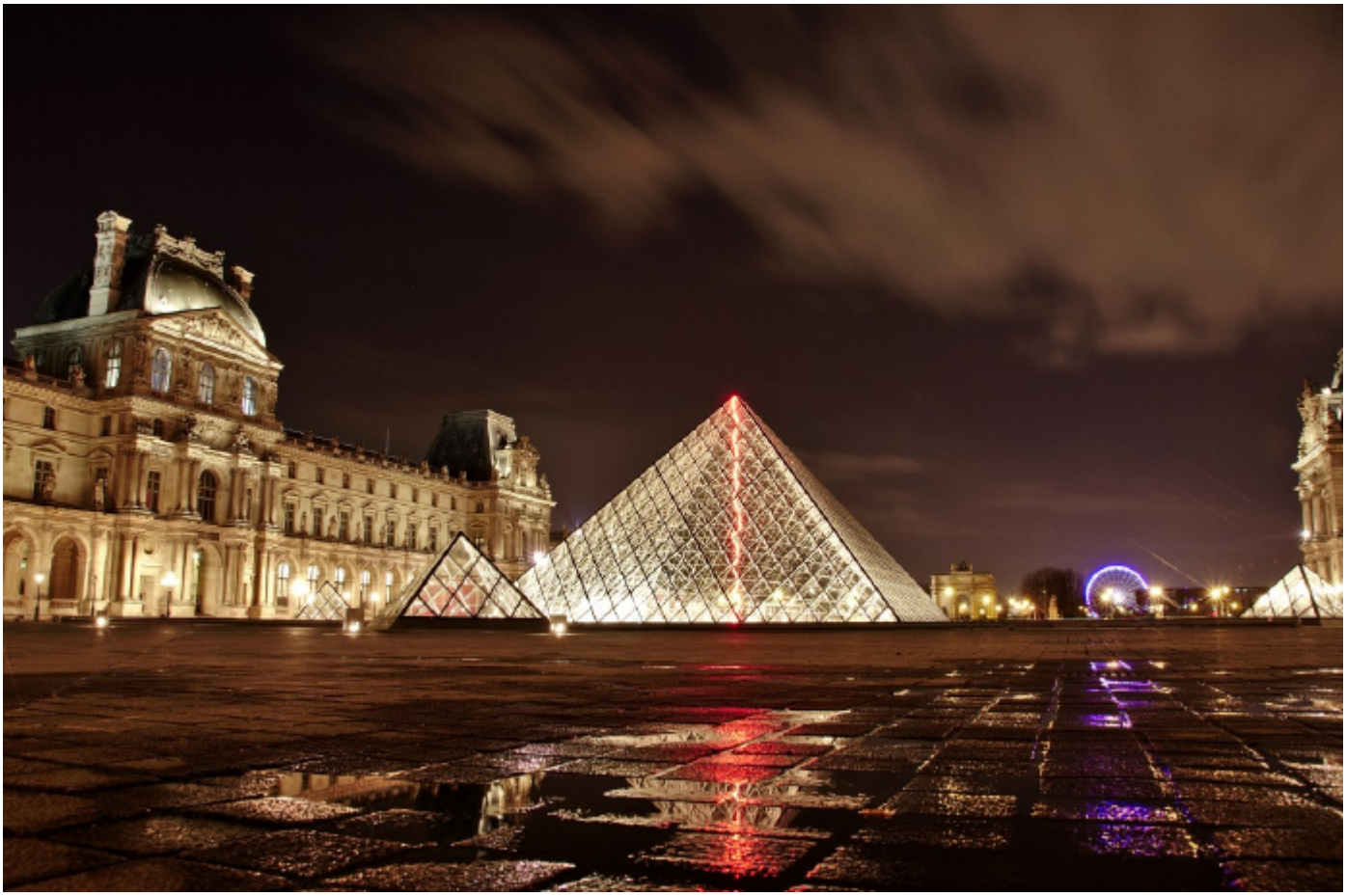
Problem A. City of Lights

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes



Paris has been called "ville lumière" (city of lights) since the 17th century. It earned this nickname in part because of the many city lights illuminating famous sites such as monuments, statues, churches, or fountains.

Those public lights in Paris are numbered from 1 to N and are all on by default. A group of hackers has gained the capability to toggle groups of lights. Every time the hackers use their program, they cause a number i (that they cannot control) to be sent to the system controlling the city lights. The lights numbered i, $2i$, $3i$, and so on (up to N) then change state instantly: lights that were on go off, and lights that were off go on.

During the night, the hackers use their programs k times. What is the greatest number of lights that are simultaneously off at the same time?

Input

The input comprises several lines, each consisting of a single integer:

- The first line contains the number N of lights. $1 \leq N \leq 1000000$.
- The second line contains the number k of uses hackers's program. $1 \leq k \leq 100$.
- The next k lines contain a number i sent to the system controlling the lights. $1 \leq i \leq N$.

Output

The output should consist of a single line, whose content is an integer, the greatest number of lights that are simultaneously off at the same time.

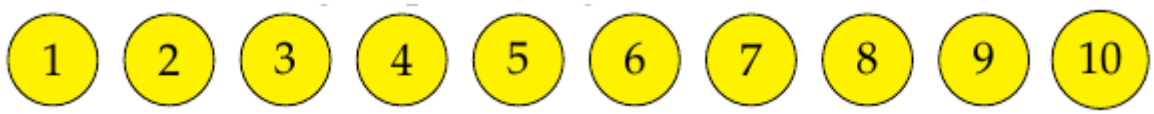
Example

standard input	standard output
10 4 6 2 1 3	6

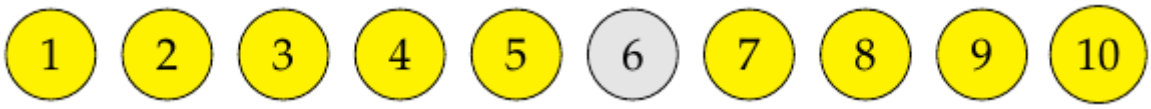
Note

Sample Explanation:

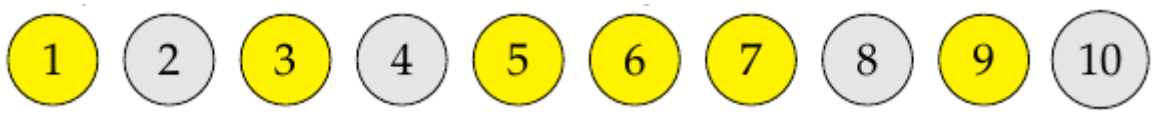
We start with a group of 10 lights which are all on.



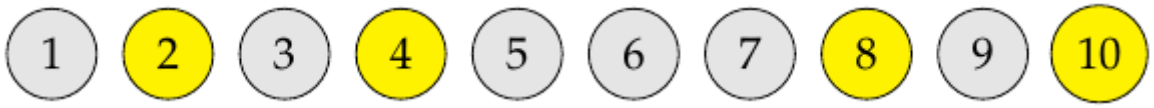
The hackers send the number 6: light 6 is toggled.



They then send the number 2: lights 2, 4, 6, 8, and 10 are toggled.



The number 1 is then sent: all lights are toggled.



They end up sending the number 3: lights 3, 6, and 9 are toggled.

The maximum number of lights off at the same time was 6.

Problem B. Blurred Pictures

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Damon loves to take photos of the places he visits during his travels, to put them into frames. All of his photos are in a square format of $N \times N$ pixels. He brought back beautiful pictures of the many monuments in Paris, such as the Eiffel Tower or the Louvre, but unfortunately, when he got back home, he realized that all his pictures were blurred on the edges. Looking closely, Damon realizes that he can easily distinguish the blurred pixels from the "good" (i.e., non-blurred) ones and that, luckily, all the non-blurred pixels are connected in such a way that any horizontal or vertical line drawn between two non-blurred pixels goes only through non-blurred pixels. In order to get the best from his failed pictures, he decides to cut out the biggest possible picture without any blurred pixel from each of his photos. And since his frames are all squares, for aesthetic reasons, the cut-out pictures have to be squares too. Damon does not want his pictures to be tilted so he wants the sides of the cut-outs to be parallel to the sides of the original picture.

Input

The input comprises several lines, each consisting of integers separated with single spaces:

- The first line contains the length N, in pixels, of the input photo;
- Each of the next N lines contains two integers a_i and b_i , the indices of the first (a_i) and the last (b_i) non-blurred pixel on the i-th line.
- $0 < N \leq 100000$,
- $0 \leq a_i \leq b_i < N$.

Output

The output should consist of a single line, whose content is an integer, the length of the largest square composed of non-blurred pixels inside the picture.

Examples

standard input	standard output
3 1 1 0 2 1 1	1
8 2 4 2 4 1 4 0 7 0 3 1 2 1 2 1 1	3

Note

- In the input picture, each row and each column has at least one non-blurred pixel.
- In any two consecutive lines, there are at least two non-blurred pixels in the same column.

Problem C. Crosswords

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

In order to attract more tourists in Paris, Anne wants to organize a huge sound and light show at night on the facades of the Louvre Palace. The facades are showing windows on several levels, and the windows are also vertically aligned. So Anne has the idea of displaying giant letters through the windows, one per window, so that it forms words both horizontally and vertically.

On a given facade, windows are organized on a rectangular grid with N rows and M columns. Anne has gathered a list of size A of N-letter words and a list of size B of M-letter words. She is wondering how many ways there are to display simultaneously N words of length M horizontally and M words of length N vertically on that grid.

Input

The input consists of the following:

- The first line contains two integers N and A separated with a single space.
- The second line contains two integers M and B separated with a single space.
- The next A lines contains N-letter words, one per line.
- The next B lines contains M-letter words, one per line.

Limits

The input satisfies $2 \leq N$, $M \leq 4$ and $1 \leq A \times B \leq 1008016$.

Words are taken from the English dictionary. Each of the two lists contains no repetition. Words consist of lowercase letters between 'a' (ASCII code 97) and 'z' (ASCII code 122).

Words from the first list will be displayed vertically, from top to bottom. Words from the second list will be displayed horizontally, from left to right. The same word can be used several times to build a grid, i.e., in several columns (resp., rows) if it belongs to the first (resp., second) list of words.

When $N = M$, it is not allowed to use words from the first list horizontally (unless they appear in the second list as well), or words from the second list vertically (unless they appear in the first list as well).

Output

The output should consist of a single line, whose content is an integer, the total number of distinct word grids.

Examples

standard input	standard output
3 4 4 5 war are yes sat says area test ways rest	2
3 7 3 7 ran age now its the set ago ran age now its the set ago	2

Note

Sample Explanation 1

s	a	y	s
a	r	e	a
t	e	s	t

w	a	y	s
a	r	e	a
r	e	s	t

The solutions are:

Sample Explanation 2

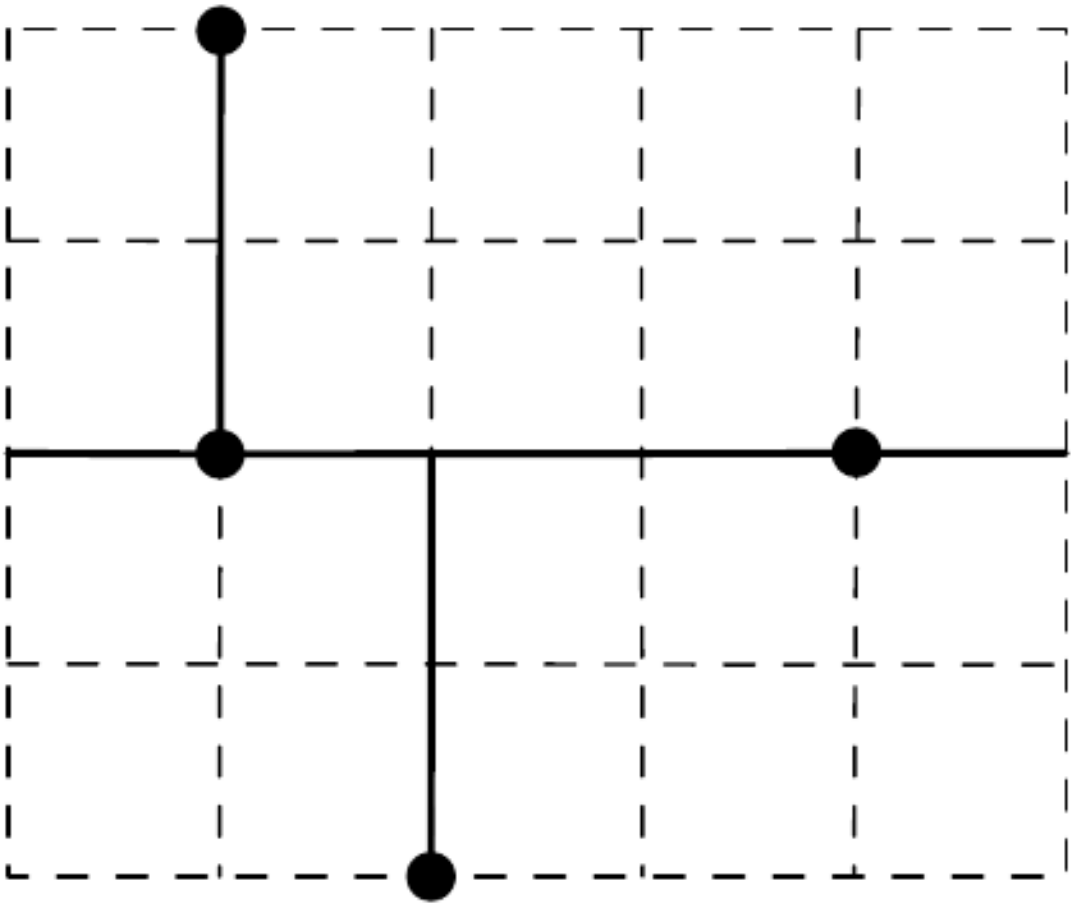
i	t	s
t	h	e
s	e	t

r	a	n
a	g	o
n	o	w

The solutions are:

Problem D. Monument Tour

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes



A tour operator proposes itineraries consisting of visits of N monuments and museums in Paris, modeled as a grid. The way the tour operates is the following: The bus enters the city from the west (on any street) and traverses the city, taking a left or a right turn to visit monuments when needed, returning to the same eastbound road it used to enter the city, and so on until it exits the city via the east.

