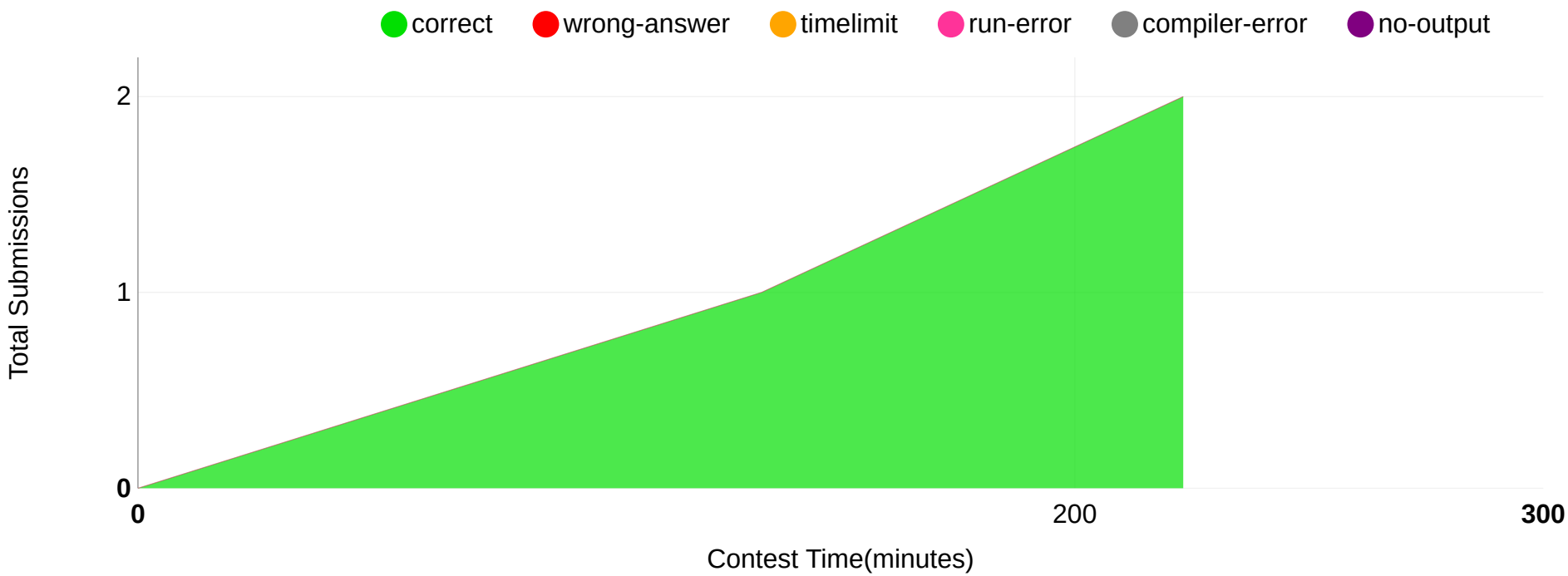
[See *The Art of Computer Programming*, volume 2, page 7, exercise 6.]

Efficiency

Precomputation: $\mathcal{O}(start + period)$.
(In our case, $period = 182\,129\,209$ and $start = 350\,125\,310$.)
Submitted solution is $\mathcal{O}(1)$.

L – River Game

Solved by 2 teams before freeze.
First solved after 133 min by **EP Chopper**.



L – River Game

Problem

Two players take turns to place cameras on a $N \times N$ grid with firm ground, wet zone and protected zone squares. Rivers are connected components of wet squares.

Rules:

- Cameras must be on firm ground, adjacent to a river.
- No two cameras on same square.
- No two cameras adjacent to the same river can be adjacent.

River properties:

- Contain at most $2N$ squares.
- Any two squares from two different rivers are at least 3 squares apart.

Who will win the game (assuming optimal play)?

