Task overview

Central Europe Regional Contest 2024

December 15, 2024

Contest statistics

Contest statistics

Number of submissions: 954

Contest statistics

Number of submissions: 954

Programming languages:

- C++: 823
- Python3: 111
- Java: 20

Contest statistics

Number of submissions: 954

Programming languages:

- C++: 823
- Python3: 111
- Java: 20

Total test runs: 18918

Contest statistics

Number of submissions: 954

Programming languages:

- C++: 823
- Python3: 111
- Java: 20

Total test runs: 18918

Total time running tests: 44 minutes

Contest statistics

Number of submissions: 954
Programming languages:

- C++: 823
- Python3: 111
- Java: 20

Total test runs: 18918
Total time running tests: 44 minutes
Total submission size: 2.1939 MB

Contest statistics

Number of submissions: 954
Programming languages:

- C++: 823
- Python3: 111
- Java: 20

Total test runs: 18918
Total time running tests: 44 minutes
Total submission size: 2.1939 MB
Smallest AC source size: 179 bytes

Contest statistics

Number of submissions: 954
Programming languages:

- C++: 823
- Python3: 111
- Java: 20

Total test runs: 18918
Total time running tests: 44 minutes
Total submission size: 2.1939 MB
Smallest AC source size: 179 bytes
Biggest AC source size: 6.83 kilobytes

Contest statistics

Number of submissions: 954
Programming languages:

- C++: 823
- Python3: 111
- Java: 20

Total test runs: 18918
Total time running tests: 44 minutes
Total submission size: 2.1939 MB
Smallest AC source size: 179 bytes
Biggest AC source size: 6.83 kilobytes
Latest contest submit: 398ms before the end, King's Festival, AGH 3

Problem C – CERC Plaques

Problem C – CERC Plaques

Accepted before freeze: 71

Teams that submitted after freeze: 0

Problem C – CERC Plaques

Problem

Given two sequences of strings $A_1, \ldots, A_N$ and $B_1, \ldots, B_N$ pair them so that every pair has the same prefix of 4 letters.

Problem C – CERC Plaques

Problem

Given two sequences of strings $A_1, \ldots, A_N$ and $B_1, \ldots, B_N$ pair them so that every pair has the same prefix of 4 letters.

- First we can sort both sequences, so assume without loss of generality that they are already sorted.
- There is a feasible solution if and only if the pairing A_i with B_i is feasible.

Problem D – Divisors

Problem D – Divisors

Accepted before freeze: 69

Teams that submitted after freeze: 0

Problem D – Divisors

Problem

Given number N find all its divisors composed of the same set of digits (without leading zeroes).

For example, $N = 819503972640$ has four such divisors:

- $N = 2 \cdot 409751986320$,
- $N = 4 \cdot 204875993160$,
- $N = 5 \cdot 163900794528$,
- $N = 8 \cdot 102437996580$.

All these numbers are composed of digits 001234567899.

Problem D – Divisors

- Every such divisor is bigger than $N/10$.
- It is enough to check numbers $N/2, N/3, \ldots, N/9$.
- The simplest solution is to convert the number to a string, sort its digits and compare.

Problem B – Bad digits

Problem B – Bad digits

Accepted before freeze: 68

Teams that submitted after freeze: 0

Problem B – Bad digits

Problem B – Bad digits

Problem

Find the Kth positive number that does not contain the digits $C_1, \ldots, C_M$ in its N-ary representation.

Problem B – Bad digits

Problem

Find the Kth positive number that does not contain the digits $C_1, \ldots, C_M$ in its N-ary representation.

- First, find the $(N - M)$-ary representation of K.

Problem B – Bad digits

Problem

Find the Kth positive number that does not contain the digits $C_1, \ldots, C_M$ in its N-ary representation.

- First, find the $(N - M)$-ary representation of K.
- Map digits $0, 1, \ldots, N - M - 1$ in $(N - M)$-ary representation of K to the remaining digits $0, D_1, \ldots, D_{N-M-1}$.

To see why this is correct, think of the numbers as strings with only some digits allowed and sort all numbers lexicographically. The only thing that matters is the relation of the digits and not which digits exactly are present.

Problem L – Labyrinth

Problem L – Labyrinth

Accepted before freeze: 54

Teams that submitted after freeze: 9

Problem L – Labyrinth

Problem

There is a grid given, where each cell can be passable or not. On the grid there is an agent, that moves in the following pattern. He check neighboring cells in order: right, front, left, behind, and moves to the first passable one. For each starting position and direction you must answer how many steps it will take for agent to leave the grid.

Problem L – Labyrinth

Problem

There is a grid given, where each cell can be passable or not. On the grid there is an agent, that moves in the following pattern. He check neighboring cells in order: right, front, left, behind, and moves to the first passable one. For each starting position and direction you must answer how many steps it will take for agent to leave the grid.

- You can simulate agent route for each query.
- If the agent visits the same state pointing to the same direction twice, it means he cycles and will never leave the grid.
- As there are many queries, you must store computed distance for each direction and position.

Problem I – Illuminati

Problem I – Illuminati

Accepted before freeze: 37

Teams that submitted after freeze: 16

Problem I – Illuminati

Problem

There is a hidden permutation $p : \{1, \ldots, N\} \to \{1, \ldots, N\}$ and a function $f : \mathbb{N} \to \{0, 1\}$. As the input we get sequences of numbers $K_0, \ldots, K_{N-1}$, where K_i is equal to $f(p^i(1)), \ldots, f(p^i(N))$. Our task is to find a lexicographically minimal p that satisfies the equations, or state that no such p exists.

Problem I – Illuminati

Problem

There is a hidden permutation $p : \{1, \ldots, N\} \to \{1, \ldots, N\}$ and a function $f : \mathbb{N} \to \{0, 1\}$. As the input we get sequences of numbers $K_0, \ldots, K_{N-1}$, where K_i is equal to $f(p^i(1)), \ldots, f(p^i(N))$. Our task is to find a lexicographically minimal p that satisfies the equations, or state that no such p exists.

Let's denote the binary sequence produced by taking x-th element of each K_i by S_x.

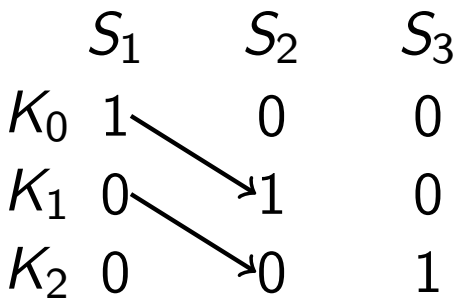
	S_1	S_2	S_3
K_0	1	0	0
K_1	0	1	0
K_2	0	0	1

Problem I – Illuminati

Observation

If $p(x) = y$, then $S_x[1, N-1] = S_y[2, N]$.

If $p(1) = 2$, then:



Problem I – Illuminati

Let's think about a graph where vertices are strings. S_x can be represented as an edge from $S_x[1, N-1]$ to $S_x[2, N]$. Note that there is a 1-1 correspondence between valid permutations and cycle decompositions of the graph.

- If for any vertex $indeg(v) \neq outdeg(v)$, no cycle decomposition exists and the answer is `NO`.
- If $indeg(v) = outdeg(v)$ for each v, we just need to match every incoming edge of v with an outgoing edge, which will give us the cycle decomposition. To achieve the lexicographically minimal permutation, we need to sort the values and match greedily.

