NCPC 2020
Presentation of solutions

2020-11-07

Problems prepared by

- Per Austrin (KTH Royal Institute of Technology)
- Bjarki Ágúst Guðmundsson (Reykjavík University)
- Nils Gustafsson (Vårdinnovation)
- Antti Laaksonen (CSES)
- Simon Lindholm (Vårdinnovation)
- Ulf Lundström (Excillum)
- Jimmy Mårdell (Spotify)
- Johan Sannemo (Huawei)
- Bergur Snorrason (University of Iceland)
- Pehr Söderman (Kattis)

M — Methodic Multiplication

Problem

Multiply two natural numbers in Peano arithmetic.

M — Methodic Multiplication

Problem

Multiply two natural numbers in Peano arithmetic.

Axiomatic Solution

```
main :- read_val(X), read_val(Y), multiply(X, Y, Z), print(Z).

multiply(_, 0, 0).
multiply(X, s(Y), Z) :- multiply(X, Y, W), add(W, X, Z).

add(X, 0, X).
add(X, s(Y), Z) :- add(X, Y, W), Z = s(W).

read_val(0)    :- peek_code(C), code_type(C, space), !, get_char(_).
read_val(s(X)) :- get_char(C), C == 'S', !, read_val(X).
read_val(X)    :- read_val(X).
```

M — Methodic Multiplication

Problem

Multiply two natural numbers in Peano arithmetic.

Non-Axiomatic Solution

```
x = input().count('S')
y = input().count('S')
z = x*y
print('S('*z + '0' + ')'*z)
```

M — Methodic Multiplication

Problem

Multiply two natural numbers in Peano arithmetic.

Non-Axiomatic Solution

```
x = input().count('S')
y = input().count('S')
z = x*y
print('S('*z + '0' + ')'*z)
```

Statistics at 4-hour mark: 317 submissions, 188 accepted, first after 00:01

C — Coin Stacks

Problem

Each stack has some number of coins. Can you remove all coins by taking one coin from two distinct stacks on each move?

C — Coin Stacks

Problem

Each stack has some number of coins. Can you remove all coins by taking one coin from two distinct stacks on each move?

Solution

1. You can remove all coins if
   (A) the number of coins is even and
   (B) no stack has more coins than all other stacks together

C — Coin Stacks

Problem

Each stack has some number of coins. Can you remove all coins by taking one coin from two distinct stacks on each move?

Solution

1. You can remove all coins if
   (A) the number of coins is even and
   (B) no stack has more coins than all other stacks together
2. One possible strategy: always choose the two stacks with largest number of coins.

C — Coin Stacks

Problem

Each stack has some number of coins. Can you remove all coins by taking one coin from two distinct stacks on each move?

Solution

1. You can remove all coins if
   (A) the number of coins is even and
   (B) no stack has more coins than all other stacks together
2. One possible strategy: always choose the two stacks with largest number of coins.
3. Easy to prove that if (A) and (B) hold before such a move, they also hold after.

C — Coin Stacks

Problem

Each stack has some number of coins. Can you remove all coins by taking one coin from two distinct stacks on each move?

Solution

1. You can remove all coins if
   (A) the number of coins is even and
   (B) no stack has more coins than all other stacks together
2. One possible strategy: always choose the two stacks with largest number of coins.
3. Easy to prove that if (A) and (B) hold before such a move, they also hold after.

Statistics at 4-hour mark: 555 submissions, 146 accepted, first after 00:07

A — Array of Discord

Problem

Given a sorted list of 100 integers, change one digit in one number to make it unsorted.

A — Array of Discord

Problem

Given a sorted list of 100 integers, change one digit in one number to make it unsorted.

Solution

1. Try all possible ways of changing a digit, check if the resulting list is unsorted.

   (Sufficient to only try changing the most significant digit to a 0, 1 or 9 but this optimization is not needed.)

A — Array of Discord

Problem

Given a sorted list of 100 integers, change one digit in one number to make it unsorted.

Solution

1. Try all possible ways of changing a digit, check if the resulting list is unsorted.

   (Sufficient to only try changing the most significant digit to a 0, 1 or 9 but this optimization is not needed.)

2. Potential pitfall: be careful with leading 0s.
   E.g. cannot change 124 into 024 but can change 4 into 0.

A — Array of Discord

Problem

Given a sorted list of 100 integers, change one digit in one number to make it unsorted.

Solution

1. Try all possible ways of changing a digit, check if the resulting list is unsorted.

   (Sufficient to only try changing the most significant digit to a 0, 1 or 9 but this optimization is not needed.)

2. Potential pitfall: be careful with leading 0s.
   E.g. cannot change 124 into 024 but can change 4 into 0.

Statistics at 4-hour mark: 811 submissions, 125 accepted, first after 00:04

D — Dams in Distress

Problem

We get a rooted tree forming a system of $n \leq 200\,000$ dams. Overflowing a dam causes it to break and release all its water downstream. What is minimum amount of water we need to add at one dam in order for w units of water to reach the root?

Solution

1. For each dam i compute how much water $f(i)$ is needed if we add water at i.

D — Dams in Distress

Problem

We get a rooted tree forming a system of $n \leq 200\,000$ dams. Overflowing a dam causes it to break and release all its water downstream. What is minimum amount of water we need to add at one dam in order for w units of water to reach the root?

Solution

1. For each dam i compute how much water $f(i)$ is needed if we add water at i.
2. For the root, $f(i) = w$.

D — Dams in Distress

Problem

We get a rooted tree forming a system of $n \leq 200\,000$ dams. Overflowing a dam causes it to break and release all its water downstream. What is minimum amount of water we need to add at one dam in order for w units of water to reach the root?

Solution

1. For each dam i compute how much water $f(i)$ is needed if we add water at i.
2. For the root, $f(i) = w$.
3. For non-root with parent p_i, capacity c_i and currently u_i water in it:
   - Need to add $c_i - u_i$ water to break the dam, this causes c_i water to go upstream.
   - Need to add $f(p_i)$ water at p_i, so need to add $f(p_i) - c_i$ more water if $f(p_i) > c_i$.

D — Dams in Distress

Problem

We get a rooted tree forming a system of $n \leq 200\,000$ dams. Overflowing a dam causes it to break and release all its water downstream. What is minimum amount of water we need to add at one dam in order for w units of water to reach the root?

Solution

1. For each dam i compute how much water $f(i)$ is needed if we add water at i.
2. For the root, $f(i) = w$.
3. For non-root with parent p_i, capacity c_i and currently u_i water in it:
   - Need to add $c_i - u_i$ water to break the dam, this causes c_i water to go upstream.
   - Need to add $f(p_i)$ water at p_i, so need to add $f(p_i) - c_i$ more water if $f(p_i) > c_i$.

Gives equation $f(i) = c_i - u_i + \max(0, f(p_i) - c_i)$.

D — Dams in Distress

Problem

We get a rooted tree forming a system of $n \leq 200\,000$ dams. Overflowing a dam causes it to break and release all its water downstream. What is minimum amount of water we need to add at one dam in order for w units of water to reach the root?

Solution

1. For each dam i compute how much water $f(i)$ is needed if we add water at i.
2. For the root, $f(i) = w$.
3. For non-root with parent p_i, capacity c_i and currently u_i water in it:
   - Need to add $c_i - u_i$ water to break the dam, this causes c_i water to go upstream.
   - Need to add $f(p_i)$ water at p_i, so need to add $f(p_i) - c_i$ more water if $f(p_i) > c_i$.

   Gives equation $f(i) = c_i - u_i + \max(0, f(p_i) - c_i)$.
4. Complexity $O(n)$ – compute top-down so $f(p_i)$ is known when computing $f(i)$.