Limits: $N \leq 10$

L – River Game

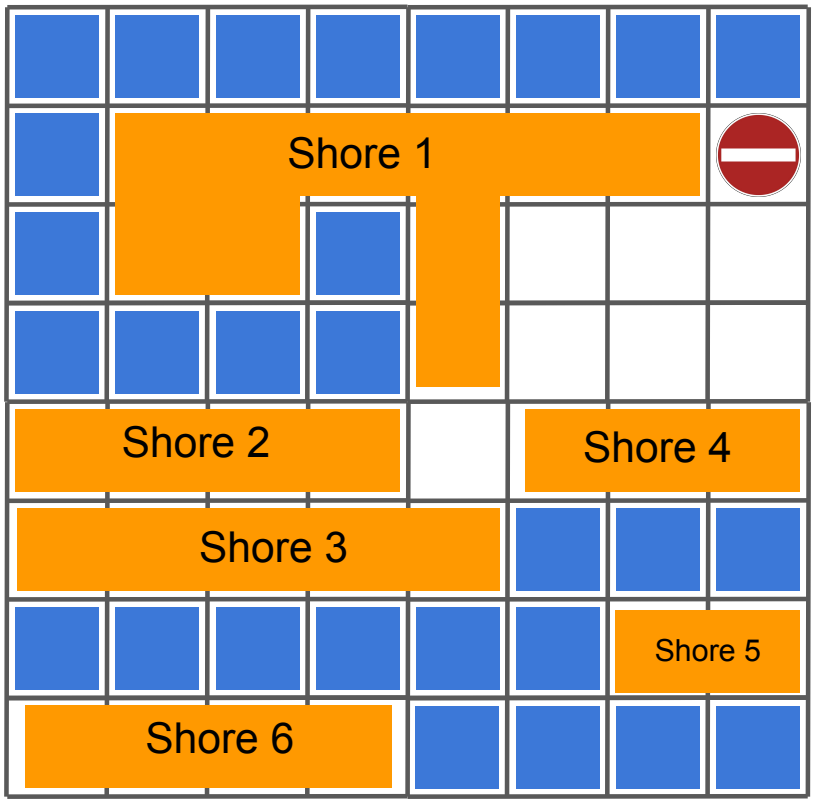
Brute force solution

- State is the $N \times N$ grid with already placed cameras marked.
- Complexity $\geq \mathcal{O}(2^{N \times N})$. Too slow.

L – River Game

Faster solution: Key idea

- **Shore**: connected component of firm ground squares adjacent to a given river.
- Cameras on one shore don't affect cameras on other shores!



L – River Game

Faster solution

- Decompose the game into K independent games (K = number of shores, $\leq N^2$ and in practice much less).
- Compute the Grundy number G_i of each shore. Computed in $\mathcal{O}(S \times 2^S)$ where S = maximum shore size.
- Position is losing iff $G_0 \oplus \ldots \oplus G_K = 0$.
- $S \leq 3N + o(3N)$. For $N = 10$ bound is tighter: $S \leq 20$.
- For $N = 10$, takes less than 100×2^{20} operations.

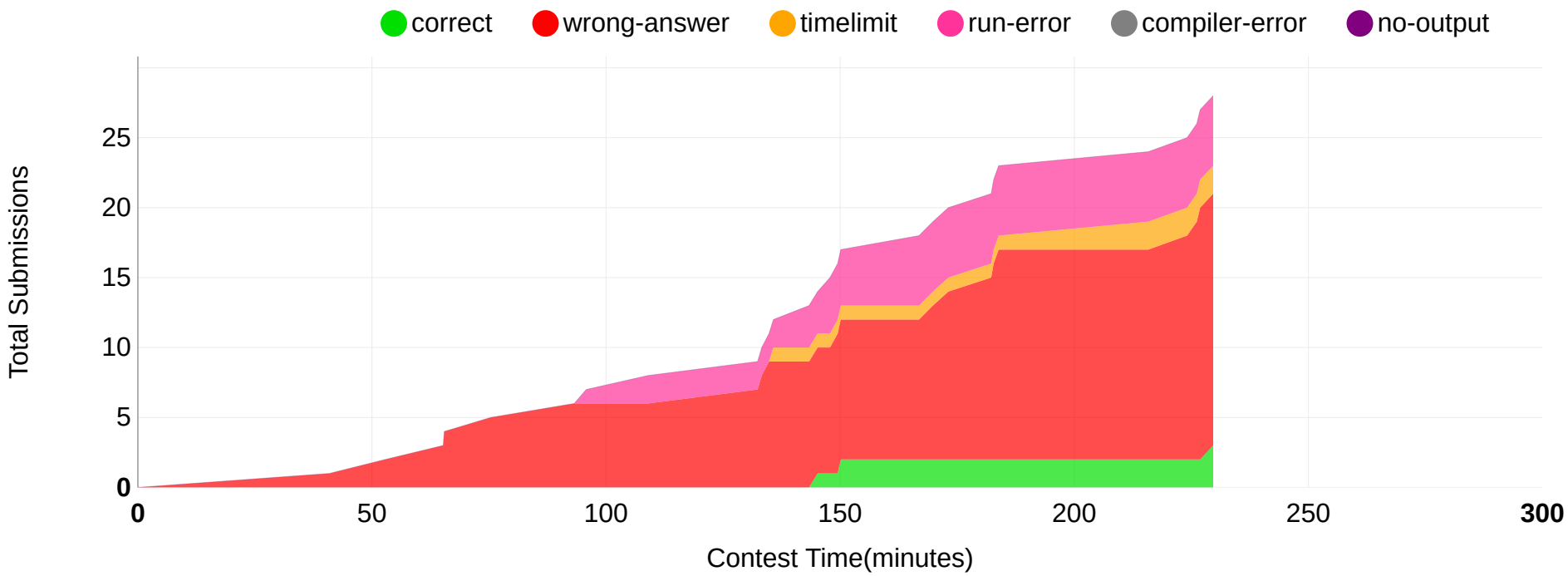
L – River Game

Grundy Numbers computation

```
def GrundyNumber(state):
    next_states = list of possible next states after
                  adding a camera
    next_grundy = set()
    for s in next_states:
        next_grundy.add(GrundyNumber(s))
    # Compute smallest non-negative integer not in
    # next_grundy (problem Ants!).
    res = 0
    while res in next_grundy: res += 1
    return res
```

G – Swapping Places

Solved by 3 teams before freeze.
First solved after 145 min by **UPC-1**.



G – Swapping Places

Problem

Given a word w on an alphabet A and a set $S \subseteq A^2$ of pairs of letters that commute with each other, find the smallest word $\overline{w}$ equivalent to w.

