Statement

Problem A: Circular Search

In this problem, you have to interactively find the secret point by covering the plane with circles.

Large Circles

In this problem, you have to interactively find the secret point by covering the plane with circles.

Point is in $[0, 10^4] \times [0, 10^4]$, but circle coordinates are up to 10^9.

Large Circles

In this problem, you have to interactively find the secret point by covering the plane with circles.

Point is in $[0, 10^4] \times [0, 10^4]$, but circle coordinates are up to 10^9.

Very large circles with very far centers act like half-planes.

Large Circles

In this problem, you have to interactively find the secret point by covering the plane with circles.

Point is in $[0, 10^4] \times [0, 10^4]$, but circle coordinates are up to 10^9.

Very large circles with very far centers act like half-planes.

Out-of-the-box trick:
Circle at $(-10^9 + c, -1)$ with radius 10^9 acts like a half-plane $x < c$.
Circle at $(-1, -10^9 + c)$ with radius 10^9 acts like a half-plane $y < c$.

Large Circles

In this problem, you have to interactively find the secret point by covering the plane with circles.

Point is in $[0, 10^4] \times [0, 10^4]$, but circle coordinates are up to 10^9.

Very large circles with very far centers act like half-planes.

Out-of-the-box trick:
Circle at $(-10^9 + c, -1)$ with radius 10^9 acts like a half-plane $x < c$.
Circle at $(-1, -10^9 + c)$ with radius 10^9 acts like a half-plane $y < c$.

Binary search!
Solve by finding coordinates x and y independently.

Large Circles

In this problem, you have to interactively find the secret point by covering the plane with circles.

Point is in $[0, 10^4] \times [0, 10^4]$, but circle coordinates are up to 10^9.

Very large circles with very far centers act like half-planes.

Out-of-the-box trick:
Circle at $(-10^9 + c, -1)$ with radius 10^9 acts like a half-plane $x < c$.
Circle at $(-1, -10^9 + c)$ with radius 10^9 acts like a half-plane $y < c$.

Binary search!
Solve by finding coordinates x and y independently.

Number of questions: $2 \cdot \lceil \log_2 10\,001 \rceil = 28$.

Statement

Problem B: Coindays Destroyed

In this problem, you have to simulate transfers between addresses and calculate destroyed potentials.

Stacks

In this problem, you have to simulate transfers between addresses and calculate destroyed potentials.

Coins at an address are last-in, first-out.

Stacks

In this problem, you have to simulate transfers between addresses and calculate destroyed potentials.

Coins at an address are last-in, first-out.

Store them as a stack of pairs (*moment*, *amount*).

Stacks

In this problem, you have to simulate transfers between addresses and calculate destroyed potentials.

Coins at an address are last-in, first-out.

Store them as a stack of pairs (*moment*, *amount*).

The whole data structure is `map <address, stack <record> >`.
Address is a string, record is a pair.

Stacks

In this problem, you have to simulate transfers between addresses and calculate destroyed potentials.

Coins at an address are last-in, first-out.

Store them as a stack of pairs (*moment*, *amount*).

The whole data structure is `map <address, stack <record> >`.
Address is a string, record is a pair.

Transfer: we want amount c at time t.
We have record (*moment*, *amount*) on top of the stack.
We use $u = \min(amount, c)$ coins.
We destroy $u \cdot (c - moment)$ coinseconds.
Now we want amount $c - u$ at time t.

Complexity

In this problem, you have to simulate transfers between addresses and calculate destroyed potentials.

One transfer creates one record.

Complexity

In this problem, you have to simulate transfers between addresses and calculate destroyed potentials.

One transfer creates one record.

If a transfer uses more than one record, all records, except maybe the oldest, are drained completely.

Complexity

In this problem, you have to simulate transfers between addresses and calculate destroyed potentials.

One transfer creates one record.

If a transfer uses more than one record, all records, except maybe the oldest, are drained completely.

The first k transfers create k records.
And use at most $2k$ records.

Implementation

In this problem, you have to simulate transfers between addresses and calculate destroyed potentials.

Initially, address 00000000 has an infinite amount.

Implementation

In this problem, you have to simulate transfers between addresses and calculate destroyed potentials.

Initially, address 00000000 has an infinite amount.

Careful with floating point!
$1 - 0.1 - 0.2 - 0.3 - 0.4$ is likely not zero.

Statement

Problem C: Edit Program

In this problem, you have to find out whether edit distance between two n-digit binary strings is n.

Case Analysis

In this problem, you have to find out whether edit distance between two n-digit binary strings is n.

Case 1: $s \neq \text{invert}(t)$
Just use n "." and "^" steps. Example:
$s = 01100$
$t = 11010$
$p =$ ^.^^.

Case Analysis

In this problem, you have to find out whether edit distance between two n-digit binary strings is n.

Case 1: $s \neq \text{invert}(t)$
Just use n "." and "^" steps. Example:
$s = 01100$
$t = 11010$
$p =$ ^.^^.

Case 2: $s = 00000$, $t = 11111$ (example)
Nothing can be done.

Case Analysis

In this problem, you have to find out whether edit distance between two n-digit binary strings is n.

Case 1: $s \neq \text{invert}(t)$
Just use n "." and "^" steps. Example:
$s = 01100$
$t = 11010$
$p =$ ^.^^.

Case 2: $s = 00000$, $t = 11111$ (example)
Nothing can be done.

Case 3: $s = 00011$, $t = 11100$ (example)
Nothing can be done.

Case Analysis

In this problem, you have to find out whether edit distance between two n-digit binary strings is n.