D — Dams in Distress

Problem

We get a rooted tree forming a system of $n \leq 200\,000$ dams. Overflowing a dam causes it to break and release all its water downstream. What is minimum amount of water we need to add at one dam in order for w units of water to reach the root?

Solution

1. For each dam i compute how much water $f(i)$ is needed if we add water at i.
2. For the root, $f(i) = w$.
3. For non-root with parent p_i, capacity c_i and currently u_i water in it:
   - Need to add $c_i - u_i$ water to break the dam, this causes c_i water to go upstream.
   - Need to add $f(p_i)$ water at p_i, so need to add $f(p_i) - c_i$ more water if $f(p_i) > c_i$.

   Gives equation $f(i) = c_i - u_i + \max(0, f(p_i) - c_i)$.
4. Complexity $O(n)$ – compute top-down so $f(p_i)$ is known when computing $f(i)$.

Statistics at 4-hour mark: 270 submissions, 90 accepted, first after 00:32

G — Gig Combinatorics

Problem

Given sequence of $n \leq 10^6$ digits 1/2/3, count how many subsequences have the form "12+3" ("2+" means 1 or more twos)

G — Gig Combinatorics

Problem

Given sequence of $n \leq 10^6$ digits 1/2/3, count how many subsequences have the form "12+3" ("2+" means 1 or more twos)

Solution

1. Let ones(i) be number of ones up to position i in the sequence.

G — Gig Combinatorics

Problem

Given sequence of $n \leq 10^6$ digits 1/2/3, count how many subsequences have the form "12+3" ("2+" means 1 or more twos)

Solution

1. Let ones(i) be number of ones up to position i in the sequence.
2. Let $p(i)$ be number of subsequences of form "12+" on the first i digits.

G — Gig Combinatorics

Problem

Given sequence of $n \leq 10^6$ digits 1/2/3, count how many subsequences have the form "12+3" ("2+" means 1 or more twos)

Solution

1. Let ones(i) be number of ones up to position i in the sequence.
2. Let $p(i)$ be number of subsequences of form "12+" on the first i digits.
   - If ith digit is 2 then $p(i) = 2 \cdot p(i-1) + \text{ones}(i)$

G — Gig Combinatorics

Problem

Given sequence of $n \leq 10^6$ digits 1/2/3, count how many subsequences have the form "12+3" ("2+" means 1 or more twos)

Solution

1. Let $\text{ones}(i)$ be number of ones up to position i in the sequence.
2. Let $p(i)$ be number of subsequences of form "12+" on the first i digits.
   - If ith digit is 2 then $p(i) = 2 \cdot p(i-1) + \text{ones}(i)$
   - Otherwise $p(i) = p(i-1)$

G — Gig Combinatorics

Problem

Given sequence of $n \leq 10^6$ digits 1/2/3, count how many subsequences have the form "12+3" ("2+" means 1 or more twos)

Solution

1. Let $\text{ones}(i)$ be number of ones up to position i in the sequence.
2. Let $p(i)$ be number of subsequences of form "12+" on the first i digits.
   - If ith digit is 2 then $p(i) = 2 \cdot p(i-1) + \text{ones}(i)$
   - Otherwise $p(i) = p(i-1)$
3. Answer is sum of $p(i)$ over all positions i where there we have a 3.

G — Gig Combinatorics

Problem

Given sequence of $n \leq 10^6$ digits 1/2/3, count how many subsequences have the form "12+3" ("2+" means 1 or more twos)

Solution

1. Let $\text{ones}(i)$ be number of ones up to position i in the sequence.
2. Let $p(i)$ be number of subsequences of form "12+" on the first i digits.
   - If ith digit is 2 then $p(i) = 2 \cdot p(i-1) + \text{ones}(i)$
   - Otherwise $p(i) = p(i-1)$
3. Answer is sum of $p(i)$ over all positions i where there we have a 3.
4. Time complexity $O(n)$.

G — Gig Combinatorics

Problem

Given sequence of $n \leq 10^6$ digits 1/2/3, count how many subsequences have the form "12+3" ("2+" means 1 or more twos)

Solution

1. Let $\text{ones}(i)$ be number of ones up to position i in the sequence.
2. Let $p(i)$ be number of subsequences of form "12+" on the first i digits.
   - If ith digit is 2 then $p(i) = 2 \cdot p(i-1) + \text{ones}(i)$
   - Otherwise $p(i) = p(i-1)$
3. Answer is sum of $p(i)$ over all positions i where there we have a 3.
4. Time complexity $O(n)$.

Statistics at 4-hour mark: 437 submissions, 78 accepted, first after 00:04

J — Joining Flows

Problem

Given $k \leq 10$ faucets with different temperatures and adjustable flow levels, determine if they can be combined to produce given flow level and temperature.

J — Joining Flows

Problem

Given $k \leq 10$ faucets with different temperatures and adjustable flow levels, determine if they can be combined to produce given flow level and temperature.

Solution

1. Check that desired flow is at least min possible flow and at most max possible flow.

J — Joining Flows

Problem

Given $k \leq 10$ faucets with different temperatures and adjustable flow levels, determine if they can be combined to produce given flow level and temperature.

Solution

1. Check that desired flow is at least min possible flow and at most max possible flow.
2. Find smallest possible temperature at the desired flow level:
   – need to use a_i flow from each faucet
   – then greedily fill up remaining desired flow using lowest temperature faucets.

J — Joining Flows

Problem

Given $k \leq 10$ faucets with different temperatures and adjustable flow levels, determine if they can be combined to produce given flow level and temperature.

Solution

1. Check that desired flow is at least min possible flow and at most max possible flow.
2. Find smallest possible temperature at the desired flow level:
   – need to use a_i flow from each faucet
   – then greedily fill up remaining desired flow using lowest temperature faucets.
3. Similarly find largest possible temperature.

J — Joining Flows

Problem

Given $k \leq 10$ faucets with different temperatures and adjustable flow levels, determine if they can be combined to produce given flow level and temperature.

Solution

1. Check that desired flow is at least min possible flow and at most max possible flow.
2. Find smallest possible temperature at the desired flow level:
   – need to use a_i flow from each faucet
   – then greedily fill up remaining desired flow using lowest temperature faucets.
3. Similarly find largest possible temperature.
4. If the desired temperature is in this range, it can be achieved.

J — Joining Flows

Problem

Given $k \leq 10$ faucets with different temperatures and adjustable flow levels, determine if they can be combined to produce given flow level and temperature.

Solution

1. Check that desired flow is at least min possible flow and at most max possible flow.
2. Find smallest possible temperature at the desired flow level:
   – need to use a_i flow from each faucet
   – then greedily fill up remaining desired flow using lowest temperature faucets.
3. Similarly find largest possible temperature.
4. If the desired temperature is in this range, it can be achieved.
5. Time complexity $O(k)$ per query. (Can also be done in $O(\log k)$ time per query.)

J — Joining Flows

Problem

Given $k \leq 10$ faucets with different temperatures and adjustable flow levels, determine if they can be combined to produce given flow level and temperature.

Solution

1. Check that desired flow is at least min possible flow and at most max possible flow.
2. Find smallest possible temperature at the desired flow level:
   – need to use a_i flow from each faucet
   – then greedily fill up remaining desired flow using lowest temperature faucets.
3. Similarly find largest possible temperature.
4. If the desired temperature is in this range, it can be achieved.
5. Time complexity $O(k)$ per query. (Can also be done in $O(\log k)$ time per query.)

Statistics at 4-hour mark: 128 submissions, 46 accepted, first after 00:44

F — Film Critics

Problem

We have one movie and n critics with opinions $x_1, \ldots, x_n$ on how good it is.

If current review average of the movie exceeds a reviewers opinion they will score it 0, otherwise they will score it m.

Order the critics so that the film ends up getting review average exactly k/n.

F — Film Critics

Problem

We have one movie and n critics with opinions $x_1, \ldots, x_n$ on how good it is.

