Winter Contest 2023

Solutions presentation

January 28, 2023

Winter Contest 2023 Jury

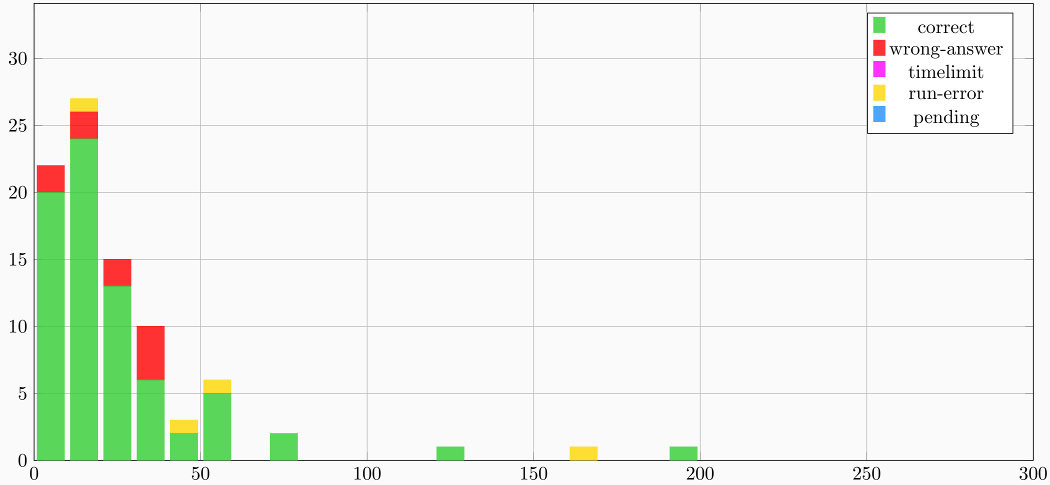
- **Florian Kothmeier**
  Friedrich–Alexander University Erlangen–Nürnberg
- **Felicia Lucke**
  CPUlm
- **Jannik Olbrich**
  CPUlm
- **Christopher Weyand**
  Karlsruhe Institute of Technology
- **Marcel Wienöbst**
  University of Lübeck
- **Wendy Yi**
  Karlsruhe Institute of Technology
- **Michael Zündorf**
  Karlsruhe Institute of Technology

Big thanks to our test solvers

- **Niko Fink**
  University of Passau
- **Michael Ruderer**
  CPUlm
- **Erik Sünderhauf**
  Technical University of Munich

I: Infinity Issues

Problem Author: Michael Zündorf



I: Infinity Issues

Problem Author: Michael Zündorf

Problem

Given a text, split it into lines of length exactly w

I: Infinity Issues

Problem Author: Michael Zündorf

Problem

Given a text, split it into lines of length exactly w

Solution

- Read the complete line containing the text
- Print it character for character
- If the position $i = 0 \bmod w$ print in addition a newline
  except if $i = 0$

I: Infinity Issues

Problem Author: Michael Zündorf

Problem

Given a text, split it into lines of length exactly w

Solution

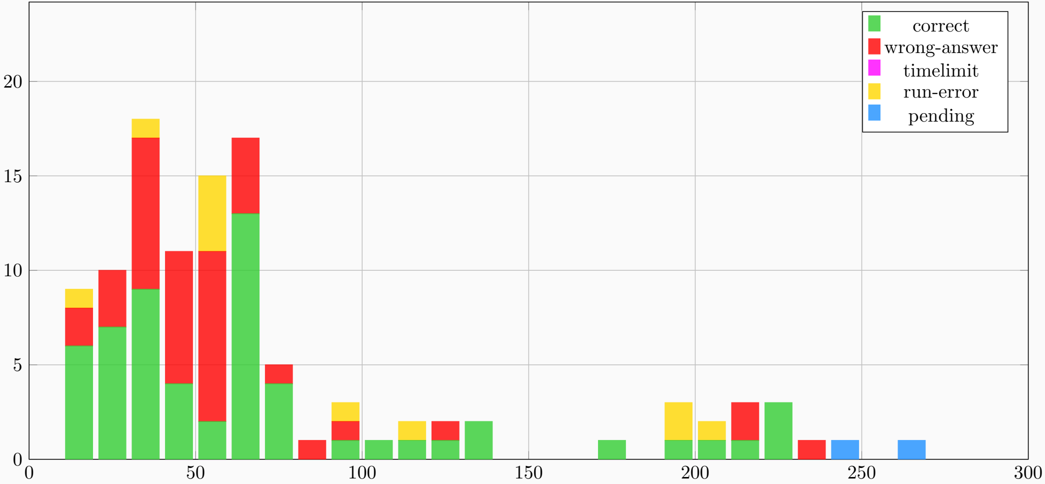
- Read the complete line containing the text
- Print it character for character
- If the position $i = 0 \mod w$ print in addition a newline
  except if $i = 0$

Tipps for Common Errors

If you combine `std::cin` and `std::getline` make sure that you read the `'\n'` character ending the previous line

C: Christmas Calories

Problem Author: Jannik Olbrich



C: Christmas Calories

Problem Author: Jannik Olbrich

Problem

Given a circle of radius r. What is the probability that a point on the circle drawn uniformly at random has distance at least ℓ from some other point on the circle?

C: Christmas Calories

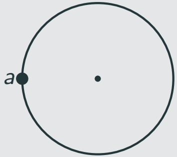
Problem Author: Jannik Olbrich

Problem

Given a circle of radius r. What is the probability that a point on the circle drawn uniformly at random has distance at least ℓ from some other point on the circle?

Solution

- Let the circle have center $(0, 0)$. Fix one point a on the circle (e.g. $(-r, 0)$)



C: Christmas Calories

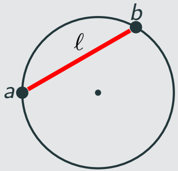
Problem Author: Jannik Olbrich

Problem

Given a circle of radius r. What is the probability that a point on the circle drawn uniformly at random has distance at least ℓ from some other point on the circle?

Solution

- Let the circle have center $(0, 0)$. Fix one point a on the circle (e.g. $(-r, 0)$)
- Consider a point b on the circle with distance ℓ to a



C: Christmas Calories

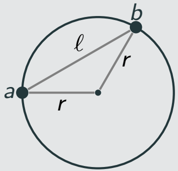
Problem Author: Jannik Olbrich

Problem

Given a circle of radius r. What is the probability that a point on the circle drawn uniformly at random has distance at least ℓ from some other point on the circle?

Solution

- Let the circle have center $(0, 0)$. Fix one point a on the circle (e.g. $(-r, 0)$)
- Consider a point b on the circle with distance ℓ to a
- a, b and $(0, 0)$ form a triangle with side lengths r, r and ℓ



C: Christmas Calories

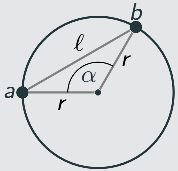
Problem Author: Jannik Olbrich

Problem

Given a circle of radius r. What is the probability that a point on the circle drawn uniformly at random has distance at least ℓ from some other point on the circle?

Solution

- Let the circle have center $(0, 0)$. Fix one point a on the circle (e.g. $(-r, 0)$)
- Consider a point b on the circle with distance ℓ to a
- a, b and $(0, 0)$ form a triangle with side lengths r, r and ℓ
- Compute the angle α: $\ell^2 = r^2 + r^2 - 2 \cdot r \cdot r \cdot \cos\alpha$ (law of cosines)



C: Christmas Calories

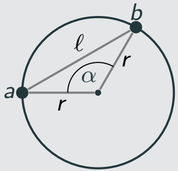
Problem Author: Jannik Olbrich

Problem

Given a circle of radius r. What is the probability that a point on the circle drawn uniformly at random has distance at least ℓ from some other point on the circle?

Solution

- Let the circle have center $(0,0)$. Fix one point a on the circle (e.g. $(-r,0)$)
- Consider a point b on the circle with distance ℓ to a
- a, b and $(0,0)$ form a triangle with side lengths r, r and ℓ
- Compute the angle α: $\ell^2 = r^2 + r^2 - 2 \cdot r \cdot r \cdot \cos\alpha$ (law of cosines)
- Answer is $1 - \alpha/\pi$



C: Christmas Calories

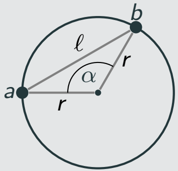
Problem Author: Jannik Olbrich

Problem

Given a circle of radius r. What is the probability that a point on the circle drawn uniformly at random has distance at least ℓ from some other point on the circle?

Solution

- Let the circle have center $(0, 0)$. Fix one point a on the circle (e.g. $(-r, 0)$)
- Consider a point b on the circle with distance ℓ to a
- a, b and $(0, 0)$ form a triangle with side lengths r, r and ℓ
- Compute the angle α: $\ell^2 = r^2 + r^2 - 2 \cdot r \cdot r \cdot \cos\alpha$ (law of cosines)
- Answer is $1 - \alpha/\pi$



Alternative solution: Binary search over α or e.g. the x-coordinate of b

C: Christmas Calories

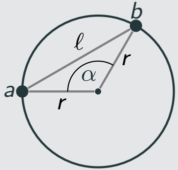
Problem Author: Jannik Olbrich

Problem

Given a circle of radius r. What is the probability that a point on the circle drawn uniformly at random has distance at least ℓ from some other point on the circle?