Case 4: $s = 011110$, $t = 100001$ (example)

```
|0 1 1 1 1 0    (x)
|1 1 1 1 0      (.)
 1|1 1 1 0      (^)
 1 0|1 1 0      (^)
 1 0 0|1 0      (^)
 1 0 0 0|0      (.)
 1 0 0 0 0|     (1)
 1 0 0 0 0|1
```

Case Analysis

In this problem, you have to find out whether edit distance between two n-digit binary strings is n.

Case 4: $s = 011110$, $t = 100001$ (example)

```
|0 1 1 1 1 0    (x)
|1 1 1 1 0      (.)
 1|1 1 1 0      (^)
 1 0|1 1 0      (^)
 1 0 0|1 0      (^)
 1 0 0 0|0      (.)
 1 0 0 0 0|     (1)
 1 0 0 0 0|1
```

We can just find any one such substring, and "^" all other digits.

Statement

Problem D: Faulty Keyboard

In this problem, given a large text offline and a small piece of it as a test, you have to find which letter disappeared around half of the time.

Letter Frequencies

In this problem, given a large text offline and a small piece of it as a test, you have to find which letter disappeared around half of the time.

Idea 1: use frequencies of letters!
$f(\text{a}) = 0.08167$, $f(\text{b}) = 0.01492$, $f(\text{c}) = 0.02782$, ...

Letter Frequencies

In this problem, given a large text offline and a small piece of it as a test, you have to find which letter disappeared around half of the time.

Idea 1: use frequencies of letters!
$f(\mathtt{a}) = 0.08167$, $f(\mathtt{b}) = 0.01492$, $f(\mathtt{c}) = 0.02782$, ...

Does not work: some characters appear prominently in some parts of the text, but disappear in other parts, messing with the frequencies of letters such as "k".

Letter Frequencies

In this problem, given a large text offline and a small piece of it as a test, you have to find which letter disappeared around half of the time.

Idea 1: use frequencies of letters!
$f(\mathrm{a}) = 0.08167$, $f(\mathrm{b}) = 0.01492$, $f(\mathrm{c}) = 0.02782$, ...

Does not work: some characters appear prominently in some parts of the text, but disappear in other parts, messing with the frequencies of letters such as "`k`".

Hard letters: `JQXZ`.

Letter Frequencies

In this problem, given a large text offline and a small piece of it as a test, you have to find which letter disappeared around half of the time.

Idea 1: use frequencies of letters!
$f(\mathtt{a}) = 0.08167$, $f(\mathtt{b}) = 0.01492$, $f(\mathtt{c}) = 0.02782$, ...

Does not work: some characters appear prominently in some parts of the text, but disappear in other parts, messing with the frequencies of letters such as "k".

Hard letters: JQXZ.

Statistics: the minimum number of occurrences in 100 000 consecutive bytes of text (hardest tests) are 27 for "j" and "z", 37 for "q", 77 for "x", 255 for "k", > 500 for other letters.

Words and Non-Words

In this problem, given a large text offline and a small piece of it as a test, you have to find which letter disappeared around half of the time.

Idea 2: look for the wrong words.

Words and Non-Words

In this problem, given a large text offline and a small piece of it as a test, you have to find which letter disappeared around half of the time.

Idea 2: look for the wrong words.

Offline:
Memoize all words from the text. For each word: try removing one letter of the alphabet in every possible way. We got a list of possible non-words.

Words and Non-Words

In this problem, given a large text offline and a small piece of it as a test, you have to find which letter disappeared around half of the time.

Idea 2: look for the wrong words.

Offline:
Memoize all words from the text. For each word: try removing one letter of the alphabet in every possible way. We got a list of possible non-words.

Good non-words are the ones which we got in a unique way.
For each of the 26 letters, store $\leqslant 500$ random good non-words as data in the solution ($\approx 130\,000$ bytes).

Words and Non-Words

In this problem, given a large text offline and a small piece of it as a test, you have to find which letter disappeared around half of the time.

Idea 2: look for the wrong words.

Offline:
Memoize all words from the text. For each word: try removing one letter of the alphabet in every possible way. We got a list of possible non-words.

Good non-words are the ones which we got in a unique way.
For each of the 26 letters, store $\leqslant 500$ random good non-words as data in the solution ($\approx 130\,000$ bytes).

Solution:
When we see a good non-word we know, we immediately know the missing letter.

Other Ideas

In this problem, given a large text offline and a small piece of it as a test, you have to find which letter disappeared around half of the time.

Idea 3: (insert your idea here)

Statement

Problem E: Free Roman Numerals

In this problem, you have to transform decimal numbers into free Roman numerals and back.

Parsing — Right to Left

In this problem, you have to transform decimal numbers into free Roman numerals and back.

`XIIVL`: parse from right to left recursively

Parsing — Right to Left

In this problem, you have to transform decimal numbers into free Roman numerals and back.

`XIIVL`: parse from right to left recursively

$f(\texttt{XIIVL}) = \texttt{L} - f(\texttt{XIIV})$

Parsing — Right to Left

In this problem, you have to transform decimal numbers into free Roman numerals and back.

`XIIVL`: parse from right to left recursively

$f(\texttt{XIIVL}) = \texttt{L} - f(\texttt{XIIV})$

$f(\texttt{XIIV}) = \texttt{V} - f(\texttt{II}) + f(\texttt{X})$

Parsing — Right to Left

In this problem, you have to transform decimal numbers into free Roman numerals and back.

`XIIVL`: parse from right to left recursively

$$f(\texttt{XIIVL}) = \texttt{L} - f(\texttt{XIIV})$$

$$f(\texttt{XIIV}) = \texttt{V} - f(\texttt{II}) + f(\texttt{X})$$

$$f(\texttt{II}) = \texttt{I} + f(\texttt{I})$$

Parsing — Right to Left

In this problem, you have to transform decimal numbers into free Roman numerals and back.