Limits

- A is small: $|A| \leqslant 200$;
- w can be long: $|w| \leqslant 100\,000$.

We can work in time $\mathcal{O}(|A|^2\,|w|)$ but not $\Omega(|w|^2)$.

G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$

w:	3	4	1	2	3	1	$\overline{w}$:
w_1:							
w_2:							
w_3:							
w_4:							

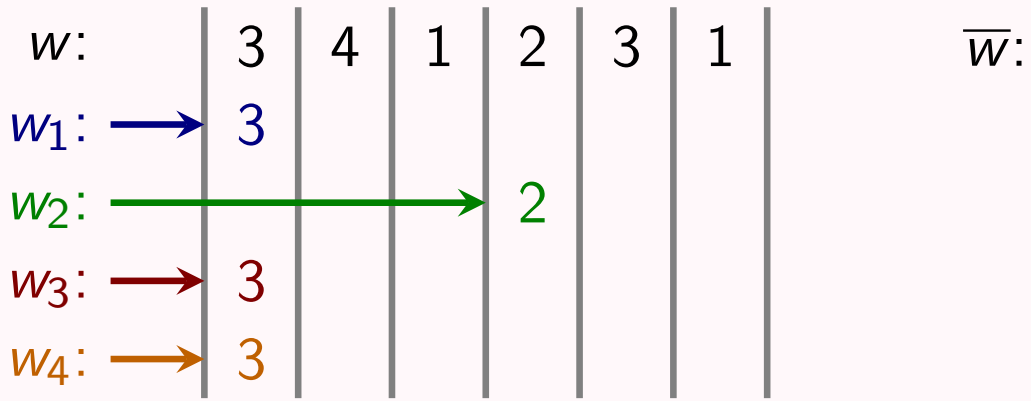
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



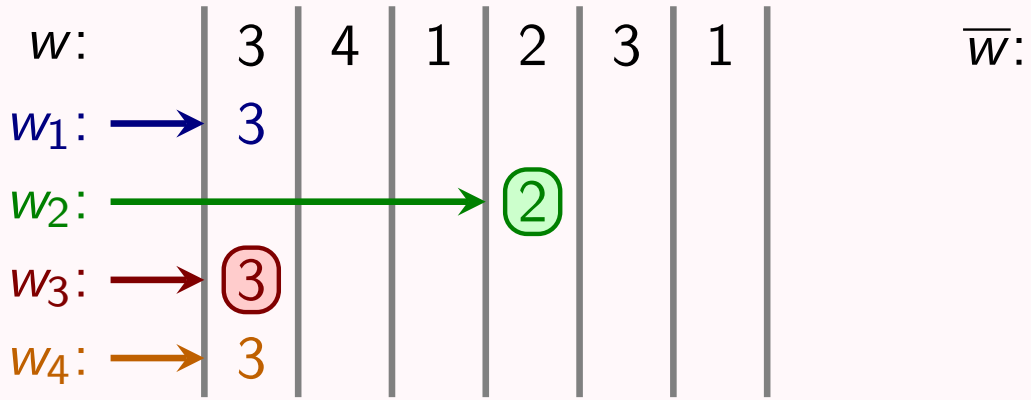
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



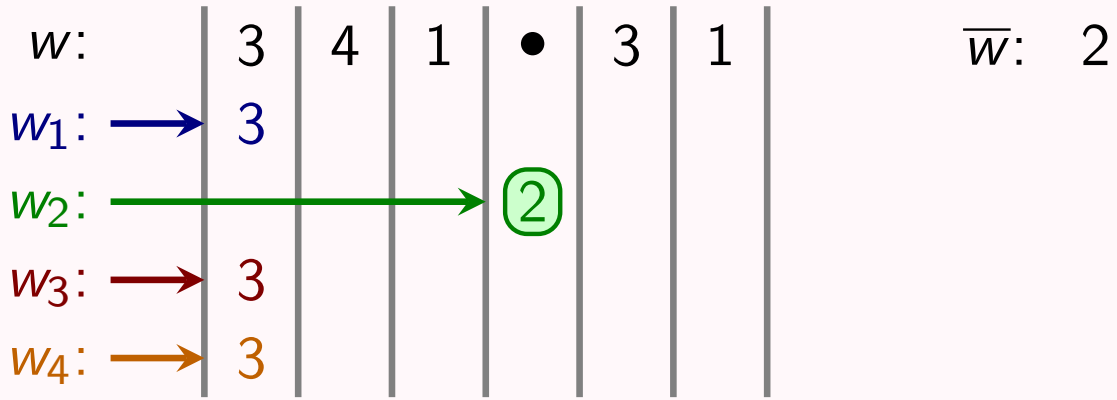
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