A tour in a 6×5 grid city might look like the figure above. On the figure, the bus enters the city at coordinate $(0, 2)$ (we consider $(0, 0)$ to be the northwest corner of the city), first visits the monument at $(1, 2)$ (already on the main road), takes a left turn and visits the monument at $(1, 0)$, returns to the main road and moves east, takes a right turn and visits the monument at $(2, 4)$, returns to the main road, visits the monument at $(4, 2)$ (again on the main road), and then exits the city at coordinate $(5, 2)$.

The bus operator counts that the traversal of one block costs 1 unit of gas. For the example above, the cost is thus $5 + 2 \times 2 + 2 \times 2 = 13$ units of gas.

Your task is to help the tour operator choose which eastbound road the bus should travel through, so that the cost of the tour is minimized while visiting all N monuments.

Input

The input comprises several lines, each consisting of integers separated with single spaces:

- The first line contains the number X of northbound streets and the number Y of eastbound streets.

- The second line contains the number N of monuments the tour needs to visit.
- The next N lines each contain the coordinates x_i and y_i of each monument.

- $1 \leq X, Y \leq 100000$;
- $1 \leq N \leq 100000$;
- $0 \leq x_i < X, \quad 0 \leq y_i < Y$.

Output

The output should consist of a single line, whose content is an integer, representing the minimal cost of a tour.

Examples

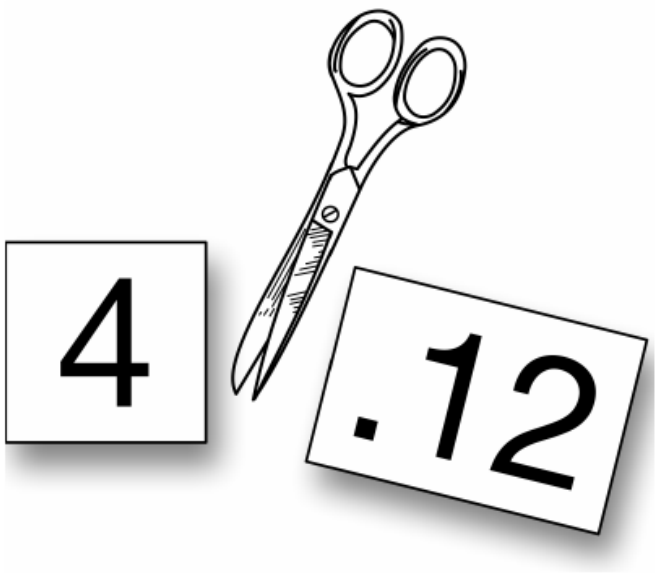
standard input	standard output
6 5 4 1 0 1 2 2 4 4 2	13
5 7 9 0 0 0 2 0 3 2 2 2 3 3 2 4 3 4 4 4 6	20

Note

- The bus operator is not allowed to return to a different, parallel, eastbound road; they have to use the same road as the one by which they entered the city.
- More than one monument can be present at the same coordinate.

Problem E. Rounding

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes



You decided to stay an extra day in Paris visiting favorite places of Parisians around Télécom ParisTech. You want to collect information about these favorite places, but asking people to fill in surveys is less fun than coding. For this reason, you asked the Parisian Agency for Really Imprecise Surveys to do it for you. You sent them a list of the P places you were interested in.

After surveying exactly 10 000 persons and asking them their favorite place (among these P places), the agency has just sent you the results. All persons surveyed answered the question. Unfortunately, the agency rounded the percentage results to the nearest integer, using the following formula: $result = \lfloor original\_value + \frac{1}{2} \rfloor$. In particular, decimal values of .50 are rounded up.

But since 10000 persons were surveyed, you should have been able to get percentage values precise to the second decimal. What a loss of precision! You want to know the range in which each original result could be.

Input

The input comprises several lines:

- The first line consists of an integer P.
- Each of the following P lines consists of the name of a place followed by an integer i, separated with a single space.

Limits

- $1 \leq P \leq 10000$;
- the name of a place is a string of between 1 and 20 characters among Latin alphabet letters ('A' to 'Z' and 'a' to 'z') and the underscore character ('_');
- no two names are the same;

- $0 \leq i \leq 100$.

Output

If the results given by the agency are not consistent, print a single line with the word IMPOSSIBLE. Otherwise the output should consist of P lines, each of them should consist of the name of a place followed by a single space and two numbers, the smallest and the largest percentage values that place could have had in the original results, as floating-point numbers with two decimals separated with a single space (each number must have at least one digit before the decimal point, even if it is 0, and exactly 2 decimals, even if the trailing ones are 0). The places must be in the same order as in the input.

Examples

standard input	standard output
4 Catacombes 32 Cite_Universitaire 22 Arenes_de_Lutece 26 Observatoire 19	Catacombes 31.53 32.49 Cite_Universitaire 21.53 22.49 Arenes_de_Lutece 25.53 26.49 Observatoire 18.53 19.49
7 Aqueduc_Medicis 11 Parc_Montsouris 40 Place_Denfert 10 Hopital_Sainte_Anne 4 Butte_aux_cailles 20 Cite_florale 12 Prison_de_la_Sante 0	Aqueduc_Medicis 11.06 11.49 Parc_Montsouris 40.06 40.49 Place_Denfert 10.06 10.49 Hopital_Sainte_Anne 4.06 4.49 Butte_aux_cailles 20.06 20.49 Cite_florale 12.06 12.49 Prison_de_la_Sante 0.06 0.49
2 Catacombes 50 Arenes_de_Lutece 49	IMPOSSIBLE

Problem F. Paris by Night

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	256 megabytes

During her trip to Paris for SWERC, Morgane graded every monument she visited. On this last night of the week, she plans to go in some hot-air balloon and take two 180° panoramic photographs of the city, thereby getting a perfect 360° view. She would like both photographs to be approximately as nice as each other.

Hence, she will proceed as follows. She chooses two monuments, that we will call the boundary monuments, and asks the balloon pilot to go between these monuments. From there, she takes two 180° pictures: each picture shows one side of Paris, both sides being delimited by the two boundary monuments. Each picture receives a grade, which is the sum of the grades of the monuments that it shows, including the boundary monuments on both pictures. Finally, if the pictures have grades A and B, the goal of Morgane is to minimize the difference between A and B (in absolute value).

The visibility from the balloon is such that all monuments can be seen in either of the two photographs.

You must assist Morgane in choosing appropriate boundary monuments. In order to do this, you are given a list of the monuments. For every monument visited by Morgane, this list contains a line which indicates the Cartesian coordinates of the monument location, along with the monument's grade. You should give the minimal possible difference, among all pairs of pictures that Morgane may take, between the grades of the two pictures.

Important Note

Morgane was so precise at locating every monument that no three monuments are on a same straight line.

Input

The input comprises several lines, each consisting of integers separated with single spaces:

- the first line contains the number N of monuments;
- each of the N next lines contains three integers associated with each monument: the X coordinate, the Y coordinate and the grade G of the monument represented on that line.

Limits

- $2 \leq N \leq 4000$;
- $0 \leq X,\ Y \leq 1000000000$ and $1 \leq G \leq 1000000000$.

Output

The output should consist of a single line, whose content is an integer, the minimal difference (in absolute value) between the grades of a pair of photographs that Morgane may take.

Example

standard input	standard output
8 0 0 10 1 1 2 2 1 3 3 2 7 2 3 8 5 2 5 1 5 12 4 5 14	2

Problem G. Strings

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

Gustave is an artist. His last project is to wrap the Eiffel Tower with a very long strip of fabric on which messages from people from all over the world are written. Obviously, the strip has to be very, very long and Gustave came up with the following way of building it. He starts with a string on which all the messages are written. Then he repeatedly builds other strings, either by concatenating copies of two strings, or by making a copy of a section of consecutive characters from another string.

Once Gustave is happy with the final string he gets, he contacts a company to have the string printed on a strip of fabric. Being meticulous, Gustave does not want the company to make a single mistake. He thus computes a checksum out of his string and has the company do the same computation as a verification.

Input

The input consists of the following lines:

- The first line contains an integer N.
- The next line contains a string $S(0)$ of lowercase alphabetic characters between 'a' and 'z'.
- The next $N-1$ lines contain instructions to build strings $S(1), \ldots, S(N-1)$. The instruction to build string $S(i)$ is either:
  - "SUB x lo hi" with x, lo, hi integers such that $0 \leq x < i$ and $0 \leq lo \leq hi \leq length(S(x))$, or
  - "APP x y" with x, y integers such that $0 \leq x, y < i$.

  Instruction "SUB x lo hi" means that $S(i)$ is formed using (a copy of) characters of $S(x)$ from index lo (inclusive) to hi (exclusive). Characters are numbered starting from 0. Instruction "APP x y" means that $S(i)$ is formed by concatenating copies of strings $S(x)$ and $S(y)$ in that order, i.e., with $S(x)$ coming first then $S(y)$.

Limits

- $1 \leq N \leq 2500$;
- $1 \leq length(S(0)) \leq 1000$;
- the length of any string $S(i)$ will never exceed $2^{63} - 1$.

Output

The output should consist of a single line, whose content is an integer, the sum of all ASCII codes of the characters of the final string $S(N-1)$, modulo 1000000007.

Example

standard input	standard output
3 foobar SUB 0 0 3 APP 1 1	648

Problem H. Travel Guide

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	256 megabytes

Paris counts many hotels. Some are very close to *Orly* airport, which is very useful to spend a night before an early flight. Some are very close to the *Notre-Dame* cathedral which allows tourists to walk around the "Île Saint-Louis" at dawn and enjoy the banks of the Seine. Finally, some are closer to the *Disneyland* Paris resort which is the most visited tourist attraction. Travelers who come to Paris usually want to stay near these three main Points Of Interest (POIs): *Orly*, *Notre-Dame*, and *Disneyland*.

You are employed by a travel agency and your manager Anna wants to prepare a list of hotels to include in her new travel guide. Her guide contains one entry per station of the metropolitan network. Anna notices that some stations do not have a convenient location with respect to the distance to the three POIs and therefore her guide should not contain a hotel section for such "useless" stations.

Anna considers that a station is *useless* when another station is closer to all the POIs. Formally a station A is useless when there exists another station B such that B is at least as close to the three POIs as A is and B is strictly closer than A to at least one of those POIs. A station is *useful* if it is not useless.

Anna asks you how many stations in her guide will have a hotel section. To compute this list you are given the plan of the metropolitan network. The plan of the metropolitan network is an undirected weighted graph. In this graph, each node corresponds to a station (note that each POI is also a station); each edge links two stations and takes a certain time to be traversed in either direction. This graph is connected and the distance between any two stations is the lowest total time of a path between the two nodes.

Input

The input comprises several lines, each consisting of integers separated with single spaces:

- The first line consists of the number N of nodes and the number E of edges.

- Each of the following E lines describes an edge with three integers A, B, and W:
  - A and B are the endpoints of the edge (numbered from 0 to $N-1$);
  - W is the weight of the edge.

In the graph:

- *Orly* corresponds to the station 0;
- *Notre-Dame* corresponds to the station 1;
- *Disneyland* corresponds to the station 2.

Limits

- $4 \leq N \leq 100000$;
- $E \leq 500000$;
- $1 \leq w \leq 100$.

Output

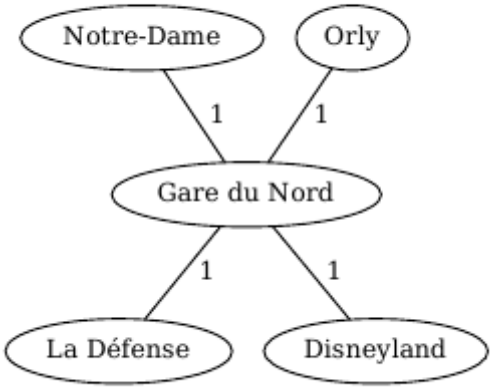
The output should consist of a single line, whose content is an integer, the number of useful stations.

Examples

standard input	standard output
5 4 0 3 1 1 3 1 2 3 1 4 3 1	4
5 6 0 3 1 1 3 1 2 3 1 4 3 1 0 1 1 0 2 1	3

Note

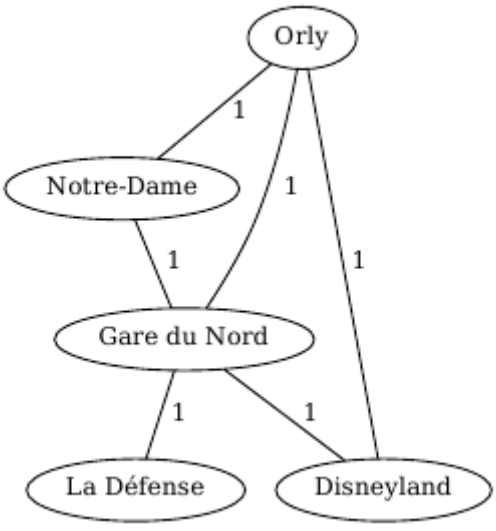
Explanation of Sample 1:



This graph is depicted on the top with *Gare du Nord* as node 3 and *La Défense* as node 4. In this graph, four stations are useful: *Orly*, *Disneyland*, *Notre-Dame*, and *Gare du Nord*. The stations corresponding to *Orly*, *Disneyland*, *Notre-Dame* are the closest stations to themselves. All stations have a POI at distance 2 except for *Gare du Nord*, which is at distance 1 to all POIs. Finally, *La Défense* is useless because it is at distance 2 from all POIs. For instance, *Orly* is as close as *La Défense* to *Notre-Dame* and *Disneyland* but strictly closer to *Orly* and thus *Orly* witnesses that the station *La Défense* is useless.