Solution

- Let the circle have center $(0, 0)$. Fix one point a on the circle (e.g. $(-r, 0)$)
- Consider a point b on the circle with distance ℓ to a
- a, b and $(0, 0)$ form a triangle with side lengths r, r and ℓ
- Compute the angle α: $\ell^2 = r^2 + r^2 - 2 \cdot r \cdot r \cdot \cos\alpha$ (law of cosines)
- Answer is $1 - \alpha/\pi$



Alternative solution: Binary search over α or e.g. the x-coordinate of b

C: Christmas Calories

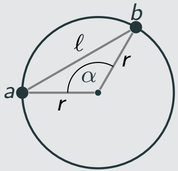
Problem Author: Jannik Olbrich

Problem

Given a circle of radius r. What is the probability that a point on the circle drawn uniformly at random has distance at least ℓ from some other point on the circle?

Solution

- Let the circle have center $(0, 0)$. Fix one point a on the circle (e.g. $(-r, 0)$)
- Consider a point b on the circle with distance ℓ to a
- a, b and $(0, 0)$ form a triangle with side lengths r, r and ℓ
- Compute the angle α: $\ell^2 = r^2 + r^2 - 2 \cdot r \cdot r \cdot \cos\alpha$ (law of cosines)
- Answer is $1 - \alpha/\pi$



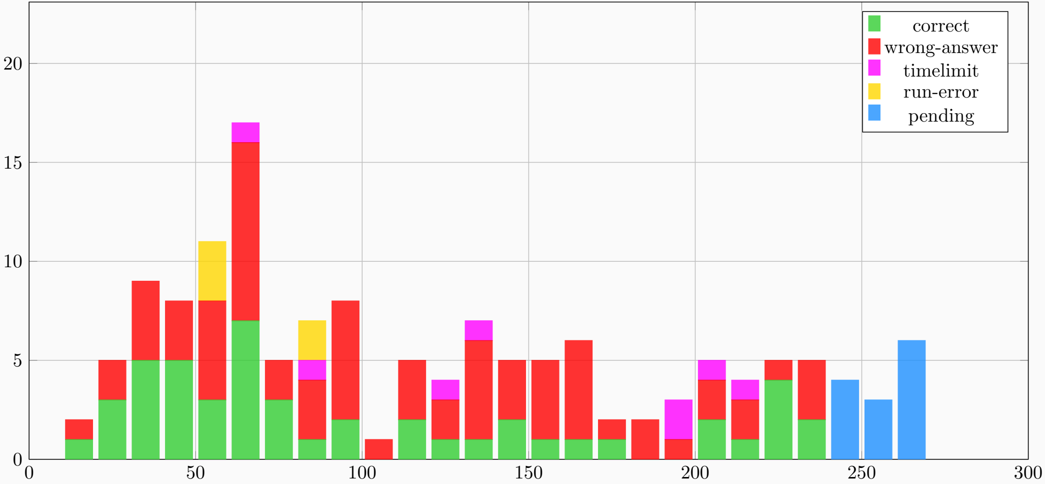
Alternative solution: Binary search over α or e.g. the x-coordinate of b

Possible pitfalls

- `float` is too imprecise
- Edge case $\ell > 2r$ may result in `NaN`

J: Jinxed Jewelry

Problem Author: Michael Zündorf

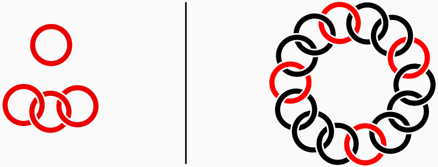


J: Jinxed Jewelry

Problem Author: Michael Zündorf

Problem

Given chains of various lengths, how many chain links do you need to open, interlock with other chain links and close again, to form a cyclic chain

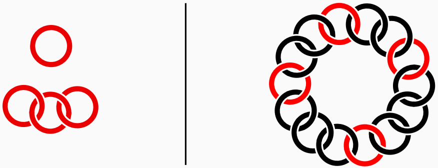


J: Jinxed Jewelry

Problem Author: Michael Zündorf

Problem

Given chains of various lengths, how many chain links do you need to open, interlock with other chain links and close again, to form a cyclic chain



Solution

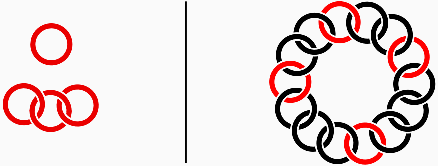
- You need to open a chain links such that you end up with $b \leq a$ chains remaining

J: Jinxed Jewelry

Problem Author: Michael Zündorf

Problem

Given chains of various lengths, how many chain links do you need to open, interlock with other chain links and close again, to form a cyclic chain



Solution

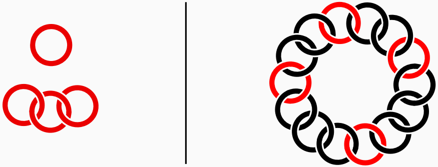
- You need to open a chain links such that you end up with $b \leq a$ chains remaining
- If $b > a$ You need to open more chain links

J: Jinxed Jewelry

Problem Author: Michael Zündorf

Problem

Given chains of various lengths, how many chain links do you need to open, interlock with other chain links and close again, to form a cyclic chain



Solution

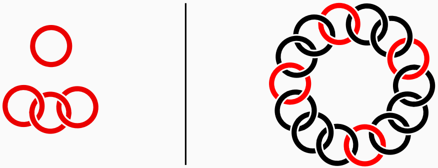
- You need to open a chain links such that you end up with $b \leq a$ chains remaining
- If $b > a$ You need to open more chain links
- If you open chain links from the shortest chain you have the chance to completely use up a chain
- This not only increases a but also decreases b

J: Jinxed Jewelry

Problem Author: Michael Zündorf

Problem

Given chains of various lengths, how many chain links do you need to open, interlock with other chain links and close again, to form a cyclic chain



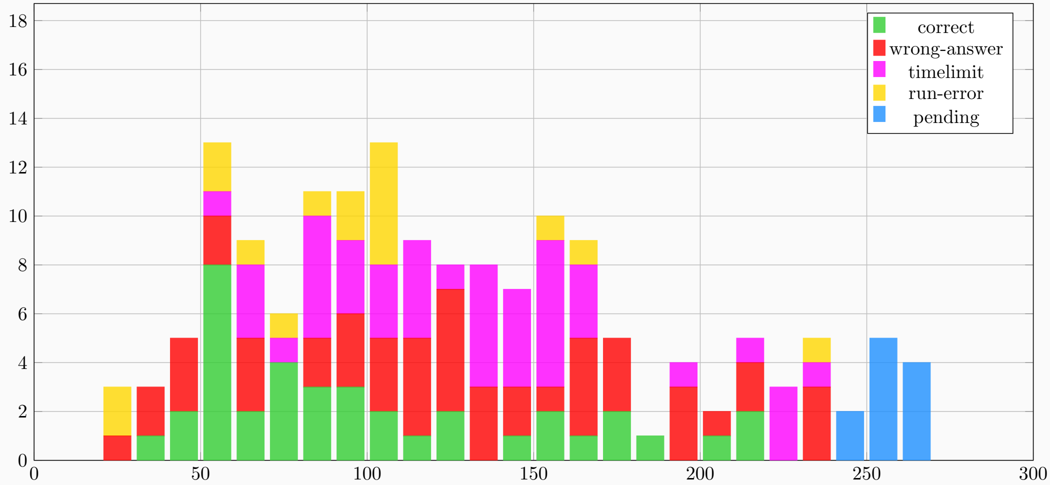
Solution

- You need to open a chain links such that you end up with $b \leq a$ chains remaining
- If $b > a$ You need to open more chain links
- If you open chain links from the shortest chain you have the chance to completely use up a chain
- This not only increases a but also decreases b

$\Rightarrow$ It is optimal to open chain links from the shortest chains first

D: Discus Domination

Problem Author: Florian Kothmeier



D: Discus Domination

Problem Author: Florian Kothmeier

Problem

Given an integers $a_1, \ldots, a_n$, maximise $a_j - a_i$, where $0 \leq j - i \leq m$ $(1 \leq n, m \leq 10^9)$

D: Discus Domination

Problem Author: Florian Kothmeier

Problem

Given an integers $a_1, \ldots, a_n$, maximise $a_j - a_i$, where $0 \leq j - i \leq m$ ($1 \leq n, m \leq 10^9$)

Solution

- Naive solution: Try each starting point and search for the highest value in range.