Problem K – King's Festival

Problem K – King's Festival

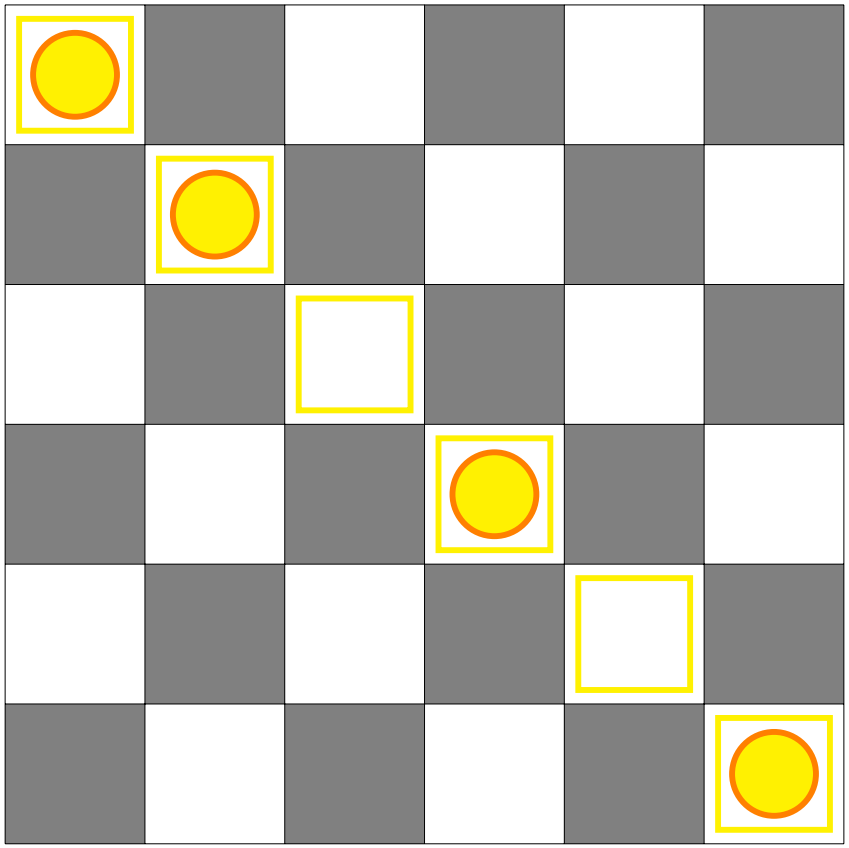
Accepted before freeze: 18

Teams that submitted after freeze: 17

Problem K – King's Festival

Problem

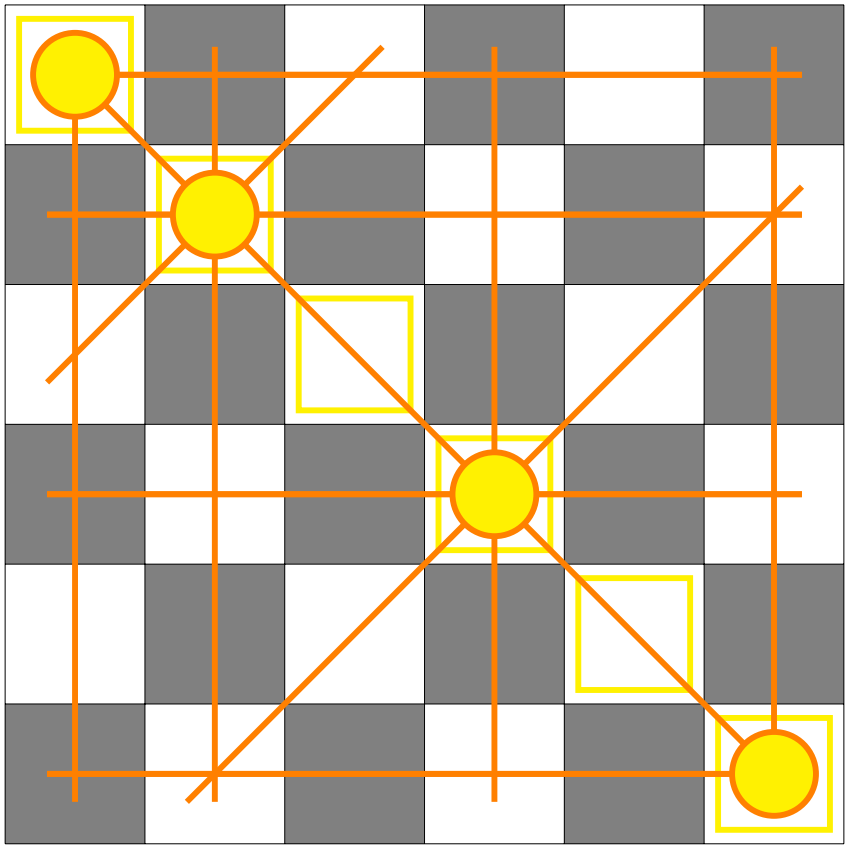
Place minimal number of queens on the main diagonal of $N \times N$ chessboard to attack all the fields (some of them are already placed).



Problem K – King's Festival

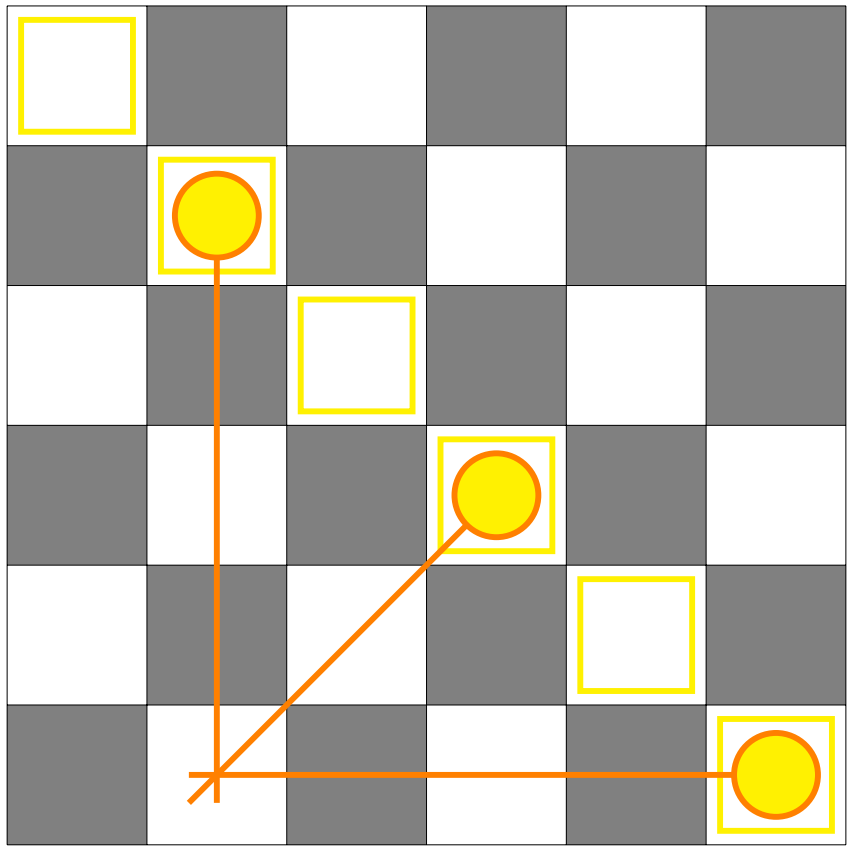
Problem

Place minimal number of queens on the main diagonal of $N \times N$ chessboard to attack all the fields (some of them are already placed).



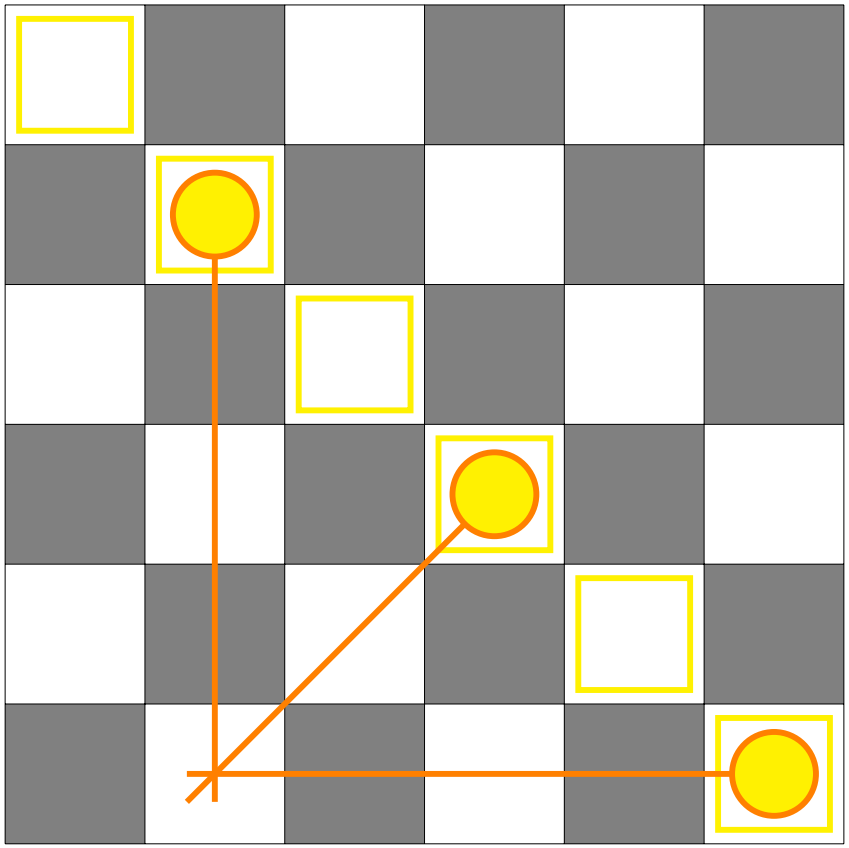
Problem K – King's Festival

- Every white field is attacked by three *arithmetic* fields.



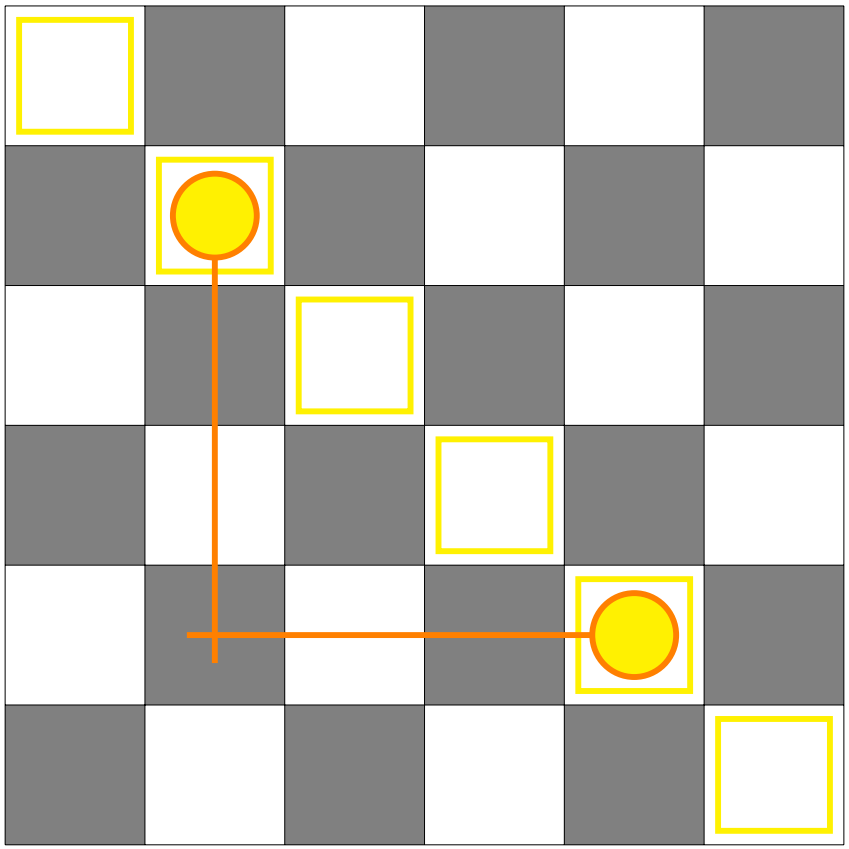
Problem K – King's Festival

- $\rightarrow$ There can be no *free three-element arithmetic sequence*.