If current review average of the movie exceeds a reviewers opinion they will score it 0, otherwise they will score it m.

Order the critics so that the film ends up getting review average exactly k/n.

Initial observations

1. Each review is either *positive* (score m) or *negative* (score 0).

F — Film Critics

Problem

We have one movie and n critics with opinions $x_1, \ldots, x_n$ on how good it is.

If current review average of the movie exceeds a reviewers opinion they will score it 0, otherwise they will score it m.

Order the critics so that the film ends up getting review average exactly k/n.

Initial observations

1. Each review is either *positive* (score m) or *negative* (score 0).
2. If there are p positive reviews, final review average is pm/n.

F — Film Critics

Problem

We have one movie and n critics with opinions $x_1, \ldots, x_n$ on how good it is.

If current review average of the movie exceeds a reviewers opinion they will score it 0, otherwise they will score it m.

Order the critics so that the film ends up getting review average exactly k/n.

Initial observations

1. Each review is either *positive* (score m) or *negative* (score 0).
2. If there are p positive reviews, final review average is pm/n.
3. So k must be divisible by m (otherwise `impossible`), and we need exactly $p = k/m$ positive reviews.

F — Film Critics

Solution

1. Key insights:

F — Film Critics

Solution

1. Key insights:we may assume that
   - the p highest x_i's will give a positive review, and the others a negative review.

F — Film Critics

Solution

1. Key insights:we may assume that
   - the p highest x_i's will give a positive review, and the others a negative review.
   - the positive reviewers will come in increasing order of x_i (and negative reviews in decreasing order)

F — Film Critics

Solution

1. Key insights:we may assume that
   - the p highest x_i's will give a positive review, and the others a negative review.
   - the positive reviewers will come in increasing order of x_i (and negative reviews in decreasing order)
2. Build the answer iteratively. Each iteration, two candidates for next reviewer:
   - lowest remaining x_i from among the p largest, or
   - largest remaining x_i form among the $n - p$ smallest

F — Film Critics

Solution

1. Key insights:we may assume that
   - the p highest x_i's will give a positive review, and the others a negative review.
   - the positive reviewers will come in increasing order of x_i (and negative reviews in decreasing order)
2. Build the answer iteratively. Each iteration, two candidates for next reviewer:
   - lowest remaining x_i from among the p largest, or
   - largest remaining x_i form among the $n - p$ smallest
3. Pick one that yields the desired outcome (positive or negative) given the current review average.

F — Film Critics

Solution

1. Key insights:we may assume that
   - the p highest x_i's will give a positive review, and the others a negative review.
   - the positive reviewers will come in increasing order of x_i (and negative reviews in decreasing order)
2. Build the answer iteratively. Each iteration, two candidates for next reviewer:
   - lowest remaining x_i from among the p largest, or
   - largest remaining x_i form among the $n - p$ smallest
3. Pick one that yields the desired outcome (positive or negative) given the current review average.
4. If no such choice then ordering is `impossible` (can happen if one of the groups has become empty and the other has remaining x_i too low/high for desired result).

F — Film Critics

Solution

1. Key insights:we may assume that
   - the p highest x_i's will give a positive review, and the others a negative review.
   - the positive reviewers will come in increasing order of x_i (and negative reviews in decreasing order)
2. Build the answer iteratively. Each iteration, two candidates for next reviewer:
   - lowest remaining x_i from among the p largest, or
   - largest remaining x_i form among the $n - p$ smallest
3. Pick one that yields the desired outcome (positive or negative) given the current review average.
4. If no such choice then ordering is `impossible` (can happen if one of the groups has become empty and the other has remaining x_i too low/high for desired result).
5. Time complexity $O(n \log n)$ for sorting then $O(n)$.

F — Film Critics

Solution

1. Key insights:we may assume that
   - the p highest x_i's will give a positive review, and the others a negative review.
   - the positive reviewers will come in increasing order of x_i (and negative reviews in decreasing order)
2. Build the answer iteratively. Each iteration, two candidates for next reviewer:
   - lowest remaining x_i from among the p largest, or
   - largest remaining x_i form among the $n - p$ smallest
3. Pick one that yields the desired outcome (positive or negative) given the current review average.
4. If no such choice then ordering is `impossible` (can happen if one of the groups has become empty and the other has remaining x_i too low/high for desired result).
5. Time complexity $O(n \log n)$ for sorting then $O(n)$.

Statistics at 4-hour mark: 62 submissions, 21 accepted, first after 00:43

K — Keep Calm and Carry Off

Problem

Given two 1 000 000-digit integers A and B, find the smallest non-negative integer X such that $A + X$ and $B - X$ (or $A - X$ and $B + X$) can be added without carry.

K — Keep Calm and Carry Off

Problem

Given two 1 000 000-digit integers A and B, find the smallest non-negative integer X such that $A + X$ and $B - X$ (or $A - X$ and $B + X$) can be added without carry.

Solution

Simplifying assumptions:

1. A and B have the same number of digits (or zero-pad)
2. it is always the first integer we add to (try both options; take the best one)

K — Keep Calm and Carry Off

Problem

Given two 1 000 000-digit integers A and B, find the smallest non-negative integer X such that $A + X$ and $B - X$ (or $A - X$ and $B + X$) can be added without carry.

Solution

1. If digits in ith position have $a_i + b_i \geq 10$ there is a carry in ith position.

K — Keep Calm and Carry Off

Problem

Given two 1 000 000-digit integers A and B, find the smallest non-negative integer X such that $A + X$ and $B - X$ (or $A - X$ and $B + X$) can be added without carry.

Solution

1. If digits in ith position have $a_i + b_i \geq 10$ there is a carry in ith position.
2. We must increment a_i and decrement b_i (mod 10) until there is no longer carry.

K — Keep Calm and Carry Off

Problem

Given two 1 000 000-digit integers A and B, find the smallest non-negative integer X such that $A + X$ and $B - X$ (or $A - X$ and $B + X$) can be added without carry.

Solution

1. If digits in ith position have $a_i + b_i \geq 10$ there is a carry in ith position.
2. We must increment a_i and decrement b_i (mod 10) until there is no longer carry.
3. This happens when a_i turns to 0 (i.e. we add $9 - a_i$).

K — Keep Calm and Carry Off

Problem

Given two 1 000 000-digit integers A and B, find the smallest non-negative integer X such that $A + X$ and $B - X$ (or $A - X$ and $B + X$) can be added without carry.

Solution

1. If digits in ith position have $a_i + b_i \geq 10$ there is a carry in ith position.
2. We must increment a_i and decrement b_i (mod 10) until there is no longer carry.
3. This happens when a_i turns to 0 (i.e. we add $9 - a_i$).
4. If i is the leftmost digit causing carry, turning it to 0 will turn all remaining digits to 0 as well and get rid of any carries there.

K — Keep Calm and Carry Off

Problem

Given two 1 000 000-digit integers A and B, find the smallest non-negative integer X such that $A + X$ and $B - X$ (or $A - X$ and $B + X$) can be added without carry.

Solution

1. If digits in ith position have $a_i + b_i \geq 10$ there is a carry in ith position.
2. We must increment a_i and decrement b_i (mod 10) until there is no longer carry.
3. This happens when a_i turns to 0 (i.e. we add $9 - a_i$).
4. If i is the leftmost digit causing carry, turning it to 0 will turn all remaining digits to 0 as well and get rid of any carries there.
5. Lets us compute $A + X$ easily, subtract A to get X.

K — Keep Calm and Carry Off

Problem

Given two 1 000 000-digit integers A and B, find the smallest non-negative integer X such that $A + X$ and $B - X$ (or $A - X$ and $B + X$) can be added without carry.

