GCPC 2024 Presentation of Solutions

The GCPC Jury

June 22, 2024

GCPC 2024 Jury

- **Sebastian Angrick**
  Hasso-Plattner-Institute Potsdam
- **Niklas Bauer**
  Georg August University of Göttingen
- **Brutenis Gliwa**
  University of Rostock, CPUlm
- **Andreas Grigorjew**
  University of Helsinki FI
- **Yvonne Kothmeier**
  Friedrich-Alexander University
  Erlangen-Nürnberg
- **Felicia Lucke**
  Fribourg University CH, CPUlm
- **Jannik Olbrich**
  Ulm University, CPUlm
- **Erik Sünderhauf**
  Technical University of Munich
- **Christopher Weyand**
  MOIA GmbH, CPUlm
- **Paul Wild**
  Friedrich-Alexander University
  Erlangen-Nürnberg, CPUlm
- **Wendy Yi**
  Karlsruhe Institute of Technology,CPUlm
- **Michael Zündorf**
  Karlsruhe Institute of Technology, CPUlm
- **Marian Zuska**
  University of Rostock

GCPC 2024 Test Solvers

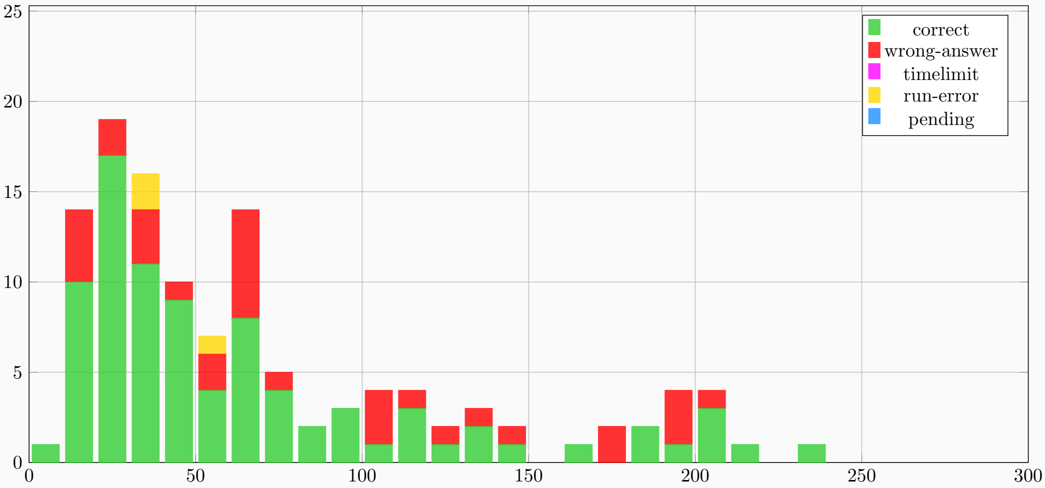
- **Khaled Ismaeel**
  freiheit.com technologies GmbH, Hamburg
- **Michael Ruderer**
  Augsburg University, CPUlm
- **Jonas Schmidt**
  Hasso-Plattner-Institute Potsdam
- **Marcel Wienöbst**
  University of Lübeck, CPUlm

GCPC 2024 Technical Team

- **Nathan Maier**
  CPUlm
- **Alexander Schmid**
  CPUlm
- **Pascal Weber**
  University of Vienna, CPUlm

B: Bookshelf Bottleneck

Problem author: Jannik Olbrich



B: Bookshelf Bottleneck

Problem author: Jannik Olbrich

Problem

Store books of size $l \times w \times h$ into a shelf of height H while minimizing the shelf width used.

B: Bookshelf Bottleneck

Problem author: Jannik Olbrich

Problem

Store books of size $l \times w \times h$ into a shelf of height H while minimizing the shelf width used.

Solution

- Put the smallest side length in shelf direction and the second smallest upwards

B: Bookshelf Bottleneck

Problem author: Jannik Olbrich

Problem

Store books of size $l \times w \times h$ into a shelf of height H while minimizing the shelf width used.

Solution

- Put the smallest side length in shelf direction and the second smallest upwards
- If this does not fit with the height H, swap the dimensions

B: Bookshelf Bottleneck

Problem author: Jannik Olbrich

Problem

Store books of size $l \times w \times h$ into a shelf of height H while minimizing the shelf width used.

Solution

- Put the smallest side length in shelf direction and the second smallest upwards
- If this does not fit with the height H, swap the dimensions
- If it still does not fit, the task is impossible

B: Bookshelf Bottleneck

Problem author: Jannik Olbrich

Problem

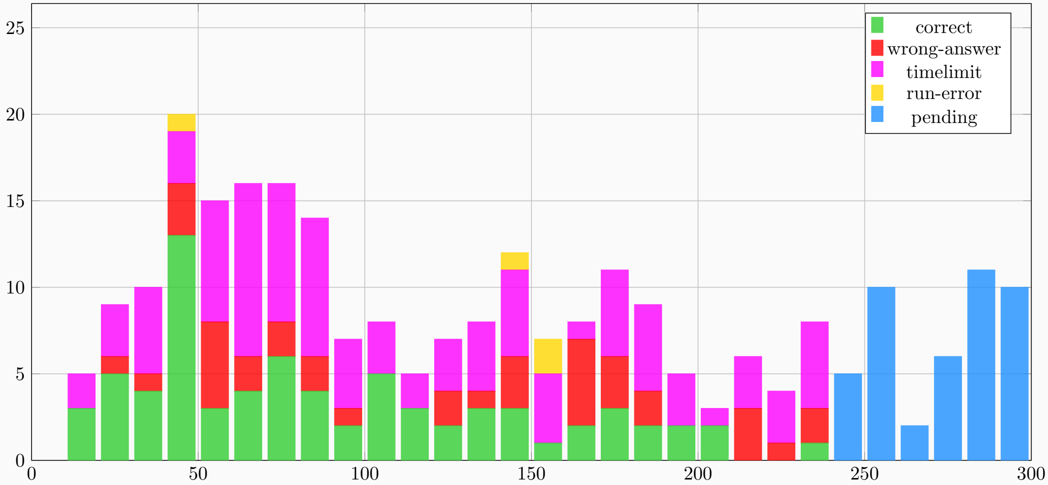
Store books of size $l \times w \times h$ into a shelf of height H while minimizing the shelf width used.

Solution

- Put the smallest side length in shelf direction and the second smallest upwards
- If this does not fit with the height H, swap the dimensions
- If it still does not fit, the task is impossible
- Otherwise, do this for every book. The sum of the lengths is the solution

K: Kitten of Chaos

Problem author: Paul Wild



K: Kitten of Chaos

Problem author: Paul Wild

Problem

Apply a bunch of rotations and reflections to a string consisting of `bdpq`:

- `h`: horizontal flip: `bbq` $\leftrightarrow$ `pdd`
- `v`: vertical flip: `bbq` $\leftrightarrow$ `ppd`
- `r`: 180 degree rotation: `bbq` $\leftrightarrow$ `bqq`

K: Kitten of Chaos

Problem author: Paul Wild

Problem

Apply a bunch of rotations and reflections to a string consisting of `bdpq`:

- `h`: horizontal flip: `bbq` $\leftrightarrow$ `pdd`
- `v`: vertical flip: `bbq` $\leftrightarrow$ `ppd`
- `r`: 180 degree rotation: `bbq` $\leftrightarrow$ `bqq`

Solution

- Applying all the transformations one by one takes $\Theta(n^2)$ time, too slow!

K: Kitten of Chaos

Problem author: Paul Wild

Problem

Apply a bunch of rotations and reflections to a string consisting of `bdpq`:

- `h`: horizontal flip: `bbq` $\leftrightarrow$ `pdd`
- `v`: vertical flip: `bbq` $\leftrightarrow$ `ppd`
- `r`: 180 degree rotation: `bbq` $\leftrightarrow$ `bqq`

Solution

- Applying all the transformations one by one takes $\Theta(n^2)$ time, too slow!
- Instead, we make some observations:
  - we may replace each `r` by `hv`
  - doing `vh` is the same as `hv` $\rightsquigarrow$ move all `h` to the front
  - we only need to know if the number of `h` is even or odd (same for `v`)

K: Kitten of Chaos

Problem author: Paul Wild

Problem

Apply a bunch of rotations and reflections to a string consisting of `bdpq`:

- `h`: horizontal flip: `bbq` $\leftrightarrow$ `pdd`
- `v`: vertical flip: `bbq` $\leftrightarrow$ `ppd`
- `r`: 180 degree rotation: `bbq` $\leftrightarrow$ `bqq`

Solution

- Applying all the transformations one by one takes $\Theta(n^2)$ time, too slow!
- Instead, we make some observations:
  - we may replace each `r` by `hv`
  - doing `vh` is the same as `hv` $\rightsquigarrow$ move all `h` to the front
  - we only need to know if the number of `h` is even or odd (same for `v`)
- Using these, we only need to do at most one `h` and at most one `v` transformation.