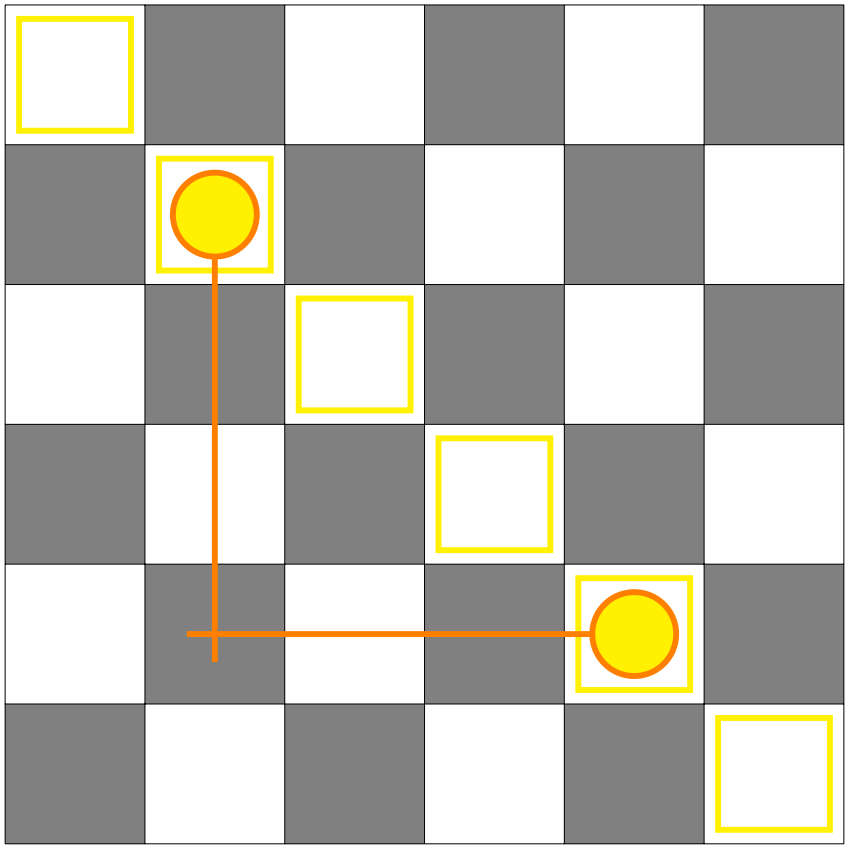
Problem K – King's Festival

- $\rightarrow$ There can be no *free three-element arithmetic sequence*.
- Every black field is attacked by two *odd-distance* fields.



Problem K – King's Festival

- $\rightarrow$ There can be no *free three-element arithmetic sequence*.
- $\rightarrow$ We need to take all even or all odd fields.



Problem K – King's Festival

- $\rightarrow$ There can be no *free three-element arithmetic sequence*.
- $\rightarrow$ We need to take all even or all odd fields.

Solution:

- We take either all even or all odd fields.
- For the remaining fields we use backtracking to find best *arithmetic-free* sequence.

Problem F – Flats

Problem F – Flats

Accepted before freeze: 8

Teams that submitted after freeze: 18

Problem F – Flats

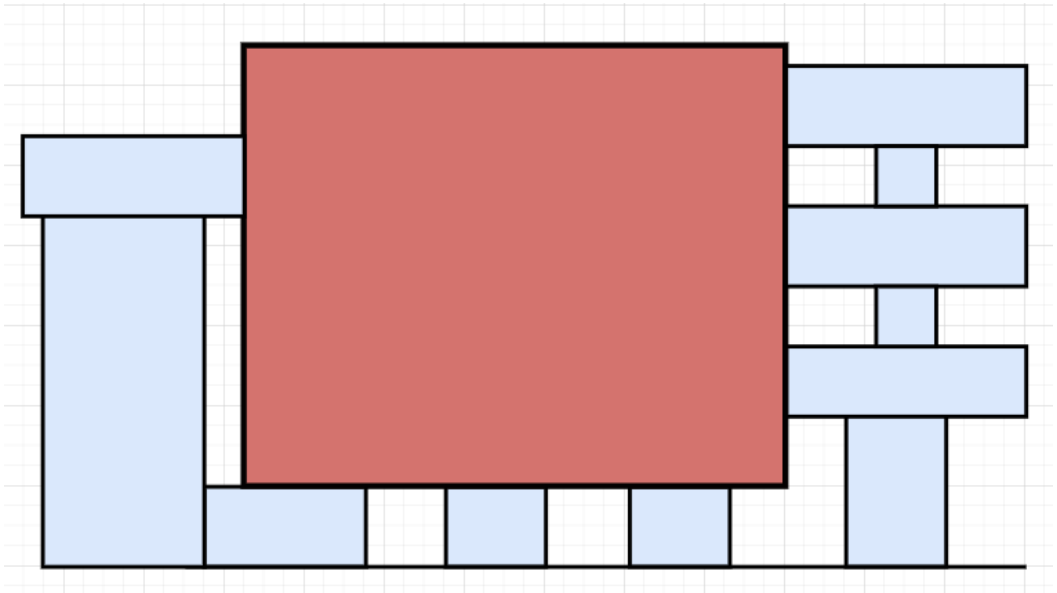
Problem

Rectangles are falling on the OX axis and stopping when they touch anything below them. Count the number of closed areas between them.

- First calculate the height at which the rectangle stops moving $\rightarrow$ segment tree.

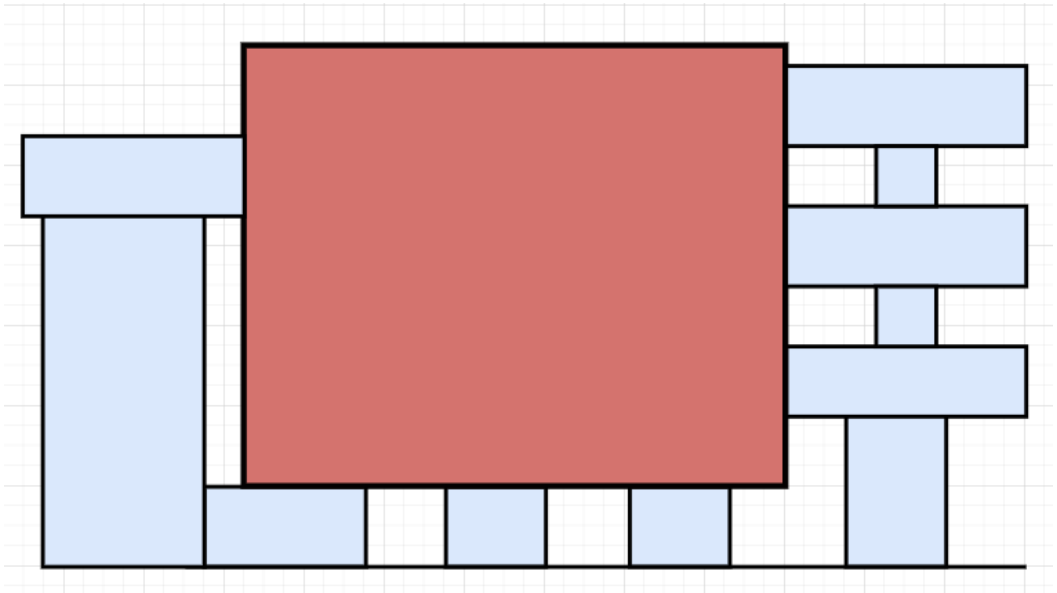
Problem F – Flats

- First, we calculate the height at which the rectangle stops moving $\rightarrow$ segment tree.
- Then we have to consider all the areas the rectangle is closing: below, to the left and to the right.