Solution

1. Caveat: turning the leftmost carry a_i into a 0 causes a_{i-1} to increase by 1, can cause a new carry in the previous digit.

K — Keep Calm and Carry Off

Problem

Given two 1 000 000-digit integers A and B, find the smallest non-negative integer X such that $A+X$ and $B-X$ (or $A-X$ and $B+X$) can be added without carry.

Solution

1. Caveat: turning the leftmost carry a_i into a 0 causes a_{i-1} to increase by 1, can cause a new carry in the previous digit.
2. Any preceding sequence of digits summing to 9 must also get their a_i's turned to 0. Example:

$$A = 811765432113$$

$$B = 111234567897$$

$$\text{Target } A+X = 812000000000$$

K — Keep Calm and Carry Off

Problem

Given two 1 000 000-digit integers A and B, find the smallest non-negative integer X such that $A+X$ and $B-X$ (or $A-X$ and $B+X$) can be added without carry.

Solution

1. Caveat: turning the leftmost carry a_i into a 0 causes a_{i-1} to increase by 1, can cause a new carry in the previous digit.
2. Any preceding sequence of digits summing to 9 must also get their a_i's turned to 0. Example:

$$A = 811765432113$$
$$B = 111234567897$$
$$\text{Target } A+X = 812000000000$$

Statistics at 4-hour mark: 115 submissions, 11 accepted, first after 00:44

B — Big Brother

Problem

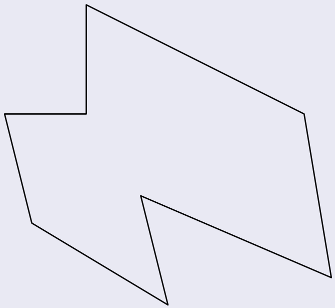
Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Solution



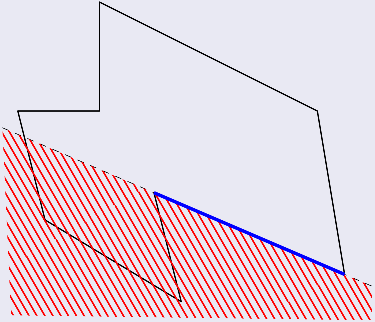
B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Solution

Line segments of the polygon induce *half-planes*:
In order for points along that wall not to be obscured, we cannot be behind that wall.



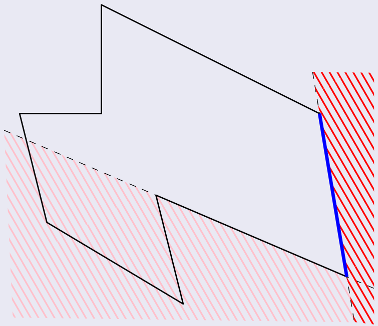
B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Solution

Line segments of the polygon induce *half-planes*:
In order for points along that wall not to be obscured, we cannot be behind that wall.



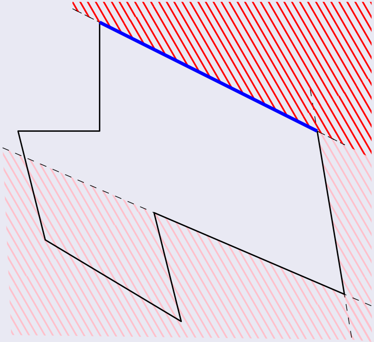
B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Solution

Line segments of the polygon induce *half-planes*:
In order for points along that wall not to be obscured, we cannot be behind that wall.



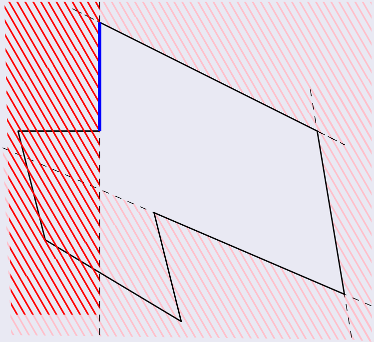
B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Solution

Line segments of the polygon induce *half-planes*:
In order for points along that wall not to be obscured, we cannot be behind that wall.



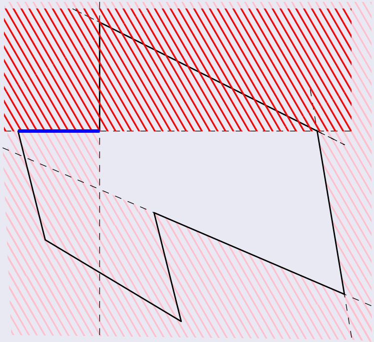
B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Solution

Line segments of the polygon induce *half-planes*: In order for points along that wall not to be obscured, we cannot be behind that wall.



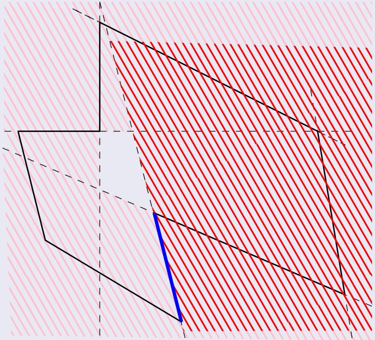
B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Solution

Line segments of the polygon induce *half-planes*:
In order for points along that wall not to be obscured, we cannot be behind that wall.



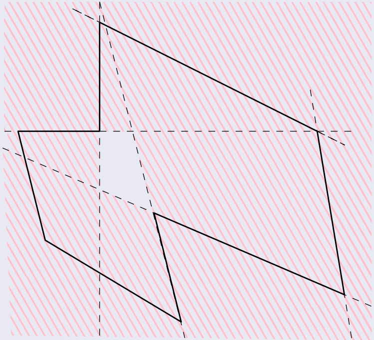
B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Solution

Line segments of the polygon induce *half-planes*: In order for points along that wall not to be obscured, we cannot be behind that wall.



Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Solution

Line segments of the polygon induce *half-planes*:
In order for points along that wall not to be obscured, we cannot be behind that wall.

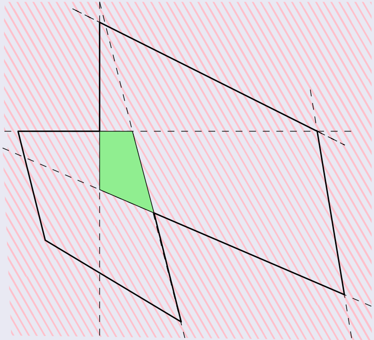
Seeing all parts of the room

$\Updownarrow$

Not “behind” any wall

$\Updownarrow$

Inside the intersection of the n half-planes.



B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Finding intersection of half-spaces

- Divide and conquer algorithm for intersection of half-spaces:

B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Finding intersection of half-spaces

- Divide and conquer algorithm for intersection of half-spaces:
  1. Split the half-spaces into two groups H_1 and H_2 of roughly $n/2$ half-spaces each.

B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Finding intersection of half-spaces

- Divide and conquer algorithm for intersection of half-spaces:
    1. Split the half-spaces into two groups H_1 and H_2 of roughly $n/2$ half-spaces each.
    2. Recursively compute intersection K_1 of half-spaces in H_1

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Finding intersection of half-spaces

- Divide and conquer algorithm for intersection of half-spaces:
  1. Split the half-spaces into two groups H_1 and H_2 of roughly $n/2$ half-spaces each.
  2. Recursively compute intersection K_1 of half-spaces in H_1
  3. Recursively compute intersection K_2 of half-spaces in H_2

B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Finding intersection of half-spaces

- Divide and conquer algorithm for intersection of half-spaces:
  1. Split the half-spaces into two groups H_1 and H_2 of roughly $n/2$ half-spaces each.
  2. Recursively compute intersection K_1 of half-spaces in H_1
  3. Recursively compute intersection K_2 of half-spaces in H_2
  4. Compute intersection of K_1 and K_2

B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Finding intersection of half-spaces

- Divide and conquer algorithm for intersection of half-spaces:
  1. Split the half-spaces into two groups H_1 and H_2 of roughly $n/2$ half-spaces each.
  2. Recursively compute intersection K_1 of half-spaces in H_1
  3. Recursively compute intersection K_2 of half-spaces in H_2
  4. Compute intersection of K_1 and K_2
- Analysis:
  1. K_1 and K_2 are convex regions (possibly unbounded), can compute their intersection in $O(n)$ time.