Explanation of Sample 2:

In this graph (depicted on the bottom), three stations are useful: *Orly*, *Disneyland*, and *Notre-Dame*. The stations corresponding to *Orly*, *Disneyland*, *Notre-Dame* are the closest stations to themselves. All stations have a POI at distance 2 except for *Gare du Nord*, which is at distance 1 to all POIs, and Orly which is at distance 1 to all POIs but at a distance of 0 to itself. Therefore *Gare du Nord* is useless.

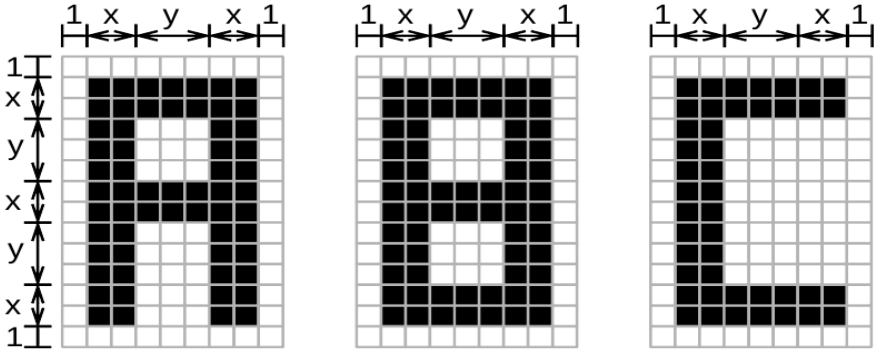


Problem I. Mason's Mark

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

A mason's mark is a symbol often found on dressed stones in buildings and other public structures. As Peter walks with his camera through Paris, he notices these marks on the wall of the Tour Jean-Sans-Peur. Every stone has one of the marks A, B or C, which are quite visible. He makes a black and white photo and observes :

- The picture shows stones, and every stone contains exactly one mark.
- All marks have one of the following shapes with x and y being arbitrary strictly positive integers, and possibly different for each mark. Note that marks are surrounded by white pixels and that marks cannot be rotated.



- The picture contains some noise, which are black pixels surrounded by 8 white pixels.
- There are 3 kinds of black pixels, corresponding respectively to the noise, the mason's marks, and the region around the stones.
- Every white pixel belongs to the surface of a stone and some of them also belong to the interior of a mark.

- The white pixels belonging to the surface of the same stone but not belonging to the interior of the mark are all connected with respect to vertical and horizontal adjacency. However, the surface of a stone may be non-convex.
- The black pixels of the region around the stones are connected with respect to vertical, horizontal, diagonal, and anti-diagonal adjacency, which means that you can go from any black pixel of the region around the stones to any other such pixel by moving from one pixel to one of its eight adjacent pixels. All pixels of the border of the picture are black and belong to this region.

You are given a rectangular matrix representing a picture made by Peter. The '#' character represents a black pixel and the '.' character a white pixel. You should count how many stones are on the picture with the respective letters A, B, and C.

Input

The first line contains two integers W and H. The next H lines each contain a string of length W. The strings are composed of '.' and '#'.

Limits

- $7 \leq W \leq 1000$;
- $9 \leq H \leq 1000$.

Output

The output should consist of a single line, whose content is three integers A, B, and C separated with single spaces, indicating the number of stones with the respective marks A, B, and C.

Example

standard input	standard output
26 15 ########################## ##........######......#..# #...###....#####..#......# #...#.#....####.........## #...###.....##....#####..# #...#.#.....#.....#####..# #...###.....#.....##.##..# #........#..#.#...#####..# #..###......#.....#####..# #..#........#...#.##.##..# #..#........#.....##.##..# #..#...#.#..#...#.##.##..# #..###......#............# ###....#....##....##.....# ##########################	1 1 0

Note

Sample Explanation

There are black pixels forming a letter C. These pixels, however, belong to the region around the stones and do not form a mark since they are not surrounded by white pixels.

Problem J. Mona Lisa

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

The Louvre museum hosts one of the most famous paintings ever made: Mona Lisa, painted by Leonardo da Vinci in the 16th century.

The painting is enclosed in a rock-solid glass chamber that can only be opened with 4 secret codes that need to be entered on 4 different keypads. The head of the museum thinks that this system is unbreakable, and your task is to prove her wrong.

To help you, a friend reverse-engineered the system. When a code (represented by a positive integer C) is entered on a keypad, the keypad sends the C-th value produced by a random number generator to a central computer. The central computer only considers the N least significant bits of the 4 pseudo-random values it receives from the 4 keypads. It computes their bitwise XOR (exclusive or) and opens the glass chamber if the result is 0. The pseudo-random number generator is described at the end of the problem statement.

Another friend found the pseudo-random seeds used by each keypad. With all this information, you think that you can retrieve the 4 secret codes unlocking Mona Lisa.

Input

The input comprises two lines, each consisting of integers separated with single spaces:

- The first line contains the integer N.
- The second line contains the four integer seeds.

Limits

- $1 \leq N \leq 50$;
- each seed is between 0 and $2^{64} - 1$.

Output

The output should consist of a single line, whose content is 4 integers, the 4 secret codes, separated with single spaces. Each code must be less than 100000000. It is guaranteed that at least one solution will exist. Multiple solutions may exist, in which case they will all be accepted.

Example

standard input	standard output
50 3641603982383516983 445363681616962640	287 17609 122886 59914 868196408185819179 1980241222855773941

Note

Pseudo-Random Generator The pseudo-random generator is described next in each programming language. You can expect that this pseudo-random generator is not biased in any way.

C/C++

```
typedef unsigned long long uint64;
uint64 state[2] = { seed, seed ^ 0x7263d9bd8409f526 };
uint64 xoroshiro128plus(uint64 s[2]) {
  uint64 s0 = s[0];
  uint64 s1 = s[1];
  uint64 result = s0 + s1;
  s1 ^= s0;
  s[0] = ((s0 << 55) | (s0 >> 9)) ^ s1 ^ (s1 << 14);
  s[1] = (s1 << 36) | (s1 >> 28);
  return result;
}
```

The i-th value of the pseudo-random sequence is the result of the i-th application of xoroshiro128plus on 'state'.

Java

```
long[] state = { seed, seed ^ 0x7263d9bd8409f526L };
long xoroshiro128plus(long[] s) {
  long s0 = s[0];
  long s1 = s[1];
  long result = s0 + s1;
  s1 ^= s0;
  s[0] = ((s0 << 55) | (s0 >>> 9)) ^ s1 ^ (s1 << 14);
  s[1] = (s1 << 36) | (s1 >>> 28);
  return result;
}
```

The i-th value of the pseudo-random sequence is the result of the i-th application of xoroshiro128plus on 'state'.

Python 2 / Python 3

```
state = [seed, seed ^ 0x7263d9bd8409f526]
def xoroshiro128plus(s):
```

```
  s0, s1 = s
  result = (s0 + s1) \% 2**64
  s1 ^= s0;
  new_state = [(((s0 << 55) | (s0 >> 9)) ^ s1 ^ (s1 << 14)) \% 2**64,
               ((s1 << 36) | (s1 >> 28)) \% 2**64]
  return result, new_state
```

The following loop yields the pseudo-random sequence starting from its first value:

```
while True:
  result, state = xoroshiro128plus(state)
  yield result
```

Problem K. Dishonest Driver

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	256 megabytes

When you arrived at Charles de Gaulle airport, you naively accepted a ride to Paris by an unauthorized driver who offered you "competitive prices". The end result was a disaster: not only was the price extremely high, but the driver made the trip much longer than necessary in order to justify that price.

You noticed the scam because you were traveling several times by the same place: indeed, you have such a good memory that you can remember very well the path you followed, including each of the loops that your scammer forced you to take.

Now, you are at the police station to file a complaint against this driver and an officer asks you to tell your story. She even asks you to give all the details of the path you took. Because you do not want to lose yet another couple of hours in doing so, you decide to give a compressed version of it.

Suppose you remember you traveled through places A, B, C, D, A, B, C, D. In this case, you prefer telling the officer "I followed twice the path ABCD", rather than "I followed the path ABCDABCD". Given that your path repeated the same sequence of places, this will significantly shorten your statement, without missing any detail.

More precisely, you have to write a program that takes as input the list of places you traveled through, and which returns the size of the shortest compressed form of this path. Such a compressed path can either be:

- a single place through which you traveled, called an "atomic path";
- the concatenation of two compressed paths;

- the repetition of a compressed path, i.e., $(\mathrm{C})^n$, meaning that you traveled through the path described by C, n times in a row.

The size of a compressed path is defined as the number of atomic paths it contains.

Input

The input consists of two lines:

- The first line contains one integer, N, the length of the path.
- The second line contains the path, described as a string of size N. Each location is described by an alphanumeric character: either a digit (from '0' to '9'), a lowercase letter (from 'a' to 'z') or an uppercase letter (from 'A' to 'Z').

Limits

$0 < N \leq 700$.

Output

The output should consist of a single line, whose content is an integer, the size of the shortest compressed path.

Examples

standard input	standard output
22 aaabaaabccdaaabaaabccd	4
4 aaba	3

Note

Sample Explanation 1

The shortest compressed form of the path is $(((a)^3b)^2(c)^2d)^2$. The atomic paths it contains are a, b, c and d. Hence, it has size 4.

Sample Explanation 2

The shortest compressed form of the path is $(a)^2ba$. The atomic paths it contains are a, b, and a. Hence, it has size 3.

Problem Analysis Session

SWERC judges

December 2, 2018

Statistics

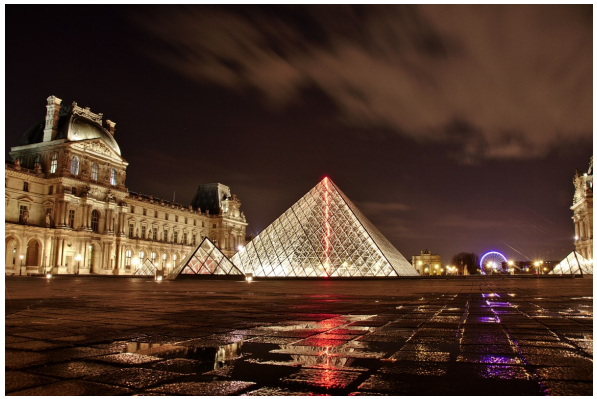
Number of submissions: about 2500
Number of clarification requests: 28 (20 answered "No comment.")

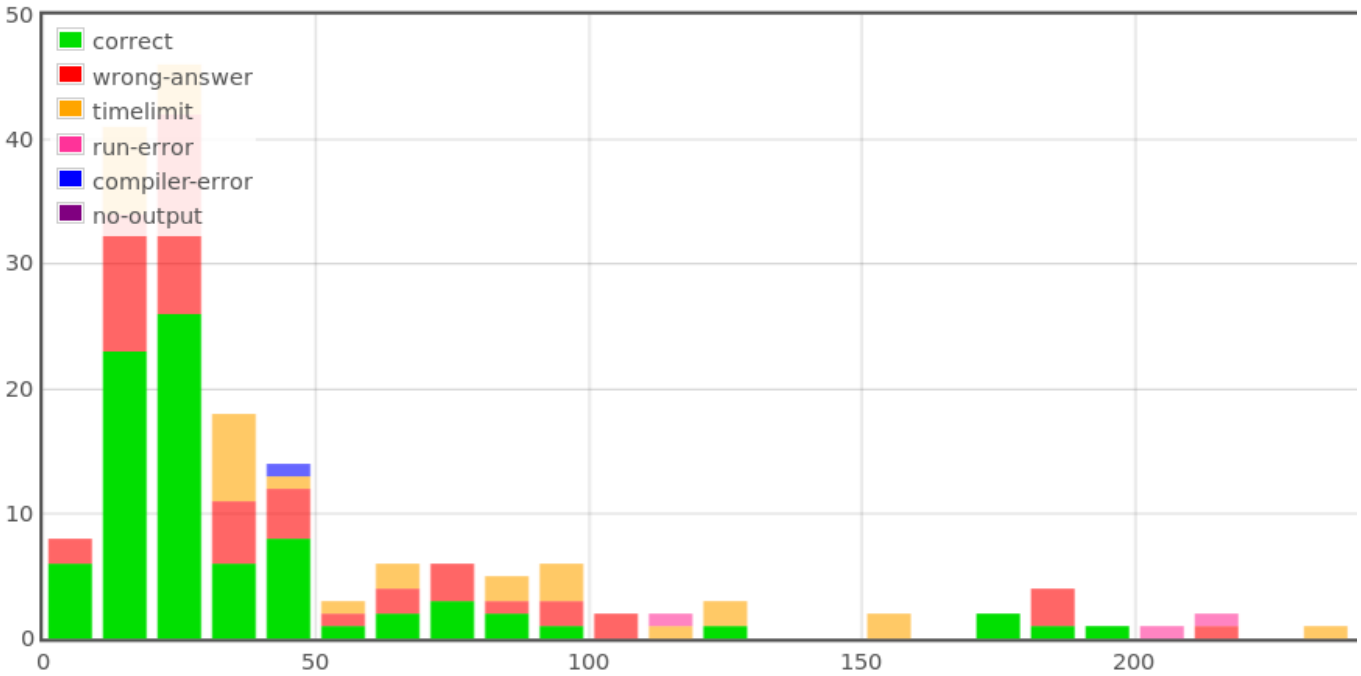
Languages:

- 1533 C++
- 34 C
- 232 Java
- 330 Python 2
- 233 Python 3

A – City of Lights

Solved by 83 teams before freeze.
First solved after 6 min by **Team RockETH**.