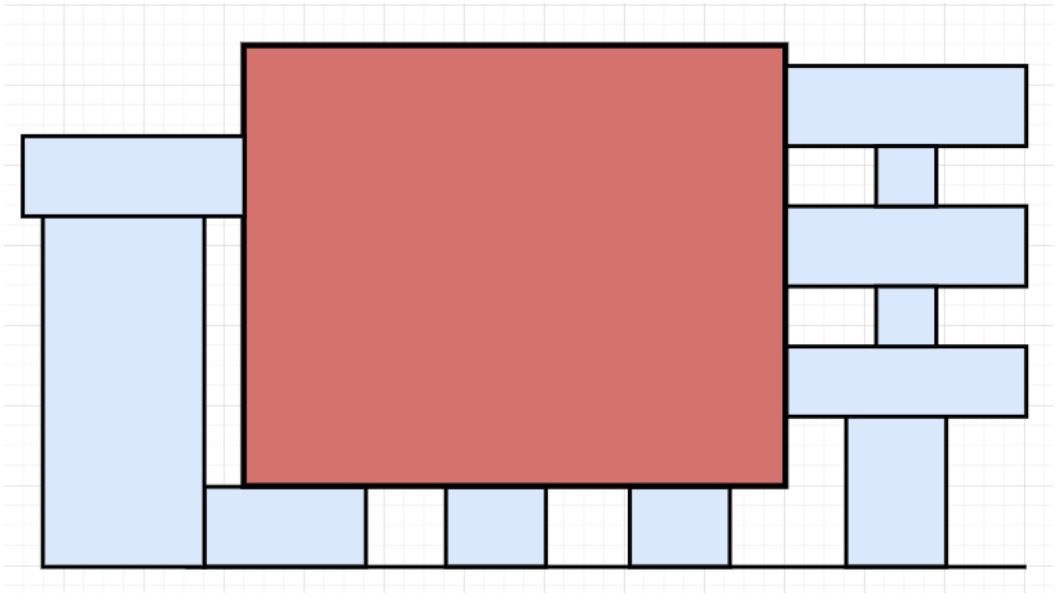
Problem F – Flats

- First, we calculate the height at which the rectangle stops moving $\rightarrow$ segment tree.
- Then we have to consider all the areas the rectangle is closing: below, to the left and to the right.
- Below: we calculate the number of rectangles it touches. Can be done e.g. using the same segment tree.



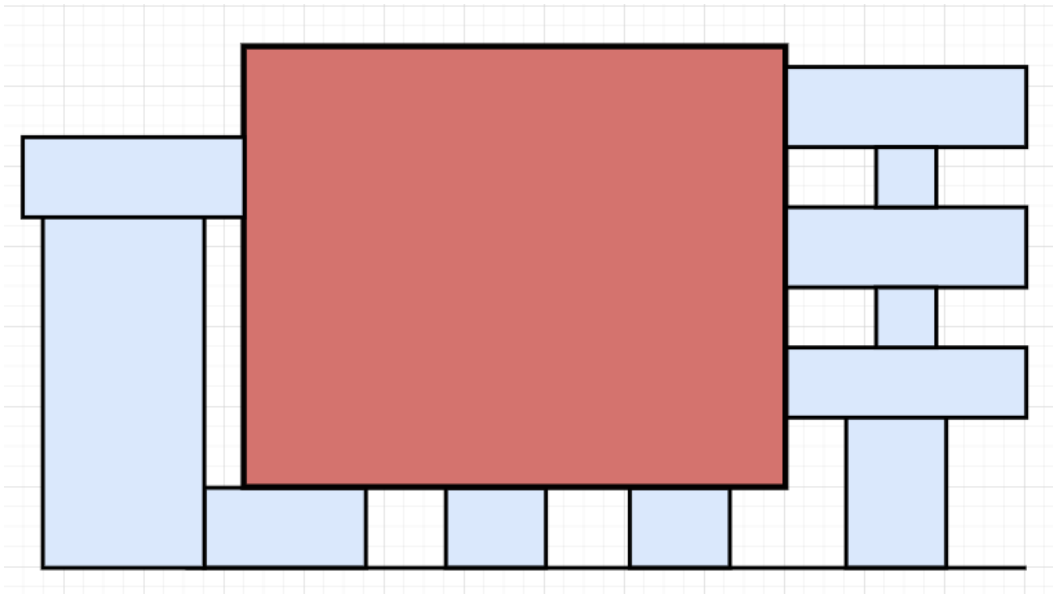
Problem F – Flats

- First, we calculate the height at which the rectangle stops moving $\rightarrow$ segment tree.
- Then we have to consider all the areas the rectangle is closing: below, to the left and to the right.
- Below: we calculate the number of rectangles it touches. Can be done e.g. using the same segment tree.
- To the left/right: for every OX coordinate keep lists of the rectangles with left/right edge there (beware of the **literal** corner cases).



Problem F – Flats

- First, we calculate the height at which the rectangle stops moving $\rightarrow$ segment tree.
- Simpler solution: sort rectangles in order of increasing bottom edge coordinate. Then you only have to worry about the areas below the rectangle.



Problem E – Expressions

Problem E – Expressions

Accepted before freeze: 3

Teams that submitted after freeze: 10

Problem E – Expressions

Problem

We are given expression using binary variables `a`, `b`, ..., `j`
and binary operators `min`, `max`, `<` and `<=`.
We need to write equivalent expression using only operators `min` and `<=`.

Example: `((max a a) <= b)` $\rightarrow$ `(a <= b)`.

Problem E – Expressions

- First step: parser...

Problem E – Expressions

- First step: parser...

Fact 1

If $\texttt{a} = \texttt{b} = \ldots = \texttt{j} = 1$, then we can not make an expression with value 0 using only `min` and `<=`, since $min(1, 1) = 1$ and $1 <= 1 = 1$.

Problem E – Expressions

- First step: parser...

Fact 1

If $\texttt{a} = \texttt{b} = \ldots = \texttt{j} = 1$, then we can not make an expression with value 0 using only `min` and <=, since $min(1, 1) = 1$ and $1 <= 1 = 1$.

Fact 2

In the other case we are able to do it!

Proof

We only need to make a construction that replaces `max` and <.

Problem E – Expressions

We need to replace < and `max` using <= and `min`.

- `L < R` $\rightarrow$ `min(not(L), R)`

Problem E – Expressions

We need to replace < and `max` using <= and `min`.

- `L < R` $\rightarrow$ `min(not(L), R)`
- `not(X)` $\rightarrow$ `X <= 0`

Problem E – Expressions

We need to replace < and `max` using <= and `min`.

- `L < R` $\rightarrow$ `min(not(L), R)`
- `not(X)` $\rightarrow$ `X <= 0`
- `0` $\rightarrow$ `min(a, min(b, min(c, ..., min(i, j))...))`

Problem E – Expressions

We need to replace < and `max` using <= and `min`.

- `L < R` $\rightarrow$ `min(not(L), R)`
- `not(X)` $\rightarrow$ `X <= 0`
- `0` $\rightarrow$ `min(a, min(b, min(c, ..., min(i, j))...))`
- `max(L, R)` $\rightarrow$ `min(not(L), not(R)) <= 0`

Problem H – Hues

Problem H – Hues

Accepted before freeze: 2

Teams that submitted after freeze: 12

Problem H – Hues

Problem

Given N circles on a plane, tell if they form a correct Venn diagram. If not, give the counterexample.

Correct Venn diagram

Incorrect Venn diagram: missing green and yellow

Problem H – Hues

Fact.

For $N \geq 4$, the answer is always `NO`.

Proof.

Two circles can intersect in at most two points (the infinite case is trivial). Adding a new circle, every intersection can create at most one region. 4th circle can intersect in at most 6 points, so there must be at most $14 < 2^4$ regions. □

This also gives intuition that there should not be more than quadratically many intersections.

Problem H – Hues

We can then search for all intersections, list all represented subsets and print any that is not present:

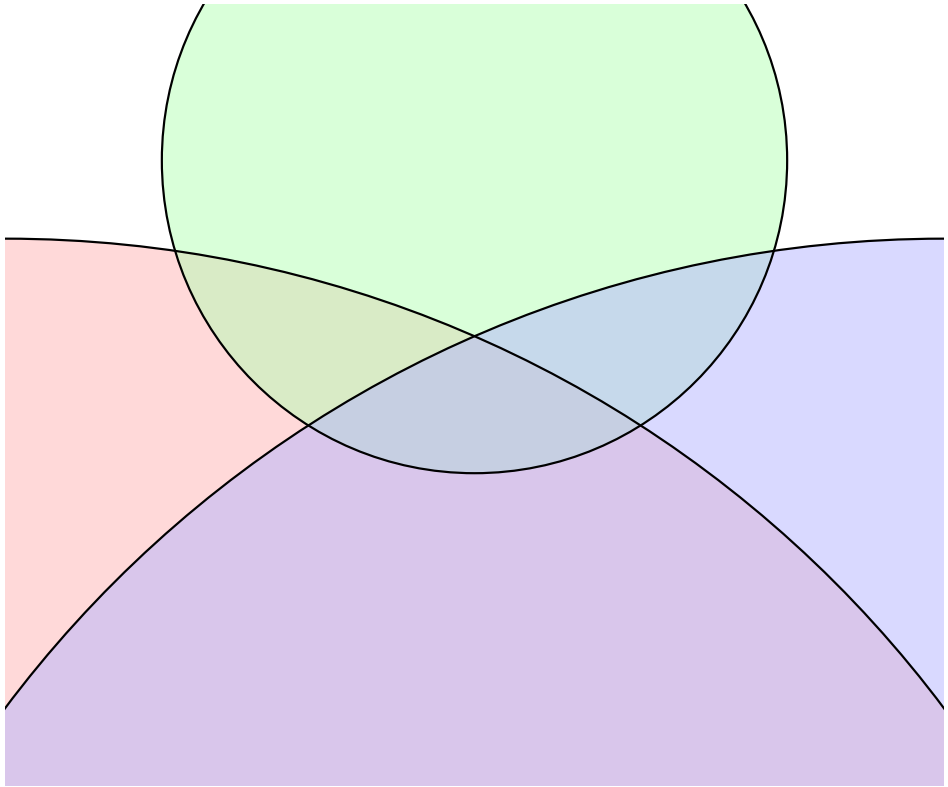
- First check if there are any two circles that do not intersect in exactly two points to get rid of any nasty corner cases.