B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Finding intersection of half-spaces

- Divide and conquer algorithm for intersection of half-spaces:
  1. Split the half-spaces into two groups H_1 and H_2 of roughly $n/2$ half-spaces each.
  2. Recursively compute intersection K_1 of half-spaces in H_1
  3. Recursively compute intersection K_2 of half-spaces in H_2
  4. Compute intersection of K_1 and K_2
- Analysis:
  1. K_1 and K_2 are convex regions (possibly unbounded), can compute their intersection in $O(n)$ time.
  2. Get recurrence $T(n) = 2T(n/2) + O(n)$, yields $O(n \log n)$ time complexity.

B — Big Brother

Problem

Given polygon-shaped map of a room, find region from which all parts of the room can be seen.

Finding intersection of half-spaces

- Divide and conquer algorithm for intersection of half-spaces:
  1. Split the half-spaces into two groups H_1 and H_2 of roughly $n/2$ half-spaces each.
  2. Recursively compute intersection K_1 of half-spaces in H_1
  3. Recursively compute intersection K_2 of half-spaces in H_2
  4. Compute intersection of K_1 and K_2
- Analysis:
  1. K_1 and K_2 are convex regions (possibly unbounded), can compute their intersection in $O(n)$ time.
  2. Get recurrence $T(n) = 2T(n/2) + O(n)$, yields $O(n \log n)$ time complexity.

Statistics at 4-hour mark: 36 submissions, 11 accepted, first after 00:28

I — Infection Estimation

Problem

Estimate within a factor 2 the number of infected people in a population, using 50 tests of the form "choose k people at random and check if at least one of them is infected".

Problem

Estimate within a factor 2 the number of infected people in a population, using 50 tests of the form "choose k people at random and check if at least one of them is infected".

Solution

1. To reduce number of possible answers, only consider answers

$$100, \quad 100 \cdot 1.01, \quad 100 \cdot 1.01^2, \quad \dots \quad 100 \cdot 1.01^i, \quad \dots \quad 5 \cdot 10^6$$

(around $\log_{1.01}(5 \cdot 10^6) \approx 1500$ different values)

Problem

Estimate within a factor 2 the number of infected people in a population, using 50 tests of the form "choose k people at random and check if at least one of them is infected".

Solution

1. To reduce number of possible answers, only consider answers

$$100, \quad 100 \cdot 1.01, \quad 100 \cdot 1.01^2, \quad \ldots \quad 100 \cdot 1.01^i, \quad \ldots \quad 5 \cdot 10^6$$

(around $\log_{1.01}(5 \cdot 10^6) \approx 1500$ different values)

2. Maintain probability distribution of likelihood of each answer (initially uniform).

Problem

Estimate within a factor 2 the number of infected people in a population, using 50 tests of the form "choose k people at random and check if at least one of them is infected".

Solution

1. To reduce number of possible answers, only consider answers
$$100, \quad 100 \cdot 1.01, \quad 100 \cdot 1.01^2, \quad \ldots \quad 100 \cdot 1.01^i, \quad \ldots \quad 5 \cdot 10^6$$
(around $\log_{1.01}(5 \cdot 10^6) \approx 1500$ different values)
2. Maintain probability distribution of likelihood of each answer (initially uniform).
3. Each round, choose k such that test result yes/no probability is close to 50-50.

I — Infection Estimation

Problem

Estimate within a factor 2 the number of infected people in a population, using 50 tests of the form "choose k people at random and check if at least one of them is infected".

Solution

1. To reduce number of possible answers, only consider answers
$$100, \quad 100 \cdot 1.01, \quad 100 \cdot 1.01^2, \quad \ldots \quad 100 \cdot 1.01^i, \quad \ldots \quad 5 \cdot 10^6$$
(around $\log_{1.01}(5 \cdot 10^6) \approx 1500$ different values)
2. Maintain probability distribution of likelihood of each answer (initially uniform).
3. Each round, choose k such that test result yes/no probability is close to 50-50.
4. Given result, update likelihoods using Bayes's theorem
$$\Pr[\text{infected} = t \mid \text{yes}] = \frac{\Pr[\text{yes} \mid \text{infected} = t] \cdot \Pr[\text{infected} = t]}{\Pr[\text{yes}]}$$

I — Infection Estimation

Problem

Estimate within a factor 2 the number of infected people in a population, using 50 tests of the form “choose k people at random and check if at least one of them is infected”.

Solution

1. To reduce number of possible answers, only consider answers
$$100, \quad 100 \cdot 1.01, \quad 100 \cdot 1.01^2, \quad \ldots \quad 100 \cdot 1.01^i, \quad \ldots \quad 5 \cdot 10^6$$
(around $\log_{1.01}(5 \cdot 10^6) \approx 1500$ different values)
2. Maintain probability distribution of likelihood of each answer (initially uniform).
3. Each round, choose k such that test result yes/no probability is close to 50-50.
4. Given result, update likelihoods using Bayes's theorem
$$\Pr[\text{infected} = t \mid \text{yes}] = \frac{\Pr[\text{yes} \mid \text{infected} = t] \cdot \Pr[\text{infected} = t]}{\Pr[\text{yes}]}$$

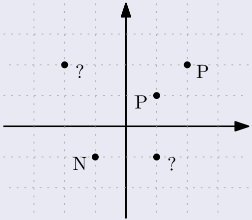
Statistics at 4-hour mark: 46 submissions, 6 accepted, first after 01:03

E — Exhaustive Experiment

Problem

We have n points which are potentially faulty. We can test points but tests only tell us if there is a faulty point within a cone above the test point. Given test results what is minimum number of faulty points?

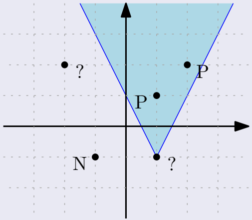
Solution (1/2)



Problem

We have n points which are potentially faulty. We can test points but tests only tell us if there is a faulty point within a cone above the test point. Given test results what is minimum number of faulty points?

Solution (1/2)



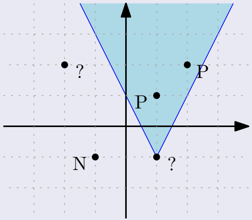
E — Exhaustive Experiment

Problem

We have n points which are potentially faulty. We can test points but tests only tell us if there is a faulty point within a cone above the test point. Given test results what is minimum number of faulty points?

Solution (1/2)

1. First let us simplify the geometry: scale x-coordinate by 2 and rotate 45 degrees



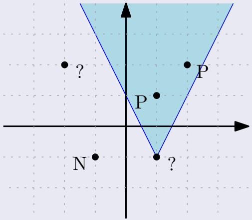
E — Exhaustive Experiment

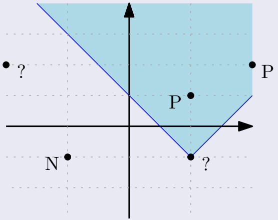
Problem

We have n points which are potentially faulty. We can test points but tests only tell us if there is a faulty point within a cone above the test point. Given test results what is minimum number of faulty points?