`XIIVL`: parse from right to left recursively

$f(\texttt{XIIVL}) = \texttt{L} - f(\texttt{XIIV})$

$f(\texttt{XIIV}) = \texttt{V} - f(\texttt{II}) + f(\texttt{X})$

$f(\texttt{II}) = \texttt{I} + f(\texttt{I})$

$f(\texttt{I}) = \texttt{I}$

Subproblems — Optimality

In this problem, you have to transform decimal numbers into free Roman numerals and back.

Add: `XXXII` is shortest $\Rightarrow$ `XXXI` is shortest

Subproblems — Optimality

In this problem, you have to transform decimal numbers into free Roman numerals and back.

Add: `XXXII` is shortest $\Rightarrow$ `XXXI` is shortest

Subtract: `IIXXL` is shortest $\Rightarrow$ `IIXX` is shortest

Subproblems — Optimality

In this problem, you have to transform decimal numbers into free Roman numerals and back.

Add: `XXXII` is shortest $\Rightarrow$ `XXXI` is shortest

Subtract: `IIXXL` is shortest $\Rightarrow$ `IIXX` is shortest

Each shortest expression is constructed by adding one Roman digit to the right of some smaller Roman numeral.

Statement

Problem F: Known Problem

In this problem, you have to interactively find the sum of 100 000 000 *pseudorandom integers.*

Guess the Seed

In this problem, you have to interactively find the sum of 100 000 000 *pseudorandom integers.*

The first x is 5. There are 2148 possibilities for s:
5, 1 000 005, 2 000 005, ..., 2 147 000 005.

Guess the Seed

In this problem, you have to interactively find the sum of 100 000 000 *pseudorandom integers.*

The first x is 5. There are 2148 possibilities for s:
5, 1 000 005, 2 000 005, ..., 2 147 000 005.

The second x is 327 695. Now, there are only two possibilities for s:
327 695 and 1 536 327 695.

Guess the Seed

In this problem, you have to interactively find the sum of 100 000 000 *pseudorandom integers.*

The first x is 5. There are 2148 possibilities for s:
5, 1 000 005, 2 000 005, ..., 2 147 000 005.

The second x is 327 695. Now, there are only two possibilities for s:
327 695 and 1 536 327 695.

The third x is 966 125. Only one possible s left: 1 966 125.

Guess the Seed

In this problem, you have to interactively find the sum of 100 000 000 *pseudorandom integers.*

The first x is 5. There are 2148 possibilities for s:
5, 1 000 005, 2 000 005, ..., 2 147 000 005.

The second x is 327 695. Now, there are only two possibilities for s:
327 695 and 1 536 327 695.

The third x is 966 125. Only one possible s left: 1 966 125.

From now on, we can construct all the following values of x without asking anything!

How Long until Unique

In this problem, you have to interactively find the sum of 100 000 000 *pseudorandom integers.*

How many times do we have to ask?

How Long until Unique

In this problem, you have to interactively find the sum of 100 000 000 *pseudorandom integers.*

How many times do we have to ask?

We have checked it in advance with all 2^{31} possible seeds.
In the majority of the cases, two questions are enough.
In the remaining cases, we are left with only two possibilities, and the third question always removes the uncertainty.

How Long until Unique

In this problem, you have to interactively find the sum of 100 000 000 *pseudorandom integers.*

How many times do we have to ask?

We have checked it in advance with all 2^{31} possible seeds.
In the majority of the cases, two questions are enough.
In the remaining cases, we are left with only two possibilities, and the third question always removes the uncertainty.

Detail: remember that s can be zero (the whole sum is zero then).

Statement

Problem G: K-th Bishop Covering

In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.

General Notes

In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.

Good news: the number of bishops is always n.

General Notes

In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.

Good news: the number of bishops is always n.

Bad news: the number k does not fit into `int64`.

General Notes

In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.

Good news: the number of bishops is always n.

Bad news: the number k does not fit into `int64`.

We will consider black and white squares independently.

General Notes

In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.

Good news: the number of bishops is always n.

Bad news: the number k does not fit into `int64`.

We will consider black and white squares independently.

Consider the squares in natural order.

General Notes

In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.

Good news: the number of bishops is always n.

Bad news: the number k does not fit into `int64`.

We will consider black and white squares independently.

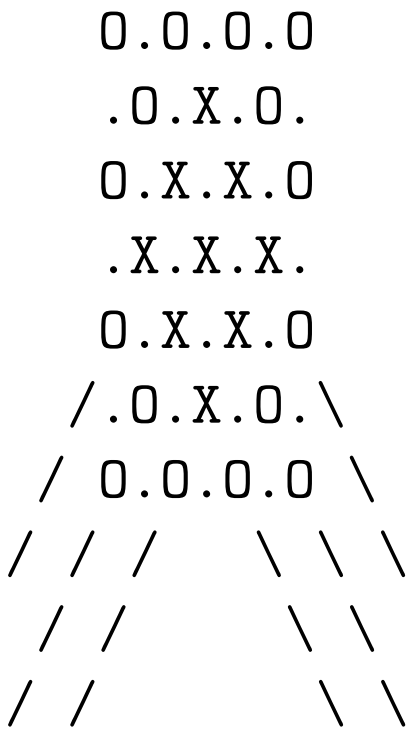
Consider the squares in natural order.

For selected color, and part of the board already considered and finalized, find a way to count the number of remaining configurations in polynomial time.