K: Kitten of Chaos

Problem author: Paul Wild

Problem

Apply a bunch of rotations and reflections to a string consisting of `bdpq`:

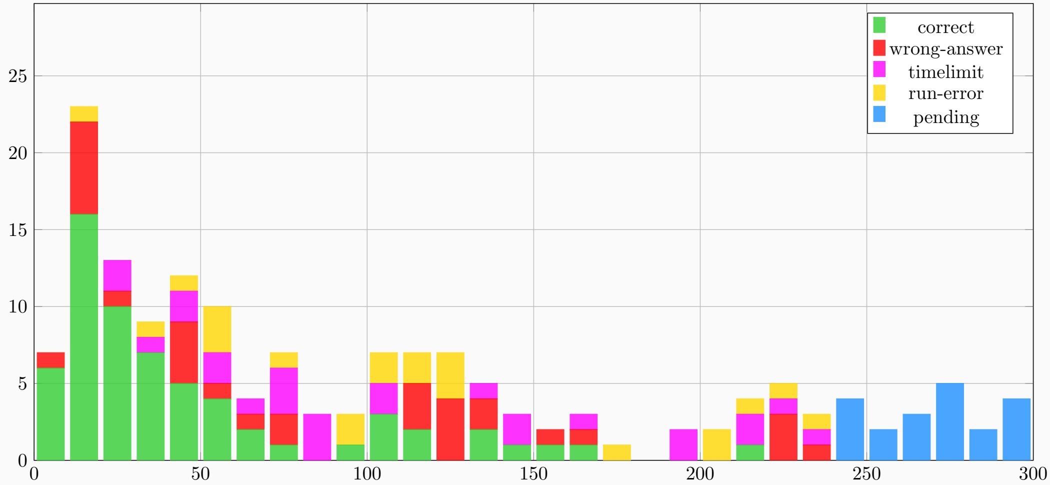
- `h`: horizontal flip: `bbq` $\leftrightarrow$ `pdd`
- `v`: vertical flip: `bbq` $\leftrightarrow$ `ppd`
- `r`: 180 degree rotation: `bbq` $\leftrightarrow$ `bqq`

Solution

- Applying all the transformations one by one takes $\Theta(n^2)$ time, too slow!
- Instead, we make some observations:
  - we may replace each `r` by `hv`
  - doing `vh` is the same as `hv` $\rightsquigarrow$ move all `h` to the front
  - we only need to know if the number of `h` is even or odd (same for `v`)
- Using these, we only need to do at most one `h` and at most one `v` transformation.
- All of this can be done in $\mathcal{O}(n)$ time.

A: Alien Attack 2

Problem author: Yvonne Kothmeier & Andreas Grigorjew



A: Alien Attack 2

Problem author: Yvonne Kothmeier & Andreas Grigorjew

Problem

- Find the size of the largest connected component in an undirected graph representing friendships.

A: Alien Attack 2

Problem author: Yvonne Kothmeier & Andreas Grigorjew

Problem

- Find the size of the largest connected component in an undirected graph representing friendships.

Solution

- Perform a graph search starting from any unvisited node.
- Utilize algorithms like Depth First Search (DFS), Breadth First Search (BFS), or Union-Find with path compression to traverse the graph.

A: Alien Attack 2

Problem author: Yvonne Kothmeier & Andreas Grigorjew

Problem

- Find the size of the largest connected component in an undirected graph representing friendships.

Solution

- Perform a graph search starting from any unvisited node.
- Utilize algorithms like Depth First Search (DFS), Breadth First Search (BFS), or Union-Find with path compression to traverse the graph.
- Count the number of nodes visited during each traversal to determine the size of the connected component.
- Repeat the process until all nodes have been visited.

A: Alien Attack 2

Problem author: Yvonne Kothmeier & Andreas Grigorjew

Problem

- Find the size of the largest connected component in an undirected graph representing friendships.

Solution

- Perform a graph search starting from any unvisited node.
- Utilize algorithms like Depth First Search (DFS), Breadth First Search (BFS), or Union-Find with path compression to traverse the graph.
- Count the number of nodes visited during each traversal to determine the size of the connected component.
- Repeat the process until all nodes have been visited.
- The size of the largest component found will dictate the size of the smallest necessary ship.

A: Alien Attack 2

Problem author: Yvonne Kothmeier & Andreas Grigorjew

Problem

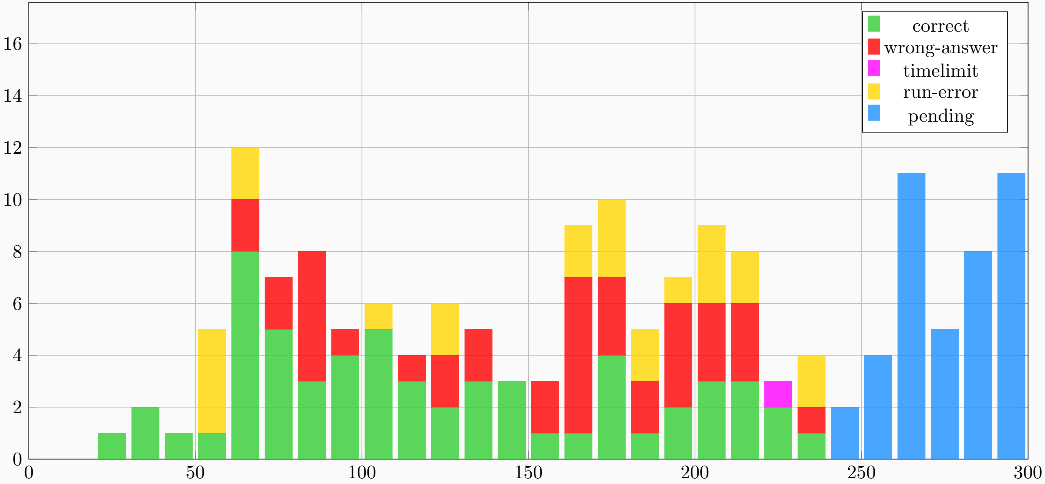
- Find the size of the largest connected component in an undirected graph representing friendships.

Solution

- Perform a graph search starting from any unvisited node.
- Utilize algorithms like Depth First Search (DFS), Breadth First Search (BFS), or Union-Find with path compression to traverse the graph.
- Count the number of nodes visited during each traversal to determine the size of the connected component.
- Repeat the process until all nodes have been visited.
- The size of the largest component found will dictate the size of the smallest necessary ship.
- Pitfalls: Inefficient graph traversal algorithms, e.g. revisiting nodes or Union-Find without path compression, may lead to time limit problems.
- For Python users: default recursion depth is low. Increase using sys.setrecursionlimit

I: Interference

Problem author: Sebastian Angrick



I: Interference

Problem author: Sebastian Angrick

Problem

- Given some alternating range updates, answer point queries

I: Interference

Problem author: Sebastian Angrick

Problem

- Given some alternating range updates, answer point queries

Solution

- Range is too large to work with, ignore it

I: Interference

Problem author: Sebastian Angrick

Problem

- Given some alternating range updates, answer point queries

Solution

- Range is too large to work with, ignore it
- $\mathcal{O}(n^2)$ is sufficient, for each query simulate all former updates.

I: Interference

Problem author: Sebastian Angrick

Problem

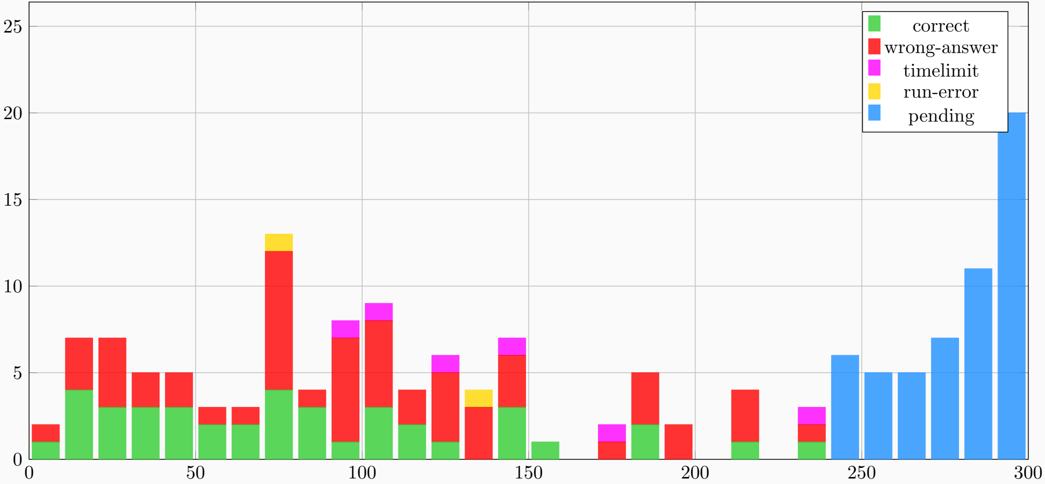
- Given some alternating range updates, answer point queries

Solution

- Range is too large to work with, ignore it
- $\mathcal{O}(n^2)$ is sufficient, for each query simulate all former updates.
- Pitfalls: Results can be large (long long may be needed) or negative, correctly deal with alternation

M: Musical Mending

Problem author: Brutenis Gliwa, Marian Zuska



M: Musical Mending

Problem author: Brutenis Gliwa, Marian Zuska

Problem

- **Problem:** Find the minimal distance from the input sequence to any sequence $x, x+1, x+2, \ldots, x+n-1$.

M: Musical Mending

Problem author: Brutenis Gliwa, Marian Zuska

Problem

- **Problem:** Find the minimal distance from the input sequence to any sequence $x, x+1, x+2, \ldots, x+n-1$.

Solution

- For a fixed x, the distance can be determined in $O(n)$.

M: Musical Mending

Problem author: Brutenis Gliwa, Marian Zuska

Problem

- **Problem:** Find the minimal distance from the input sequence to any sequence $x, x+1, x+2, \ldots, x+n-1$.