D: Discus Domination

Problem Author: Florian Kothmeier

Problem

Given an integers $a_1, \ldots, a_n$, maximise $a_j - a_i$, where $0 \leq j - i \leq m$ ($1 \leq n, m \leq 10^9$)

Solution

- Naive solution: Try each starting point and search for the highest value in range.
  $\Rightarrow \mathcal{O}(n \cdot m) \Rightarrow$ **too slow!**

D: Discus Domination

Problem Author: Florian Kothmeier

Problem

Given an integers $a_1, \ldots, a_n$, maximise $a_j - a_i$, where $0 \leq j - i \leq m$ $(1 \leq n, m \leq 10^9)$

Solution

- Naive solution: Try each starting point and search for the highest value in range.
  $\Rightarrow \mathcal{O}(n \cdot m) \Rightarrow$ **too slow!**
- Idea: maximum value for a point a_j (discus landing point) is given by the smallest starting position a_i in the last m values.

D: Discus Domination

Problem Author: Florian Kothmeier

Problem

Given an integers $a_1, \dots, a_n$, maximise $a_j - a_i$, where $0 \leq j - i \leq m$ ($1 \leq n, m \leq 10^9$)

Solution

- Naive solution: Try each starting point and search for the highest value in range.
  $\Rightarrow \mathcal{O}(n \cdot m) \Rightarrow$ **too slow!**
- Idea: maximum value for a point a_j (discus landing point) is given by the smallest starting position a_i in the last m values.
  - Can be queried in $\mathcal{O}(\log m)$ using a min-heap.
    - C++: use `std::multiset` or `std::map`
    - Java: use `java.util.TreeMap`
    - Python: `from queue import PriorityQueue`

D: Discus Domination

Problem Author: Florian Kothmeier

Problem

Given an integers $a_1, \dots, a_n$, maximise $a_j - a_i$, where $0 \leq j - i \leq m$ $(1 \leq n, m \leq 10^9)$

Solution

- Naive solution: Try each starting point and search for the highest value in range.
  $\Rightarrow \mathcal{O}(n \cdot m) \Rightarrow$ **too slow!**
- Idea: maximum value for a point a_j (discus landing point) is given by the smallest starting position a_i in the last m values.
  - Can be queried in $\mathcal{O}(\log m)$ using a min-heap.
    - C++: use `std::multiset` or `std::map`
    - Java: use `java.util.TreeMap`
    - Python: `from queue import PriorityQueue`
  - For $j = 1 \to n$:
    - insert a_j into the (multi) set.
    - query $a_i = min(set)$.
    - delete a_{j-m} from the set.

  $\Rightarrow \mathcal{O}(n \cdot log\ m)$

D: Discus Domination

Problem Author: Florian Kothmeier

Possible Pitfall

- Be careful of duplicate elements, e.g. use `std::multiset` instead of `std::set`.

$\Rightarrow$ If not available (e.g. in Java), use a map (e.g. `TreeMap`) and count their occurrences. Delete entries only when they appear 0-times in the map.

D: Discus Domination

Problem Author: Florian Kothmeier

Possible Pitfall

- Be careful of duplicate elements, e.g. use `std::multiset` instead of `std::set`.

⇒ If not available (e.g. in Java), use a map (e.g. `TreeMap`) and count their occurrences. Delete entries only when they appear 0-times in the map.

Alternative Solution

- Use a `Deque` for $\mathcal{O}(1)$ insertion and deletion from both ends.

D: Discus Domination

Problem Author: Florian Kothmeier

Possible Pitfall

- Be careful of duplicate elements, e.g. use `std::multiset` instead of `std::set`.

$\Rightarrow$ If not available (e.g. in Java), use a map (e.g. `TreeMap`) and count their occurrences. Delete entries only when they appear 0-times in the map.

Alternative Solution

- Use a `Deque` for $\mathcal{O}(1)$ insertion and deletion from both ends.
- Add the current value and position (a_j, j) at the end and remove the preceding entry while its value is higher. $\Rightarrow$ Smallest element will always be the first

D: Discus Domination

Problem Author: Florian Kothmeier

Possible Pitfall

- Be careful of duplicate elements, e.g. use `std::multiset` instead of `std::set`.

⇒ If not available (e.g. in Java), use a map (e.g. `TreeMap`) and count their occurrences. Delete entries only when they appear 0-times in the map.

Alternative Solution

- Use a `Deque` for $\mathcal{O}(1)$ insertion and deletion from both ends.
- Add the current value and position (a_j, j) at the end and remove the preceding entry while its value is higher. ⇒ Smallest element will always be the first
- Remove elements from the front when their position is less than $j - m$

D: Discus Domination

Problem Author: Florian Kothmeier

Possible Pitfall

- Be careful of duplicate elements, e.g. use `std::multiset` instead of `std::set`.

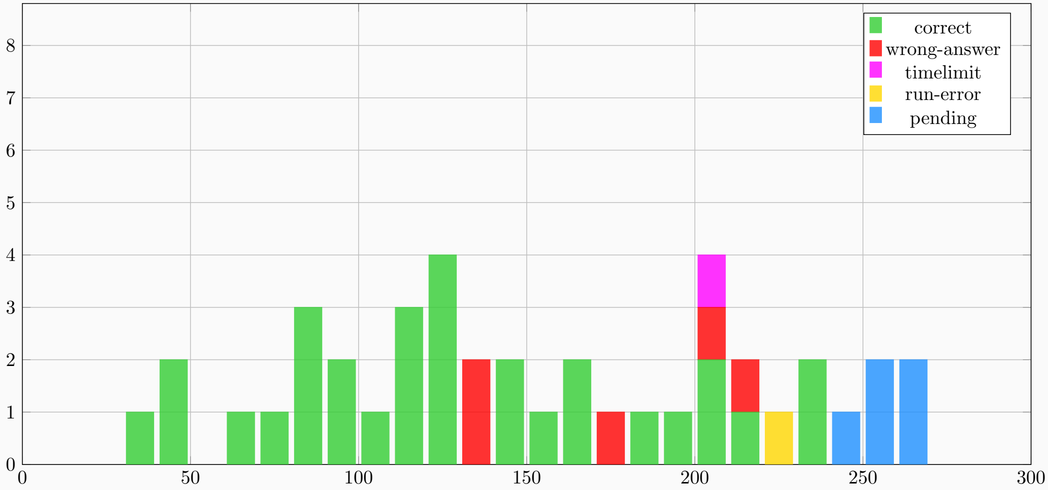
$\Rightarrow$ If not available (e.g. in Java), use a map (e.g. `TreeMap`) and count their occurrences. Delete entries only when they appear 0-times in the map.

Alternative Solution

- Use a `Deque` for $\mathcal{O}(1)$ insertion and deletion from both ends.
- Add the current value and position (a_j, j) at the end and remove the preceding entry while its value is higher. $\Rightarrow$ Smallest element will always be the first
- Remove elements from the front when their position is less than $j - m$
- Each element will be added once (and deleted once) from the list $\Rightarrow \mathcal{O}(n)$

E: Elegant Exterior

Problem Author: Marcel Wienöbst



E: Elegant Exterior

Problem Author: Marcel Wienöbst

Problem

Compute the maximum area of a Haus vom Nikolaus with total line length n.

E: Elegant Exterior

Problem Author: Marcel Wienöbst

Problem

Compute the maximum area of a Haus vom Nikolaus with total line length n.

Solution

- Ternary search over w/h to find the optimal ratio of width and height. For a fixed ratio, one can compute w and h and thus the maximum area by binary search.

E: Elegant Exterior

Problem Author: Marcel Wienöbst

Problem

Compute the maximum area of a Haus vom Nikolaus with total line length n.

Solution

- Ternary search over w/h to find the optimal ratio of width and height. For a fixed ratio, one can compute w and h and thus the maximum area by binary search.
- This is fast enough. However, there is a simpler solution.

E: Elegant Exterior

Problem Author: Marcel Wienöbst

Problem

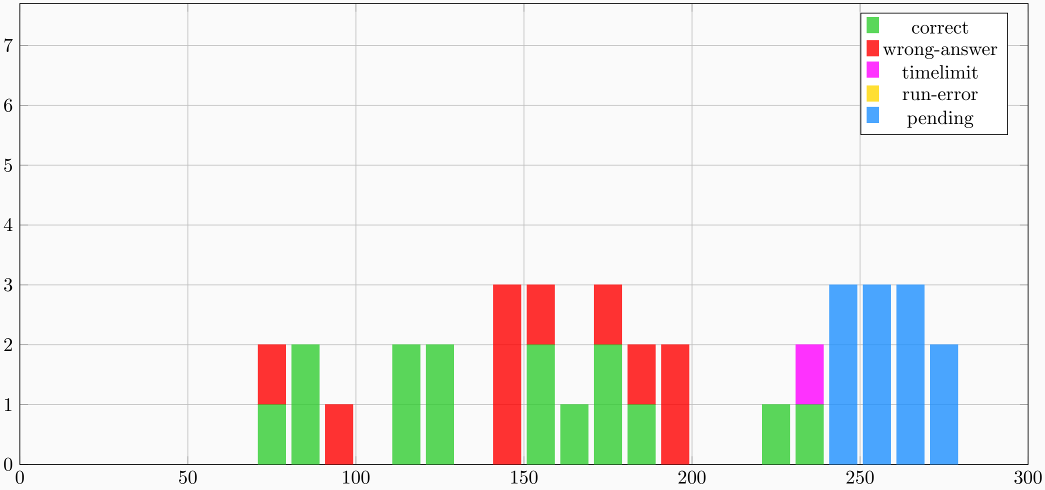
Compute the maximum area of a Haus vom Nikolaus with total line length n.