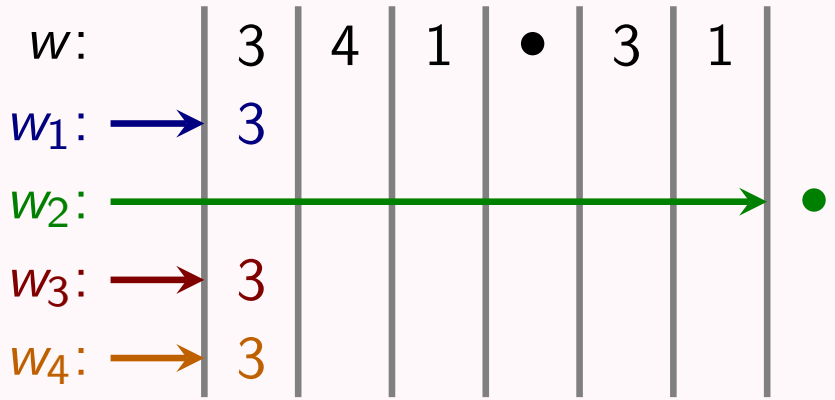
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



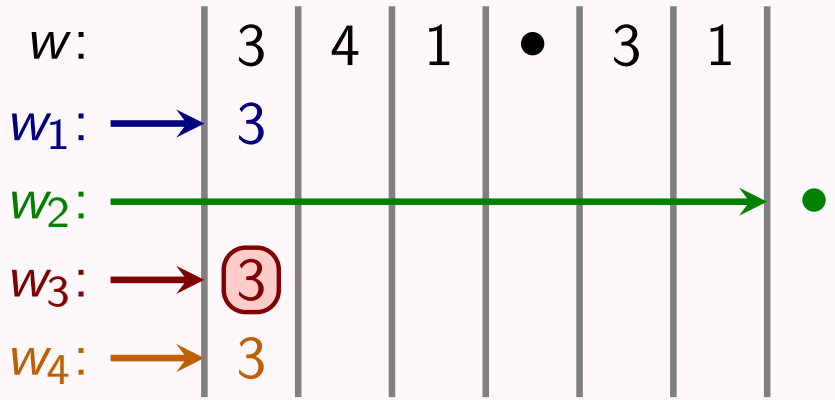
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



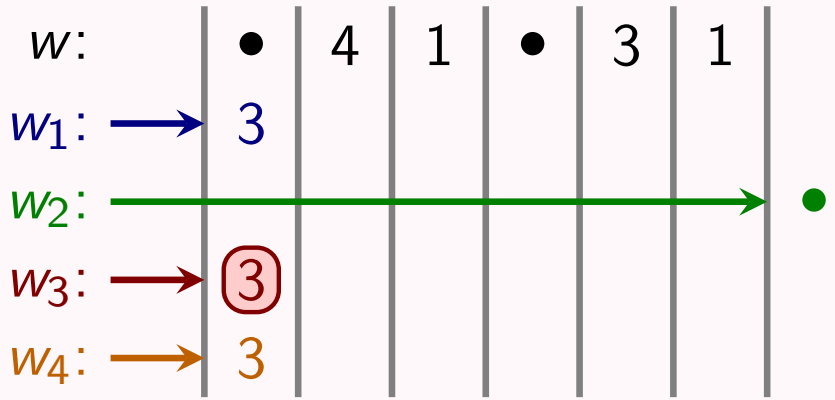
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



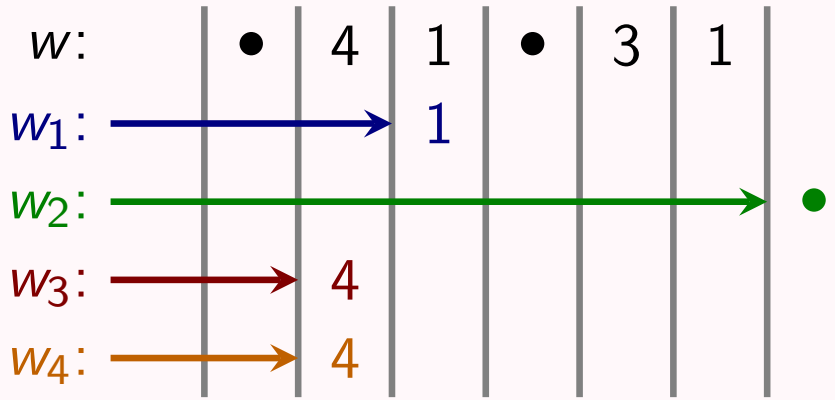
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



$\overline{w}$: 2 3

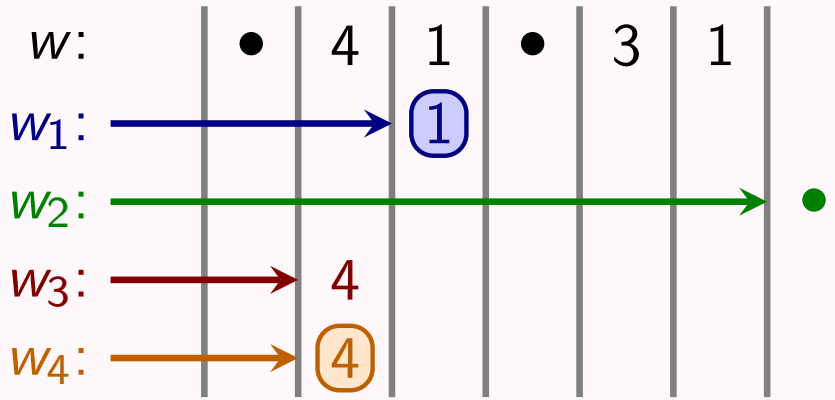
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