Solution

- For a fixed x, the distance can be determined in $O(n)$.
- Naive solution: Compute the distance for all possible $x \in [-200\,000, 200\,000]$. $\mathcal{O}(v \cdot n)$ is too slow!

M: Musical Mending

Problem author: Brutenis Gliwa, Marian Zuska

Problem

- **Problem:** Find the minimal distance from the input sequence to any sequence $x, x+1, x+2, \ldots, x+n-1$.

Solution

- For a fixed x, the distance can be determined in $O(n)$.
- Naive solution: Compute the distance for all possible $x \in [-200\,000, 200\,000]$. $\mathcal{O}(v \cdot n)$ is too slow!
- Binary searching x does not work, as the score is not a monotonic function.

M: Musical Mending

Problem author: Brutenis Gliwa, Marian Zuska

Problem

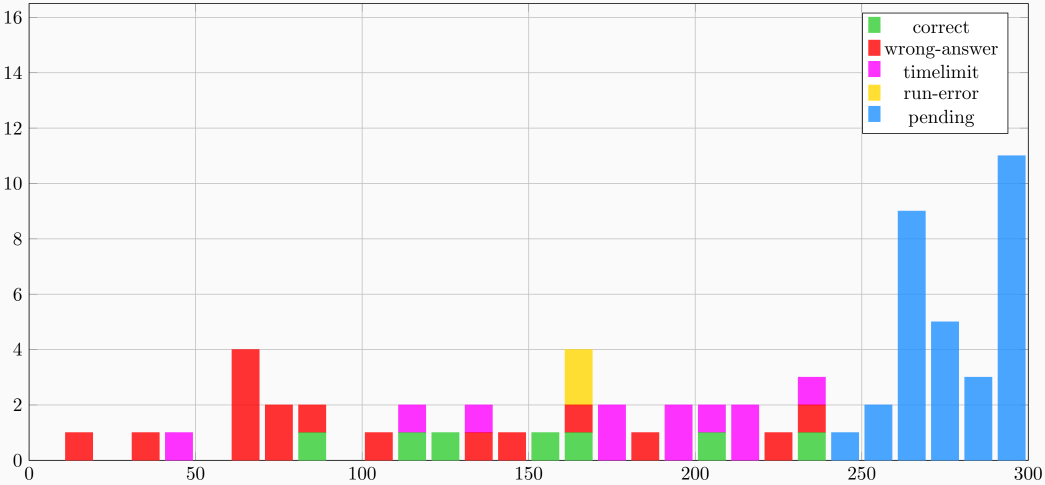
- **Problem:** Find the minimal distance from the input sequence to any sequence $x, x+1, x+2, \ldots, x+n-1$.

Solution

- For a fixed x, the distance can be determined in $O(n)$.
- Naive solution: Compute the distance for all possible $x \in [-200\,000, 200\,000]$. $\mathcal{O}(v \cdot n)$ is too slow!
- Binary searching x does not work, as the score is not a monotonic function.
- Ternary search the answer over all possible x! $\mathcal{O}(\log(v) \cdot n)$

C: Copycat Catcher

Problem author: Jannik Olbrich



C: Copycat Catcher

Problem author: Jannik Olbrich

Problem

- Determine whether pieces of code can be obtained by renaming variables from substrings of a reference

```
for i in list do print i j
    k in list do print k a
                 print b c
```

C: Copycat Catcher

Problem author: Jannik Olbrich

Problem

- Determine whether pieces of code can be obtained by renaming variables from substrings of a reference

```
for i in list do print i j
    k in list do print k a
                 print b c
```

Solution

- Transform the code:
  Replace each occurrence of a variable V with

C: Copycat Catcher

Problem author: Jannik Olbrich

Problem

- Determine whether pieces of code can be obtained by renaming variables from substrings of a reference

```
for i in list do print i j
    k in list do print k a
                 print b c
```

Solution

- Transform the code:
  Replace each occurrence of a variable V with
  - the distance to the previous occurrence of V, or

C: Copycat Catcher

Problem author: Jannik Olbrich

Problem

- Determine whether pieces of code can be obtained by renaming variables from substrings of a reference

```
for i in list do print i j
    k in list do print k a
                 print b c
```

Solution

- Transform the code:
  Replace each occurrence of a variable V with
  - the distance to the previous occurrence of V, or
  - 0 if there is no previous occurrence

C: Copycat Catcher

Problem author: Jannik Olbrich

Problem

- Determine whether pieces of code can be obtained by renaming variables from substrings of a reference

```
for i in list do print i j
    k in list do print k a
                 print b c
```

Solution

- Transform the code:
  Replace each occurrence of a variable V with
  - the distance to the previous occurrence of V, or
  - 0 if there is no previous occurrence

C: Copycat Catcher

Problem author: Jannik Olbrich

Problem

- Determine whether pieces of code can be obtained by renaming variables from substrings of a reference

```
for i in list do print i j
    k in list do print k a
                 print b c
```

Solution

- Transform the code:
  Replace each occurrence of a variable V with
  - the distance to the previous occurrence of V, or
  - 0 if there is no previous occurrence

```
for 0 in list do print 5 0
    0 in list do print 5 0
                 print 0 0
```

- Do this for all suffixes of the reference:

```
for 0 in list do print 5 0      list do print 0 0      0 0
    0 in list do print 5 0           do print 0 0        0
      in list do print 0 0              print 0 0
```

C: Copycat Catcher

Problem author: Jannik Olbrich

Problem

- Determine whether pieces of code can be obtained by renaming variables from substrings of a reference

```
for i in list do print i j
    k in list do print k a
                 print b c
```

Solution

- Transform the code:
  Replace each occurrence of a variable V with
  - the distance to the previous occurrence of V, or
  - 0 if there is no previous occurrence

```
for 0 in list do print 5 0
    0 in list do print 5 0
                 print 0 0
```

- Do this for all suffixes of the reference:

```
for 0 in list do print 5 0      list do print 0 0      0 0
    0 in list do print 5 0           do print 0 0        0
      in list do print 0 0              print 0 0
```

- Sort these transformed suffixes lexicographically, use binary search to find the transformed query

C: Copycat Catcher

Problem author: Jannik Olbrich

Problem

- Determine whether pieces of code can be obtained by renaming variables from substrings of a reference

```
for i in list do print i j
    k in list do print k a
                 print b c
```

Solution

- Transform the code:
  Replace each occurrence of a variable V with
  - the distance to the previous occurrence of V, or
  - 0 if there is no previous occurrence

```
for 0 in list do print 5 0
    0 in list do print 5 0
                 print 0 0
```

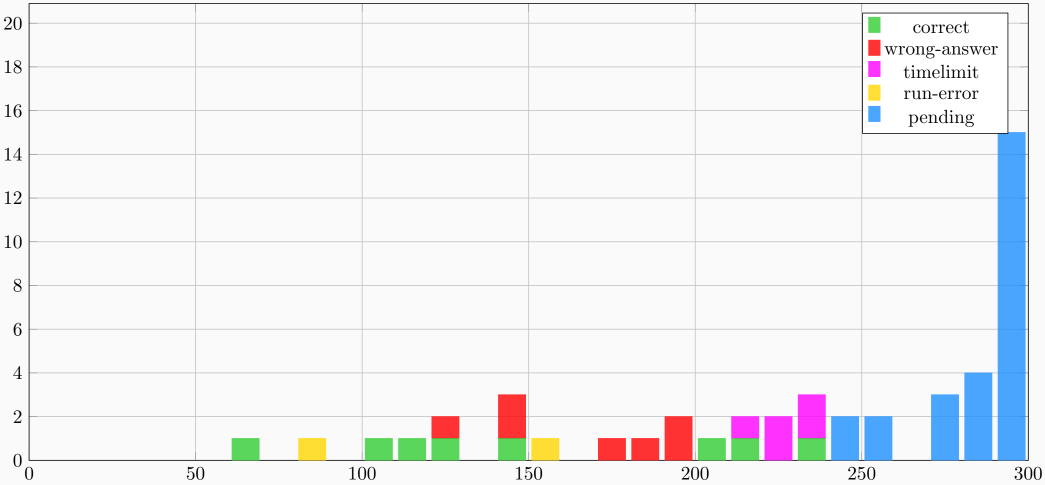
- Do this for all suffixes of the reference:

```
for 0 in list do print 5 0      list do print 0 0      0 0
    0 in list do print 5 0           do print 0 0        0
      in list do print 0 0              print 0 0
```

- Sort these transformed suffixes lexicographically, use binary search to find the transformed query
- Time complexity: $\mathcal{O}(n^2 + q \cdot qlen \cdot \log n)$, where $qlen \leq 2\,000$ is the max. length of a query

E: Even Odd Game

Problem author: Paul Wild



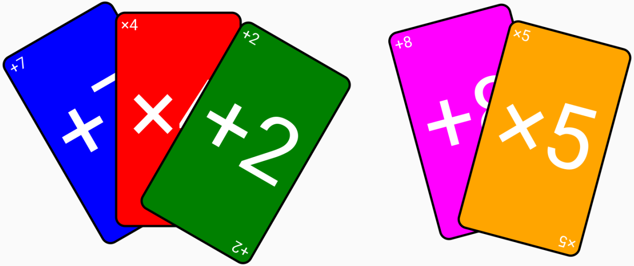
E: Even Odd Game

Problem author: Paul Wild

Problem

Find and interactively execute a winning strategy in the following game:

- There are some cards containing math operations $+n$ and $\times n$.
- Two players alternate picking cards until no cards are left.
- These operations are applied to a given number in the order they are picked.
- One player wins if the final result is even, the other wins if it is odd.