A – City of Lights

This was the easiest problem of the contest.

Problem

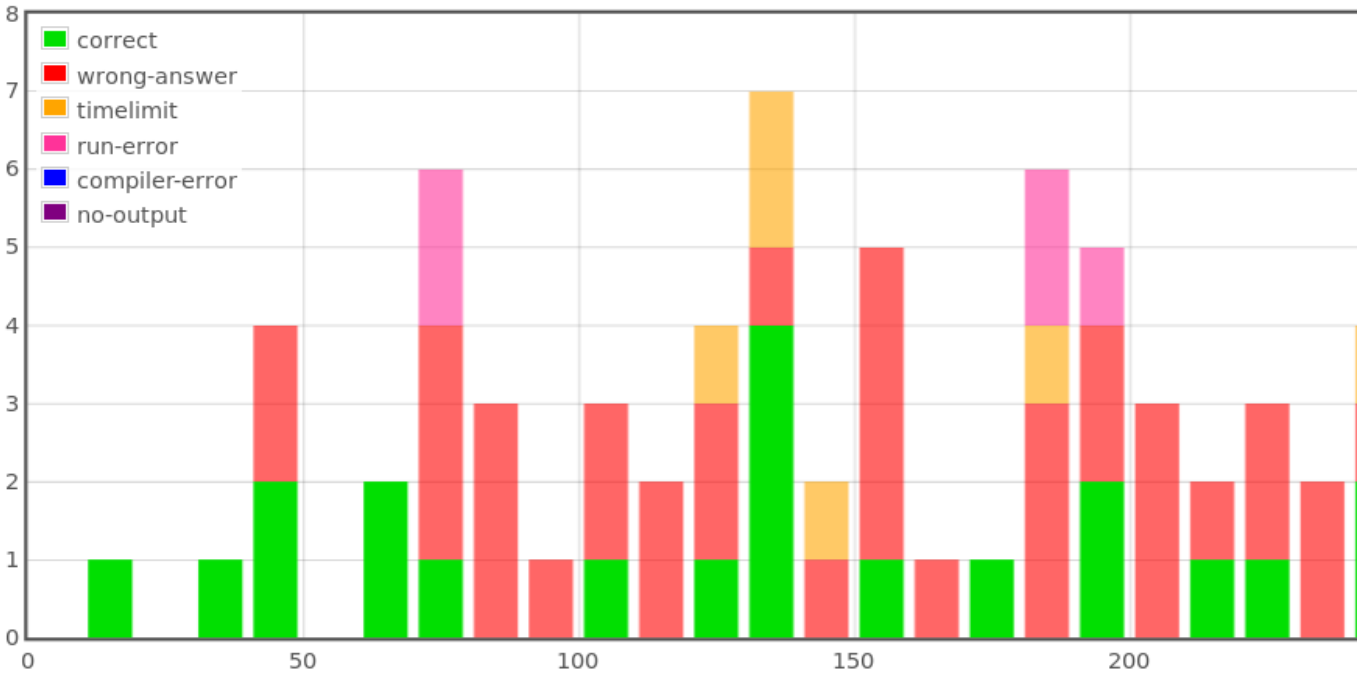
Toggle regularly spaced lights at every step, and print the maximum number of turned-off lights.

Straightforward solution

- Keep an array with the light status (or a bit set).
- Keep the number of currently turned-off lights in a variable.

K – Dishonest Driver

Solved by 18 teams before freeze.
First solved after 17 min by **Team RaclETH**.



K – Dishonest Driver

Problem

Given a string, compute the length of its shortest compressed form.
How to build a compressed form:

- one character c (size: $|c| = 1$),
- concatenation $w_1 w_2$ (size: $|w_1 w_2| = |w_1| + |w_2|$),
- repetition $(w)^n$ (size: $|(w)^n| = |w|$).

K – Dishonest Driver

Solution in time $\mathcal{O}(N^3)$

Dynamic programming on:

$$F(i,j) = \text{size of compressed form of substring } u_{ij} = u_i \dots u_{j-1}$$

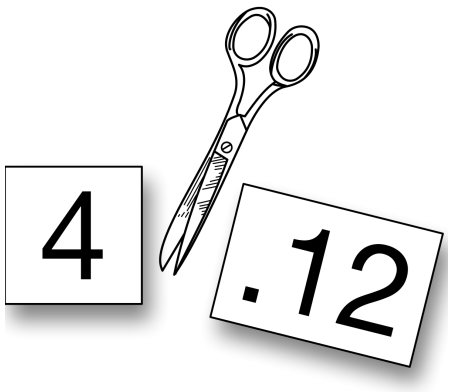
If $j = i+1$, then $F(i,j) = 1$. Otherwise:

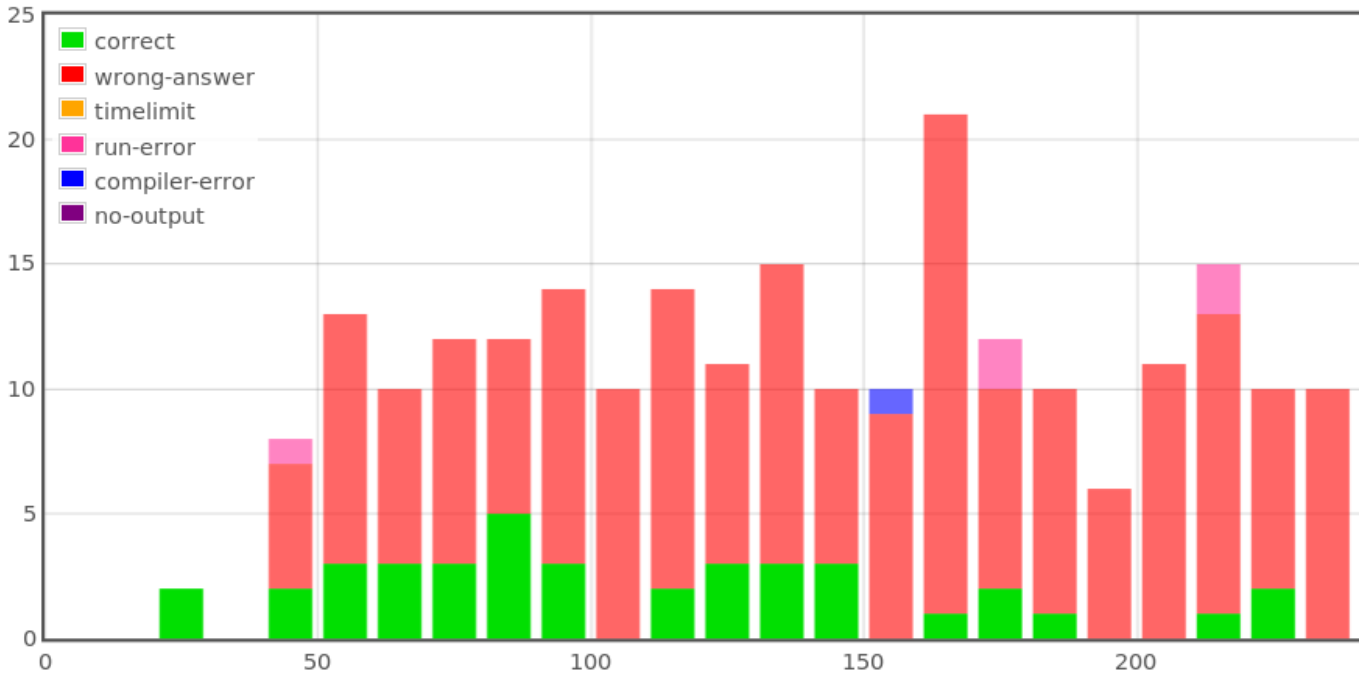
- Try splitting $u_{ij} = u_{ik}u_{kj}$ for any position $k \in [i+1, j-1]$;
- Try factorizing u_{ij} into $u_{ij} = u_{ik}^n$:
  - What are the factorizations of u_{ij}?
  - Trick: search second occurence of u_{ij} in $u_{ij}u_{ij}$
  - $\mathcal{O}(N)$ with KMP (e.g., use C++ stdlib `find` function)

Note: we also have a $\mathcal{O}(N^2 \log N)$ algorithm

E – Rounding

Solved by 39 teams before freeze.
First solved after 23 min by **SNS 1**.





E – Rounding

First bounds

Each monument **m** with rounded value $\mathbf{round_m}$ had an original value $\mathbf{origin_m}$ such that:

- $\mathbf{origin_m} \geqslant \mathbf{min_m}$, with $\mathbf{min_m} = \max\{0, \mathbf{round_m} - 0.50\}$;
- $\mathbf{origin_m} \leqslant \mathbf{max_m}$, with $\mathbf{max_m} = \min\{100, \mathbf{round_m} + 0.49\}$.

Possible or not?

Possible **if and only if**

$$\sum_{\mathbf{m}} \mathbf{min_m} \leqslant 100 \leqslant \sum_{\mathbf{m}} \mathbf{max_m}.$$

E – Rounding

Solution

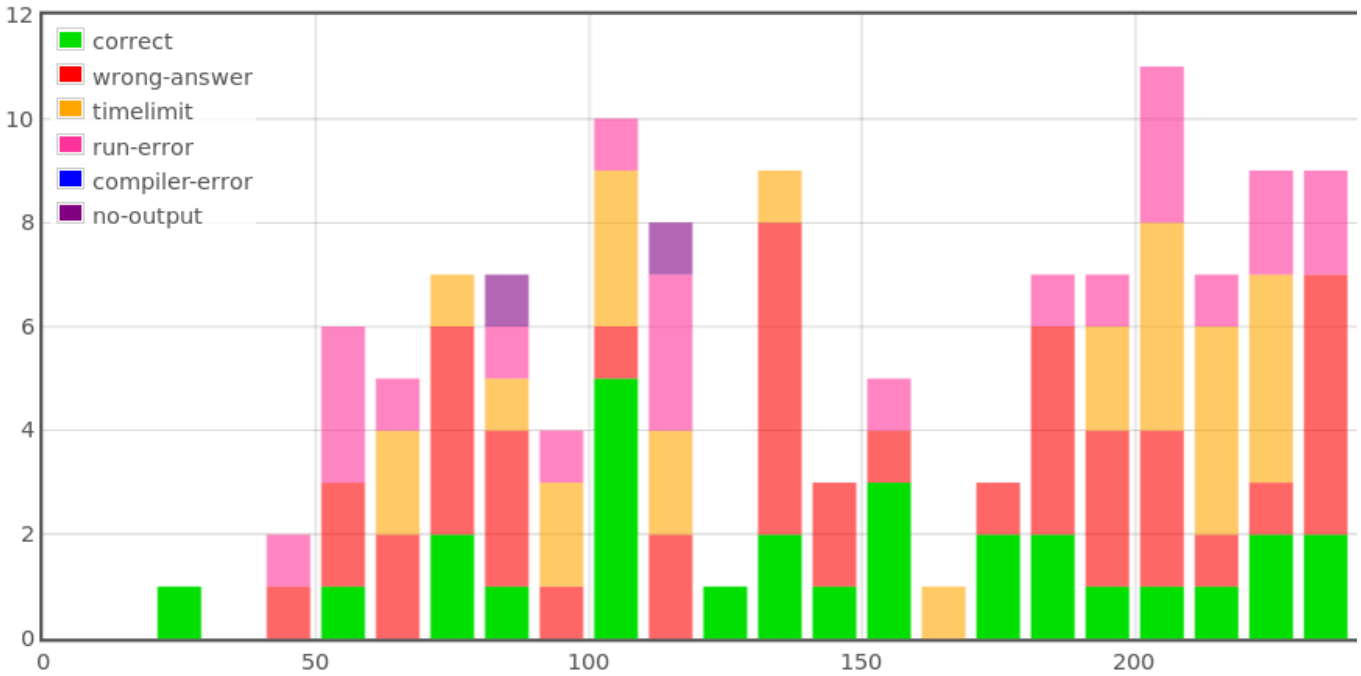
- Compute **minSum** $= \sum_{\mathbf{m}}$ **min**$_{\mathbf{m}}$ and **maxSum** $= \sum_{\mathbf{m}}$ **max**$_{\mathbf{m}}$
- Return `IMPOSSIBLE` if **minSum** > 100 or **maxSum** < 100
- **Real minimal value** for monument **m**:
  realMin$_{\mathbf{m}} = \max\{$**min**$_{\mathbf{m}}$, **max**$_{\mathbf{m}} - ($**maxSum** $- 100)\}$
- **Real maximal value** for monument **m**:
  realMax$_{\mathbf{m}} = \min\{$**max**$_{\mathbf{m}}$, **min**$_{\mathbf{m}} + (100 -$ **minSum**$)\}$

Main causes for wrong answers

- Allowing original values < 0 or > 100
- Using floating point numbers
- Result formatting issues

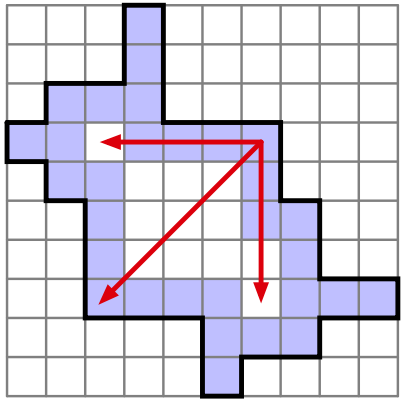
B – Blurred pictures

Solved by 28 teams before freeze.
First solved after 29 min by **UPC-1**.