Solution

- Ternary search over w/h to find the optimal ratio of width and height. For a fixed ratio, one can compute w and h and thus the maximum area by binary search.
- This is fast enough. However, there is a simpler solution.
- There is an optimal ratio of w and h independent of n (around 1.2221, but that's not even necessary to know). Thus, the maximal area scales with n^2 and the answer is simply $0.0185303 \cdot n^2$, where 0.0185303 is the solution to Sample Input 1.

G: Gorgeous Garment

Problem Author: Wendy Yi



G: Gorgeous Garment

Problem Author: Wendy Yi

Problem

You are given

- the number of stitches of each round (which are increasing)
- and the amount and order of the colours.

How many rounds can you crochet such that each colour stripe is at least as wide as the last?

G: Gorgeous Garment

Problem Author: Wendy Yi

Problem

You are given

- the number of stitches of each round (which are increasing)
- and the amount and order of the colours.

How many rounds can you crochet such that each colour stripe is at least as wide as the last?

Solution

- If you can crochet i rounds of the pattern, you can crochet fewer rounds as well.
- Use binary search to determine the maximum number of rounds.

G: Gorgeous Garment

Problem Author: Wendy Yi

Problem

You are given

- the number of stitches of each round (which are increasing)
- and the amount and order of the colours.

How many rounds can you crochet such that each colour stripe is at least as wide as the last?

Solution

- If you can crochet i rounds of the pattern, you can crochet fewer rounds as well.
- Use binary search to determine the maximum number of rounds.
- Test if it is possible to crochet i rounds:
  - Colour stripes are wider towards the outer edge
    $\Longrightarrow$ use outermost colour for as many rounds as possible.

G: Gorgeous Garment

Problem Author: Wendy Yi

Problem

You are given

- the number of stitches of each round (which are increasing)
- and the amount and order of the colours.

How many rounds can you crochet such that each colour stripe is at least as wide as the last?

Solution

- If you can crochet i rounds of the pattern, you can crochet fewer rounds as well.
- Use binary search to determine the maximum number of rounds.
- Test if it is possible to crochet i rounds:
  - Colour stripes are wider towards the outer edge
    $\implies$ use outermost colour for as many rounds as possible.
  - Working from the outermost to the innermost round, greedily crochet as many rounds as possible with each colour.

G: Gorgeous Garment

Problem Author: Wendy Yi

Problem

You are given

- the number of stitches of each round (which are increasing)
- and the amount and order of the colours.

How many rounds can you crochet such that each colour stripe is at least as wide as the last?

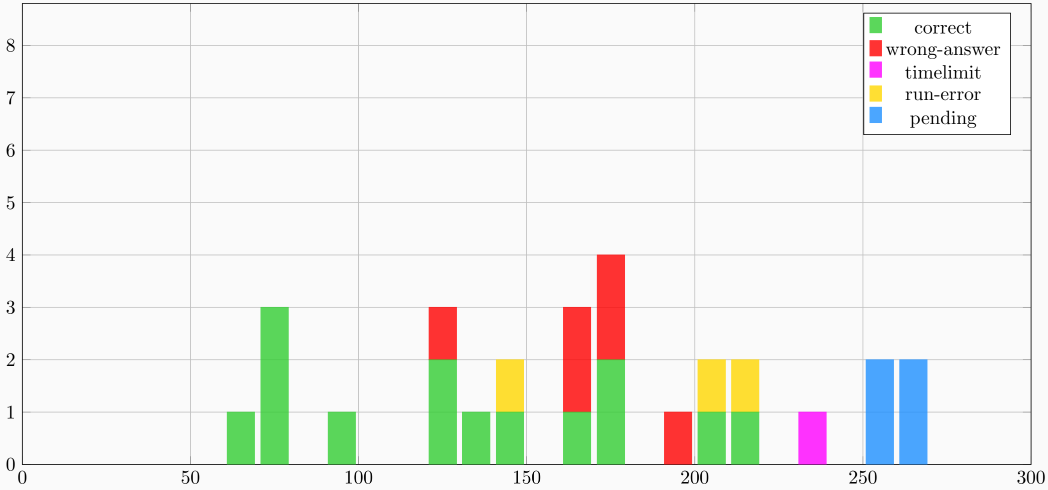
Solution

- If you can crochet i rounds of the pattern, you can crochet fewer rounds as well.
- Use binary search to determine the maximum number of rounds.
- Test if it is possible to crochet i rounds:
  - Colour stripes are wider towards the outer edge
    $\Longrightarrow$ use outermost colour for as many rounds as possible.
  - Working from the outermost to the innermost round, greedily crochet as many rounds as possible with each colour.

Running time: $\mathcal{O}(n \log(n))$

L: Legendary Lanparty

Problem Author: Michael Zündorf



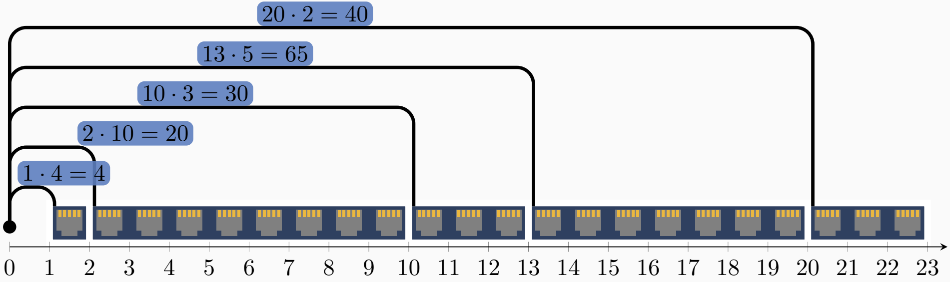
L: Legendary Lanparty

Problem Author: Michael Zündorf

Problem

Given n tuples (c_i, m_i), reorder them such that the following sum is minimized

$$\sum_{i=1}^{n} c_i \cdot \sum_{j=1}^{i-1} m_j \ .$$



L: Legendary Lanparty

Problem Author: Michael Zündorf

Solution

- Observe that swapping two adjacent tuples changes the cost by

$$\delta = c_i \cdot m_{i+1} - c_{i+1} \cdot m_i$$

$\Rightarrow$ δ must be positive for all adjacent tuples

L: Legendary Lanparty

Problem Author: Michael Zündorf

Solution

- Observe that swapping two adjacent tuples changes the cost by

$$\delta = c_i \cdot m_{i+1} - c_{i+1} \cdot m_i$$

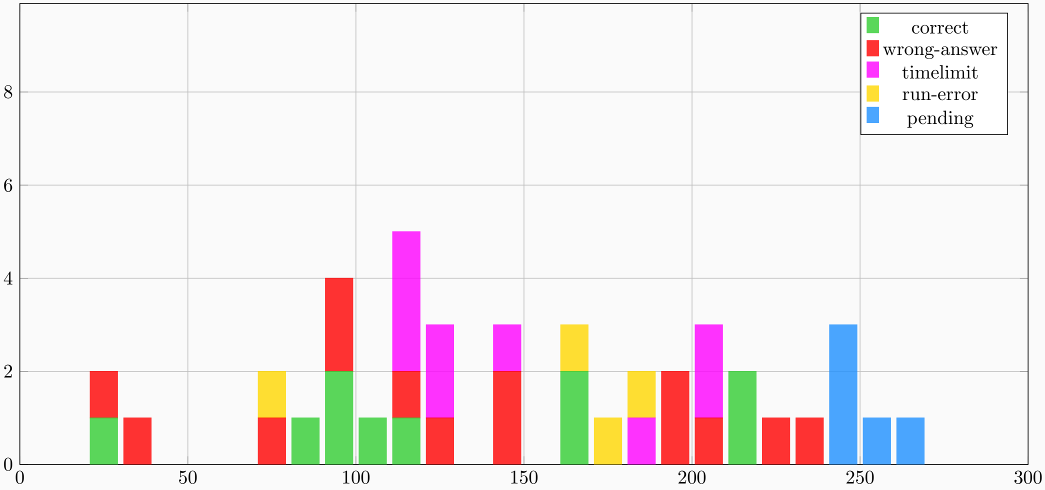
$\Rightarrow$ δ must be positive for all adjacent tuples

- This already implies a total order

$\Rightarrow$ We can sort by δ, and compare non adjacent elements with it

A: Alien Attack

Problem Author: Christopher Weyand



A: Alien Attack

Problem Author: Christopher Weyand

Problem

Given an undirected, connected graph. Each time step the following happens:

- the vertex of highest degree (id as tiebreaker) is deleted. Vertex 1 is never deleted.
- any vertex that cannot reach vertex 1 is deleted.

How many steps until only vertex 1 remains?