E: Even Odd Game

Problem author: Paul Wild

Solution

- As we only care about parity, reduce all numbers mod 2.

E: Even Odd Game

Problem author: Paul Wild

Solution

- As we only care about parity, reduce all numbers mod 2.
- So there are only three types of cards: $+0$, $+1$, $\times 0$
  - Note that $+0$ is the same as $\times 1$.

E: Even Odd Game

Problem author: Paul Wild

Solution

- As we only care about parity, reduce all numbers mod 2.
- So there are only three types of cards: $+0$, $+1$, $\times 0$
  - Note that $+0$ is the same as $\times 1$.
- There are at most $n \leq 300$ cards, so we can use $\Theta(n^3)$ dynamic programming:

$$\texttt{dp[who][cur][a][b][c]} = \text{Does player } \texttt{who} \text{ win when the current value is } \texttt{cur} \text{ and there are } \texttt{a}, \texttt{b} \text{ and } \texttt{c} \text{ operations of the respective types remaining?}$$

E: Even Odd Game

Problem author: Paul Wild

Solution

- As we only care about parity, reduce all numbers mod 2.
- So there are only three types of cards: $+0$, $+1$, $\times 0$
  - Note that $+0$ is the same as $\times 1$.
- There are at most $n \leq 300$ cards, so we can use $\Theta(n^3)$ dynamic programming:

$$\texttt{dp[who][cur][a][b][c]} = \text{Does player } \texttt{who} \text{ win when the current value is } \texttt{cur} \text{ and there are } \texttt{a}, \texttt{b} \text{ and } \texttt{c} \text{ operations of the respective types remaining?}$$

- The game can be played by following along the values in the DP table.

E: Even Odd Game

Problem author: Paul Wild

Solution

- As we only care about parity, reduce all numbers mod 2.
- So there are only three types of cards: $+0$, $+1$, $\times 0$
  - Note that $+0$ is the same as $\times 1$.
- There are at most $n \leq 300$ cards, so we can use $\Theta(n^3)$ dynamic programming:

  `dp[who][cur][a][b][c]` = Does player `who` win when the current value is `cur` and there are `a`, `b` and `c` operations of the respective types remaining?

- The game can be played by following along the values in the DP table.

E: Even Odd Game

Problem author: Paul Wild

Solution

- As we only care about parity, reduce all numbers mod 2.
- So there are only three types of cards: $+0$, $+1$, $\times 0$
  - Note that $+0$ is the same as $\times 1$.
- There are at most $n \leq 300$ cards, so we can use $\Theta(n^3)$ dynamic programming:

  `dp[who][cur][a][b][c]` = Does player `who` win when the current value is `cur` and there are `a`, `b` and `c` operations of the respective types remaining?

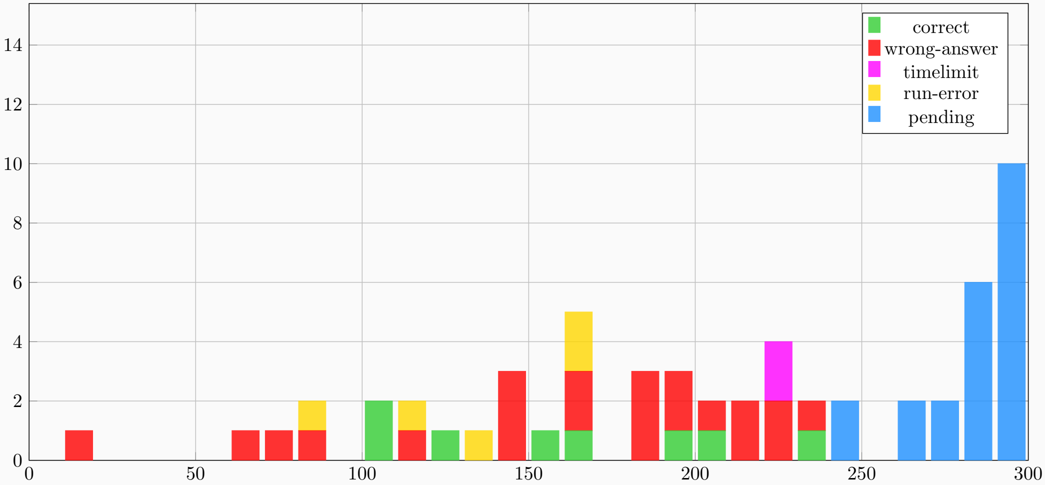
- The game can be played by following along the values in the DP table.

Challenge

Can you also solve the problem for $n \leq 10^5$?

G: Geometric Gridlock

Problem author: Paul Wild



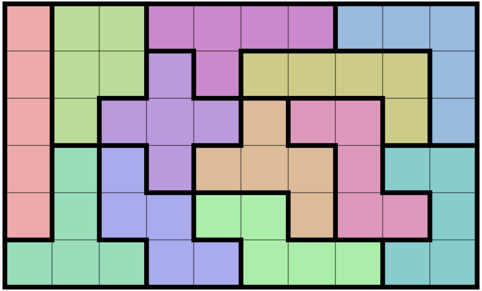
G: Geometric Gridlock

Problem author: Paul Wild

Problem

Construct a valid *Pentominous* grid of a given size:

- Divide an $h \times w$ grid into regions of size 5 (pentominoes)...
- ...such that no two adjacent regions have the same shape.



G: Geometric Gridlock

Problem author: Paul Wild

Insights and corner cases

- One of h or w must be a multiple of 5; by symmetry, assume it's w.

G: Geometric Gridlock

Problem author: Paul Wild

Insights and corner cases

- One of h or w must be a multiple of 5; by symmetry, assume it's w.
- $1 \times w$ is only solvable for $w = 5$:

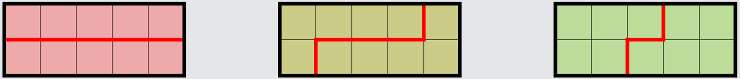
G: Geometric Gridlock

Problem author: Paul Wild

Insights and corner cases

- One of h or w must be a multiple of 5; by symmetry, assume it's w.
- $1 \times w$ is only solvable for $w = 5$:

- 2×5 is not solvable:



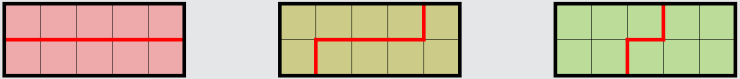
G: Geometric Gridlock

Problem author: Paul Wild

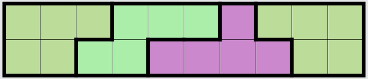
Insights and corner cases

- One of h or w must be a multiple of 5; by symmetry, assume it's w.
- $1 \times w$ is only solvable for $w = 5$:

- 2×5 is not solvable:



- $2 \times w$ is solvable in all other cases:



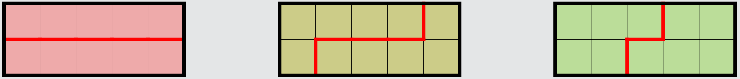
G: Geometric Gridlock

Problem author: Paul Wild

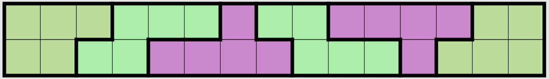
Insights and corner cases

- One of h or w must be a multiple of 5; by symmetry, assume it's w.
- $1 \times w$ is only solvable for $w = 5$:

- 2×5 is not solvable:



- $2 \times w$ is solvable in all other cases:



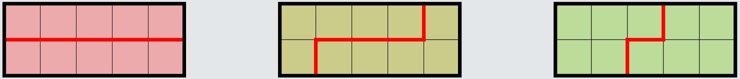
G: Geometric Gridlock

Problem author: Paul Wild

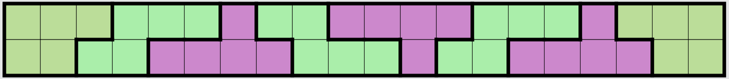
Insights and corner cases

- One of h or w must be a multiple of 5; by symmetry, assume it's w.
- $1 \times w$ is only solvable for $w = 5$:

- 2×5 is not solvable:



- $2 \times w$ is solvable in all other cases:

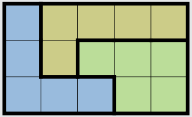


G: Geometric Gridlock

Problem author: Paul Wild

Solution

- For 3×5 we can come up with a solution that can be repeated to achieve any width:

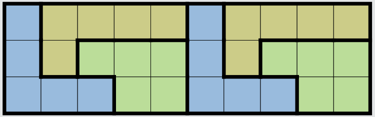


G: Geometric Gridlock

Problem author: Paul Wild

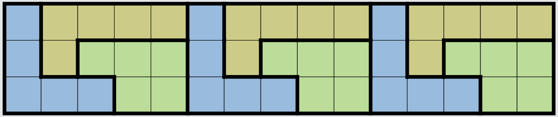
Solution

- For 3×5 we can come up with a solution that can be repeated to achieve any width:



Solution

- For 3×5 we can come up with a solution that can be repeated to achieve any width:

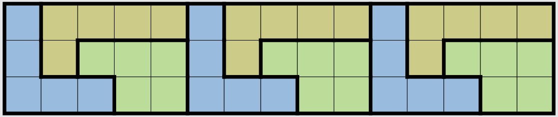


G: Geometric Gridlock

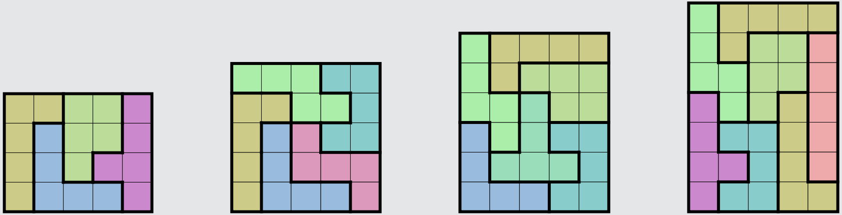
Problem author: Paul Wild

Solution

- For 3×5 we can come up with a solution that can be repeated to achieve any width:

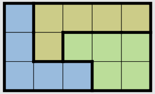


- Similar repeatable patterns exist for heights 4, 5, 6 and 7:



Solution (continued)

- With some care, these patterns can be chosen so that they tile along both directions:

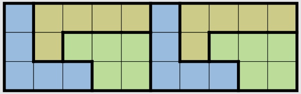


G: Geometric Gridlock

Problem author: Paul Wild

Solution (continued)

- With some care, these patterns can be chosen so that they tile along both directions:

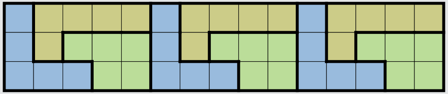


G: Geometric Gridlock

Problem author: Paul Wild

Solution (continued)

- With some care, these patterns can be chosen so that they tile along both directions:

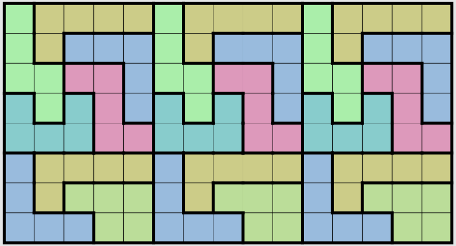


G: Geometric Gridlock

Problem author: Paul Wild

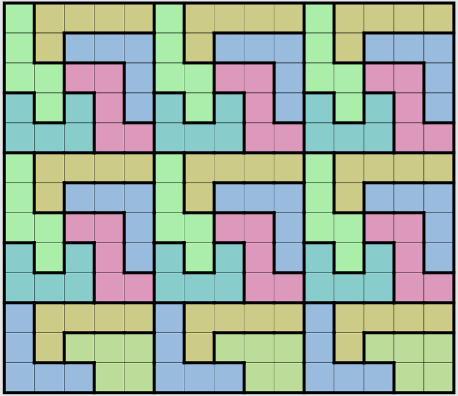
Solution (continued)

- With some care, these patterns can be chosen so that they tile along both directions:



Solution (continued)

- With some care, these patterns can be chosen so that they tile along both directions:

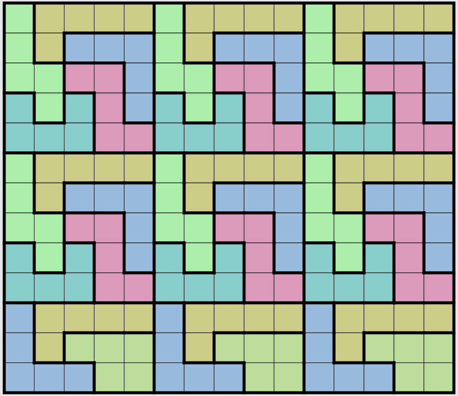


G: Geometric Gridlock

Problem author: Paul Wild

Solution (continued)

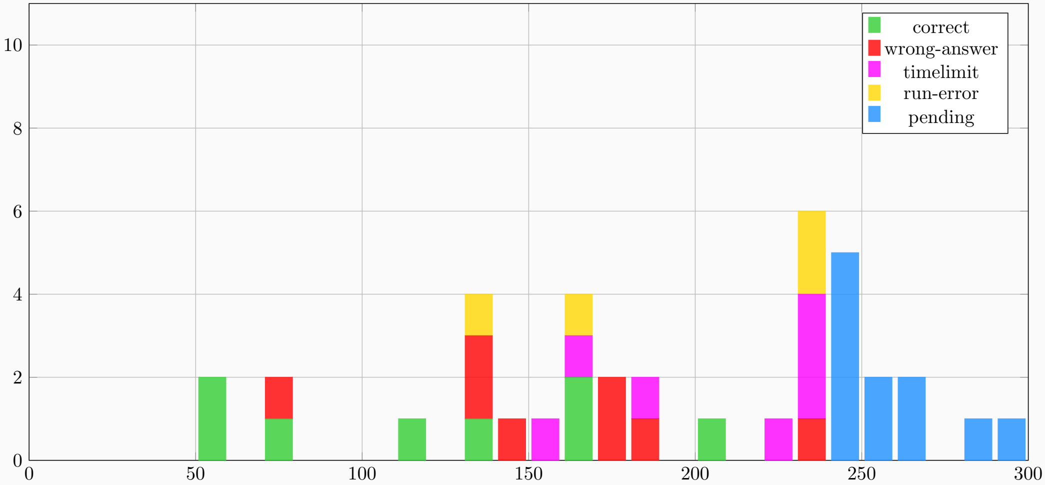
- With some care, these patterns can be chosen so that they tile along both directions:



- This way, we can reduce any height h to one of the base cases 3, 4, 5, 6 or 7.

D: Dark Alley

Problem author: Michael Zündorf



D: Dark Alley

Problem author: Michael Zündorf

Problem

Process the following queries:

- **+ b x:** place a lamp with brightness b at position x.
- **- b x:** remove a lamp with brightness b at position x.
- **? x:** calculate the brightness at position x.

Note that the light reduces by a factor of $\tilde{p} = 1 - p$ every metre.

0	0	0	0	0	0

D: Dark Alley

Problem author: Michael Zündorf

Problem

Process the following queries:

- `+ b x:` place a lamp with brightness b at position x.
- `- b x:` remove a lamp with brightness b at position x.
- `? x:` calculate the brightness at position x.

Note that the light reduces by a factor of $\tilde{p} = 1 - p$ every metre.

0.25	0.5	1	0.5	0.25	0.125

D: Dark Alley

Problem author: Michael Zündorf

Solution

0.25	0.5	1	0.5	0.25	0.125

D: Dark Alley

Problem author: Michael Zündorf

Solution

- Split light into two directions and store it in two data structures.
  $\Longrightarrow$ only consider light to the right for now

0	0	1	0.5	0.25	0.125

D: Dark Alley

Problem author: Michael Zündorf

Solution

- Split light into two directions and store it in two data structures.
  $\implies$ only consider light to the right for now
- A light with brightness b at position x contributes $b \cdot \tilde{p}^{y-x}$ at $y \geq x$.

0	0	1	0.5	0.25	0.125

D: Dark Alley

Problem author: Michael Zündorf

Solution

- Split light into two directions and store it in two data structures.
  $\Longrightarrow$ only consider light to the right for now
- A light with brightness b at position x contributes $b \cdot \tilde{p}^{y-x}$ at $y \geq x$.
- A light with brightness $b \cdot \tilde{p}^{x}$ at position 0 has the same contribution at y.

4	2	1	0.5	0.25	0.125

D: Dark Alley

Problem author: Michael Zündorf

Solution

- Split light into two directions and store it in two data structures.
  $\Longrightarrow$ only consider light to the right for now
- A light with brightness b at position x contributes $b \cdot \tilde{p}^{y-x}$ at $y \geq x$.
- A light with brightness $b \cdot \tilde{p}^{x}$ at position 0 has the same contribution at y.
- Do not propagate light.
- Place bulbs at positions $x, x+1, \ldots, n$ with constant brightness $b \cdot \tilde{p}^{x}$.

$0 \cdot \tilde{p}^{-0}$	$0 \cdot \tilde{p}^{-1}$	$4 \cdot \tilde{p}^{-2}$	$4 \cdot \tilde{p}^{-3}$	$4 \cdot \tilde{p}^{-4}$	$4 \cdot \tilde{p}^{-5}$

D: Dark Alley

Problem author: Michael Zündorf

Solution

- Split light into two directions and store it in two data structures.
  $\implies$ only consider light to the right for now
- A light with brightness b at position x contributes $b \cdot \tilde{p}^{y-x}$ at $y \geq x$.
- A light with brightness $b \cdot \tilde{p}^{x}$ at position 0 has the same contribution at y.
- Do not propagate light.
- Place bulbs at positions $x, x+1, \ldots, n$ with constant brightness $b \cdot \tilde{p}^{x}$.
- The light at position y is now too bright by a constant factor p^{y}.
- For queries of type `? x`, answer with $\ell_x \cdot \tilde{p}^{-x}$.