B – Blurred pictures

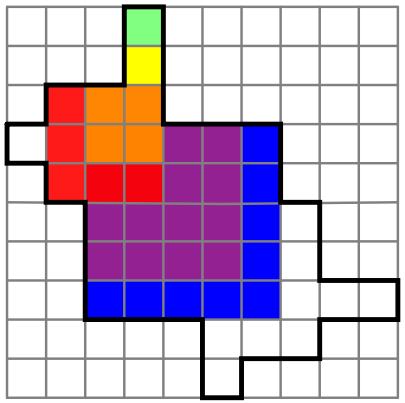
Dynamic programming **on the grid** would take time $\mathcal{O}(N \times N) \longrightarrow$ time limit exceeded



Note that perimeter is in $\mathcal{O}(N)$ and use it to compute only the mandatory extreme values in time $\mathcal{O}(N)$.

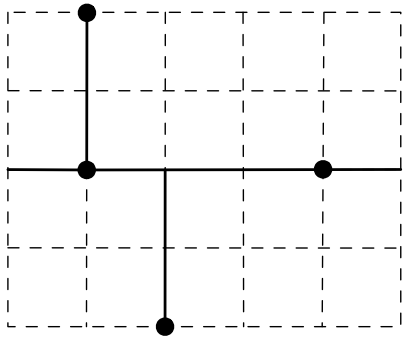
Even simpler:

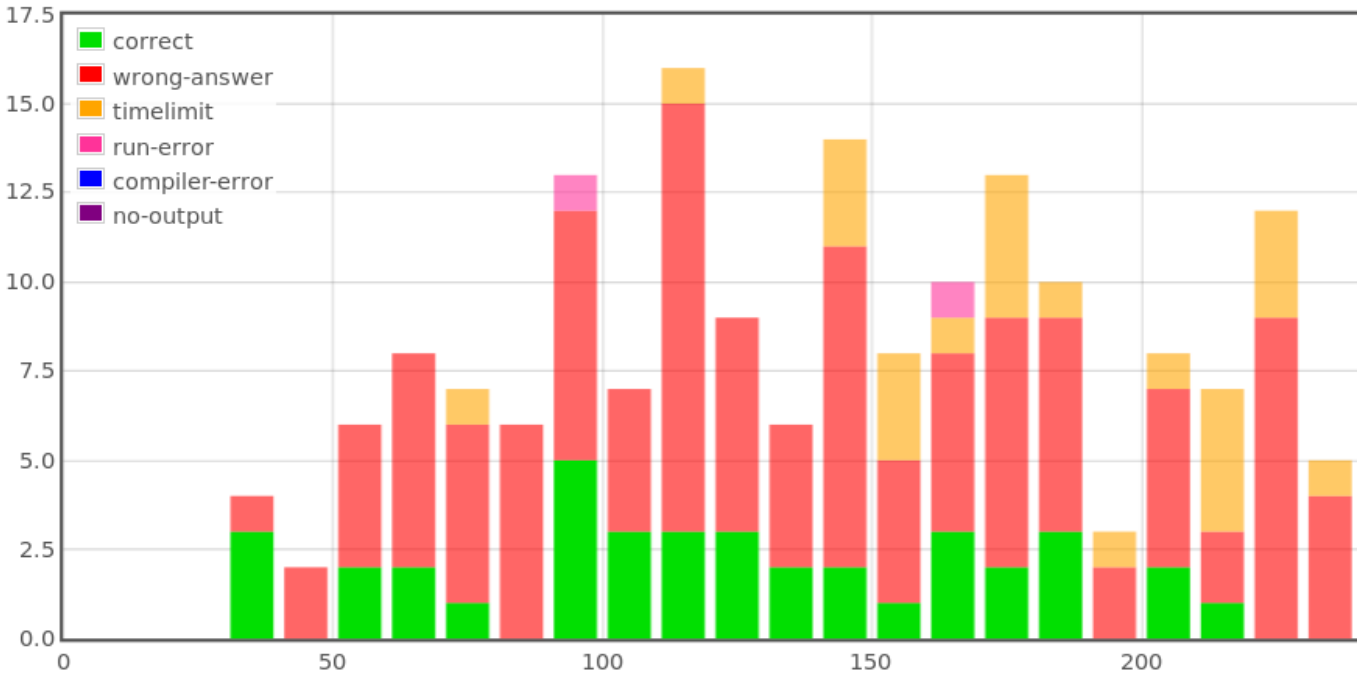
- You only need to keep track of the size of the largest square.
- Start from the first line and grow the maximum square from there, increasing its size at each new line when possible, else changing the starting line.



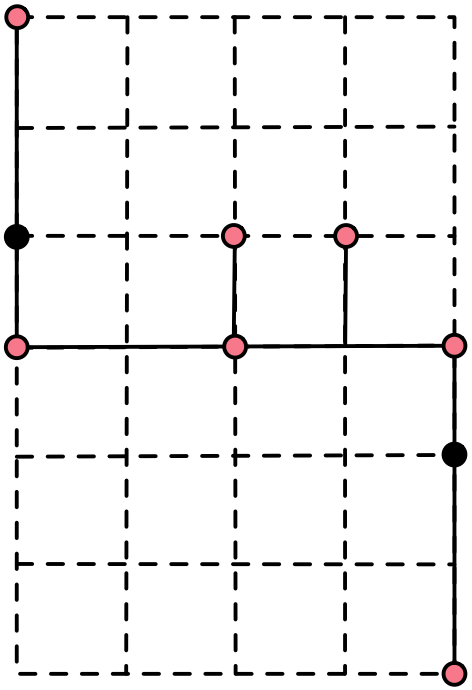
D – Monument Tour

Solved by 38 teams before freeze.
First solved after 37 min by **Blaise1**.





D – Monument Tour



Coordinates

0, 2, 2, 2, 3, 3, 3, 6

Solution

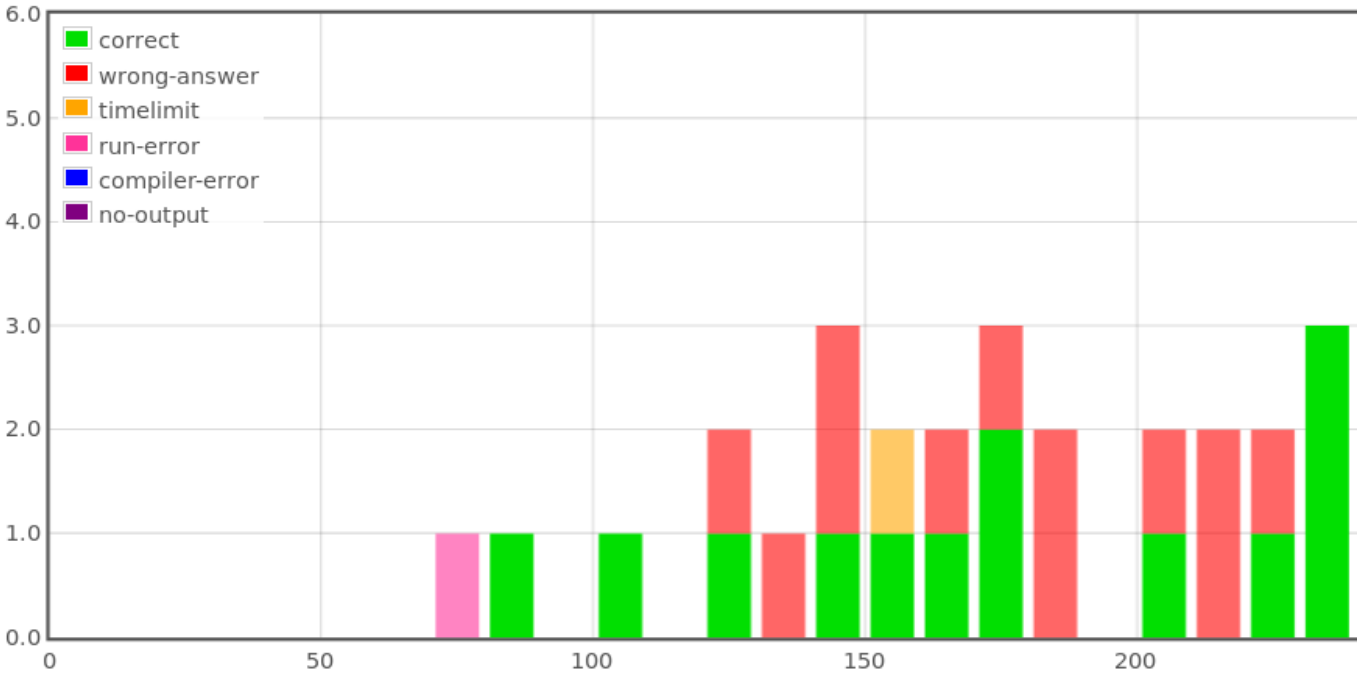
- the main road will always pass through at least one monument
- the best placement is the median of the y coordinates of the extreme points of "monument segments"

Monument Segment

- keep only the extremes of y coordinates corresponding to the same x
- count single points as a segment (i.e., count y coordinate twice)

F – Paris by Night

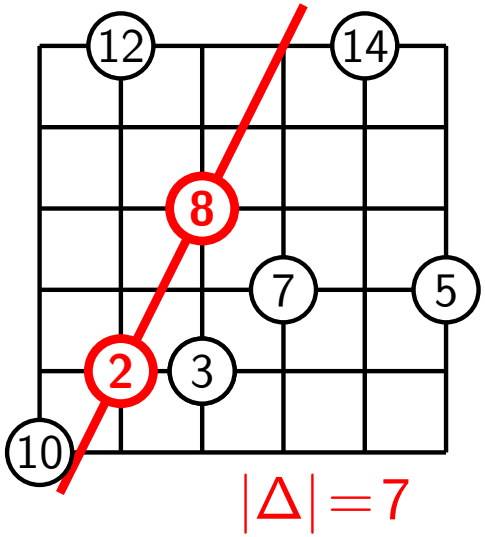
Solved by 13 teams before freeze.
First solved after 83 min by **Team RaclETH**.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

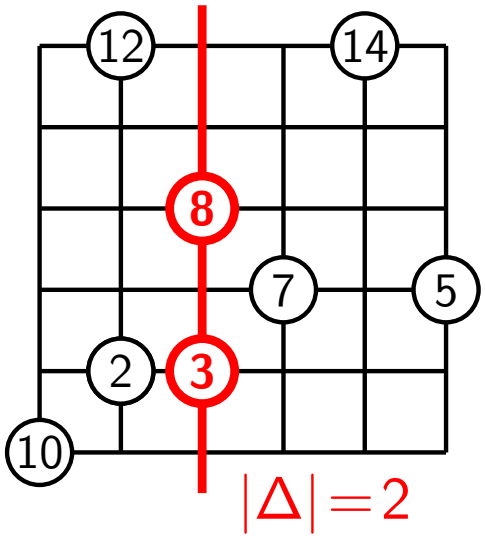
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M'}$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M'}}$ from scratch.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

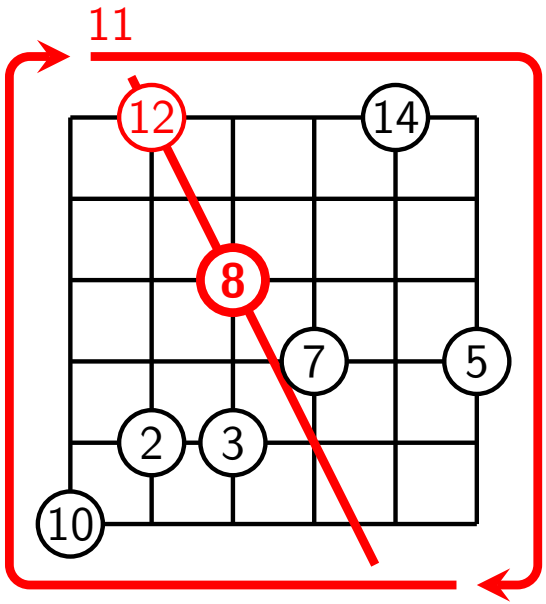
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M}'$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M}'}$ from scratch.