Case 1: Easy

In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.

Easy case: odd board, one color.



Example: $n = 7$.

Three bishops are enough.

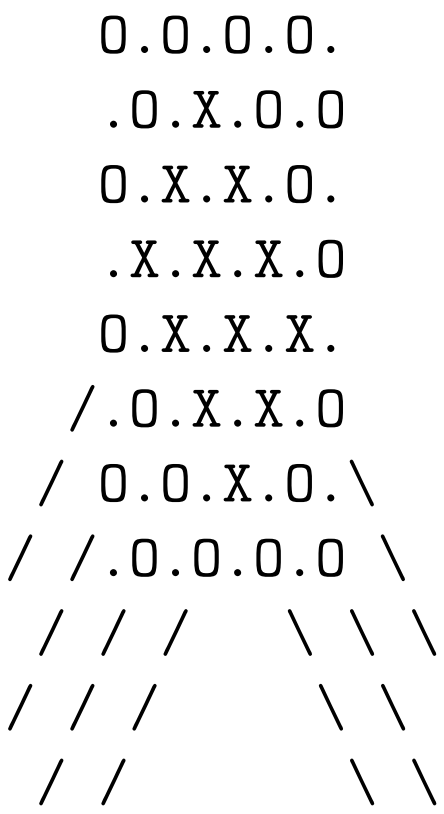
The two $r - c$ corners are covered if each of the three $r - c$ diagonals (/) is covered.

The two $r + c$ corners are covered if each of the three $r + c$ diagonals (\) is covered.

Case 2: Medium

In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.

Medium case: even board, any color.



Example: $n = 8$.

Four bishops are enough.

The two $r - c$ corners are covered if each of the four $r - c$ diagonals (/) is covered.

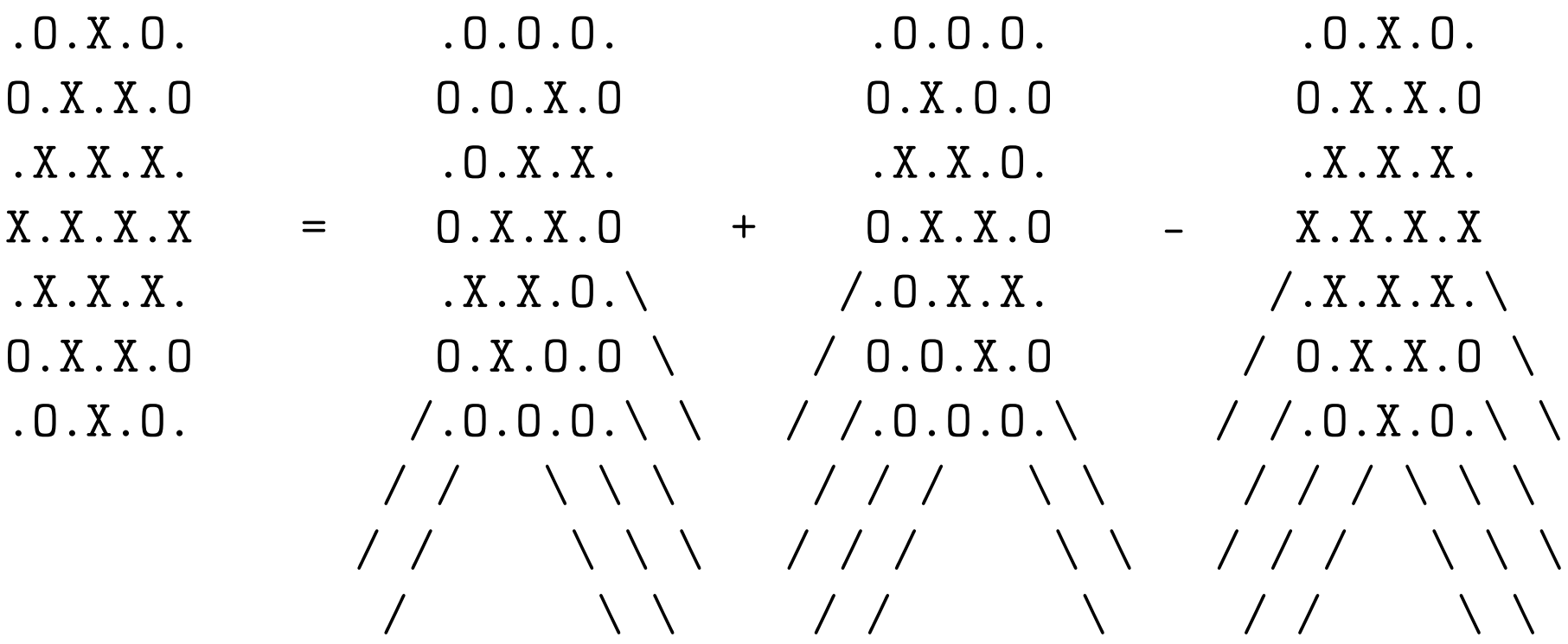
The two $r + c$ corners are covered if each of the three $r + c$ diagonals (\) is covered.

One bishop can freely choose its $r + c$ position!

Case 3: Hard

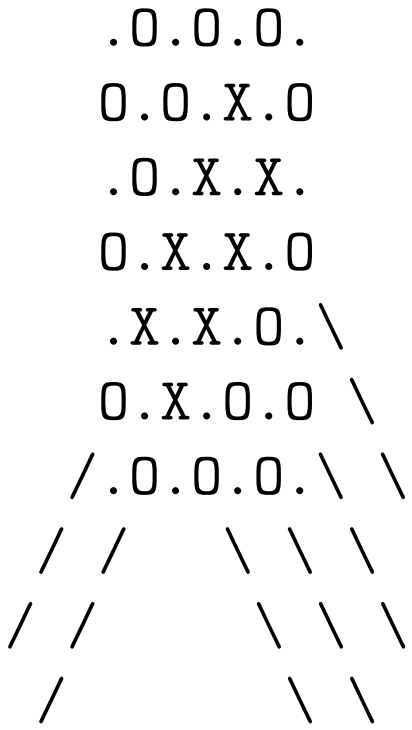
In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.