$\overline{w}$: 2 3

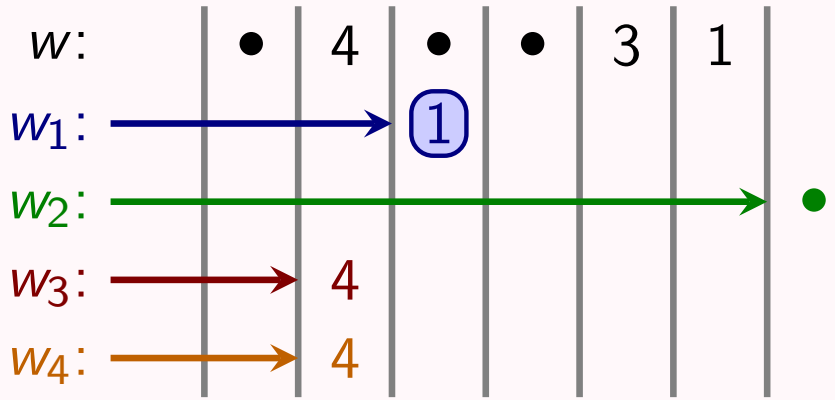
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



$\overline{w}$: 2 3 1

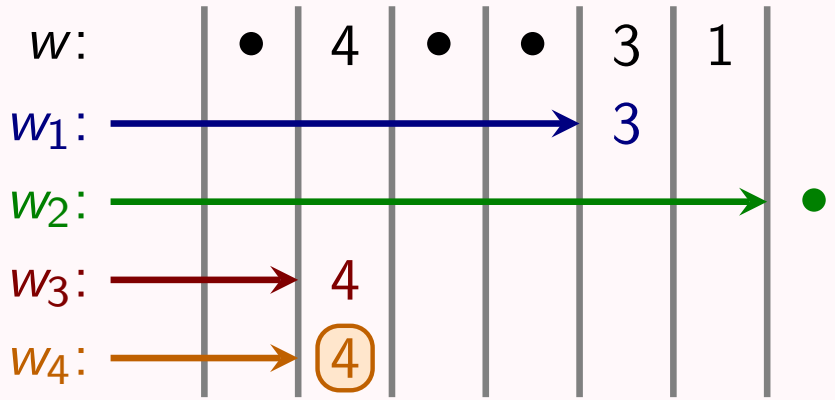
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



$\overline{w}$: 2 3 1

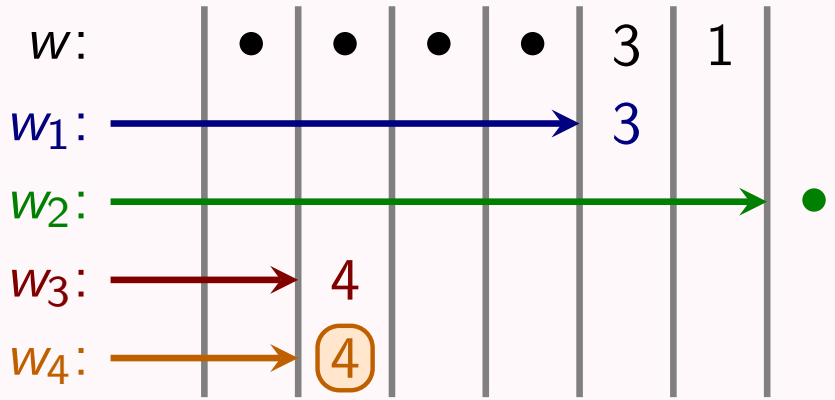
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



$\overline{w}$: 2 3 1 4

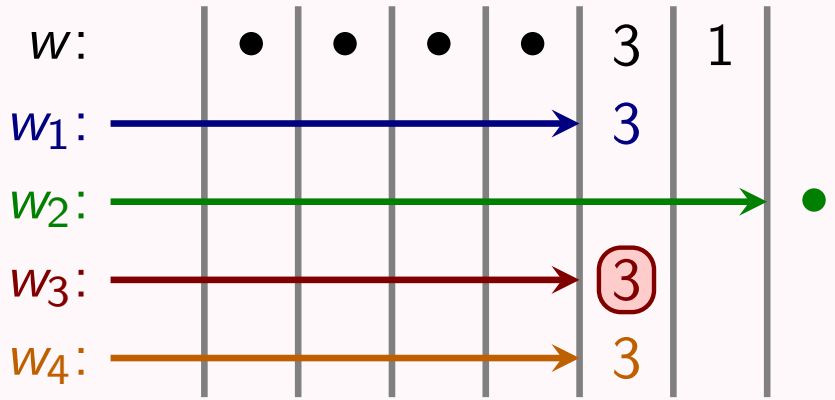
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