Better approach in time $\mathcal{O}(N^2 \log(N))$

For all limiting monuments $\mathbf{M}$:

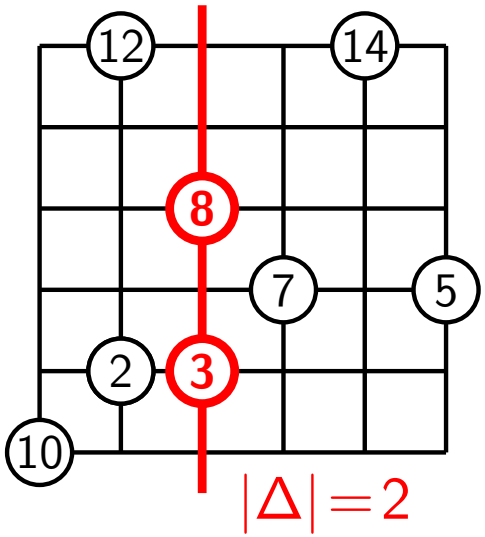
- order monuments $\mathbf{M}' \neq \mathbf{M}$ clockwise, based on the direction of $(\mathbf{M}\,\mathbf{M}')$;
- compute differences $\Delta_{\mathbf{M},\mathbf{M}'}$ incrementally.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

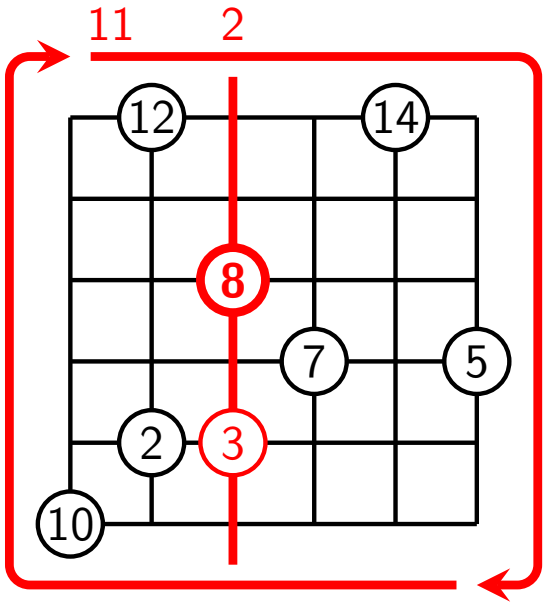
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M}'$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M}'}$ from scratch.



Better approach in time $\mathcal{O}(N^2 \log(N))$

For all limiting monuments $\mathbf{M}$:

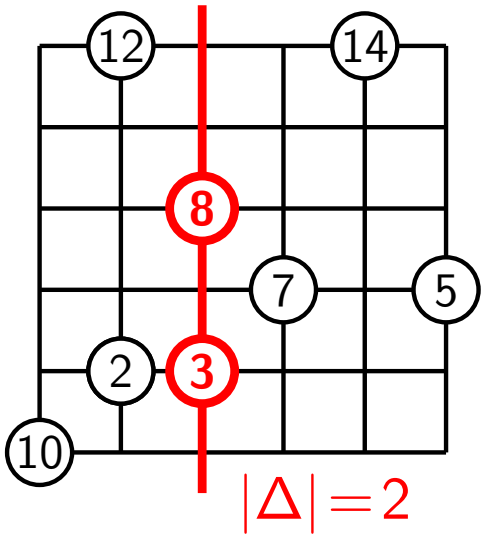
- order monuments $\mathbf{M}' \neq \mathbf{M}$ clockwise, based on the direction of $(\mathbf{M}\,\mathbf{M}')$;
- compute differences $\Delta_{\mathbf{M},\mathbf{M}'}$ incrementally.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

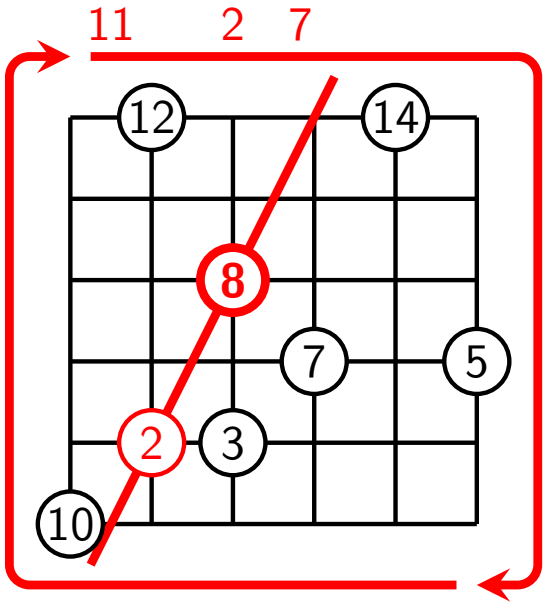
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M}'$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M}'}$ from scratch.



Better approach in time $\mathcal{O}(N^2 \log(N))$

For all limiting monuments $\mathbf{M}$:

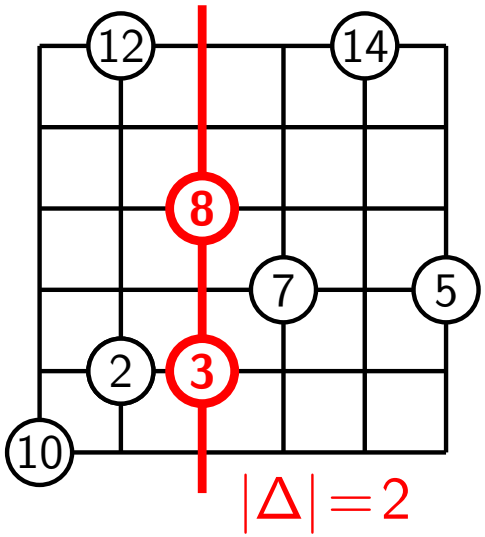
- order monuments $\mathbf{M}' \neq \mathbf{M}$ clockwise, based on the direction of $(\mathbf{M}\,\mathbf{M}')$;
- compute differences $\Delta_{\mathbf{M},\mathbf{M}'}$ incrementally.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

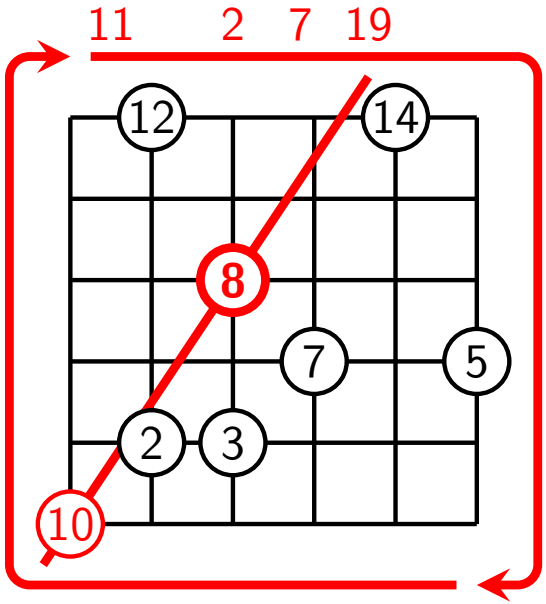
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M}'$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M}'}$ from scratch.



Better approach in time $\mathcal{O}(N^2 \log(N))$

For all limiting monuments $\mathbf{M}$:

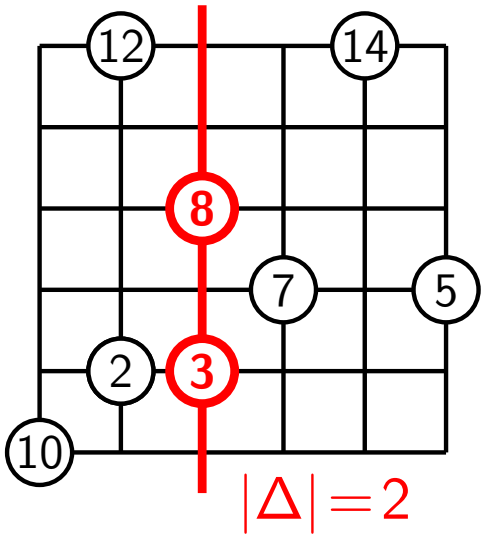
- order monuments $\mathbf{M}' \neq \mathbf{M}$ clockwise, based on the direction of $(\mathbf{M}\,\mathbf{M}')$;
- compute differences $\Delta_{\mathbf{M},\mathbf{M}'}$ incrementally.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

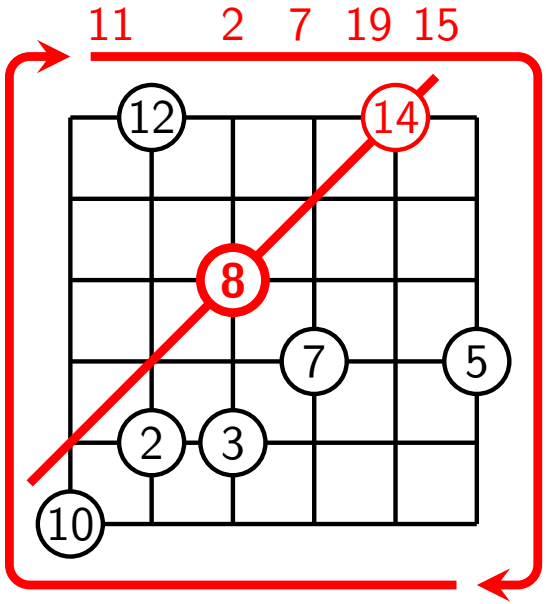
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M}'$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M}'}$ from scratch.



Better approach in time $\mathcal{O}(N^2 \log(N))$

For all limiting monuments $\mathbf{M}$:

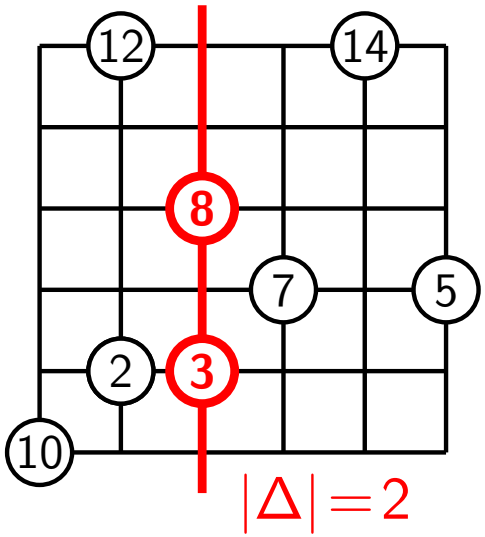
- order monuments $\mathbf{M}' \neq \mathbf{M}$ clockwise, based on the direction of $(\mathbf{M}\,\mathbf{M}')$;
- compute differences $\Delta_{\mathbf{M},\mathbf{M}'}$ incrementally.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

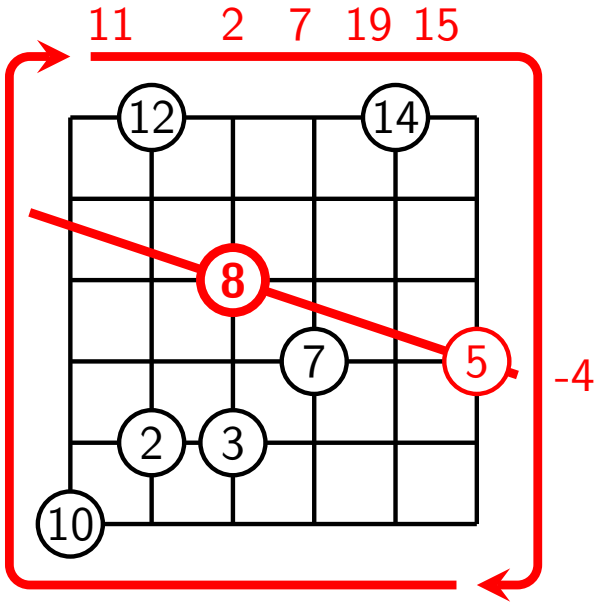
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M}'$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M}'}$ from scratch.



Better approach in time $\mathcal{O}(N^2 \log(N))$

For all limiting monuments $\mathbf{M}$:

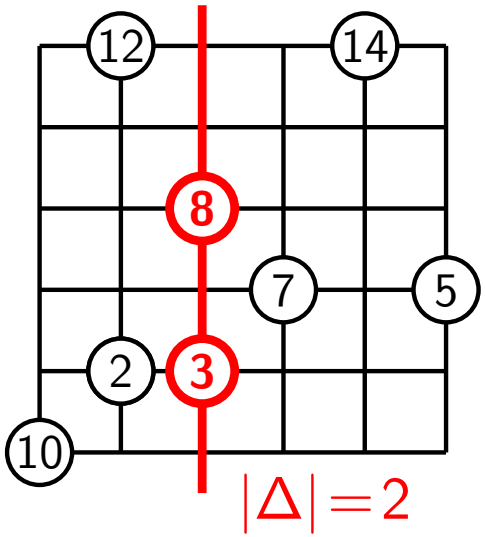
- order monuments $\mathbf{M}' \neq \mathbf{M}$ clockwise, based on the direction of $(\mathbf{M}\,\mathbf{M}')$;
- compute differences $\Delta_{\mathbf{M},\mathbf{M}'}$ incrementally.



F – Paris by Night

Naive approach in time $\mathcal{O}(N^3)$

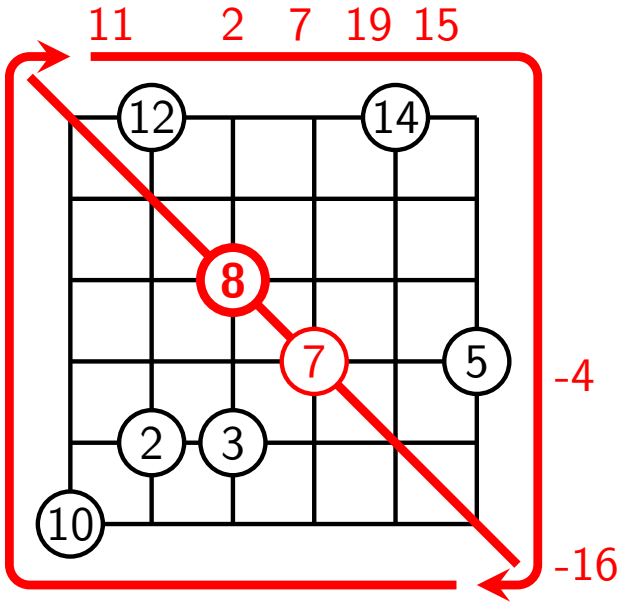
For all pairs of limiting monuments $\mathbf{M} \neq \mathbf{M}'$, compute the grade difference $\Delta_{\mathbf{M},\mathbf{M}'}$ from scratch.