Hard case: odd board, other color. Use inclusion-exclusion principle.



Case 3: Hard

In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.



Example: $n = 7$.

Four bishops are enough.

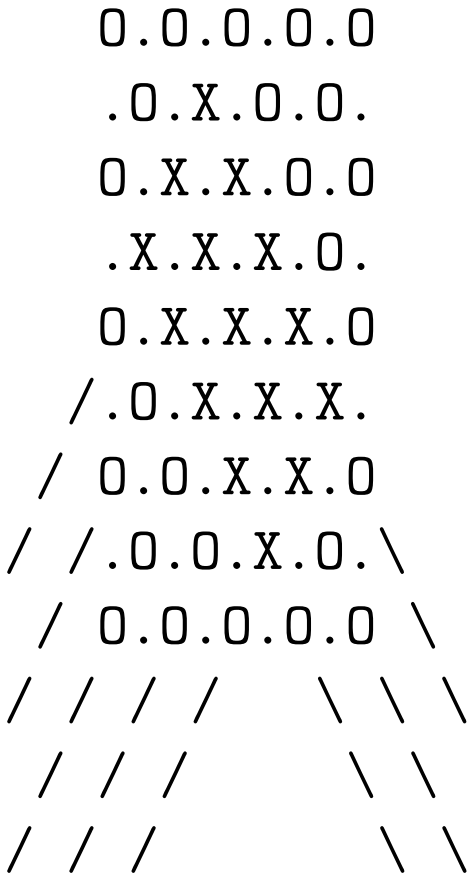
The two $r - c$ corners are covered if each of the two $r - c$ diagonals (/) is covered.

The two $r + c$ corners are covered if each of the four $r + c$ diagonals (\) is covered.

Two bishops can freely choose their $r - c$ positions!

Case 3: Hard

In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.



Example: $n = 9$.

Five bishops are enough.

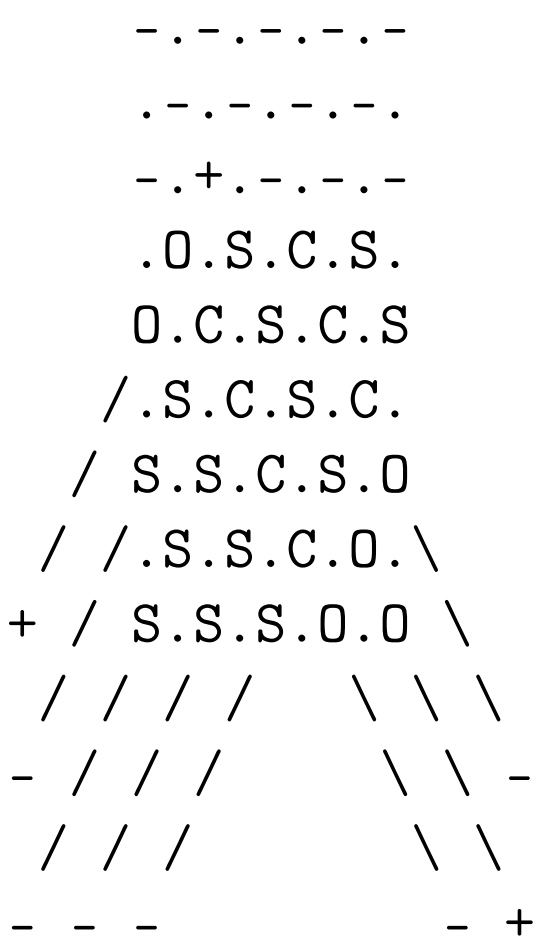
The two $r - c$ corners are covered if each of the five $r - c$ diagonals (/) is covered.

The two $r + c$ corners are covered if each of the three $r + c$ diagonals (\) is covered.

Two bishops can freely choose their $r - c$ positions!

Case 3: Hard

In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.



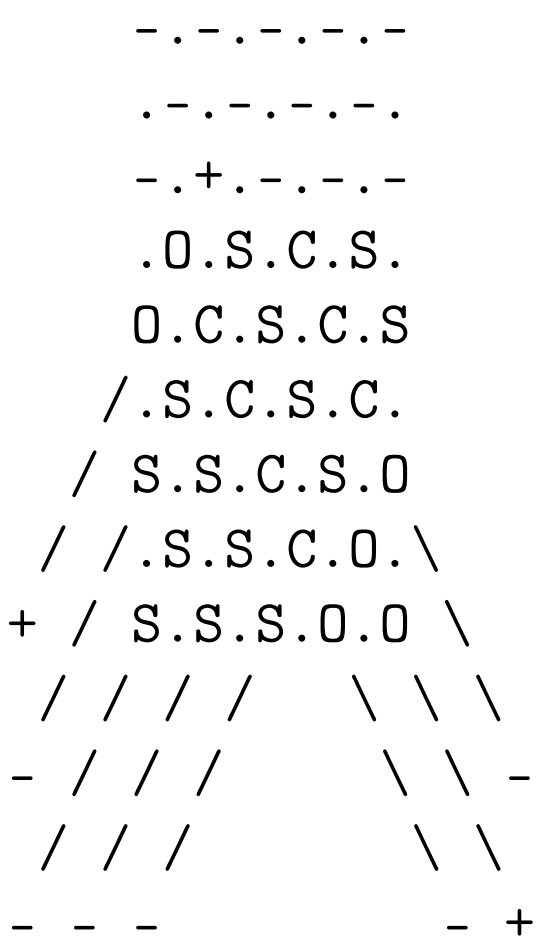
Example: $n = 9$, $free = 2$

C are common squares which contribute to both their diagonals.
S are special squares which contribute only to the essential (five) diagonals.