$\overline{w}$: 2 3 1 4

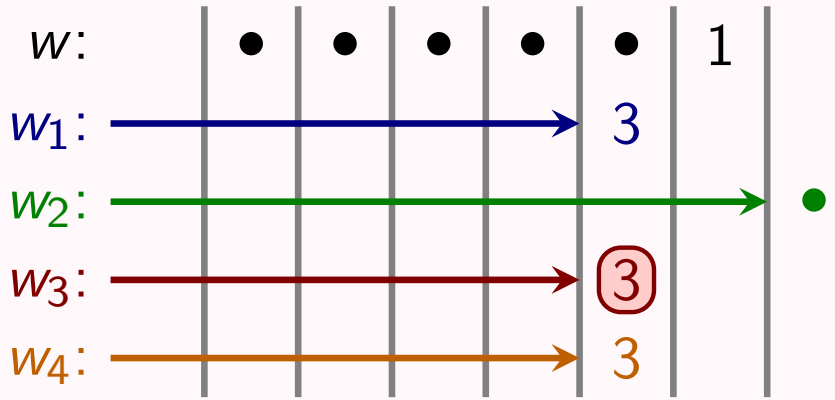
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



$\overline{w}$: 2 3 1 4 3

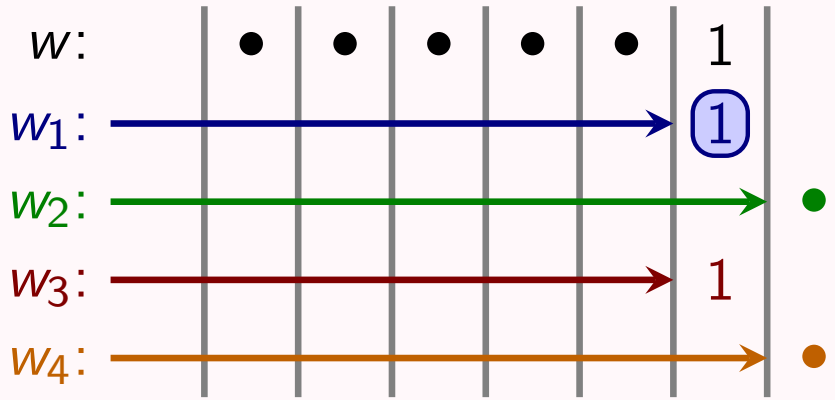
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



$\overline{w}$: 2 3 1 4 3

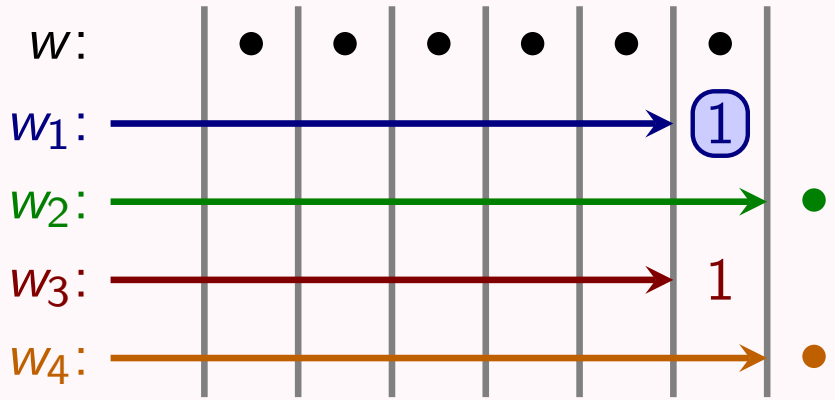
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



$\overline{w}$: 2 3 1 4 3 1

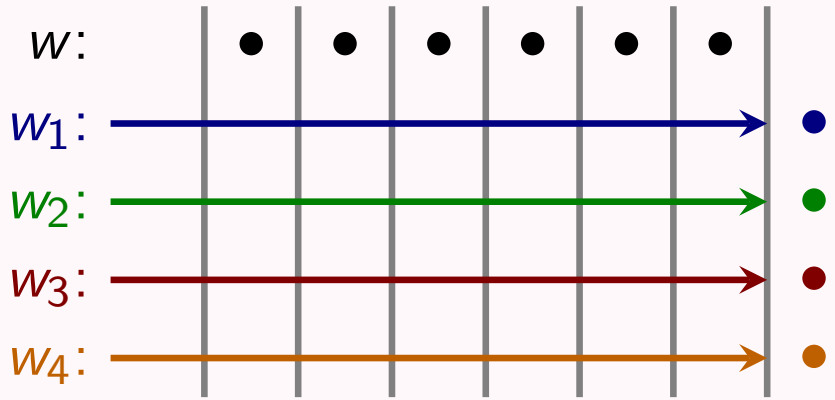
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$



$\overline{w}$: 2 3 1 4 3 1

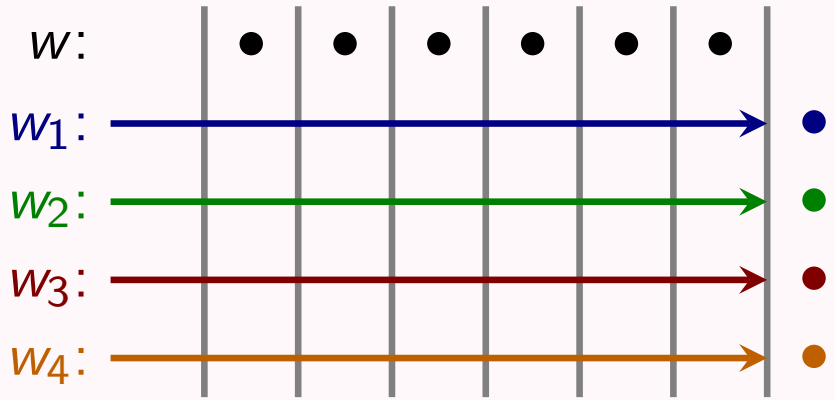
G – Swapping Places

Idea

Find the letters of $\overline{w}$ one by one, from left to right:

- For each letter λ, find the longest prefix w_λ of w that commutes with λ and does not contain λ.
- The first letter of $\overline{w}$ is the smallest μ such that $w_\mu\mu$ is a prefix of w.
- Erase the leftmost occurrence of μ in w and update all prefixes w_λ.