Better approach in time $\mathcal{O}(N^2 \log(N))$

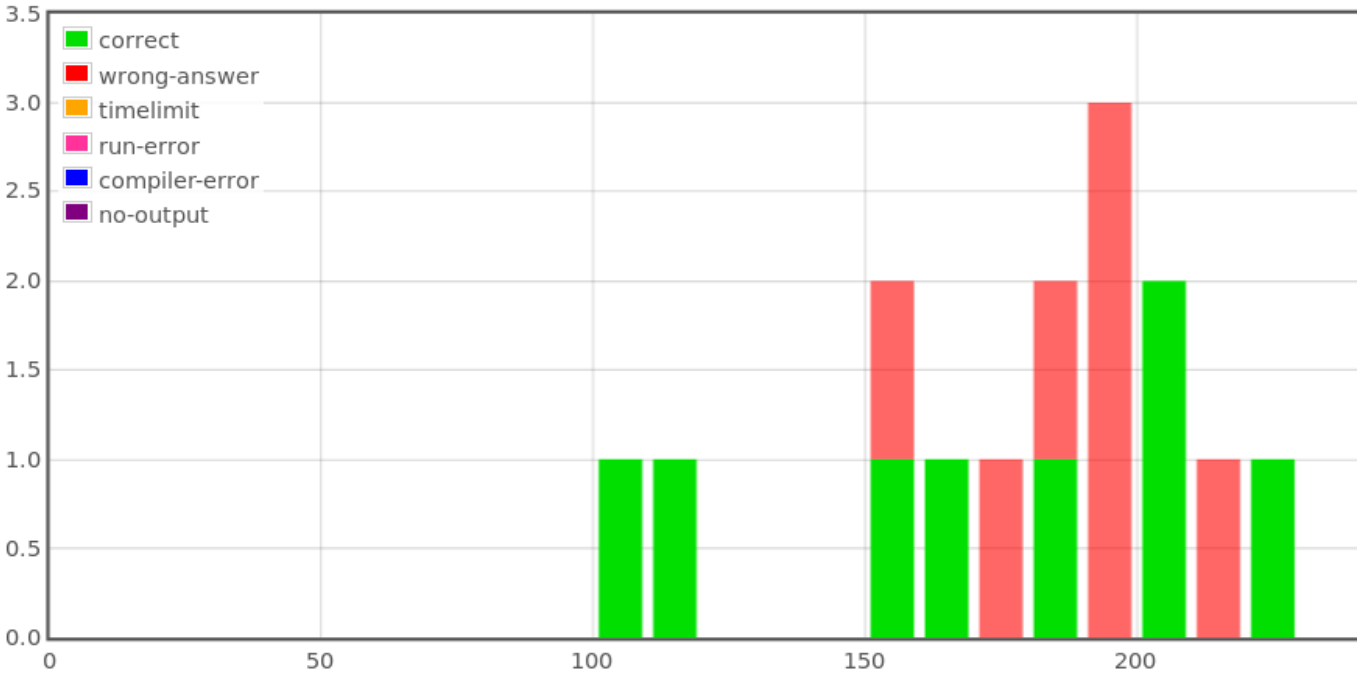
For all limiting monuments $\mathbf{M}$:

- order monuments $\mathbf{M}' \neq \mathbf{M}$ clockwise, based on the direction of $(\mathbf{M}\,\mathbf{M}')$;
- compute differences $\Delta_{\mathbf{M},\mathbf{M}'}$ incrementally.



I – Mason's Mark

Solved by 8 teams before freeze.
First solved after 100 min by **ENS Ulm 1**.

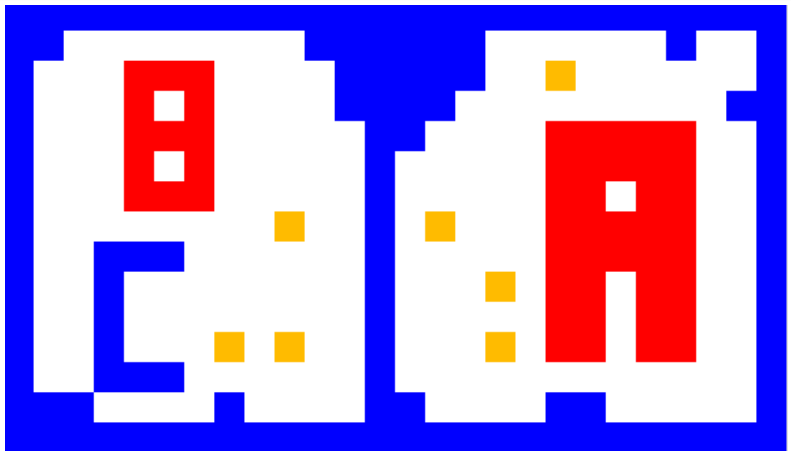


I – Mason's Mark

Many solutions are possible. For example:

Find connected components in a grid

Black dots form connected components, one of them contains the frame, others are single noise dots, and the remaining correspond to marks.

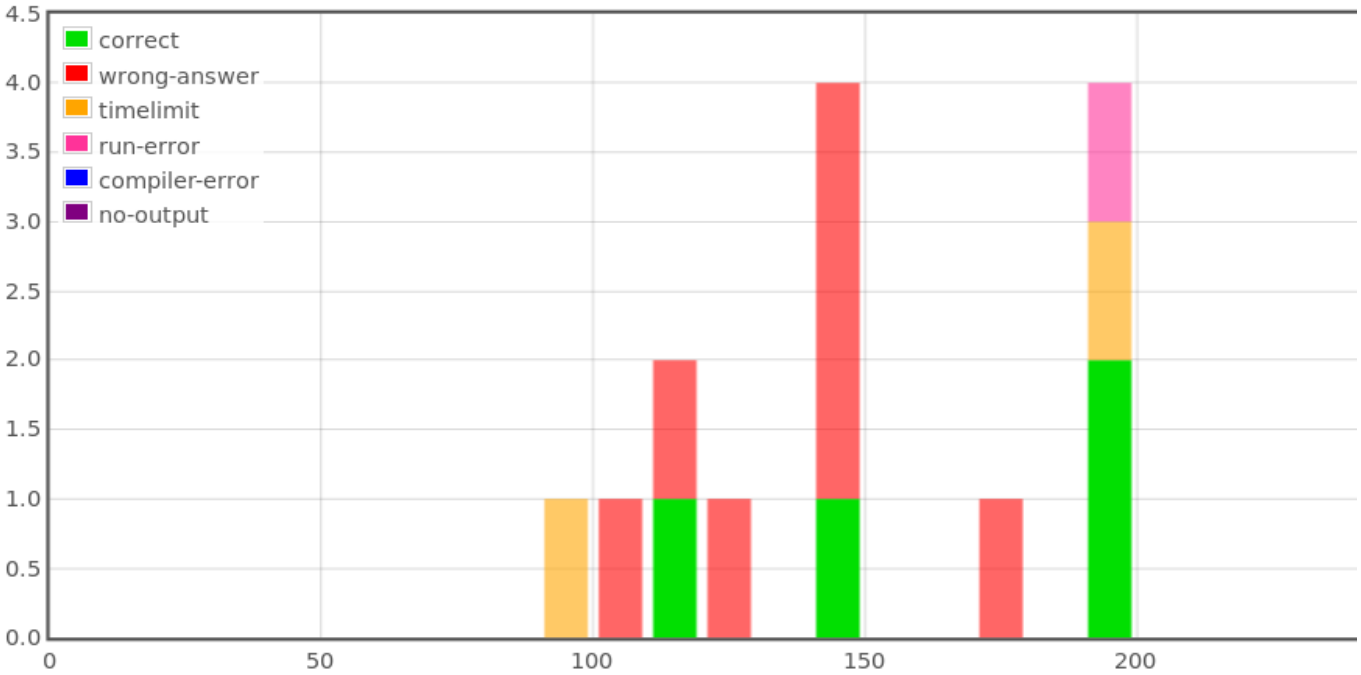


One possibility

Let M be manson's mark. Determining its bounding box. Now either inspect two particular points, or comparing the size of M with a threshold, in order to determine the type of M.

H – Travel Guide

Solved by 4 teams before freeze.
First solved after 118 min by **Team RaclETH**.



H – Travel Guide

Moving from a graph problem towards a vector problem

Three passes of Dijkstra algorithm to compute the distance from each POI to each node. $\mathcal{O}(|E| \times \log(|E|))$

We sort the vectors by lexicographical order.

$$
\begin{array}{ccc}
x_1 & y_1 & z_1 \\
x_2 & y_2 & z_2 \\
x_3 & y_3 & z_3 \\
\dots & & \\
x_n & y_n & z_n
\end{array}
$$

Key observation

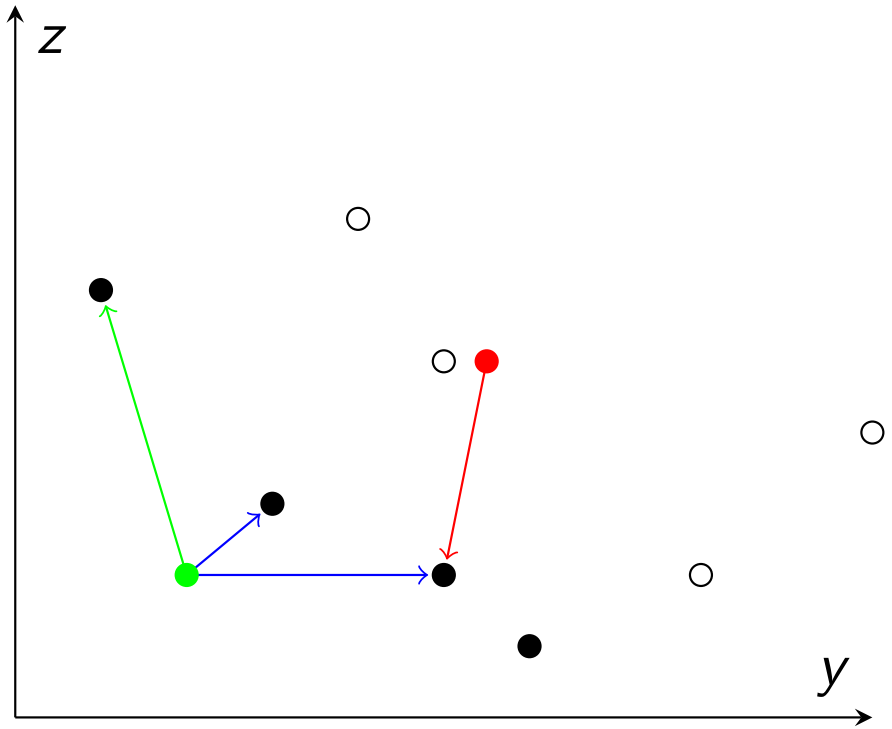
A vector v_i is minimal *iff* it is minimal among the vectors $v_1, \dots, v_i$ without considering the x coordinate.

H – Travel Guide

Idea: Maintain the 2D minimal vectors

Maintain a list of minimal vectors sorted by increasing y with a tree.

Note that it is sorted by decreasing z!



Checking that (y, z) is minimal

Is $z < z'$ for all (y', z') with $y' < y$?

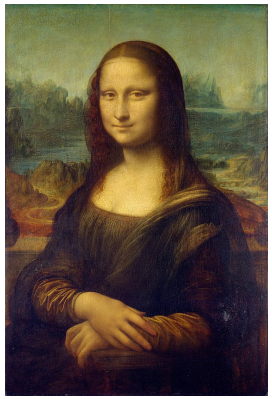
Inserting (y, z) as a minimal

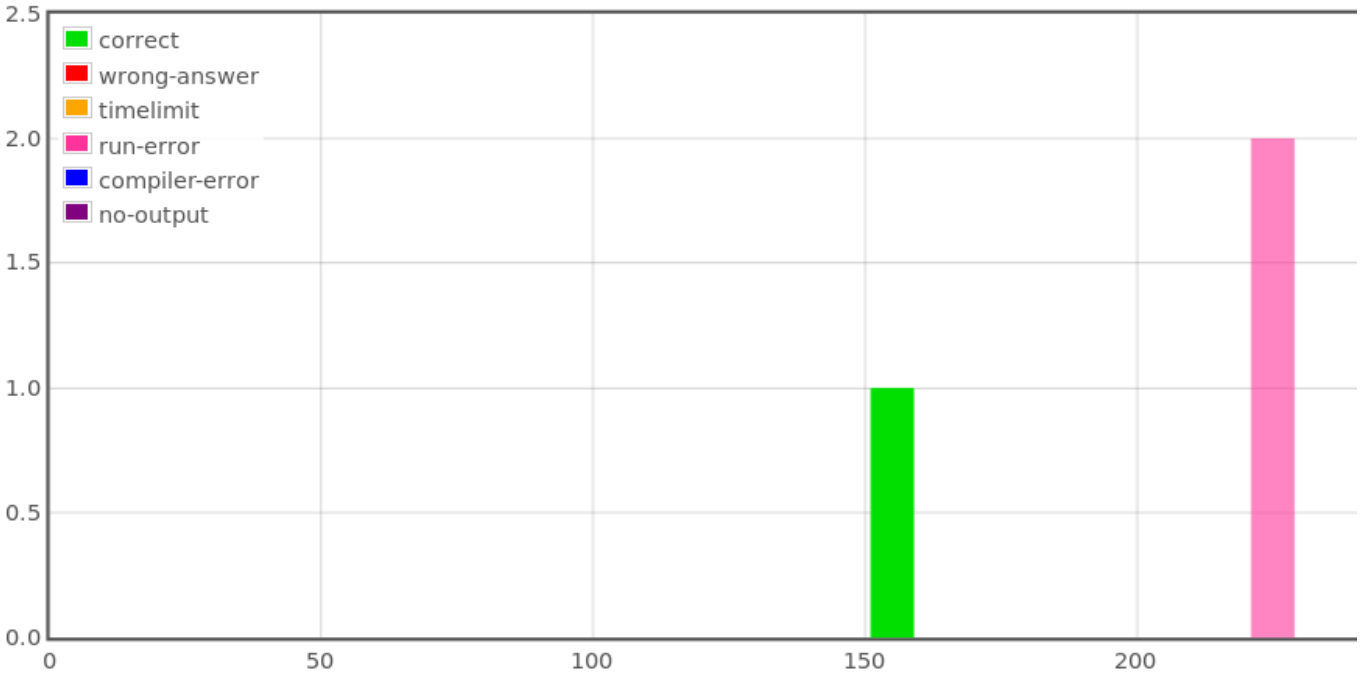
Remove all $z < z'$ and $y' < y$?