We select a position for a free bishop in S, or a position for a common bishop in C.

Case 3: Hard

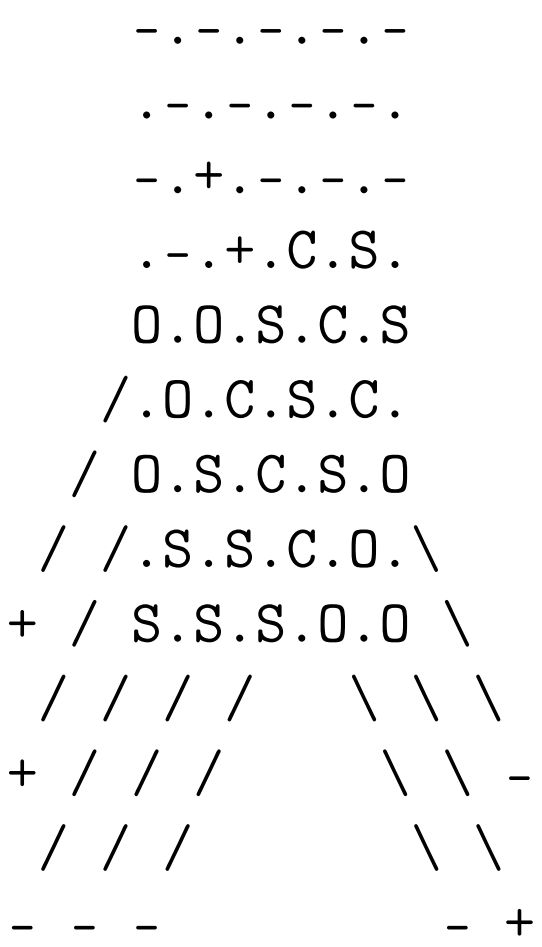
In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.



$$f(\text{pos}) =$$
$$3 \cdot 4 \cdot 2 \cdot 1 + \quad (\texttt{SSCC})$$
$$3 \cdot 2 \cdot 5 \cdot 1 + \quad (\texttt{SCSC})$$
$$3 \cdot 2 \cdot 1 \cdot 5 + \quad (\texttt{SCCS})$$
$$1 \cdot 5 \cdot 5 \cdot 1 + \quad (\texttt{CSSC})$$
$$1 \cdot 5 \cdot 1 \cdot 5 + \quad (\texttt{CSCS})$$
$$1 \cdot 1 \cdot 6 \cdot 5 = \quad (\texttt{CCSS})$$
$$24 + 30 + 30 + 25 + 25 + 30 = 164.$$

Case 3: Hard

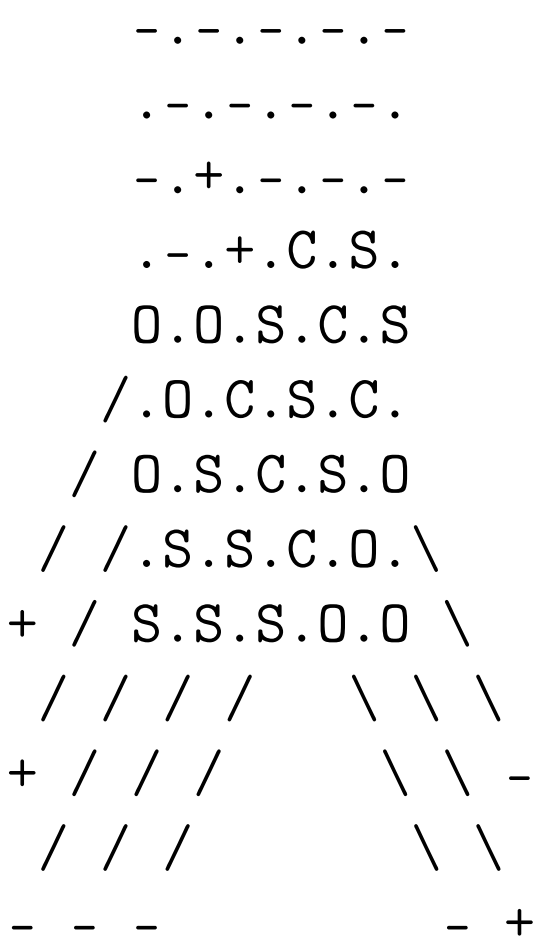
In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.