Example: $w = 341231$, with $12 = 21$, $14 = 41$, $23 = 32$ and $24 = 42$

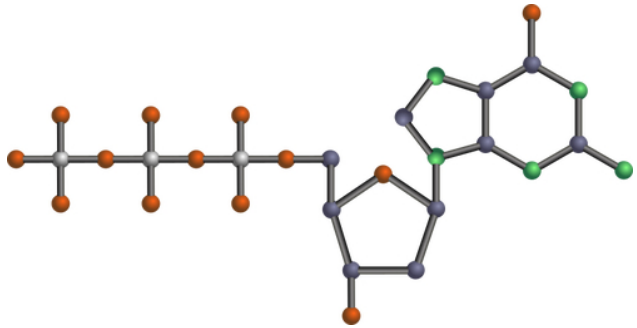


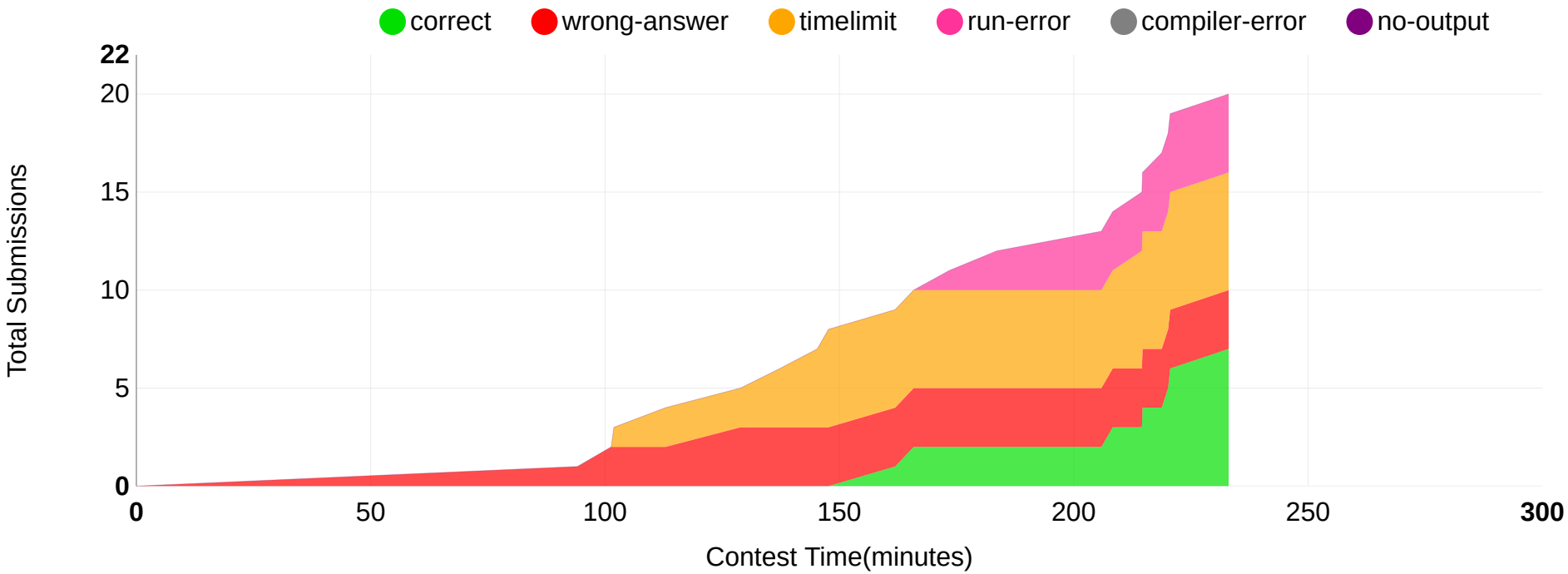
$\overline{w}$: 2 3 1 4 3 1

Time complexity: $\mathcal{O}(|A|\,|w|)$

D – Gnalcats

Solved by 8 teams before freeze.
First solved after 161 min by **mETH**.





D – Gnalcats

Problem

Stack language, inspired by Tezos' smart contract language Michelson.

Programs work on an infinite stack of values.

Values are either a pair of values or a non-pair.

Prove the equivalence of two programs on input stacks of non-pair values.

Instructions

COPY	Copy top value (`DUP`)
DROP	Drop top value (`DROP`)
SWAP	Swap top two values (`SWAP`)
PAIR	Construct pair from top two values (`PAIR`)
UNPAIR	Destruct top pair (`UNPAIR`), **FAIL** on non-pair values
LEFT	Replace top pair by its left component (`CAR`) $\equiv$ **USD**
RIGHT	Replace top pair by its right component (`CDR`) $\equiv$ **UD**

D – Gnalcats

Solution

Symbolic evaluation

- give a unique identifier to elements of the input stack
  (first $10^5 + 2$ elements are enough)
- evaluate both programs on this symbolic input stack
  (linear in program size)
- compare symbolic output stacks
  (linear in output stack overall sizes)

But...