A: Alien Attack

Problem Author: Christopher Weyand

Problem

Given an undirected, connected graph. Each time step the following happens:

- the vertex of highest degree (id as tiebreaker) is deleted. Vertex 1 is never deleted.
- any vertex that cannot reach vertex 1 is deleted.

How many steps until only vertex 1 remains?

Solution

- consider an alternative process that deletes the highest degree node each step (nothing else)

A: Alien Attack

Problem Author: Christopher Weyand

Problem

Given an undirected, connected graph. Each time step the following happens:

- the vertex of highest degree (id as tiebreaker) is deleted. Vertex 1 is never deleted.
- any vertex that cannot reach vertex 1 is deleted.

How many steps until only vertex 1 remains?

Solution

- consider an alternative process that deletes the highest degree node each step (nothing else)
- deletions in the connected component (CC) of vertex 1 remain as in the original process

A: Alien Attack

Problem Author: Christopher Weyand

Problem

Given an undirected, connected graph. Each time step the following happens:

- the vertex of highest degree (id as tiebreaker) is deleted. Vertex 1 is never deleted.
- any vertex that cannot reach vertex 1 is deleted.

How many steps until only vertex 1 remains?

Solution

- consider an alternative process that deletes the highest degree node each step (nothing else)
- deletions in the connected component (CC) of vertex 1 remain as in the original process
- and deletions outside this CC are irrelevant to the original process anyway

A: Alien Attack

Problem Author: Christopher Weyand

Problem

Given an undirected, connected graph. Each time step the following happens:

- the vertex of highest degree (id as tiebreaker) is deleted. Vertex 1 is never deleted.
- any vertex that cannot reach vertex 1 is deleted.

How many steps until only vertex 1 remains?

Solution

- consider an alternative process that deletes the highest degree node each step (nothing else)
- deletions in the connected component (CC) of vertex 1 remain as in the original process
- and deletions outside this CC are irrelevant to the original process anyway
- the answer is thus the number of nodes that can reach vertex 1 at the time of their deletion

A: Alien Attack

Problem Author: Christopher Weyand

Problem

Given an undirected, connected graph. Each time step the following happens:

- the vertex of highest degree (id as tiebreaker) is deleted. Vertex 1 is never deleted.
- any vertex that cannot reach vertex 1 is deleted.

How many steps until only vertex 1 remains?

Solution

- consider an alternative process that deletes the highest degree node each step (nothing else)
- deletions in the connected component (CC) of vertex 1 remain as in the original process
- and deletions outside this CC are irrelevant to the original process anyway
- the answer is thus the number of nodes that can reach vertex 1 at the time of their deletion
- deletion order can be computed by maintaining degrees with a priority queue in $O(m \log n)$

A: Alien Attack

Problem Author: Christopher Weyand

Problem

Given an undirected, connected graph. Each time step the following happens:

- the vertex of highest degree (id as tiebreaker) is deleted. Vertex 1 is never deleted.
- any vertex that cannot reach vertex 1 is deleted.

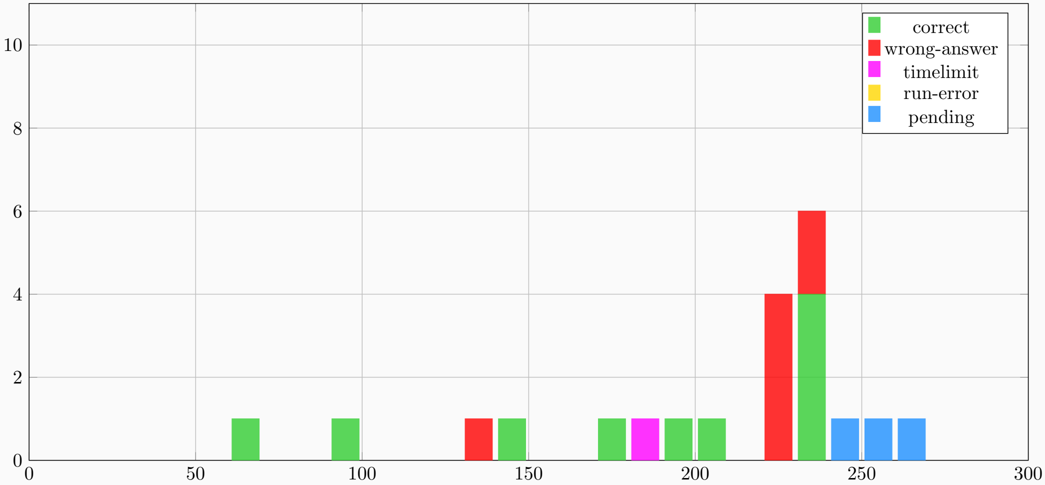
How many steps until only vertex 1 remains?

Solution

- consider an alternative process that deletes the highest degree node each step (nothing else)
- deletions in the connected component (CC) of vertex 1 remain as in the original process
- and deletions outside this CC are irrelevant to the original process anyway
- the answer is thus the number of nodes that can reach vertex 1 at the time of their deletion
- deletion order can be computed by maintaining degrees with a priority queue in $O(m \log n)$
- by simulating the process in reverse (deletions become insertions) reachability checks can be done with a union-find data structure

M: Massive Mountains

Problem Author: The Winter Contest Jury, Julian Baldus



M: Massive Mountains

Problem Author: The Winter Contest Jury, Julian Baldus

Problem

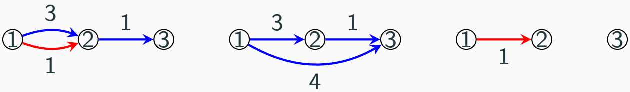
Given a weighted, directed graph with red and blue edges. A and B want to get from vertex 1 to vertex n. They are not allowed to use an edge of the same colour at the same time. How long does it take them at least to get to vertex n.

M: Massive Mountains

Problem Author: The Winter Contest Jury, Julian Baldus

Problem

Given a weighted, directed graph with red and blue edges. A and B want to get from vertex 1 to vertex n. They are not allowed to use an edge of the same colour at the same time. How long does it take them at least to get to vertex n.

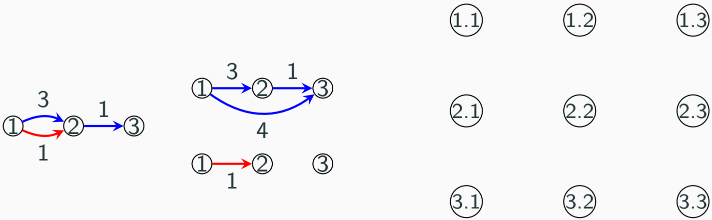


Solution

- Assume A starts with a red edge and B with a blue one.
- When A and B swap colours they both have to be on a vertex.
- Between swapping colours A and B walk through the subgraph with red/blue edges.
- We may assume that they use only shortest paths. (They are allowed to wait.)
- Step 1: Compute all shortest paths in the subgraph with red/blue edges. (Floyd Warshall)

M: Massive Mountains

Problem Author: The Winter Contest Jury, Julian Baldus

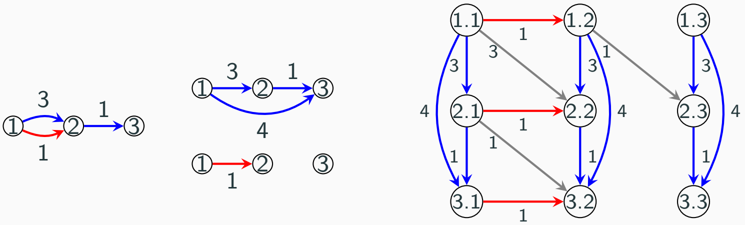


Solution (continued)

- Step 2: Consider the product graph where every vertex is a tuple (b,r) corresponding to the position in the orginal graph of the person using red/blue edges.

M: Massive Mountains

Problem Author: The Winter Contest Jury, Julian Baldus

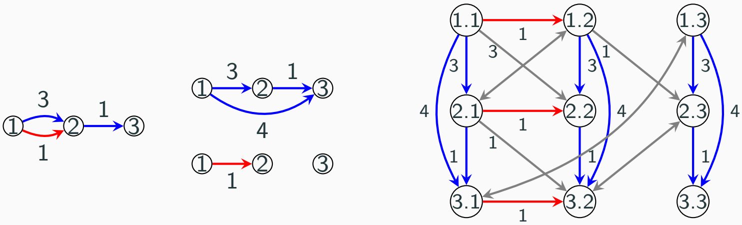


Solution (continued)

- Step 2: Consider the product graph where every vertex is a tuple (b,r) corresponding to the position in the orginal graph of the person using red/blue edges.
- Add edges:
  - If there are paths (r, r') and (b, b') in G, add an arc $((r, b), (r', b'))$ with cost $\max(cost(r, r'), cost(b, b'))$.