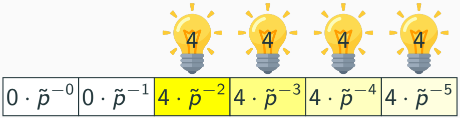
$0 \cdot \tilde{p}^{-0}$	$0 \cdot \tilde{p}^{-1}$	$4 \cdot \tilde{p}^{-2}$	$4 \cdot \tilde{p}^{-3}$	$4 \cdot \tilde{p}^{-4}$	$4 \cdot \tilde{p}^{-5}$

D: Dark Alley

Problem author: Michael Zündorf

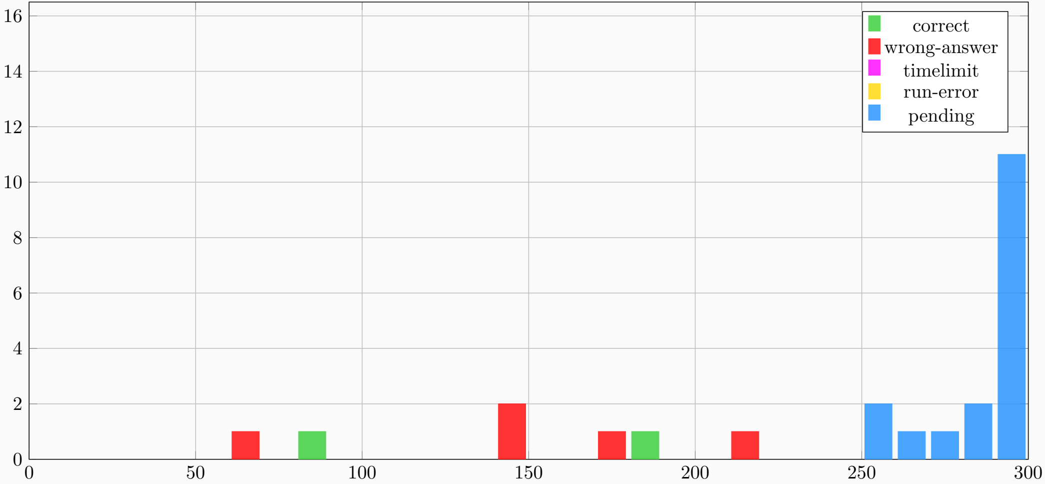
Solution

- Split light into two directions and store it in two data structures.
  $\Longrightarrow$ only consider light to the right for now
- A light with brightness b at position x contributes $b \cdot \tilde{p}^{y-x}$ at $y \geq x$.
- A light with brightness $b \cdot \tilde{p}^{x}$ at position 0 has the same contribution at y.
- Do not propagate light.
- Place bulbs at positions $x, x+1, \ldots, n$ with constant brightness $b \cdot \tilde{p}^{x}$.
- The light at position y is now too bright by a constant factor p^{y}.
- For queries of type `? x`, answer with $\ell_x \cdot \tilde{p}^{-x}$.
- Use segment tree or fenwick tree to maintain ℓ in $\mathcal{O}(q \log(n))$.



L: Laundry

Problem author: Wendy Yi



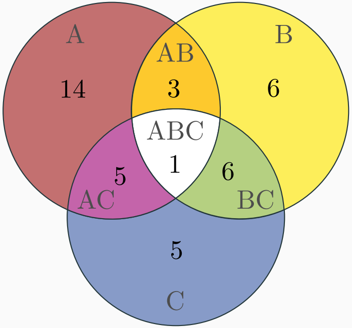
L: Laundry

Problem author: Wendy Yi

Problem

Given the capacity $1 \leq k \leq 10^9$ of a washing machine with three programmes, and how many items can be washed with which programmes.

What is the minimum number of loads needed to wash all items?



$k = 10$

L: Laundry

Problem author: Wendy Yi

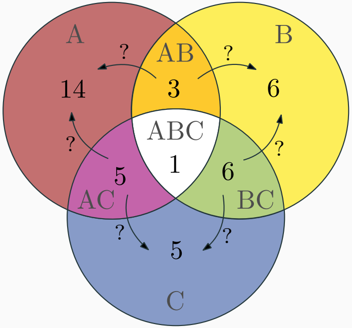
Problem

Given the capacity $1 \leq k \leq 10^9$ of a washing machine with three programmes, and how many items can be washed with which programmes.

What is the minimum number of loads needed to wash all items?

Insights

- Items with no choice and items with all choices are easy.



$k = 10$

$A : 14$

$B : 6$

$C : 5$

L: Laundry

Problem author: Wendy Yi

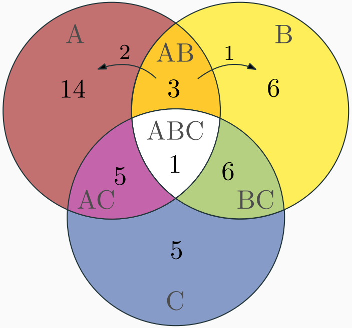
Problem

Given the capacity $1 \leq k \leq 10^9$ of a washing machine with three programmes, and how many items can be washed with which programmes.

What is the minimum number of loads needed to wash all items?

Insights

- Items with no choice and items with all choices are easy.
- If we assign all items of one set with two choices, an optimal solution for the rest can be determined greedily.



$k = 10$

$A : 16$

$B : 7$

$C : 5$

L: Laundry

Problem author: Wendy Yi

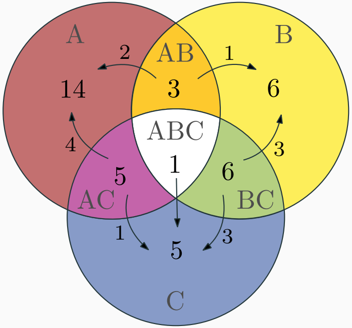
Problem

Given the capacity $1 \leq k \leq 10^9$ of a washing machine with three programmes, and how many items can be washed with which programmes.

What is the minimum number of loads needed to wash all items?

Insights

- Items with no choice and items with all choices are easy.
- If we assign all items of one set with two choices, an optimal solution for the rest can be determined greedily.



$k = 10$

$A : 20$

$B : 10$

$C : 10$

L: Laundry

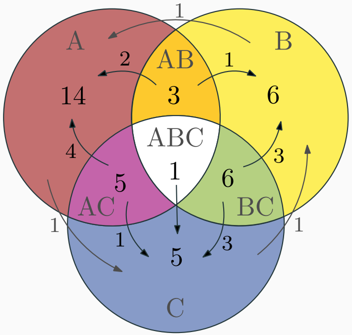
Problem author: Wendy Yi

Problem

Given the capacity $1 \leq k \leq 10^9$ of a washing machine with three programmes, and how many items can be washed with which programmes.
What is the minimum number of loads needed to wash all items?

Insights

- Items with no choice and items with all choices are easy.
- If we assign all items of one set with two choices, an optimal solution for the rest can be determined greedily.
- There is an optimal solution where there is one set of items with two choices that is assigned to the same programme.



$k = 10$

$A : 20$

$B : 10$

$C : 10$

L: Laundry

Problem author: Wendy Yi

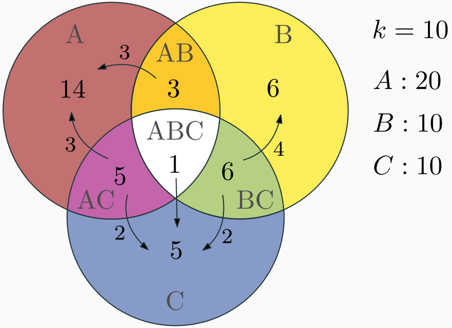
Problem

Given the capacity $1 \leq k \leq 10^9$ of a washing machine with three programmes, and how many items can be washed with which programmes.

What is the minimum number of loads needed to wash all items?

Insights

- Items with no choice and items with all choices are easy.
- If we assign all items of one set with two choices, an optimal solution for the rest can be determined greedily.
- There is an optimal solution where there is one set of items with two choices that is assigned to the same programme.



L: Laundry

Problem author: Wendy Yi

Problem

Given the capacity $1 \leq k \leq 10^9$ of a washing machine with three programmes, and how many items can be washed with which programmes.
What is the minimum number of loads needed to wash all items?

Solution

- Process items that can only be washed with one programme first.

L: Laundry

Problem author: Wendy Yi

Problem

Given the capacity $1 \leq k \leq 10^9$ of a washing machine with three programmes, and how many items can be washed with which programmes.
What is the minimum number of loads needed to wash all items?

Solution

- Process items that can only be washed with one programme first.
- For each set with two choices:

L: Laundry

Problem author: Wendy Yi

Problem

Given the capacity $1 \leq k \leq 10^9$ of a washing machine with three programmes, and how many items can be washed with which programmes.

What is the minimum number of loads needed to wash all items?

Solution

- Process items that can only be washed with one programme first.
- For each set with two choices:
  - Try to assign the whole set to one of the two choices.

L: Laundry

Problem author: Wendy Yi

Problem

Given the capacity $1 \leq k \leq 10^9$ of a washing machine with three programmes, and how many items can be washed with which programmes.

What is the minimum number of loads needed to wash all items?

Solution

- Process items that can only be washed with one programme first.
- For each set with two choices:
  - Try to assign the whole set to one of the two choices.
  - Determine the optimal solution for the other sets with two choices.

L: Laundry

Problem author: Wendy Yi

Problem

Given the capacity $1 \leq k \leq 10^9$ of a washing machine with three programmes, and how many items can be washed with which programmes.

What is the minimum number of loads needed to wash all items?

Solution

- Process items that can only be washed with one programme first.
- For each set with two choices:
  - Try to assign the whole set to one of the two choices.
  - Determine the optimal solution for the other sets with two choices.
  - Distribute the items with three choices optimally.

L: Laundry

Problem author: Wendy Yi

Problem

Given the capacity $1 \leq k \leq 10^9$ of a washing machine with three programmes, and how many items can be washed with which programmes.
What is the minimum number of loads needed to wash all items?

Solution

- Process items that can only be washed with one programme first.
- For each set with two choices:
  - Try to assign the whole set to one of the two choices.
  - Determine the optimal solution for the other sets with two choices.
  - Distribute the items with three choices optimally.
- Take the minimum over all such assignments (6 in total).

L: Laundry

Problem author: Wendy Yi

Problem

Given the capacity $1 \leq k \leq 10^9$ of a washing machine with three programmes, and how many items can be washed with which programmes.

What is the minimum number of loads needed to wash all items?