Values can grow exponentially!

E.g. Pair Copy Pair Copy Pair Copy ...

D – Gnalcats

But...

Values can grow exponentially! E.g. Pair Copy Pair Copy ...

Solution

Hash-consing

- give the same identifier to all pairs constructed from the same elements
- use a hash table $\langle left\_id, right\_id \rangle \to pair\_id$

Complexity of comparison becomes linear in the size of stacks ($\leq 10^5$).

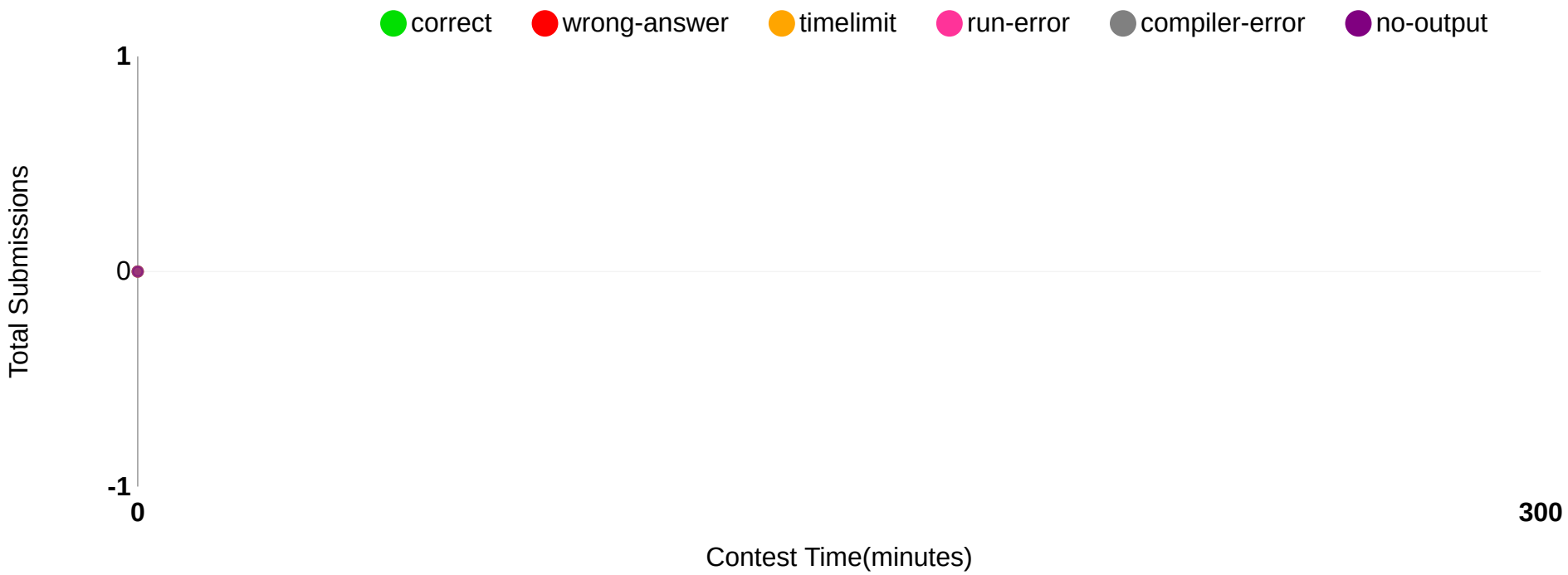
Even better (not necessary here)

- Represent stacks as pairs $\langle top, rest \rangle$
- Allows comparison in $\mathcal{O}(1)$

Or worse: use congruence-closure with a union-find

E – Pixels

Not solved before freeze.



E – Pixels

Problem

You are given:

- a grid g of black/white pixels: all pixels are white at start;
- a family of controllers: pressing c switches the pixels in a set S_c;
- a target grid t.

Can you draw the grid t? If yes, by pressing which controllers?

Limits

- g and t can be long: $|g| = |t| = KL \leqslant 100\,000$;
- for each controller, $|S_c| \leqslant 5$;
- controllers are arranged along a grid: sets S_c are **very** regular.

We can work in time $\mathcal{O}(\min\{K, L\}KL) \leqslant \mathcal{O}((KL)^{3/2})$ but not $\Omega((KL)^2)$.

E – Pixels

Idea: Reduce the problem to solving a linear equation in $\mathbb{F}_2^{KL}$

- One pixel = one element of $\mathbb{F}_2$
- Grids v and t = vectors in $\mathbb{F}_2^{KL}$
- Family of sets S_c = sparse $(KL) \times (KL)$ matrix M
- Pressing a set C of controllers = Obtaining the vector $M \cdot C$

Solution

Use Gaussian elimination, starting from the controllers associated with top-left pixels, and find a C such that $M \cdot C = t$ (if any).

Time complexity: $\mathcal{O}(\min\{K, L\}KL)$