Solution (1/2)

1. First let us simplify the geometry: scale x-coordinate by 2 and rotate 45 degrees





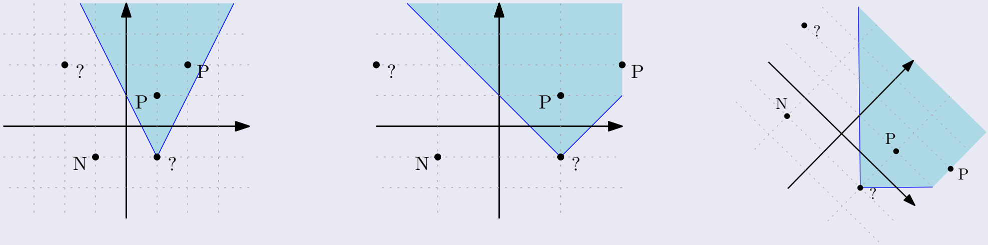
E — Exhaustive Experiment

Problem

We have n points which are potentially faulty. We can test points but tests only tell us if there is a faulty point within a cone above the test point. Given test results what is minimum number of faulty points?

Solution (1/2)

1. First let us simplify the geometry: scale x-coordinate by 2 and rotate 45 degrees



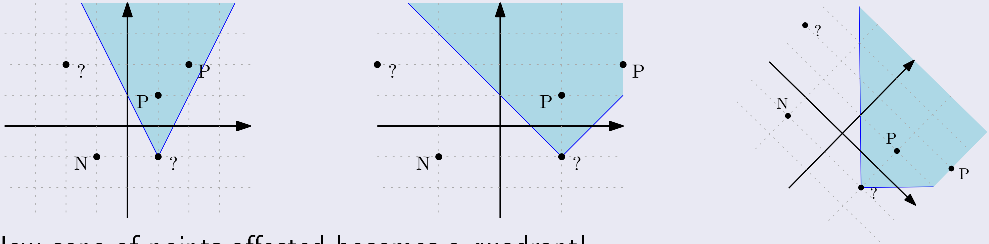
E — Exhaustive Experiment

Problem

We have n points which are potentially faulty. We can test points but tests only tell us if there is a faulty point within a cone above the test point. Given test results what is minimum number of faulty points?

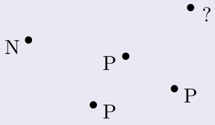
Solution (1/2)

1. First let us simplify the geometry: scale x-coordinate by 2 and rotate 45 degrees



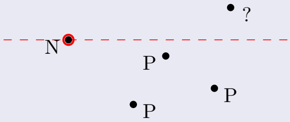
2. Now cone of points affected becomes a quadrant!

Solution (2/2)



- Negative tests: nothing in upper right quadrant of negative test can be faulty.

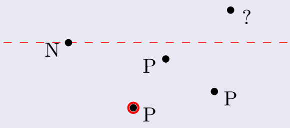
Solution (2/2)



- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen

E — Exhaustive Experiment

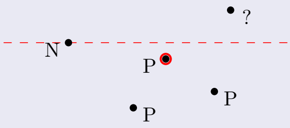
Solution (2/2)



- Negative tests: nothing in upper right quadrant of negative test can be faulty.
    1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen

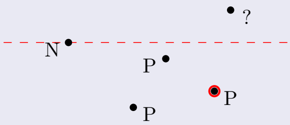
E — Exhaustive Experiment

Solution (2/2)



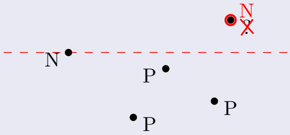
- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen

Solution (2/2)



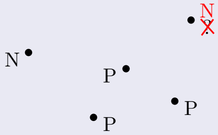
- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen

Solution (2/2)



- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen
  2. Points above current minimum must also be negative (if marked P $\Rightarrow$ "impossible")

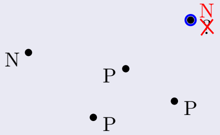
Solution (2/2)



- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen
  2. Points above current minimum must also be negative (if marked P $\Rightarrow$ "impossible")
- Greedily find smallest possible number of positive points:

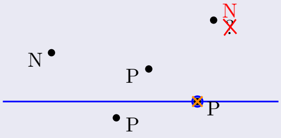
E — Exhaustive Experiment

Solution (2/2)



- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen
  2. Points above current minimum must also be negative (if marked P $\Rightarrow$ "impossible")
- Greedily find smallest possible number of positive points:
  1. Sweep from right to left, maintaining maximum y-coordinate of a faulty point, and maximum y-coordinate of a *potentially faulty* point.

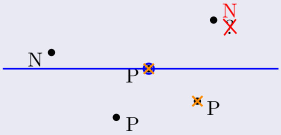
Solution (2/2)



- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen
  2. Points above current minimum must also be negative (if marked P $\Rightarrow$ “impossible”)
- Greedily find smallest possible number of positive points:
  1. Sweep from right to left, maintaining maximum y-coordinate of a faulty point, and maximum y-coordinate of a *potentially faulty* point.

E — Exhaustive Experiment

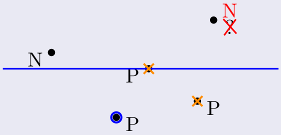
Solution (2/2)



- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen
  2. Points above current minimum must also be negative (if marked P $\Rightarrow$ “impossible”)
- Greedily find smallest possible number of positive points:
  1. Sweep from right to left, maintaining maximum y-coordinate of a faulty point, and maximum y-coordinate of a *potentially faulty* point.

E — Exhaustive Experiment

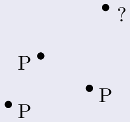
Solution (2/2)



- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen
  2. Points above current minimum must also be negative (if marked P $\Rightarrow$ "impossible")
- Greedily find smallest possible number of positive points:
  1. Sweep from right to left, maintaining maximum y-coordinate of a faulty point, and maximum y-coordinate of a *potentially faulty* point.

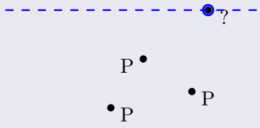
E — Exhaustive Experiment

Solution (2/2)



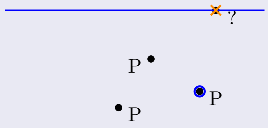
- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum *y*-coordinate of a negative point seen
  2. Points above current minimum must also be negative (if marked P $\Rightarrow$ "impossible")
- Greedily find smallest possible number of positive points:
  1. Sweep from right to left, maintaining maximum *y*-coordinate of a faulty point, and maximum *y*-coordinate of a *potentially faulty* point.

Solution (2/2)



- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen
  2. Points above current minimum must also be negative (if marked P $\Rightarrow$ "impossible")
- Greedily find smallest possible number of positive points:
  1. Sweep from right to left, maintaining maximum y-coordinate of a faulty point, and maximum y-coordinate of a *potentially faulty* point.

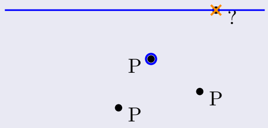
Solution (2/2)



- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen
  2. Points above current minimum must also be negative (if marked P $\Rightarrow$ "impossible")
- Greedily find smallest possible number of positive points:
  1. Sweep from right to left, maintaining maximum y-coordinate of a faulty point, and maximum y-coordinate of a *potentially faulty* point.
  2. When explanation for positive test needed, use best *potentially faulty* one seen so far.

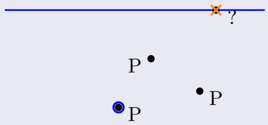
E — Exhaustive Experiment

Solution (2/2)



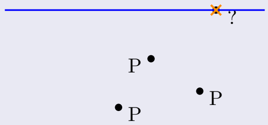
- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen
  2. Points above current minimum must also be negative (if marked P $\Rightarrow$ "impossible")
- Greedily find smallest possible number of positive points:
  1. Sweep from right to left, maintaining maximum y-coordinate of a faulty point, and maximum y-coordinate of a *potentially faulty* point.
  2. When explanation for positive test needed, use best *potentially faulty* one seen so far.