Solution

- Process items that can only be washed with one programme first.
- For each set with two choices:
  - Try to assign the whole set to one of the two choices.
  - Determine the optimal solution for the other sets with two choices.
  - Distribute the items with three choices optimally.
- Take the minimum over all such assignments (6 in total).

L: Laundry

Problem author: Wendy Yi

Problem

Given the capacity $1 \leq k \leq 10^9$ of a washing machine with three programmes, and how many items can be washed with which programmes.

What is the minimum number of loads needed to wash all items?

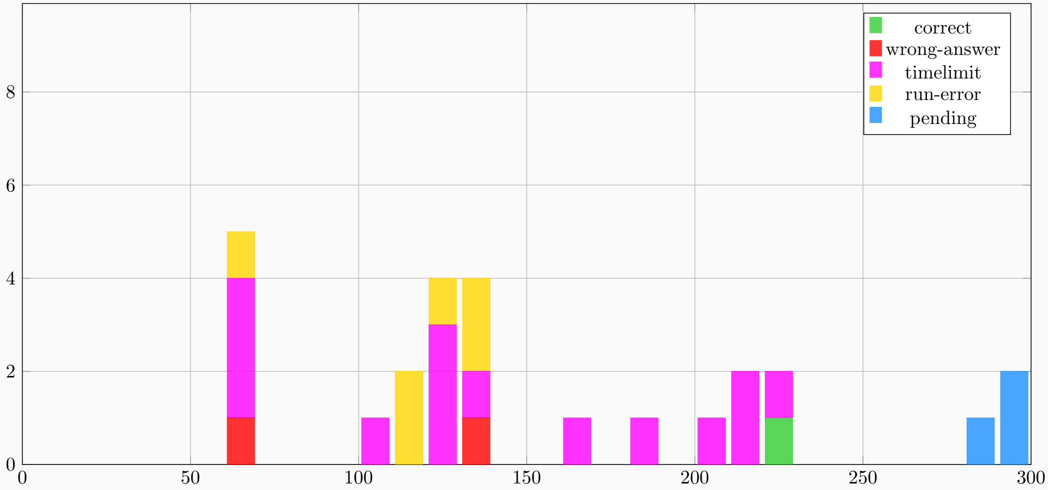
Solution

- Process items that can only be washed with one programme first.
- For each set with two choices:
  - Try to assign the whole set to one of the two choices.
  - Determine the optimal solution for the other sets with two choices.
  - Distribute the items with three choices optimally.
- Take the minimum over all such assignments (6 in total).

Running time: $\mathcal{O}(1)$ per test case

H: Headline Heat

Problem author: Christopher Weyand



H: Headline Heat

Problem author: Christopher Weyand

Problem

- Given $n < 10^5$ university names, $m < 10^5$ rivalries between universities, and $k < 10^5$ texts. For each text, answer if there are two rivalling universities with different number of occurrences. The summed length of all names and texts is $W < 10^6$.

H: Headline Heat

Problem author: Christopher Weyand

Problem

- Given $n < 10^5$ university names, $m < 10^5$ rivalries between universities, and $k < 10^5$ texts. For each text, answer if there are two rivalling universities with different number of occurrences. The summed length of all names and texts is $W < 10^6$.

Solution

First a solution in time $\mathcal{O}(mW)$.

H: Headline Heat

Problem author: Christopher Weyand

Problem

- Given $n < 10^5$ university names, $m < 10^5$ rivalries between universities, and $k < 10^5$ texts. For each text, answer if there are two rivalling universities with different number of occurrences. The summed length of all names and texts is $W < 10^6$.

Solution

First a solution in time $\mathcal{O}(mW)$.

- build Aho-Corasick automaton out of all the names

H: Headline Heat

Problem author: Christopher Weyand

Problem

- Given $n < 10^5$ university names, $m < 10^5$ rivalries between universities, and $k < 10^5$ texts. For each text, answer if there are two rivalling universities with different number of occurrences. The summed length of all names and texts is $W < 10^6$.

Solution

First a solution in time $\mathcal{O}(mW)$.

- build Aho-Corasick automaton out of all the names
- each state stores m long vector that tracks for each rivalry the difference in occurrence, initially 0

H: Headline Heat

Problem author: Christopher Weyand

Problem

- Given $n < 10^5$ university names, $m < 10^5$ rivalries between universities, and $k < 10^5$ texts. For each text, answer if there are two rivalling universities with different number of occurrences. The summed length of all names and texts is $W < 10^6$.

Solution

First a solution in time $\mathcal{O}(mW)$.

- build Aho-Corasick automaton out of all the names
- each state stores m long vector that tracks for each rivalry the difference in occurrence, initially 0
- for rivalry i between u and v, add $+1$ to the ith entry of the vector of states that accept u and -1 in states that accept v

H: Headline Heat

Problem author: Christopher Weyand

Problem

- Given $n < 10^5$ university names, $m < 10^5$ rivalries between universities, and $k < 10^5$ texts. For each text, answer if there are two rivalling universities with different number of occurrences. The summed length of all names and texts is $W < 10^6$.

Solution

First a solution in time $\mathcal{O}(mW)$.

- build Aho-Corasick automaton out of all the names
- each state stores m long vector that tracks for each rivalry the difference in occurrence, initially 0
- for rivalry i between u and v, add $+1$ to the ith entry of the vector of states that accept u and -1 in states that accept v
- to process an article, feed the text into the automaton and add the m long vectors up element wise

H: Headline Heat

Problem author: Christopher Weyand

Problem

- Given $n < 10^5$ university names, $m < 10^5$ rivalries between universities, and $k < 10^5$ texts. For each text, answer if there are two rivalling universities with different number of occurrences. The summed length of all names and texts is $W < 10^6$.

Solution

First a solution in time $\mathcal{O}(mW)$.

- build Aho-Corasick automaton out of all the names
- each state stores m long vector that tracks for each rivalry the difference in occurrence, initially 0
- for rivalry i between u and v, add $+1$ to the ith entry of the vector of states that accept u and -1 in states that accept v
- to process an article, feed the text into the automaton and add the m long vectors up element wise
- a text is safe if we get a vector with all 0s

H: Headline Heat

Problem author: Christopher Weyand

Problem

- Given $n < 10^5$ university names, $m < 10^5$ rivalries between universities, and $k < 10^5$ texts. For each text, answer if there are two rivalling universities with different number of occurrences. The summed length of all names and texts is $W < 10^6$.

Solution

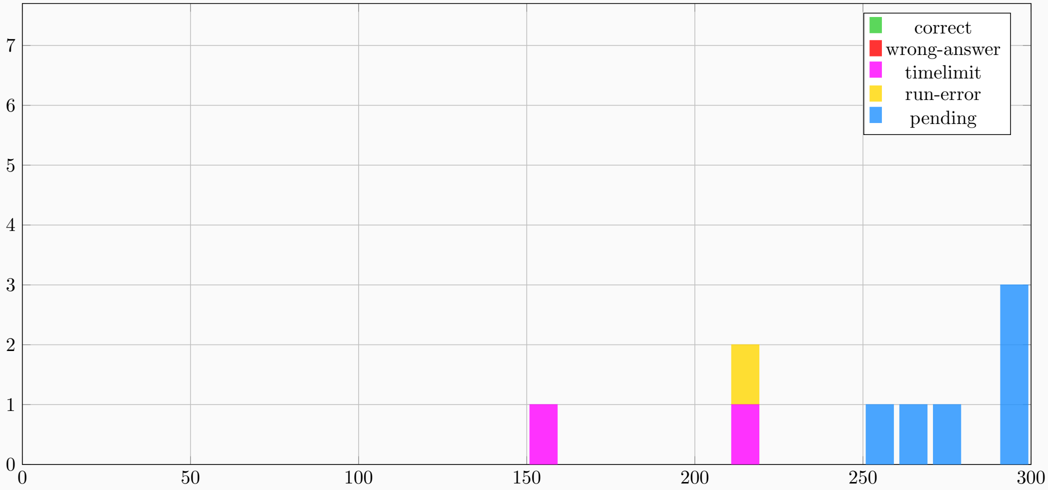
First a solution in time $\mathcal{O}(mW)$.

- build Aho-Corasick automaton out of all the names
- each state stores m long vector that tracks for each rivalry the difference in occurrence, initially 0
- for rivalry i between u and v, add $+1$ to the ith entry of the vector of states that accept u and -1 in states that accept v
- to process an article, feed the text into the automaton and add the m long vectors up element wise
- a text is safe if we get a vector with all 0s

To avoid quadratic time, hash the vectors. **Runtime**: $\mathcal{O}(n + m + k)$

J: Jigsaw Present

Problem author: Erik Sünderhauf



J: Jigsaw Present

Problem author: Erik Sünderhauf

Problem

- Given $n \leq 4\,096$ pairs of numbers (x_i, y_i), with $1 \leq x_i \leq 4\,096 =: C$ and $|y_i| \leq C$. Find two distinct subsets with the same sum or report that this is not possible.

J: Jigsaw Present

Problem author: Erik Sünderhauf

Problem

- Given $n \leq 4\,096$ pairs of numbers (x_i, y_i), with $1 \leq x_i \leq 4\,096 =: C$ and $|y_i| \leq C$. Find two distinct subsets with the same sum or report that this is not possible.

Solution

- The coordinates are too large to run a dp solution, so we should look for a brute force approach.