M: Massive Mountains

Problem Author: The Winter Contest Jury, Julian Baldus

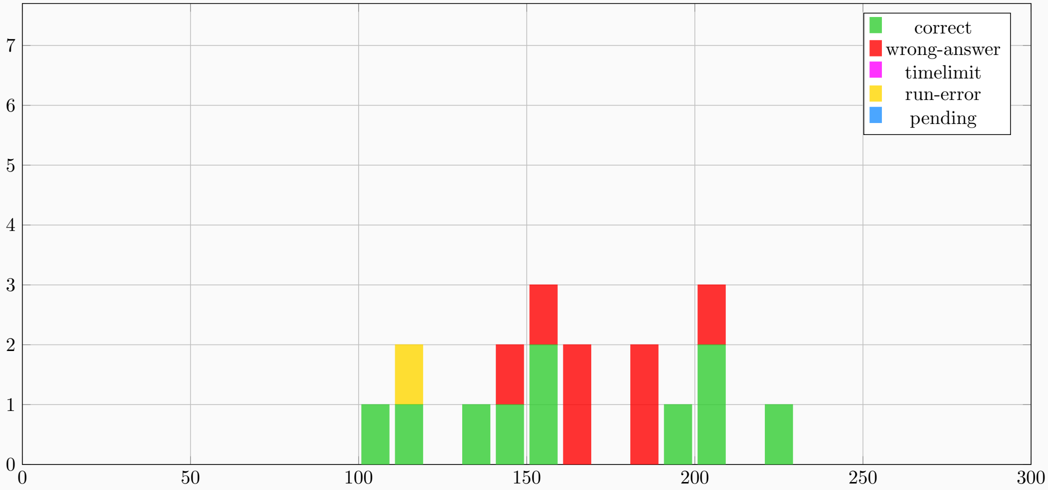


Solution (continued)

- Step 2: Consider the product graph where every vertex is a tuple (b,r) corresponding to the position in the orginal graph of the person using red/blue edges.
- Add edges:
  - If there are paths (r, r') and (b, b') in G, add an arc $((r, b), (r', b'))$ with cost $\max(cost(r, r'), cost(b, b'))$.
  - A and B may swap colours. Add a bidirectional arc $((r, b), (b, r))$ with cost 0.
- Find a shortest path from $(1, 1)$ to (n, n) in G' (e.g. with Dijkstra).

K: K.O. Kids II

Problem Author: Marcel Wienöbst



K: K.O. Kids II

Problem Author: Marcel Wienöbst

Problem

Given probabilities $a_1, \dots, a_k$ of overcoming an unbeaten obstacle (already beaten obstacles are overcome every time) and a queue of n participants, calculate the maximum probability to be the first finisher.

K: K.O. Kids II

Problem Author: Marcel Wienöbst

Problem

Given probabilities $a_1, \ldots, a_k$ of overcoming an unbeaten obstacle (already beaten obstacles are overcome every time) and a queue of n participants, calculate the maximum probability to be the first finisher.

Solution

- Main Idea: For participant i, compute the probability of making it up to obstacle j and failing there:

$$P(i,j) = \sum_{k \leq j} P(i-1,k) \cdot \prod_{k \leq l < j} a_l \cdot (1 - a_j).$$

  In words, multiply the probability for each possible position of the previous participant by the probability to make it from there exactly to obstacle j and not further.

K: K.O. Kids II

Problem Author: Marcel Wienöbst

Problem

Given probabilities $a_1, \dots, a_k$ of overcoming an unbeaten obstacle (already beaten obstacles are overcome every time) and a queue of n participants, calculate the maximum probability to be the first finisher.

Solution

- Main Idea: For participant i, compute the probability of making it up to obstacle j and failing there:

$$P(i,j) = \sum_{k \leq j} P(i-1,k) \cdot \prod_{k \leq l < j} a_l \cdot (1 - a_j).$$

  In words, multiply the probability for each possible position of the previous participant by the probability to make it from there exactly to obstacle j and not further.
- Evaluating this naively takes time $O(nk^2)$, which is too slow.

K: K.O. Kids II

Problem Author: Marcel Wienöbst

Problem

Given probabilities $a_1, \dots, a_k$ of overcoming an unbeaten obstacle (already beaten obstacles are overcome every time) and a queue of n participants, calculate the maximum probability to be the first finisher.

Solution

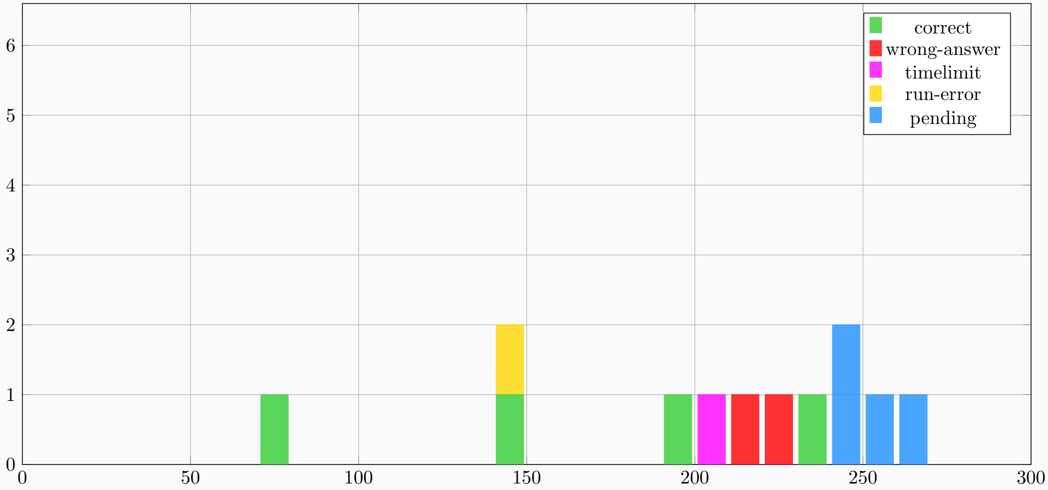
- Main Idea: For participant i, compute the probability of making it up to obstacle j and failing there:

  $$P(i,j) = \sum_{k \leq j} P(i-1,k) \cdot \prod_{k \leq l < j} a_l \cdot (1 - a_j).$$

  In words, multiply the probability for each possible position of the previous participant by the probability to make it from there exactly to obstacle j and not further.
- Evaluating this naively takes time $O(nk^2)$, which is too slow.
- Instead, dynamically build up the term $T(i,j) = \sum_{k \leq j} P(i-1,k) \cdot \prod_{k \leq l < j} a_l$. It holds that $T(i,j) = (T(i,j-1) + P(i-1,j)) \cdot a_j$ and clearly $P(i,j) = T(i,j-1) \cdot (1 - a_j)$.
- This can be implemented in $O(nk)$ time.

H: Hungry Hunting

Problem Author: The Winter Contest Jury, Julian Baldus



H: Hungry Hunting

Problem Author: The Winter Contest Jury, Julian Baldus

Problem

Given n item types with values $c_1, \ldots, c_n$. If we double c_i, how many items do we have to take to obtain a value of exactly w? Print the answer for every i.

H: Hungry Hunting

Problem Author: The Winter Contest Jury, Julian Baldus

Problem

Given n item types with values $c_1, \dots, c_n$. If we double c_i, how many items do we have to take to obtain a value of exactly w? Print the answer for every i.

Solution

- Without doubling this problem is the classic coin change problem: Let $dp_\ell(k, j)$ be the minimum number of items that have total value j, given that only types $1, \dots, k$ are allowed
- For each i double the value c_i and do the classic coin change DP

H: Hungry Hunting

Problem Author: The Winter Contest Jury, Julian Baldus

Problem

Given n item types with values $c_1, \dots, c_n$. If we double c_i, how many items do we have to take to obtain a value of exactly w? Print the answer for every i.

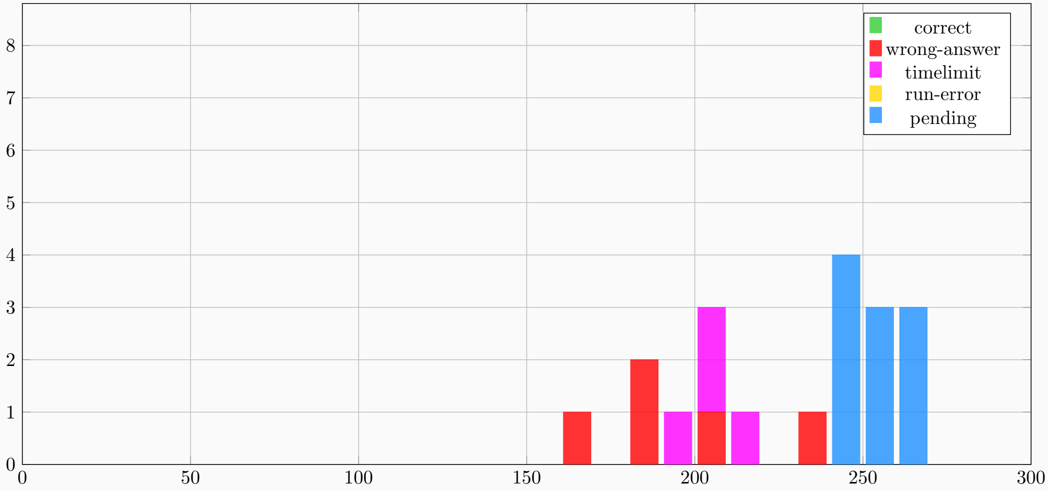
Solution

- Without doubling this problem is the classic coin change problem: Let $dp_\ell(k, j)$ be the minimum number of items that have total value j, given that only types $1, \dots, k$ are allowed
- For each i double the value c_i and do the classic coin change DP $\Rightarrow \mathcal{O}(n^2 \cdot w) \Rightarrow$ **Too slow!**
- Insight: $dp_\ell(k, j)$ for $k < i$ is independent of whether c_i is doubled or not.
- The same property holds when the DP works from the other direction: Only types $k, \dots, n$ are allowed for $dp_r(k, j)$; c_i is irrelevant for $k > i$.
- For each i, compute $double(i, j) = \min\{double(i, j - 2c_i), dp_\ell(i - 1, j)\}$:
  "number of items with total value j, given that only types $1, \dots, i$ are allowed and c_i is doubled"
- For each i, find $\min_j\{double(i, j) + dp_r(i + 1, w - j)\}$.