Another example: $n = 9$, $free = 1$

Case 3: Hard

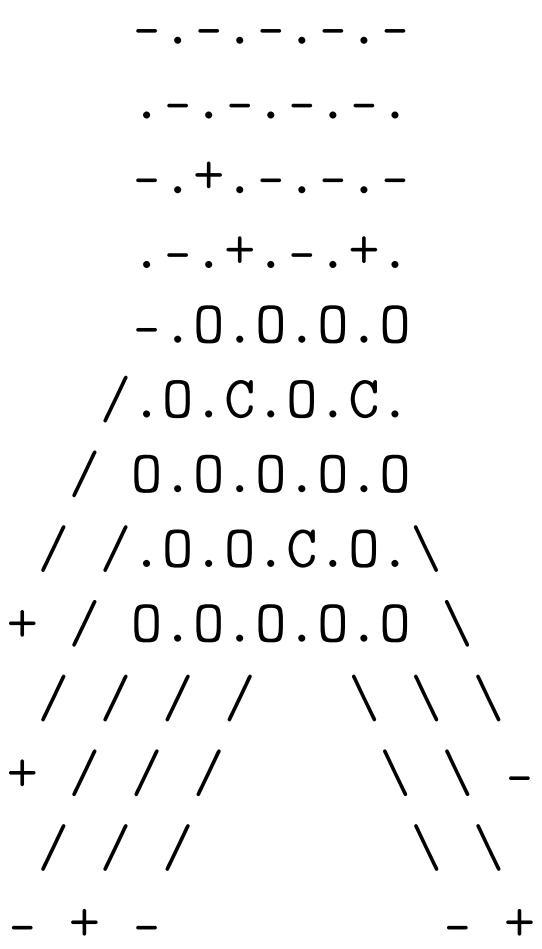
In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.



$$f(\text{pos}) =$$
$$4 \cdot 2 \cdot 1 + \quad (\texttt{SCC})$$
$$2 \cdot 5 \cdot 1 + \quad (\texttt{CSC})$$
$$2 \cdot 1 \cdot 5 = \quad (\texttt{CCS})$$
$$8 + 10 + 10 = 28.$$

Case 3: Hard

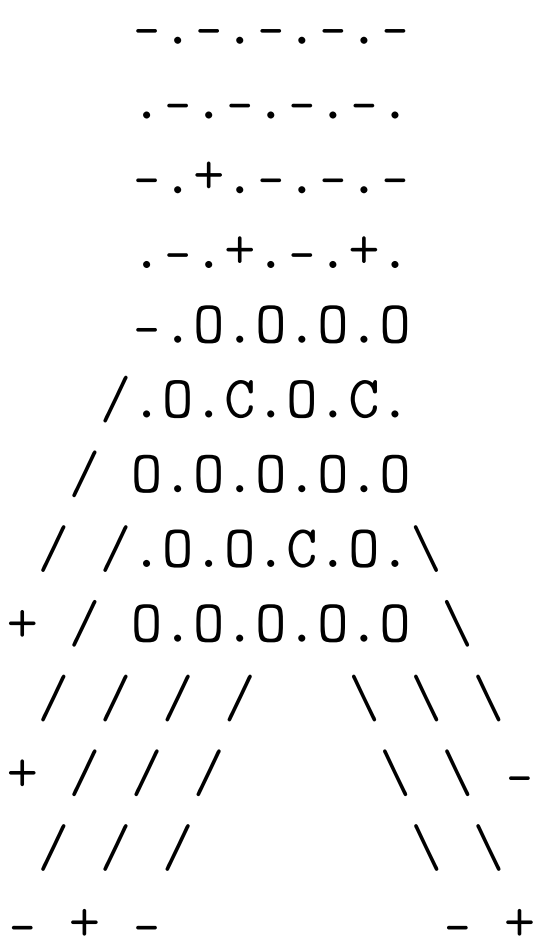
In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.



Yet another example: $n = 9$, $free = 0$

Case 3: Hard

In this problem, you have to find the lexicographically k-th minimal covering of an $n \times n$ chessboard by bishops.



$f(\text{pos}) =$

$1 \cdot 1 =$ (CC)

$1.$

Statement

Problem H: Luck

In this problem, you have to distinguish between 16 *possible distributions of throwing dice.*

Find the Distributions

In this problem, you have to distinguish between 16 *possible distributions of throwing dice.*

Phase 1: find the 16 possible distributions.

Find the Distributions

In this problem, you have to distinguish between 16 *possible distributions of throwing dice.*

Phase 1: find the 16 possible distributions.

Way 1: simulation.
For each distribution, generate many random outcomes.
For each possible outcome, remember how many times it happened.

Find the Distributions

In this problem, you have to distinguish between 16 *possible distributions of throwing dice.*

Phase 1: find the 16 possible distributions.

Way 1: simulation.
For each distribution, generate many random outcomes.
For each possible outcome, remember how many times it happened.

Way 2: calculation.
Maintain the probabilities of each possible outcome.
Write functions to take the sum, minimum and maximum of two distributions.

Find the Most Likely

In this problem, you have to distinguish between 16 *possible distributions of throwing dice.*

Phase 2: compare to the distribution we got as input.

Find the Most Likely

In this problem, you have to distinguish between 16 *possible distributions of throwing dice.*

Phase 2: compare to the distribution we got as input.

Way 1: moments.
Calculate mean and variance, norm appropriately.
Find the closest point on the mean $\times$ variance plane.
Warning: only one of them is not enough!
Two are sufficient.

Find the Most Likely

In this problem, you have to distinguish between 16 *possible distributions of throwing dice.*

Phase 2: compare to the distribution we got as input.

Way 2: maximum likelihood.
For each event we observe, for each of the 16 hypotheses, we know the probability this event happens under this hypothesis.
So, for each of the 16 hypotheses, calculate the total probability of all observed events: the product of individual probabilities.
The hypothesis with the greatest total probability wins.

Find the Most Likely

In this problem, you have to distinguish between 16 *possible distributions of throwing dice.*

Phase 2: compare to the distribution we got as input.

Way 2: maximum likelihood.
For each event we observe, for each of the 16 hypotheses, we know the probability this event happens under this hypothesis.
So, for each of the 16 hypotheses, calculate the total probability of all observed events: the product of individual probabilities.
The hypothesis with the greatest total probability wins.