Solution (2/2)



- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen
  2. Points above current minimum must also be negative (if marked P $\Rightarrow$ "impossible")
- Greedily find smallest possible number of positive points:
  1. Sweep from right to left, maintaining maximum y-coordinate of a faulty point, and maximum y-coordinate of a *potentially faulty* point.
  2. When explanation for positive test needed, use best *potentially faulty* one seen so far.

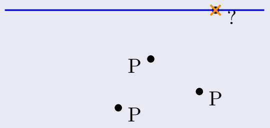
Solution (2/2)



- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen
  2. Points above current minimum must also be negative (if marked P $\Rightarrow$ "impossible")
- Greedily find smallest possible number of positive points:
  1. Sweep from right to left, maintaining maximum y-coordinate of a faulty point, and maximum y-coordinate of a *potentially faulty* point.
  2. When explanation for positive test needed, use best *potentially faulty* one seen so far.
- Time complexity $O(n \log n)$ for sorting then $O(n)$.

E — Exhaustive Experiment

Solution (2/2)



- Negative tests: nothing in upper right quadrant of negative test can be faulty.
  1. Sweep from left to right, maintaining minimum y-coordinate of a negative point seen
  2. Points above current minimum must also be negative (if marked P $\Rightarrow$ "impossible")
- Greedily find smallest possible number of positive points:
  1. Sweep from right to left, maintaining maximum y-coordinate of a faulty point, and maximum y-coordinate of a *potentially faulty* point.
  2. When explanation for positive test needed, use best *potentially faulty* one seen so far.
- Time complexity $O(n \log n)$ for sorting then $O(n)$.

Statistics at 4-hour mark: 18 submissions, 3 accepted, first after 01:14

H — Hiring and Firing

Problem

A company hires and fires workers over up to 100 000 days. Assign an HR employee to each day so that for each worker a different HR employee is assigned the day they are hired and the day they are fired. Minimize number of HR employees used.

Solution (1/3)

H — Hiring and Firing

Problem

A company hires and fires workers over up to 100 000 days. Assign an HR employee to each day so that for each worker a different HR employee is assigned the day they are hired and the day they are fired. Minimize number of HR employees used.

Solution (1/3)

1. The hiring/firings form a graph: each day is a vertex, and each worker is an edge between the day they are hired and the day they are fired.

H — Hiring and Firing

Problem

A company hires and fires workers over up to 100 000 days. Assign an HR employee to each day so that for each worker a different HR employee is assigned the day they are hired and the day they are fired. Minimize number of HR employees used.

Solution (1/3)

1. The hiring/firings form a graph: each day is a vertex, and each worker is an edge between the day they are hired and the day they are fired.
2. We seek the chromatic number of this graph.

H — Hiring and Firing

Problem

A company hires and fires workers over up to 100 000 days. Assign an HR employee to each day so that for each worker a different HR employee is assigned the day they are hired and the day they are fired. Minimize number of HR employees used.

Solution (1/3)

1. The hiring/firings form a graph: each day is a vertex, and each worker is an edge between the day they are hired and the day they are fired.
2. We seek the chromatic number of this graph.
3. Graph coloring is NP-hard in general and even in planar graphs.

H — Hiring and Firing

Problem

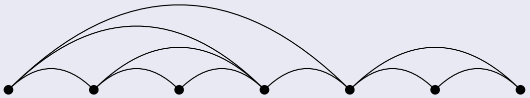
A company hires and fires workers over up to 100 000 days. Assign an HR employee to each day so that for each worker a different HR employee is assigned the day they are hired and the day they are fired. Minimize number of HR employees used.

Solution (1/3)

1. The hiring/firings form a graph: each day is a vertex, and each worker is an edge between the day they are hired and the day they are fired.
2. We seek the chromatic number of this graph.
3. Graph coloring is NP-hard in general and even in planar graphs.
   But these graphs are special... what do they look like?

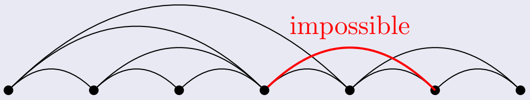
Solution (2/3)

1. Mark the days on a timeline and draw an arc for each worker:



Solution (2/3)

1. Mark the days on a timeline and draw an arc for each worker:

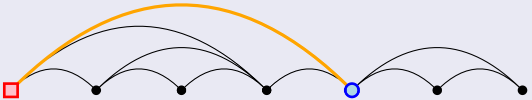


2. Because of last-in-first-out order of hirings and firings, cannot be any crossing arcs.

H — Hiring and Firing

Solution (2/3)

1. Mark the days on a timeline and draw an arc for each worker:



2. Because of last-in-first-out order of hirings and firings, cannot be any crossing arcs.
3. Gives rise to a recursive structure that we can use to color the graph:
   start with longest edge from leftmost vertex and color its endpoints arbitrarily

H — Hiring and Firing

Solution (2/3)

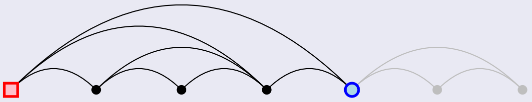
1. Mark the days on a timeline and draw an arc for each worker:

2. Because of last-in-first-out order of hirings and firings, cannot be any crossing arcs.
3. Gives rise to a recursive structure that we can use to color the graph:
   start with longest edge from leftmost vertex and color its endpoints arbitrarily
4. No-crossing property $\Rightarrow$ separation into two independent subproblems.

H — Hiring and Firing

Solution (2/3)

1. Mark the days on a timeline and draw an arc for each worker:

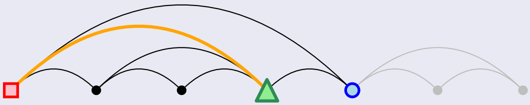


2. Because of last-in-first-out order of hirings and firings, cannot be any crossing arcs.
3. Gives rise to a recursive structure that we can use to color the graph:
   start with longest edge from leftmost vertex and color its endpoints arbitrarily
4. No-crossing property $\Rightarrow$ separation into two independent subproblems.

H — Hiring and Firing

Solution (2/3)

1. Mark the days on a timeline and draw an arc for each worker:

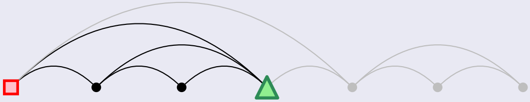


2. Because of last-in-first-out order of hirings and firings, cannot be any crossing arcs.
3. Gives rise to a recursive structure that we can use to color the graph:
   start with longest edge from leftmost vertex and color its endpoints arbitrarily
4. No-crossing property $\Rightarrow$ separation into two independent subproblems.
5. Recurse on subproblems and repeat. Use a valid color for the middle vertex.

H — Hiring and Firing

Solution (2/3)

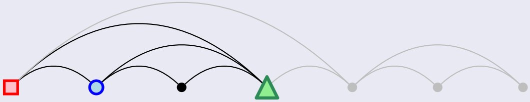
1. Mark the days on a timeline and draw an arc for each worker:



2. Because of last-in-first-out order of hirings and firings, cannot be any crossing arcs.
3. Gives rise to a recursive structure that we can use to color the graph: start with longest edge from leftmost vertex and color its endpoints arbitrarily
4. No-crossing property $\Rightarrow$ separation into two independent subproblems.
5. Recurse on subproblems and repeat. Use a valid color for the middle vertex.

Solution (2/3)

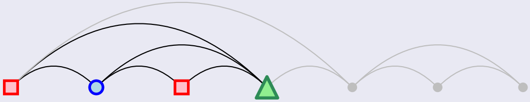
1. Mark the days on a timeline and draw an arc for each worker:



2. Because of last-in-first-out order of hirings and firings, cannot be any crossing arcs.
3. Gives rise to a recursive structure that we can use to color the graph: start with longest edge from leftmost vertex and color its endpoints arbitrarily
4. No-crossing property $\Rightarrow$ separation into two independent subproblems.
5. Recurse on subproblems and repeat. Use a valid color for the middle vertex.