Total time complexity: $\mathcal{O}(n \cdot w)$

B: Broken Borders

Problem Author: Jannik Olbrich

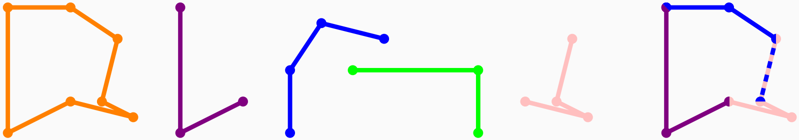


B: Broken Borders

Problem Author: Jannik Olbrich

Problem

Given a simple polygon and many polylines. Can the polylines can be aligned to the polygon such that every line segment of the polygon is covered? Polylines can be used arbitrarily often.

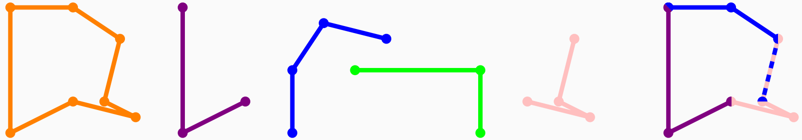


B: Broken Borders

Problem Author: Jannik Olbrich

Problem

Given a simple polygon and many polylines. Can the polylines can be aligned to the polygon such that every line segment of the polygon is covered? Polylines can be used arbitrarily often.



Solution

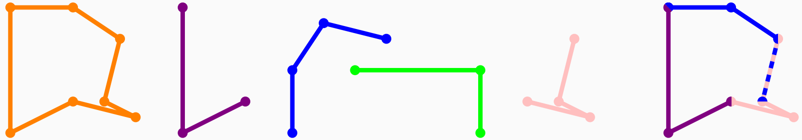
- A polyline can be aligned to a part of the polygon iff
  - the ith segment of the polyline and the ith segment of the polygon part have equal length
  - the ith angle of the polyline and the ith angle of the polygon part are equal

B: Broken Borders

Problem Author: Jannik Olbrich

Problem

Given a simple polygon and many polylines. Can the polylines can be aligned to the polygon such that every line segment of the polygon is covered? Polylines can be used arbitrarily often.



Solution

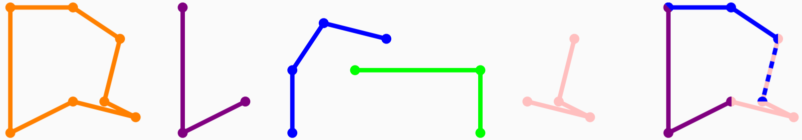
- A polyline can be aligned to a part of the polygon iff
  - the ith segment of the polyline and the ith segment of the polygon part have equal length
  - the ith angle of the polyline and the ith angle of the polygon part are equal
- Transform the polygon and polylines into strings of integers by enumerating all segment lengths and angles

B: Broken Borders

Problem Author: Jannik Olbrich

Problem

Given a simple polygon and many polylines. Can the polylines can be aligned to the polygon such that every line segment of the polygon is covered? Polylines can be used arbitrarily often.



Solution

- A polyline can be aligned to a part of the polygon iff
  - the ith segment of the polyline and the ith segment of the polygon part have equal length
  - the ith angle of the polyline and the ith angle of the polygon part are equal
- Transform the polygon and polylines into strings of integers by enumerating all segment lengths and angles
- We now have the following problem: Given a (circular) string and a set of patterns. Is every length-id (i.e. line segment) in the string covered by some match of a pattern.

Solution (cont.)

- We now have the following problem: Given a (circular) string and a set of patterns. Is every length-id (i.e. line segment) in the string covered by some match of a pattern.
- Find every match of every pattern using your favourite string matching algorithm

B: Broken Borders

Problem Author: Jannik Olbrich

Solution (cont.)

- We now have the following problem: Given a (circular) string and a set of patterns. Is every length-id (i.e. line segment) in the string covered by some match of a pattern.
- Find every match of every pattern using your favourite string matching algorithm
  too slow! every pattern can have n matches

B: Broken Borders

Problem Author: Jannik Olbrich

Solution (cont.)

- We now have the following problem: Given a (circular) string and a set of patterns. Is every length-id (i.e. line segment) in the string covered by some match of a pattern.
- Find every match of every pattern using your favourite string matching algorithm
  too slow! every pattern can have n matches
- Find matches using the suffix array: All matches of a pattern form an interval in the suffix array

Solution (cont.)

- We now have the following problem: Given a (circular) string and a set of patterns. Is every length-id (i.e. line segment) in the string covered by some match of a pattern.
- Find every match of every pattern using your favourite string matching algorithm
  too slow! every pattern can have n matches
- Find matches using the suffix array: All matches of a pattern form an interval in the suffix array
- For each position in the suffix array determine the length of the longest pattern whose interval contains this position

B: Broken Borders

Problem Author: Jannik Olbrich

Solution (cont.)

- We now have the following problem: Given a (circular) string and a set of patterns. Is every length-id (i.e. line segment) in the string covered by some match of a pattern.
- Find every match of every pattern using your favourite string matching algorithm
  too slow! every pattern can have n matches
- Find matches using the suffix array: All matches of a pattern form an interval in the suffix array
- For each position in the suffix array determine the length of the longest pattern whose interval contains this position
- Finally, use a sweep-line algorithm to mark all covered positions in the string

B: Broken Borders

Problem Author: Jannik Olbrich

Solution (cont.)

- We now have the following problem: Given a (circular) string and a set of patterns. Is every length-id (i.e. line segment) in the string covered by some match of a pattern.
- Find every match of every pattern using your favourite string matching algorithm
  too slow! every pattern can have n matches
- Find matches using the suffix array: All matches of a pattern form an interval in the suffix array
- For each position in the suffix array determine the length of the longest pattern whose interval contains this position
- Finally, use a sweep-line algorithm to mark all covered positions in the string

Alternative solution: Use Aho-Corasick (lazy or with a persistent array data structure) to find the longest match at every position of the string, then proceed with the sweep-line as above

B: Broken Borders

Problem Author: Jannik Olbrich

Solution (cont.)

- We now have the following problem: Given a (circular) string and a set of patterns. Is every length-id (i.e. line segment) in the string covered by some match of a pattern.
- Find every match of every pattern using your favourite string matching algorithm
  too slow! every pattern can have n matches
- Find matches using the suffix array: All matches of a pattern form an interval in the suffix array
- For each position in the suffix array determine the length of the longest pattern whose interval contains this position
- Finally, use a sweep-line algorithm to mark all covered positions in the string

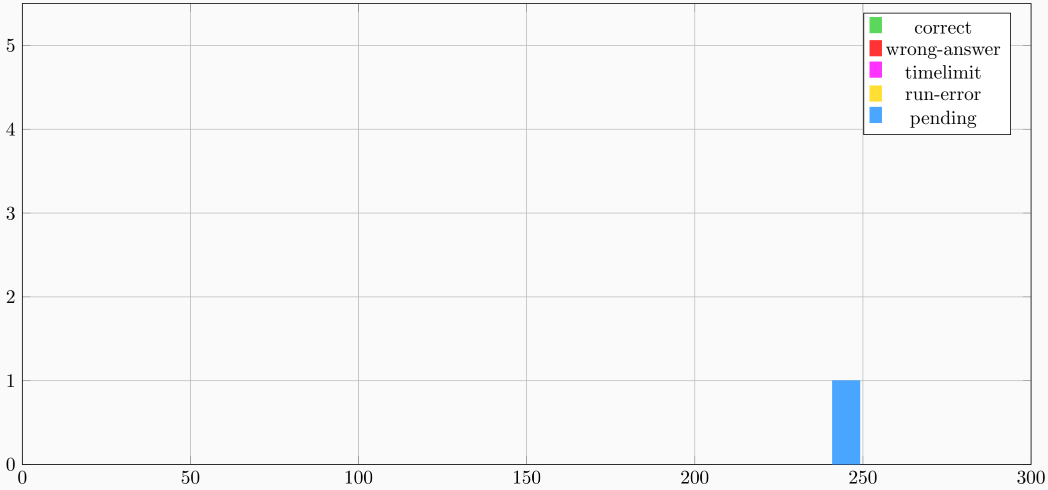
Alternative solution: Use Aho-Corasick (lazy or with a persistent array data structure) to find the longest match at every position of the string, then proceed with the sweep-line as above

Possible pitfalls

- You need integer-save angle comparison, (`long`) `double` is not precise enough
- $\mathcal{O}(n \log^2 n)$ suffix array construction may be too slow
- $\mathcal{O}(n^{1.5})$ hashing solutions can be too slow

F: Fragmented Floor

Problem Author: Jannik Olbrich

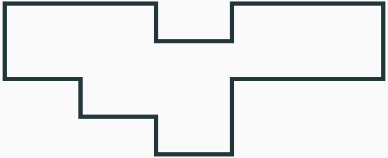


F: Fragmented Floor

Problem Author: Jannik Olbrich

Problem

Given a simple axis-aligned polygon. Find the minimum number of rectangles that cover it exactly.