Problem H – Hues

We can then search for all intersections, list all represented subsets and print any that is not present:

- First check if there are any two circles that do not intersect in exactly two points to get rid of any nasty corner cases.
- For an intersection point of circles C_i and C_j, find all circles that cointains that point. Let's say that those circles are $C_{i_1}, \ldots, C_{i_k}$. This gives us exactly 4 regions, representing subsets:
  - $\{i_1, \ldots, i_k\}$
  - $\{i, i_1, \ldots, i_k\}$
  - $\{j, i_1, \ldots, i_k\}$
  - $\{i, j, i_1, \ldots, i_k\}$

Problem H – Hues



Green circle contributes to all regions around the intersection point.

Problem H – Hues

We can then search for all intersections, list all represented subsets and print any that is not present:

- First check if there are any two circles that do not intersect in exactly two points to get rid of any nasty corner cases.
- For an intersection point of circles C_i and C_j, find all circles that contains that point. Let's say that those circles are $C_{i_1}, \ldots, C_{i_k}$. This gives us exactly 4 regions, representing subsets:
  - $\{i_1, \ldots, i_k\}$
  - $\{i, i_1, \ldots, i_k\}$
  - $\{j, i_1, \ldots, i_k\}$
  - $\{i, j, i_1, \ldots, i_k\}$
- All regions have some adjacent intersection point, hence this approach finds all of them.

Complexity of this solution: $\mathcal{O}(N^3)$.

Problem G – Groceries

Problem G – Groceries

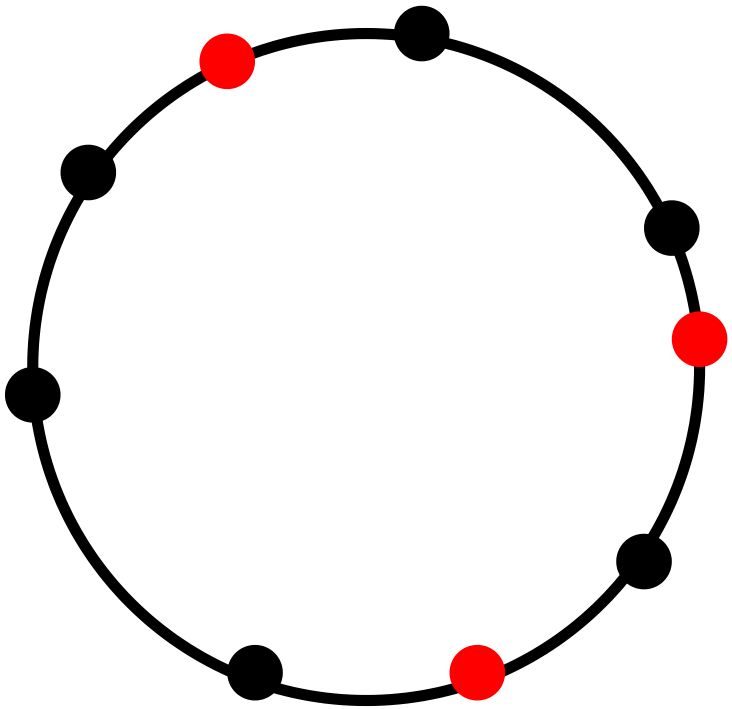
Accepted before freeze: 2

Teams that submitted after freeze: 8

Problem G – Groceries

Problem

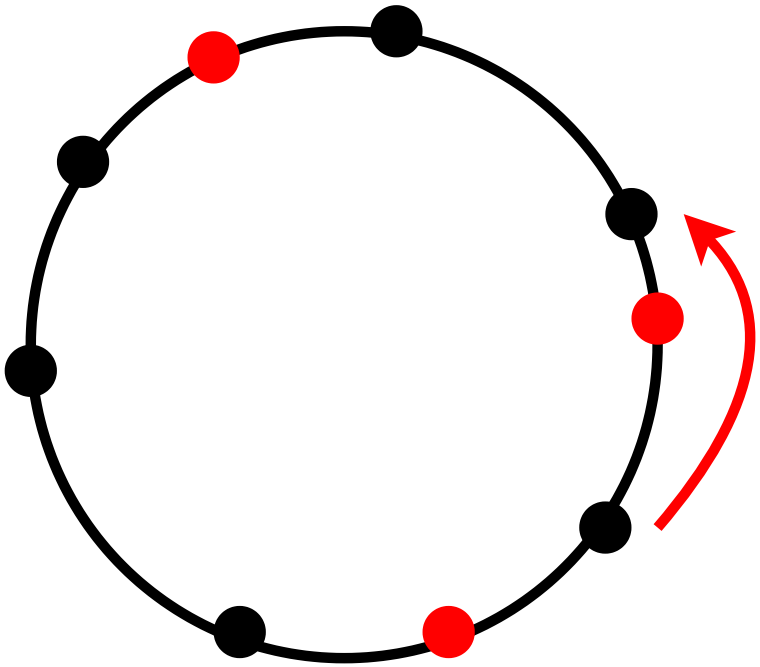
Given N black points on a circular road, place K red points on it to minimize the maximal distance travelled between any two neighbouring black points stopping by the closest red point along the way.
Input: distances between the black points: $d_1, d_2, \ldots d_n$.



Problem G – Groceries

Problem

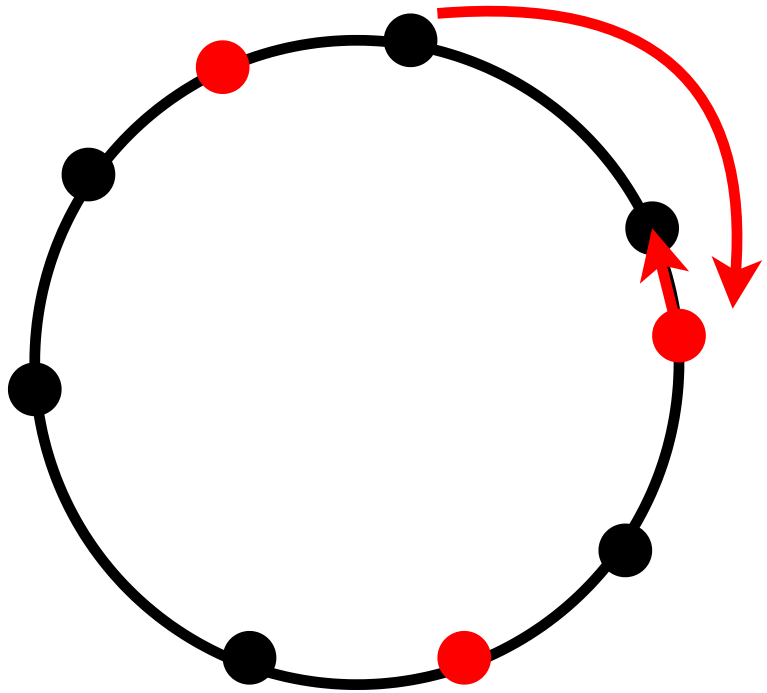
Given N black points on a circular road, place K red points on it to minimize the maximal distance travelled between any two neighbouring black points stopping by the closest red point along the way.
Input: distances between the black points: $d_1, d_2, \ldots d_n$.



Problem G – Groceries

Problem

Given N black points on a circular road, place K red points on it to minimize the maximal distance travelled between any two neighbouring black points stopping by the closest red point along the way.
Input: distances between the black points: $d_1, d_2, \ldots d_n$.



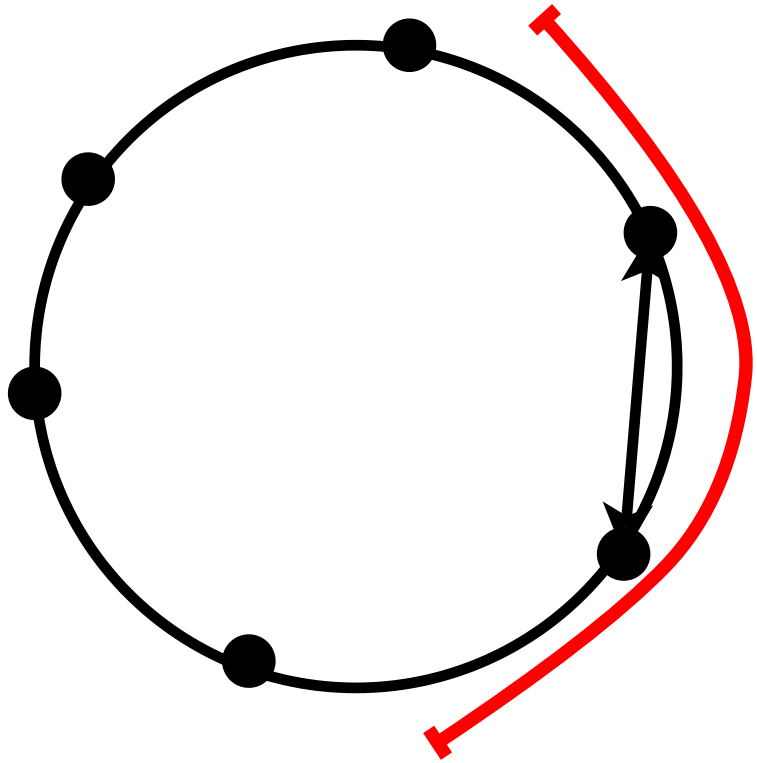
Observation

For $K = 1$, the result is $\mathrm{Len} - \max_i d_i$.

Observation

For $K > 2$, the result is at most $\min\left(\mathrm{Len} - \max_i d_i, \frac{\mathrm{Len}}{2}\right)$.