Solution (2/3)

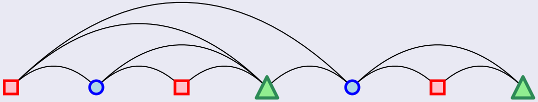
1. Mark the days on a timeline and draw an arc for each worker:



2. Because of last-in-first-out order of hirings and firings, cannot be any crossing arcs.
3. Gives rise to a recursive structure that we can use to color the graph: start with longest edge from leftmost vertex and color its endpoints arbitrarily
4. No-crossing property $\Rightarrow$ separation into two independent subproblems.
5. Recurse on subproblems and repeat. Use a valid color for the middle vertex.

Solution (2/3)

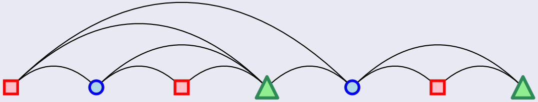
1. Mark the days on a timeline and draw an arc for each worker:



2. Because of last-in-first-out order of hirings and firings, cannot be any crossing arcs.
3. Gives rise to a recursive structure that we can use to color the graph: start with longest edge from leftmost vertex and color its endpoints arbitrarily
4. No-crossing property $\Rightarrow$ separation into two independent subproblems.
5. Recurse on subproblems and repeat. Use a valid color for the middle vertex.

Solution (2/3)

1. Mark the days on a timeline and draw an arc for each worker:

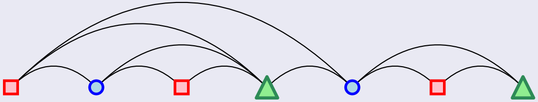


2. Because of last-in-first-out order of hirings and firings, cannot be any crossing arcs.
3. Gives rise to a recursive structure that we can use to color the graph: start with longest edge from leftmost vertex and color its endpoints arbitrarily
4. No-crossing property $\Rightarrow$ separation into two independent subproblems.
5. Recurse on subproblems and repeat. Use a valid color for the middle vertex.
6. In each subproblem, only left & right of current range have already been colored.

H — Hiring and Firing

Solution (2/3)

1. Mark the days on a timeline and draw an arc for each worker:

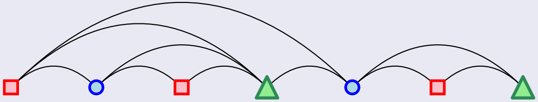


2. Because of last-in-first-out order of hirings and firings, cannot be any crossing arcs.
3. Gives rise to a recursive structure that we can use to color the graph: start with longest edge from leftmost vertex and color its endpoints arbitrarily
4. No-crossing property ⇒ separation into two independent subproblems.
5. Recurse on subproblems and repeat. Use a valid color for the middle vertex.
6. In each subproblem, only left & right of current range have already been colored. So if we have three colors, will always be a choice available for the middle vertex.

H — Hiring and Firing

Solution (2/3)

1. Mark the days on a timeline and draw an arc for each worker:



2. Because of last-in-first-out order of hirings and firings, cannot be any crossing arcs.
3. Gives rise to a recursive structure that we can use to color the graph:
   start with longest edge from leftmost vertex and color its endpoints arbitrarily
4. No-crossing property $\Rightarrow$ separation into two independent subproblems.
5. Recurse on subproblems and repeat. Use a valid color for the middle vertex.
6. In each subproblem, only left & right of current range have already been colored.
   So if we have three colors, will always be a choice available for the middle vertex.
   In other words, 3 colors is always enough!

H — Hiring and Firing

Solution (3/3)

1. Check if graph is 2-colorable (or 1-colorable), standard algorithm.

H — Hiring and Firing

Solution (3/3)

1. Check if graph is 2-colorable (or 1-colorable), standard algorithm.
2. If not, construct a 3-coloring using the procedure described before (or in some other way, many similar strategies work)

Solution (3/3)

1. Check if graph is 2-colorable (or 1-colorable), standard algorithm.
2. If not, construct a 3-coloring using the procedure described before (or in some other way, many similar strategies work)
3. Can be implemented in $O(n)$ time.

Solution (3/3)

1. Check if graph is 2-colorable (or 1-colorable), standard algorithm.
2. If not, construct a 3-coloring using the procedure described before (or in some other way, many similar strategies work)
3. Can be implemented in $O(n)$ time.

Statistics at 4-hour mark: 22 submissions, 0 accepted, first after N/A

L — Language Survey

Problem

Find three connected regions in a grid. For each cell prescribed whether it should contain a single region or at least two different regions.

L — Language Survey

Problem

Find three connected regions in a grid. For each cell prescribed whether it should contain a single region or at least two different regions.

Solution (1/2)

1. Try to partition grid into three *disjoint* connected regions with following property: for every cell (i, j) there is a neighboring cell (i', j') belonging to a different region.

Problem

Find three connected regions in a grid. For each cell prescribed whether it should contain a single region or at least two different regions.

Solution (1/2)

1. Try to partition grid into three *disjoint* connected regions with following property:

   for every cell (i, j) there is a neighboring cell (i', j') belonging to a different region.

2. If we manage to find this, problem becomes easy:

L — Language Survey

Problem

Find three connected regions in a grid. For each cell prescribed whether it should contain a single region or at least two different regions.

Solution (1/2)

1. Try to partition grid into three *disjoint* connected regions with following property:

   for every cell (i, j) there is a neighboring cell (i', j') belonging to a different region.

2. If we manage to find this, problem becomes easy:
   - Use the partition as a starting point.

L — Language Survey

Problem

Find three connected regions in a grid. For each cell prescribed whether it should contain a single region or at least two different regions.

Solution (1/2)

1. Try to partition grid into three *disjoint* connected regions with following property:

   for every cell (i, j) there is a neighboring cell (i', j') belonging to a different region.

2. If we manage to find this, problem becomes easy:
   - Use the partition as a starting point.
   - For every cell (i, j) where two or more regions should overlap, extend the region from (i', j') into (i, j).

L — Language Survey

Problem

Find three connected regions in a grid. For each cell prescribed whether it should contain a single region or at least two different regions.

Solution (2/2)

1. How to find the starting point partition?

Problem

Find three connected regions in a grid. For each cell prescribed whether it should contain a single region or at least two different regions.

Solution (2/2)

1. How to find the starting point partition?
2. If dimensions not too small, one possible pattern ⟶

```
ACCCCCCCC
ACBBBBBBB
ACBCCCCCB
ACBCCCBCB
ACBBBBBCB
ACCCCCCCB
```

L — Language Survey

Problem

Find three connected regions in a grid. For each cell prescribed whether it should contain a single region or at least two different regions.

Solution (2/2)

1. How to find the starting point partition?
2. If dimensions not too small, one possible pattern ⟶
3. Case when dimensions are small is left as exercise...
   – when at least two rows and columns the above works
   – case with only one row or one column easy to solve directly

```
ACCCCCCCC
ACBBBBBBB
ACBCCCCCB
ACBCCCBCB
ACBBBBBCB
ACCCCCCCB
```

L — Language Survey

Problem

Find three connected regions in a grid. For each cell prescribed whether it should contain a single region or at least two different regions.

Solution (2/2)

1. How to find the starting point partition?
2. If dimensions not too small, one possible pattern ⟶
3. Case when dimensions are small is left as exercise...
   – when at least two rows and columns the above works
   – case with only one row or one column easy to solve directly

```
ACCCCCCC
ACBBBBBB
ACBCCCCB
ACBCCCBCB
ACBBBBCB
ACCCCCCB
```

Statistics at 4-hour mark: 5 submissions, 0 accepted, first after N/A

Results!