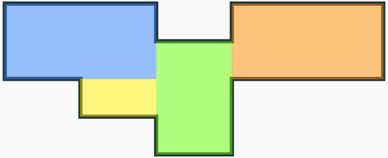


F: Fragmented Floor

Problem Author: Jannik Olbrich

Problem

Given a simple axis-aligned polygon. Find the minimum number of rectangles that cover it exactly.

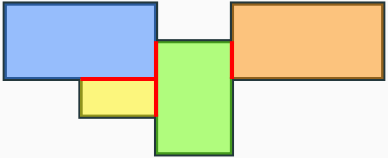


F: Fragmented Floor

Problem Author: Jannik Olbrich

Problem

Given a simple axis-aligned polygon. Find the minimum number of rectangles that cover it exactly.



Solution

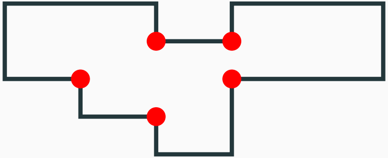
- Any solution (i.e. list of rectangles) is characterised by the axis-parallel diagonals that split the polygon into the rectangles. Call those *dissection edges*
- Number of rectangles is $1 + \#$dissection edges

F: Fragmented Floor

Problem Author: Jannik Olbrich

Problem

Given a simple axis-aligned polygon. Find the minimum number of rectangles that cover it exactly.



Solution

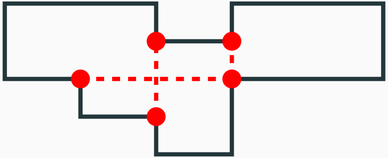
- Any solution (i.e. list of rectangles) is characterised by the axis-parallel diagonals that split the polygon into the rectangles. Call those *dissection edges*
- Number of rectangles is $1 + \#$dissection edges
- Each concave corner of the polygon must be met by (at least) one dissection edge

F: Fragmented Floor

Problem Author: Jannik Olbrich

Problem

Given a simple axis-aligned polygon. Find the minimum number of rectangles that cover it exactly.

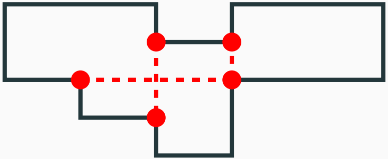


Solution

- Any solution (i.e. list of rectangles) is characterised by the axis-parallel diagonals that split the polygon into the rectangles. Call those *dissection edges*
- Number of rectangles is $1 + \#\text{dissection edges}$
- Each concave corner of the polygon must be met by (at least) one dissection edge
- A dissection edge meets at most two concave corners. Call edges incident to two corners *critical*
- Number of rectangles is $1 + \#\text{concave corners} - \#\text{used non-intersecting critical edges}$

F: Fragmented Floor

Problem Author: Jannik Olbrich

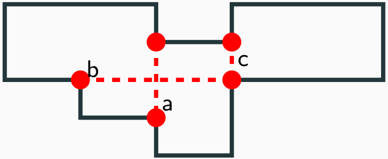


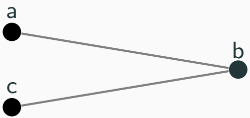
Solution (cont.)

- Number of rectangles is $1 + \#$concave corners $- \#$used non-intersecting critical edges
  $\Rightarrow$ Maximise number of non-intersecting critical edges

F: Fragmented Floor

Problem Author: Jannik Olbrich



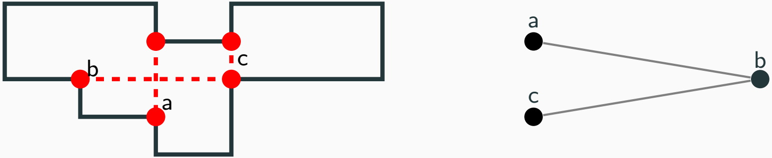


Solution (cont.)

- Number of rectangles is $1 + \#$concave corners $- \#$used non-intersecting critical edges
  $\Rightarrow$ Maximise number of non-intersecting critical edges
- Build graph: nodes are critical edges; connect nodes iff edges intersect
  $\Rightarrow$ Max. set of critical edges $\widehat{=}$ max. independent set

F: Fragmented Floor

Problem Author: Jannik Olbrich

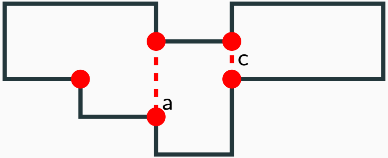


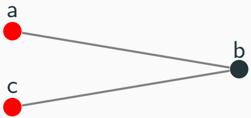
Solution (cont.)

- Number of rectangles is $1 + \#$concave corners $-$ $\#$used non-intersecting critical edges
  $\Rightarrow$ Maximise number of non-intersecting critical edges
- Build graph: nodes are critical edges; connect nodes iff edges intersect
  $\Rightarrow$ Max. set of critical edges $\widehat{=}$ max. independent set
- Only horizontal and vertical edges intersect $\Rightarrow$ Graph is bipartite
  $\Rightarrow$ solve using bipartite matching (Kőnig's theorem)

F: Fragmented Floor

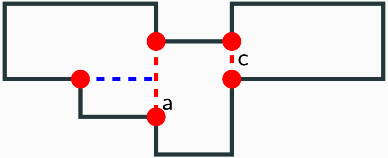
Problem Author: Jannik Olbrich

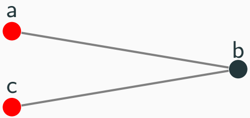




Solution (cont.)

- Number of rectangles is $1 + \#\text{concave corners} - \#\text{used non-intersecting critical edges}$
  $\Rightarrow$ Maximise number of non-intersecting critical edges
- Build graph: nodes are critical edges; connect nodes iff edges intersect
  $\Rightarrow$ Max. set of critical edges $\widehat{=}$ max. independent set
- Only horizontal and vertical edges intersect $\Rightarrow$ Graph is bipartite
  $\Rightarrow$ solve using bipartite matching (Kőnig's theorem)



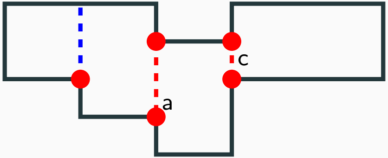


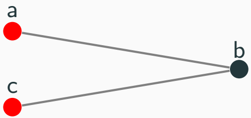
Solution (cont.)

- Number of rectangles is $1 + \#\text{concave corners} - \#\text{used non-intersecting critical edges}$
  $\Rightarrow$ Maximise number of non-intersecting critical edges
- Build graph: nodes are critical edges; connect nodes iff edges intersect
  $\Rightarrow$ Max. set of critical edges $\widehat{=}$ max. independent set
- Only horizontal and vertical edges intersect $\Rightarrow$ Graph is bipartite
  $\Rightarrow$ solve using bipartite matching (Kőnig's theorem)
- (To complete the dissection, draw a chord from each remaining convex vertex; the direction does not matter.)

F: Fragmented Floor

Problem Author: Jannik Olbrich





Solution (cont.)

- Number of rectangles is $1 + \#\text{concave corners} - \#\text{used non-intersecting critical edges}$
  $\Rightarrow$ Maximise number of non-intersecting critical edges
- Build graph: nodes are critical edges; connect nodes iff edges intersect
  $\Rightarrow$ Max. set of critical edges $\widehat{=}$ max. independent set
- Only horizontal and vertical edges intersect $\Rightarrow$ Graph is bipartite
  $\Rightarrow$ solve using bipartite matching (Kőnig's theorem)
- (To complete the dissection, draw a chord from each remaining convex vertex; the direction does not matter.)

Random facts

Jury work

- 263 commits

Random facts

Jury work

- 263 commits
- 369 secret test cases ($\approx$ 28 per problem)

Random facts

Jury work

- 263 commits
- 369 secret test cases ($\approx$ 28 per problem)
- 110 jury solutions

Random facts

Jury work

- 263 commits
- 369 secret test cases ($\approx$ 28 per problem)
- 110 jury solutions
- The minimum number of lines the jury needed to solve all problems is

$$39 + 67 + 4 + 8 + 1 + 52 + 28 + 23 + 4 + 17 + 16 + 14 + 25 = 298$$

  On average 23 lines per problem