Note that you need to deal with duplicates.

J – Mona Lisa

Solved by 1 team before freeze.
First solved after 154 min by **ENS Ulm 1**.





J – Mona Lisa

Problem

Given 4 streams X_1, X_2, X_3, X_4 of pseudo-random n-bit integers, find $x_1 \in X_1, \ldots, x_4 \in X_4$ such that $x_1 \oplus x_2 \oplus x_3 \oplus x_4 = 0$.

Naive solution in $\mathcal{O}(2^{n/2})$ (exceeds time limit)

- Store $\mathcal{O}(2^{n/2})$ values from X_1 in a hashmap.
- Pick $x_3 \in X_3$ and $x_4 \in X_4$ arbitrarily.
- Iterate over $x_{2,i} \in X_2$, look for $x_{2,i} \oplus x_3 \oplus x_4$ in the hashmap.
- We expect to find a match after $\mathcal{O}(2^{n/2})$ steps by Birthday Paradox.

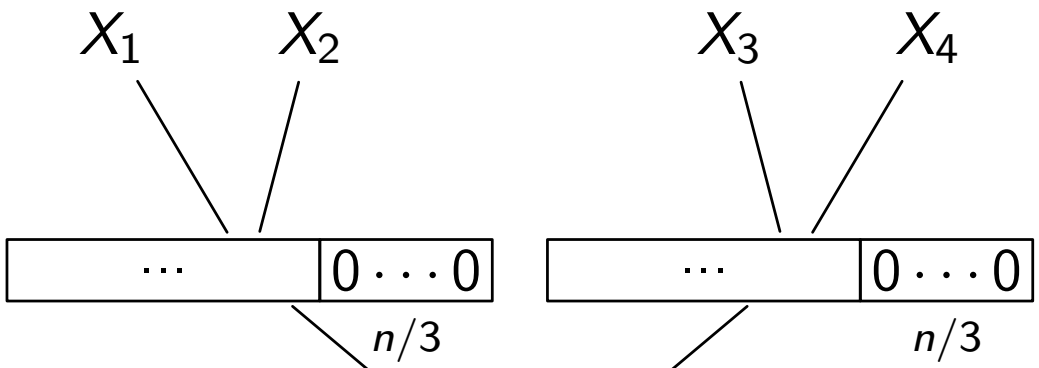
J – Mona Lisa

Solution in $\mathcal{O}(2^{n/3})$ (space and time)

- Build a list of $x_1 \oplus x_2$ when x_1 and x_2 match on their $n/3$ least significant bits. When using $\mathcal{O}(2^{n/3})$ values from X_1 and X_2, the list has $\mathcal{O}(2^{n/3})$ elements by Birthday Paradox.
- Do the same on X_3, X_4.
- The two lists generated have $\mathcal{O}(2^{n/3})$ elements of only $2n/3$ bits. By Birthday paradox, we expect $\mathcal{O}(1)$ matches.

J – Mona Lisa

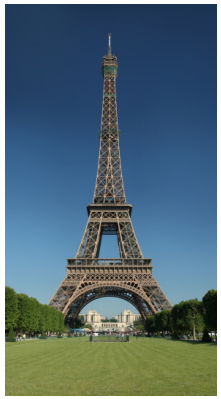
$\mathcal{O}(2^{n/3})$ collisions

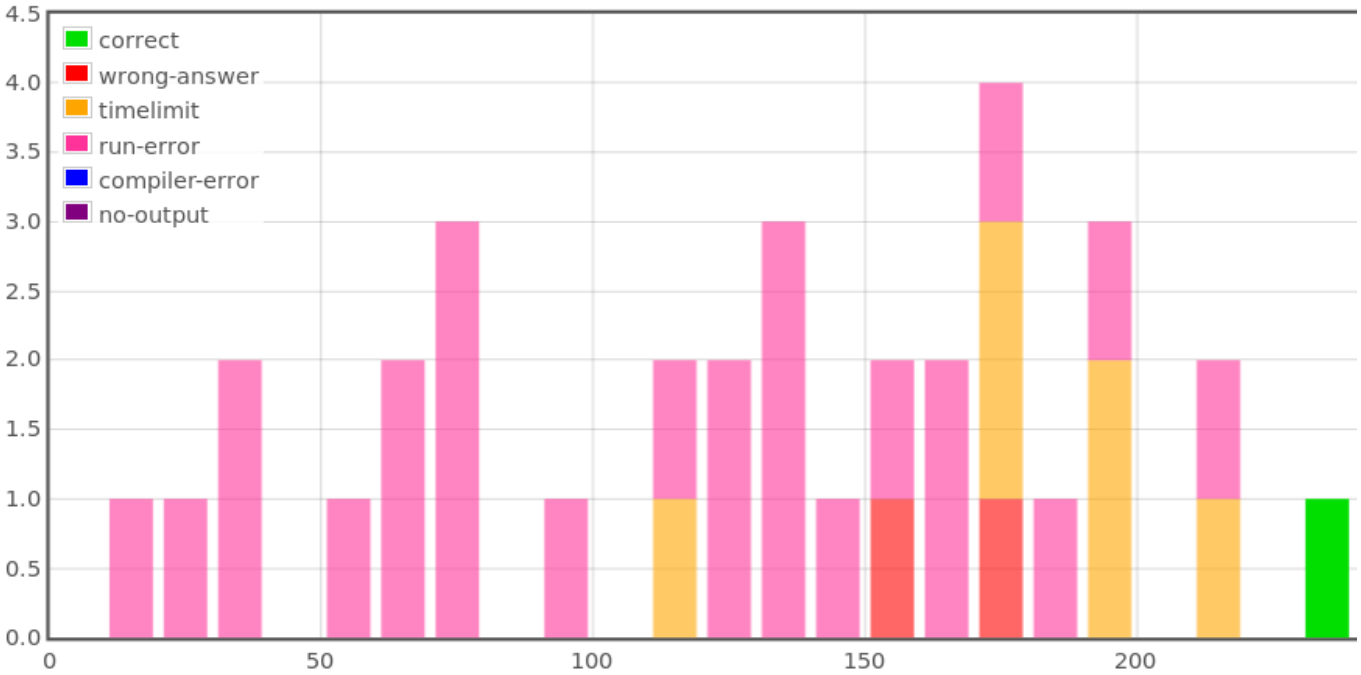


$\mathcal{O}(1)$ collisions

G – Strings

Solved by 1 team before freeze.
First solved after 235 min by **ENS Ulm 1**.





G – Strings

Source

Ropes: an Alternative to Strings
Boehm, Atkinson, Plass, 1995

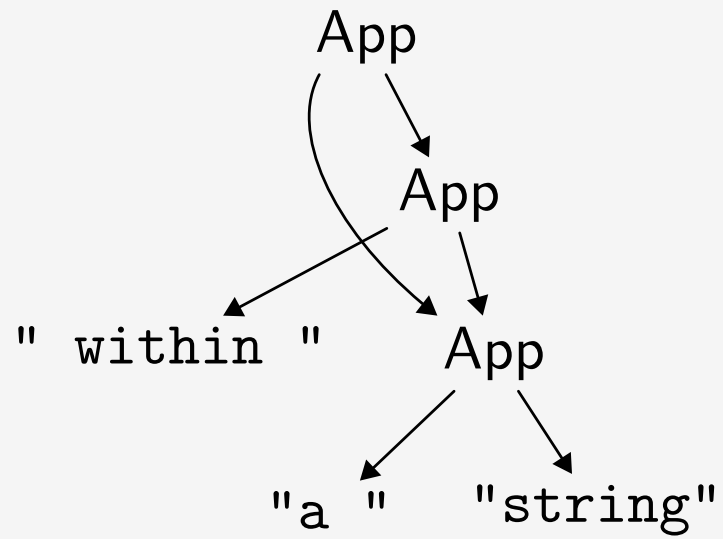
Main ideas

- do not concatenate strings, build binary trees instead
- ropes are immutable, thus sharing is possible

Implementation

- rope length in $\mathcal{O}(1)$
- substring of a leaf in $\mathcal{O}(1)$, else recursively in $\mathcal{O}(N)$

Example

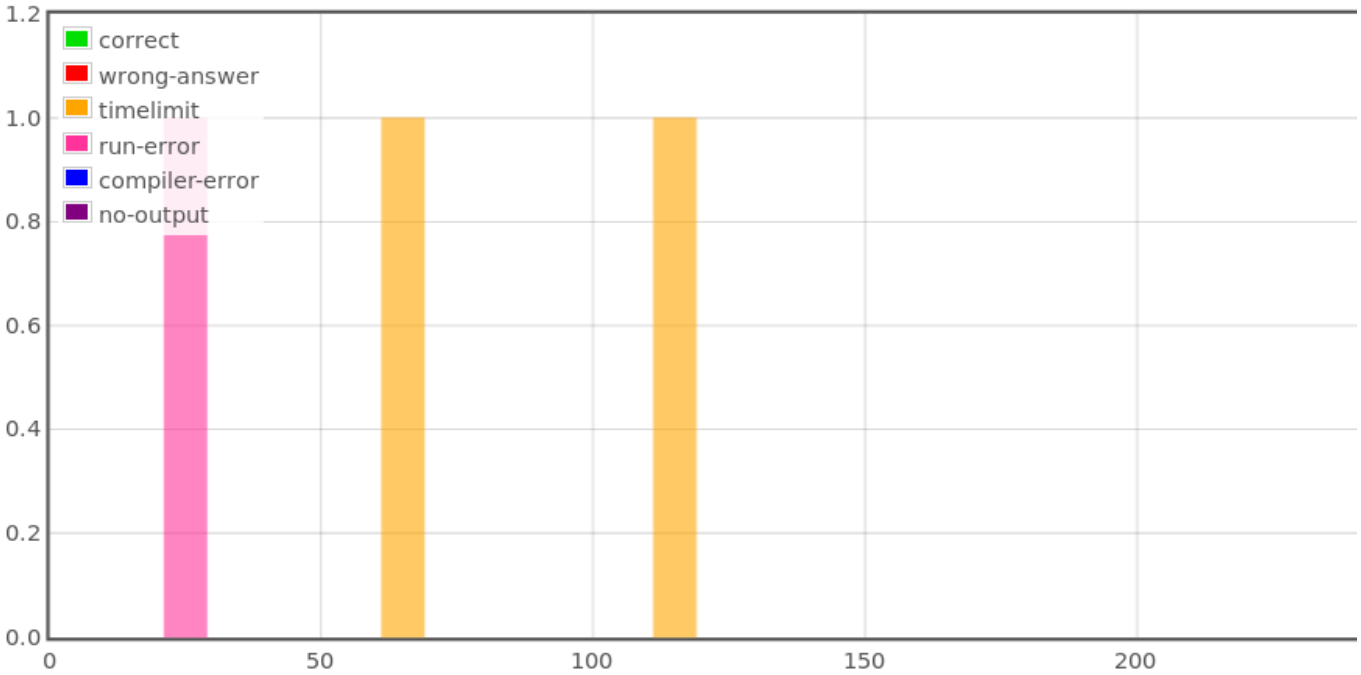


Overall complexity

$\mathcal{O}(N^2)$

C – Crosswords

Not solved before freeze.



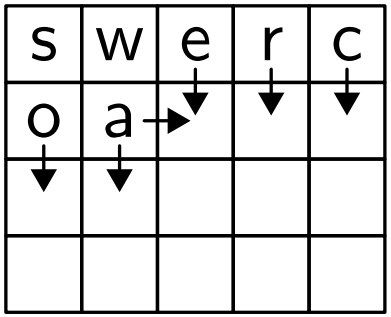
C – Crosswords

Source

Knuth, The Art of Computer Programming
forthcoming volume 4B, pre-fascicle 5b Introduction to Backtracking
Word Rectangles (page 8)

Backtracking Algorithm

- fill the grid, in any order
- $+1$ when completely filled



Data Structure

- build two tries, for horizontal and vertical words
- maintain pointers into these tries, for the columns and the row
- speed up the lookup at the intersection with sparse, sorted branches in your tries (see ex. 28)