Let's binary search:



This reduces the problem to segment cutting. W.l.o.g. red points will be placed at the clockwise-furthest endpoints of the segments.

Let's assume we know where to put the first red point:

- Greedily move clockwise ignoring the segments that are already satisfied,
- Put a new red point at the end of the first unsatisfied segment.

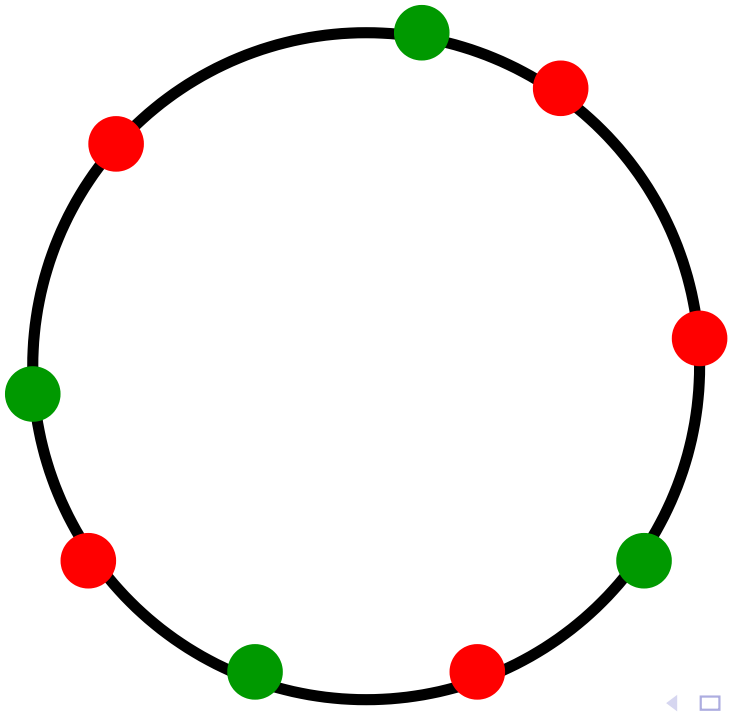
But, how to find the starting point?

Observation

Starting from a point that is a part of an optimal solution, we will always jump to a point that also belongs to an optimal solution.

Observation

If we cycle back to the point we have already visited, this vertex must belong to an optimal solution.

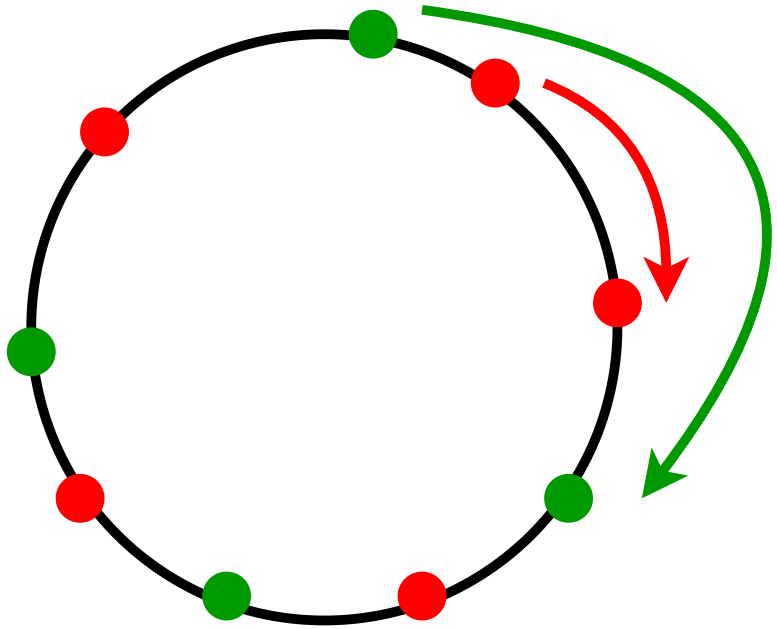


Observation

Starting from a point that is a part of an optimal solution, we will always jump to a point that also belongs to an optimal solution.

Observation

If we cycle back to the point we have already visited, this vertex must belong to an optimal solution.



Observation

Starting from a point that is a part of an optimal solution, we will always jump to a point that also belongs to an optimal solution.

Observation

If we cycle back to the point we have already visited, this vertex must belong to an optimal solution.

Corollary

After N jumps we must end up in an optimal starting point.

Bringing it all together:

- Binary search on the maximum distance: $\mathcal{O}(\log \mathrm{Len})$
- For each of the N segment endpoints, calculate an optimal next position using two pointers: $\mathcal{O}(N)$
- Find the optimal starting point and calculate the number of red points needed: $\mathcal{O}(N)$.

Time complexity: $\mathcal{O}(N \log \mathrm{Len})$.

Problem J - Just Mining

Problem J - Just Mining

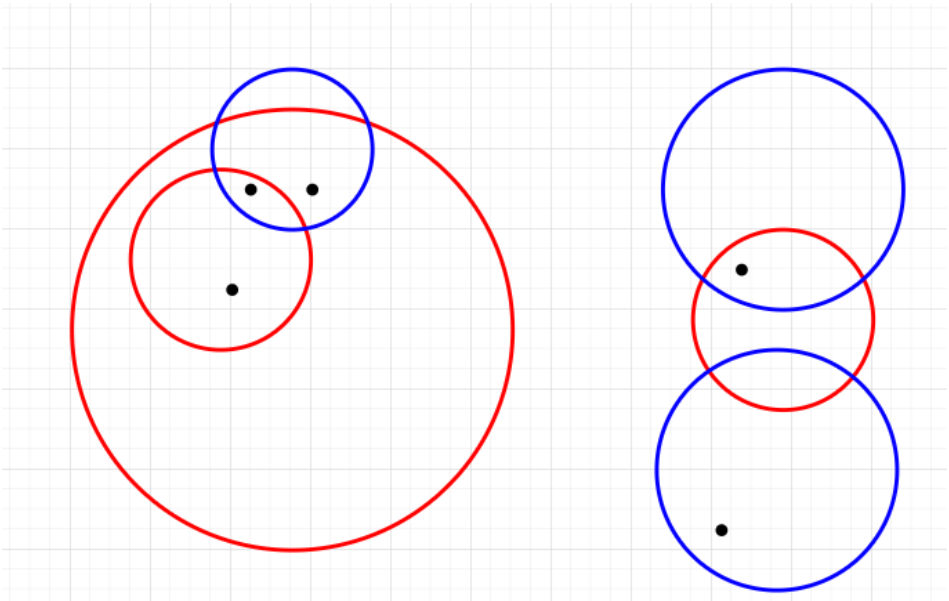
Accepted before freeze: 1

Teams that submitted after freeze: 2

Problem J - Just Mining

Problem

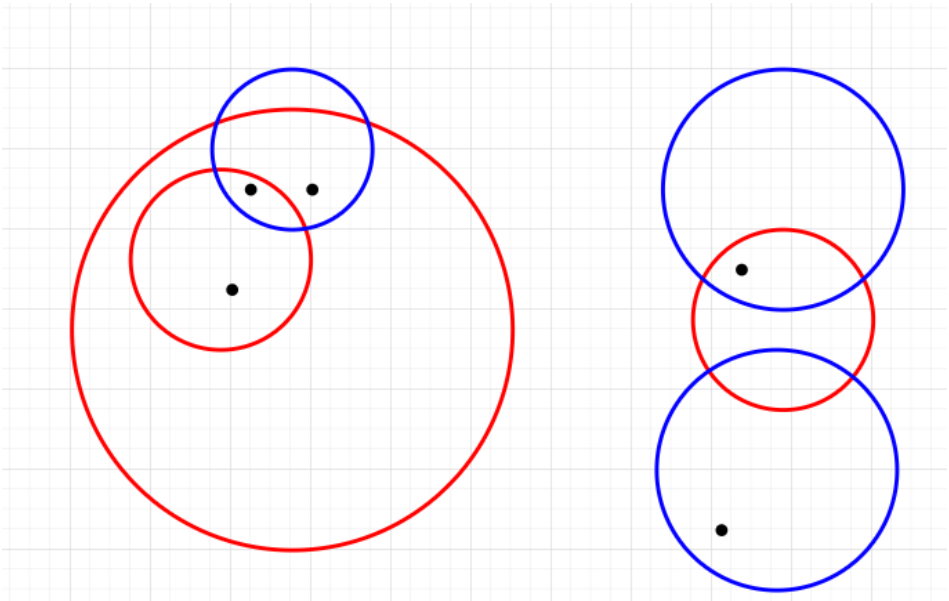
We are given a set of circles and a set of points on the plane. Circles have capacities, points have values. Circles can be divided into two laminar families. We have to choose a subset of points with maximum value so that no capacities are violated.