In practice, we sum the logarithms of probabilities instead of multiplying probabilities themselves.

Statement

Problem I: Polyomino Packing

In this problem, you have to put the given polyominoes tightly in the given box.

Remarks

In this problem, you have to put the given polyominoes tightly in the given box.

Only a few pointers here, not a full solution outline!

Remarks

In this problem, you have to put the given polyominoes tightly in the given box.

Only a few pointers here, not a full solution outline!

Fill the board in the natural order: row by row, each row from left to right.

Remarks

In this problem, you have to put the given polyominoes tightly in the given box.

Only a few pointers here, not a full solution outline!

Fill the board in the natural order: row by row, each row from left to right.

Try to use the larger polyominoes first, so that in the end, we will have to deal only with 3-square polyominoes.

Remarks

In this problem, you have to put the given polyominoes tightly in the given box.

Only a few pointers here, not a full solution outline!

Fill the board in the natural order: row by row, each row from left to right.

Try to use the larger polyominoes first, so that in the end, we will have to deal only with 3-square polyominoes.

Maintain good diversity of the polyominoes left.

Remarks

In this problem, you have to put the given polyominoes tightly in the given box.

It is better to put a polyomino which has more corners.

Remarks

In this problem, you have to put the given polyominoes tightly in the given box.

It is better to put a polyomino which has more corners.

It is better to put a polyomino so that the resulting figure has less corners.

Remarks

In this problem, you have to put the given polyominoes tightly in the given box.

It is better to put a polyomino which has more corners.

It is better to put a polyomino so that the resulting figure has less corners.

Holes isolated from above and below are bad.

Remarks

In this problem, you have to put the given polyominoes tightly in the given box.

Randomize the search, and roll back if there is no way to proceed.

Remarks

In this problem, you have to put the given polyominoes tightly in the given box.

Randomize the search, and roll back if there is no way to proceed.

If there is still no way to proceed, roll back some more.

Remarks

In this problem, you have to put the given polyominoes tightly in the given box.

Randomize the search, and roll back if there is no way to proceed.

If there is still no way to proceed, roll back some more.

...Experiment! You have half the tests.

Statement

Problem J: Switches

In this problem, you have to count the number of possible configurations of width k after using the given manipulators.

Invertibility

In this problem, you have to count the number of possible configurations of width k after using the given manipulators.

All moves are invertible: just do them again in any order.

Invertibility

In this problem, you have to count the number of possible configurations of width k after using the given manipulators.

All moves are invertible: just do them again in any order.

Two manipulators a and b are equivalent to a and $a \oplus b$.

Invertibility

In this problem, you have to count the number of possible configurations of width k after using the given manipulators.

All moves are invertible: just do them again in any order.

Two manipulators a and b are equivalent to a and $a \oplus b$.

Two manipulators a and b are equivalent to a and $a \oplus (b$ shifted $\pm s)$.

Greatest Common Divisor

In this problem, you have to count the number of possible configurations of width k after using the given manipulators.

Two manipulators a and b are equivalent to one: $\mathrm{GCD}(a, b)$.

Greatest Common Divisor

In this problem, you have to count the number of possible configurations of width k after using the given manipulators.

Two manipulators a and b are equivalent to one: $\mathrm{GCD}(a, b)$.

Example:
111001 and 1001
011101 and 1001
001111 and 1001
000110 and 1001
000011 and 1001
000011 and 0101
000011 and 0011
000011 and 0000

Greatest Common Divisor

In this problem, you have to count the number of possible configurations of width k after using the given manipulators.

Two manipulators a and b are equivalent to one: $\mathrm{GCD}(a, b)$.

Example:
111001 and 1001
011101 and 1001
001111 and 1001
000110 and 1001
000011 and 1001
000011 and 0101
000011 and 0011
000011 and 0000

See also: manipulators as polynomials, binary (Stein's) GCD algorithm.

Remaining Cases

In this problem, you have to count the number of possible configurations of width k after using the given manipulators.

So, we found the GCD of all manipulators.
Example: $\mathrm{GCD} = 101$. It has length $r = 3$.

Remaining Cases

In this problem, you have to count the number of possible configurations of width k after using the given manipulators.

So, we found the GCD of all manipulators.
Example: $\mathrm{GCD} = 101$. It has length $r = 3$.

Case 1: $r < k$ (example: $k = 2$). No way!

Remaining Cases

In this problem, you have to count the number of possible configurations of width k after using the given manipulators.

So, we found the GCD of all manipulators.
Example: $\mathrm{GCD} = 101$. It has length $r = 3$.

Case 1: $r < k$ (example: $k = 2$). No way!

Case 2: $r = k$ (example: $k = 3$). One way.

Remaining Cases

In this problem, you have to count the number of possible configurations of width k after using the given manipulators.

So, we found the GCD of all manipulators.
Example: $\mathrm{GCD} = 101$. It has length $r = 3$.

Case 1: $r < k$ (example: $k = 2$). No way!

Case 2: $r = k$ (example: $k = 3$). One way.

Case 3: $r > k$ (example: $k = 6$). 2^{k-r-1} ways:

`101...`	(must put)
`.???..`	(two possibilities)
`..???.`	(two possibilities)
`...101`	(must put)

Contest Developer:

- Ivan Kazmenko

Special Thanks:

- Natalya Ginzburg
- Oleg Hristenko
- Alisa Kazmenko
- Dasha Kazmenko
- Pavel Kunyavskiy
- Andrei Lopatin
- Roman Soshkin
- Andrew Stankevich

Solutions, checkers, interactors, and generators written in the D programming language (`https://dlang.org`).

Questions?