Problem J - Just Mining

Problem

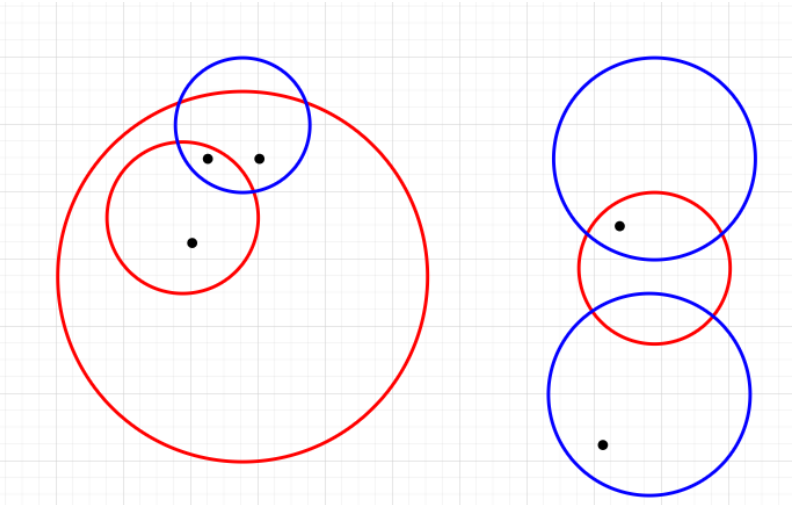
We are given a set of circles and a set of points on the plane. Circles have capacities, points have values. Circles can be divided into two laminar families. We have to choose a subset of points with maximum value so that no capacities are violated.

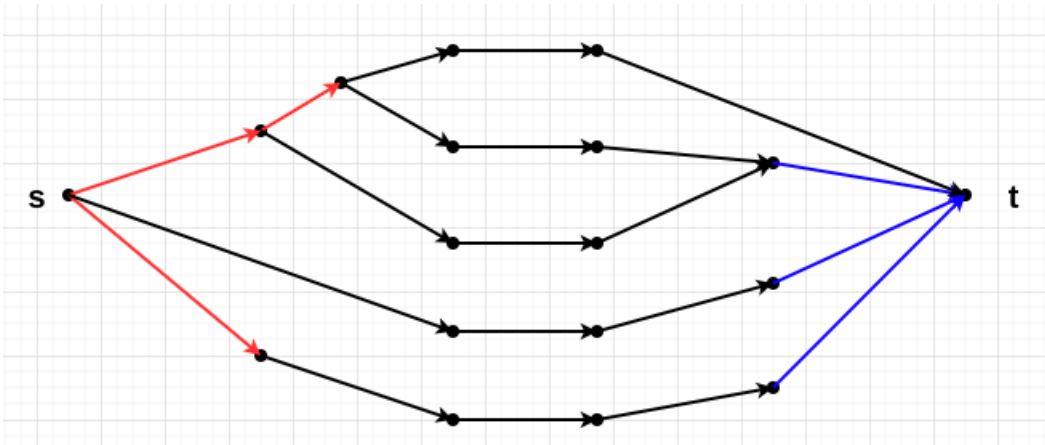


- Step 1: divide circles into two laminar families.

Problem J - Just Mining

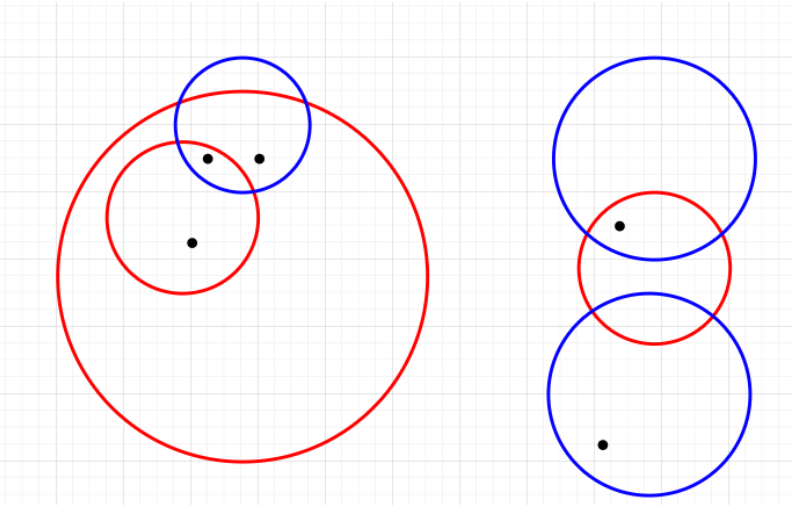
- Step 2: Create a network graph.

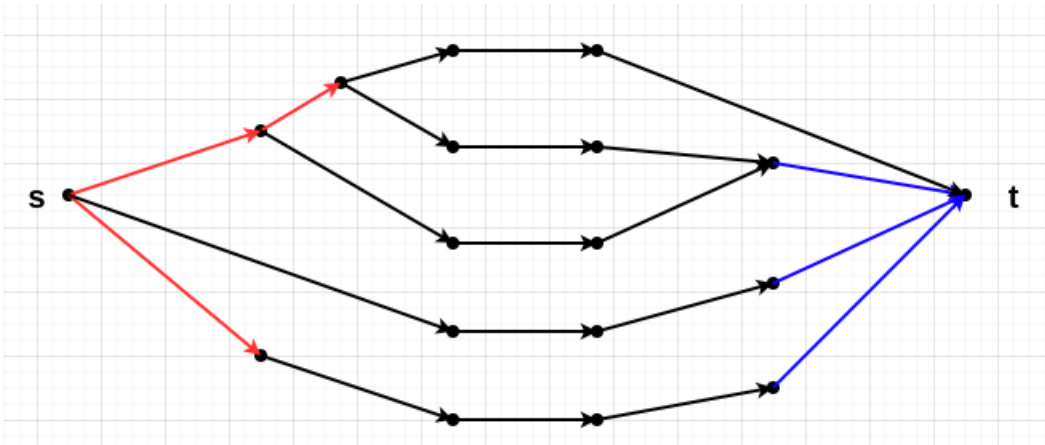




Problem J - Just Mining

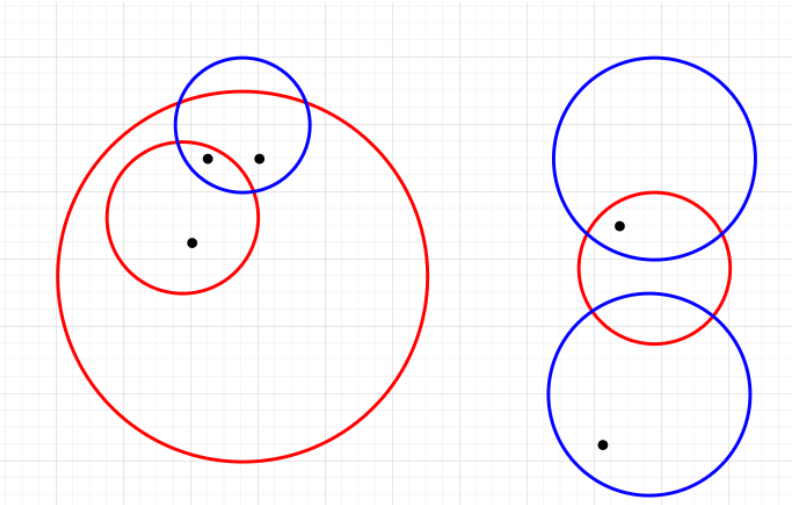
- Step 2: Create a network graph.
  - capacities: equal to capacities of circles; for other edges equal to one

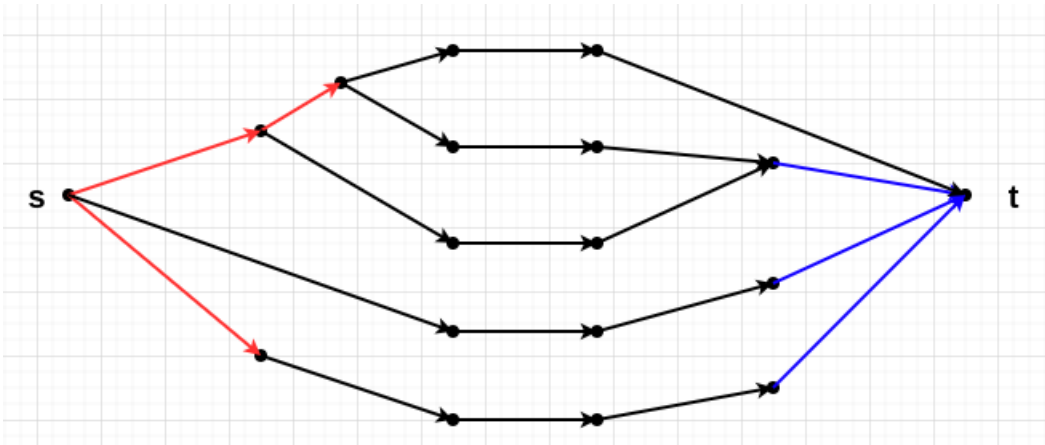




Problem J - Just Mining

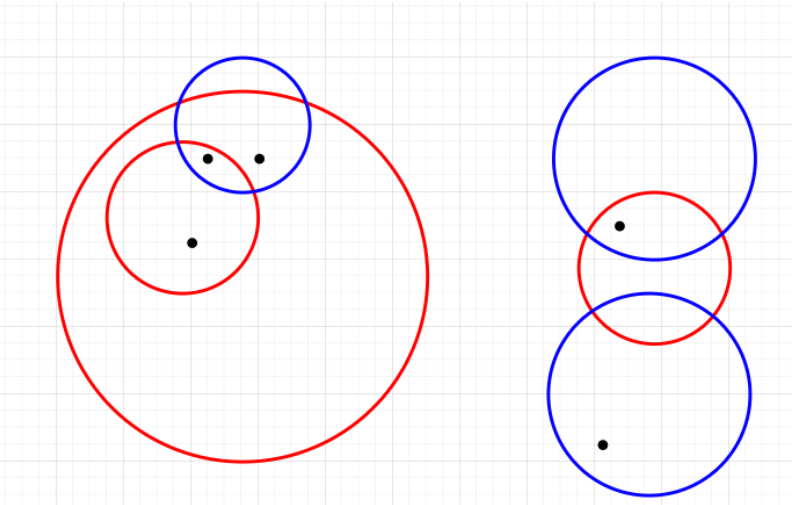
- Step 2: Create a network graph.
  - capacities: equal to capacities of circles; for other edges equal to one
  - values: all zeroes except between "vertex nodes"

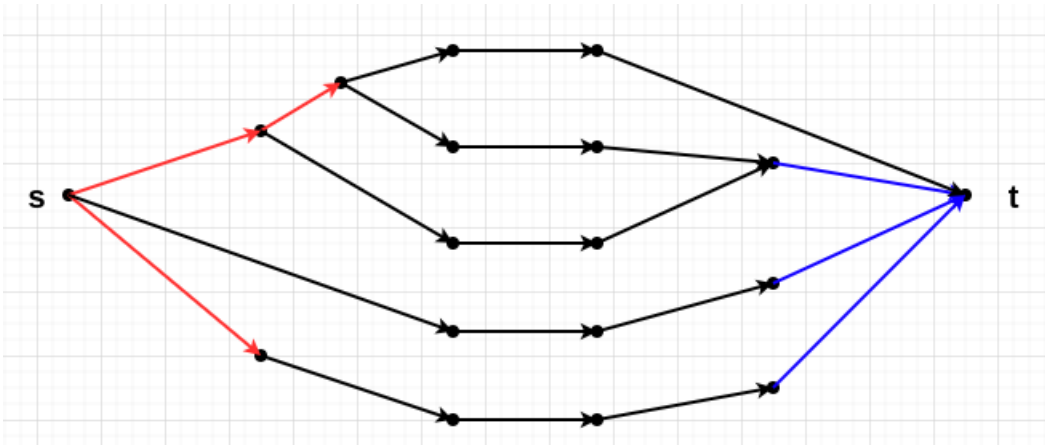




Problem J - Just Mining

- Step 2: Create a network graph.
  - capacities: equal to capacities of circles; for other edges equal to one
  - values: all zeroes except between "vertex nodes"
- Step 3: Run a max-cost max flow.





Problem A – Anthem

Problem A – Anthem

Accepted before freeze: 0

Teams that submitted after freeze: 5

Problem A – Anthem

Problem

Given a string s, we can generate string t by walking over s:

1. We choose a starting position in s and make the letter at this position the first letter in t.
2. Then in each step we can either stay on the same letter, or move to the left or right, and append the letter we end on to t.

For a given word w we have to find its smallest generator.

Problem A – Anthem

Definition

We will say that string s has non-trivial generator t if t generates s and $|t| < |s|$

Lemma

String s has non-trivial generator if and only if s has one of the following forms (where x, y, z are (possibly empty) words, and a, b are letters):

1. $xaay$
2. $b\tilde{x}axby$
3. $ybxa\tilde{x}b$
4. $yaxb\tilde{x}axbz$

Proof follows from the walking definition.

Problem A – Anthem

Simplification

We can simplify w iteratively, by reducing above forms to:

1. $xaay \rightarrow xay$
2. $b\tilde{x}axby \rightarrow axby$
3. $ybxa\tilde{x}b \rightarrow ybxa$
4. $yaxb\tilde{x}axbz \rightarrow yaxbz$

This way we will obtain minimal generator, which turns out to be also the shortest one. For the proof see Ian Pratt-Hartmann, *Walking on Words*

Problem A – Anthem

- We can simplify 1. by simply skipping repeated letters as the first step.

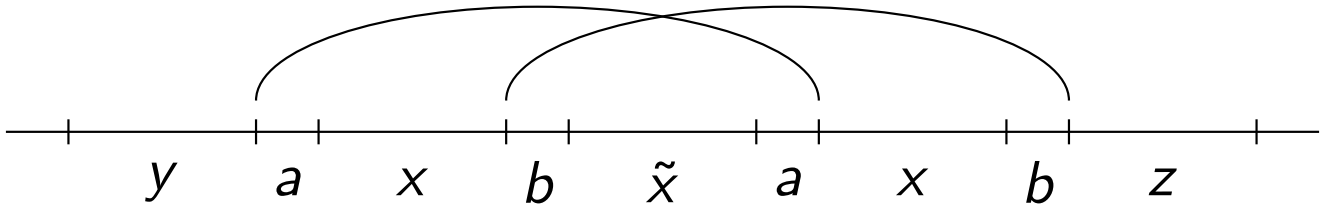
Problem A – Anthem

- We can simplify 1. by simply skipping repeated letters as the first step.
- Forms 2. and 3. can be detected and simplified with the help of Manacher's algorithm - we can simplify palindromes on the prefix and suffix of the word.

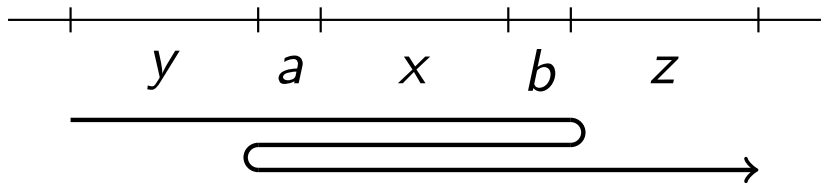
Observation

Simplifying forms 2. and 3. will not create new patterns of form 4., so this can be done as the last step.

Problem A – Anthem



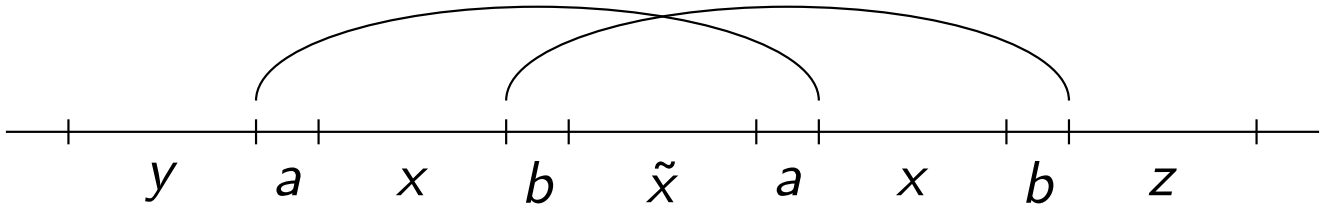
Form 4. before reduction



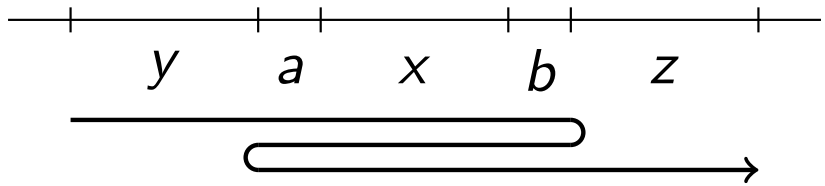
Form 4. after reduction

- Form 4 is an intersection of two odd palindromes containing each other's middle. We will go from left to right, and every time Form 4 occurs, we will perform the simplification.

Problem A – Anthem



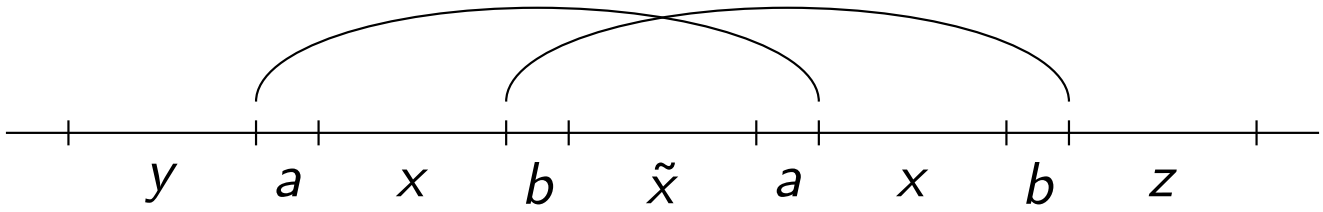
Form 4. before reduction



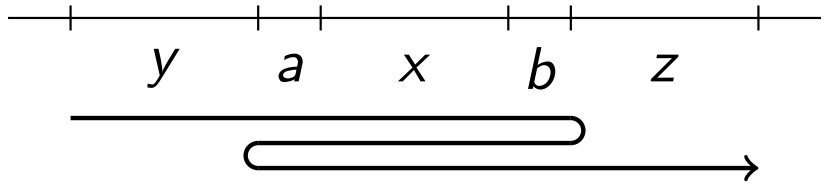
Form 4. after reduction

- Form 4 is an intersection of two odd palindromes containing each other's middle. We will go from left to right, and every time Form 4 occurs, we will perform the simplification.
- For every position i, let us store a set of *active* palindromes, i.e., the indices j such that the palindrome with middle j ends exactly at i and can be extended to the right.

Problem A – Anthem



Form 4. before reduction



Form 4. after reduction

- Extending to $i+1$: If any of the active palindromes can be extended at this point and it contains the middle of another palindrome on the left, which also contains this palindrome's middle, we can remove this palindrome entirely and move back to the middle of the previous palindrome.

Problem A – Anthem

How fast is this? For each pair s, t of palindromes in the active set, s and t cannot contain each other's middle points (otherwise, we would have simplified them). If we sort the palindromes by size, each will be at least twice as large as the previous one. Therefore, there will be at most $O(\log n)$ palindromes in the active set. The entire solution will run in $O(n \log n)$.

Thank you