J: Jigsaw Present

Problem author: Erik Sünderhauf

Problem

- Given $n \leq 4\,096$ pairs of numbers (x_i, y_i), with $1 \leq x_i \leq 4\,096 =: C$ and $|y_i| \leq C$. Find two distinct subsets with the same sum or report that this is not possible.

Solution

- The coordinates are too large to run a dp solution, so we should look for a brute force approach.
- We could solve this in $\mathcal{O}(3^n)$ by deciding for each element whether it should go in the 1st subset, the 2nd subset, or none of the subsets.

J: Jigsaw Present

Problem author: Erik Sünderhauf

Problem

- Given $n \leq 4\,096$ pairs of numbers (x_i, y_i), with $1 \leq x_i \leq 4\,096 =: C$ and $|y_i| \leq C$. Find two distinct subsets with the same sum or report that this is not possible.

Solution

- The coordinates are too large to run a dp solution, so we should look for a brute force approach.
- We could solve this in $\mathcal{O}(3^n)$ by deciding for each element whether it should go in the 1st subset, the 2nd subset, or none of the subsets.
- Optimize with meet-in-the-middle to $\mathcal{O}(3^{n/2})$ by computing all possible sums of the form

$$\sum_{i=1}^{n/2} s_i \cdot (x_i, y_i), \sum_{i=n/2+1}^{n} s_i \cdot (x_i, y_i), \quad s_i \in \{-1, 0, 1\}$$

and finding a collision.

J: Jigsaw Present

Problem author: Erik Sünderhauf

Problem

- Given $n \leq 4\,096$ pairs of numbers (x_i, y_i), with $1 \leq x_i \leq 4\,096 =: C$ and $|y_i| \leq C$. Find two distinct subsets with the same sum or report that this is not possible.

Solution

- The coordinates are too large to run a dp solution, so we should look for a brute force approach.
- We could solve this in $\mathcal{O}(3^n)$ by deciding for each element whether it should go in the 1st subset, the 2nd subset, or none of the subsets.
- Optimize with meet-in-the-middle to $\mathcal{O}(3^{n/2})$ by computing all possible sums of the form

$$\sum_{i=1}^{n/2} s_i \cdot (x_i, y_i), \sum_{i=n/2+1}^{n} s_i \cdot (x_i, y_i), \quad s_i \in \{-1, 0, 1\}$$

  and finding a collision.
- But n is waaay too large for this approach...

J: Jigsaw Present

Problem author: Erik Sünderhauf

Insights

- We can have up to 2^n possible subset sums, which is exponential in n.

J: Jigsaw Present

Problem author: Erik Sünderhauf

Insights

- We can have up to 2^n possible subset sums, which is exponential in n.
- However, the absolute value of the coordinates in any subset sum are always $\leq n \cdot C$, which is polynomial in n!

J: Jigsaw Present

Problem author: Erik Sünderhauf

Insights

- We can have up to 2^n possible subset sums, which is exponential in n.
- However, the absolute value of the coordinates in any subset sum are always $\leq n \cdot C$, which is polynomial in n!

Solution

- If n is large enough we can always find two distinct subsets with the same sum. Just do `n = min(n, N)` at the beginning of your code. ($N \sim 28 - 32$)

J: Jigsaw Present

Problem author: Erik Sünderhauf

Insights

- We can have up to 2^n possible subset sums, which is exponential in n.
- However, the absolute value of the coordinates in any subset sum are always $\leq n \cdot C$, which is polynomial in n!

Solution

- If n is large enough we can always find two distinct subsets with the same sum. Just do `n = min(n, N)` at the beginning of your code. ($N \sim 28 - 32$)
- One can prove that for $n \geq 32$ there always is a collision (short sketch on next slide).

J: Jigsaw Present

Problem author: Erik Sünderhauf

Insights

- We can have up to 2^n possible subset sums, which is exponential in n.
- However, the absolute value of the coordinates in any subset sum are always $\leq n \cdot C$, which is polynomial in n!

Solution

- If n is large enough we can always find two distinct subsets with the same sum. Just do `n = min(n, N)` at the beginning of your code. ($N \sim 28 - 32$)
- One can prove that for $n \geq 32$ there always is a collision (short sketch on next slide).
- Challenge: Construct test cases without collision and with a large n. The best case we could achieve has $n = 27$. Hint: powers of 2 are not useful.

J: Jigsaw Present

Problem author: Erik Sünderhauf

Proof sketch

Let (X, Y) be the total sum of all pairs. Pick a random subset with sum $(\tilde{x}, \tilde{y})$. Using Chebyshev's inequality you can show that the probability that we are "close"' to the total sum

$$\left|(\tilde{x}, \tilde{y}) - \frac{1}{2}(X, Y)\right| \lesssim \sqrt{n}C$$

happens with probability $\geq 1/2$. Note that there are $\mathcal{O}(nC^2)$ possible sums that are "close". If all subset sums that are "close" to the total sum are distinct, then this requires

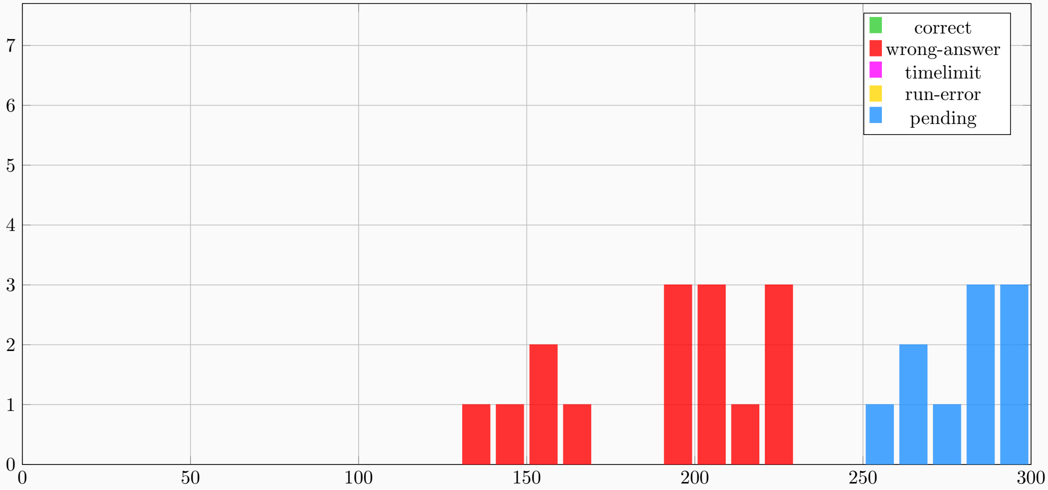
$$nC^2 \cdot 2^{-n} \gtrsim \frac{1}{2} \Rightarrow C \gtrsim \frac{2^{n/2}}{\sqrt{n}}.$$

Inserting numbers and more details[1] shows that we always have a collision for $n \geq 32$.

[1] search for "Probabilistic method"

F: Fair Fruitcake Fragmenting

Problem author: Jannik Olbrich

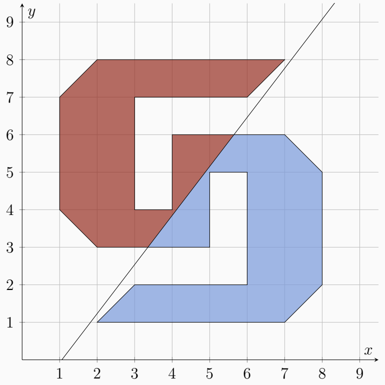


F: Fair Fruitcake Fragmenting

Problem author: Jannik Olbrich

Problem

Given a point symmetric polygon, check if it can be cut into exactly two pieces of equal size along an infinite line.



F: Fair Fruitcake Fragmenting

Problem author: Jannik Olbrich

Solution

- Polygon is point symmetric
- Parts must have equal size

  $\implies$ Line has to go through centre of mass = point of symmetry

F: Fair Fruitcake Fragmenting

Problem author: Jannik Olbrich

Solution

- Polygon is point symmetric
- Parts must have equal size

  $\Longrightarrow$ Line has to go through centre of mass $=$ point of symmetry

  $\Longrightarrow$ We can do a sweepline around centre of mass

F: Fair Fruitcake Fragmenting

Problem author: Jannik Olbrich

Solution

- Polygon is point symmetric
- Parts must have equal size

  $\Longrightarrow$ Line has to go through centre of mass $=$ point of symmetry

  $\Longrightarrow$ We can do a sweepline around centre of mass
- Due to symmetry, it is sufficient to keep track of the upper half of the polygon

F: Fair Fruitcake Fragmenting

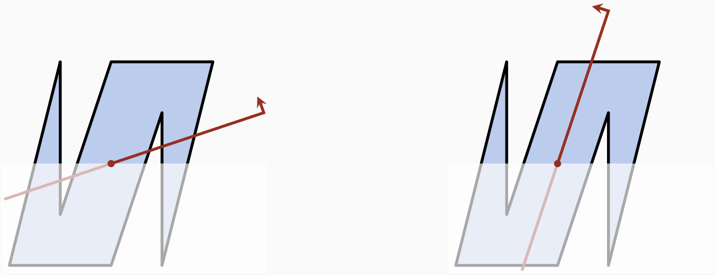
Problem author: Jannik Olbrich

Solution

- Polygon is point symmetric
- Parts must have equal size

  $\Longrightarrow$ Line has to go through centre of mass $=$ point of symmetry

  $\Longrightarrow$ We can do a sweepline around centre of mass
- Due to symmetry, it is sufficient to keep track of the upper half of the polygon
- Sweepline is valid answer $\Longleftrightarrow$ sweepline intersects the polygon exactly once

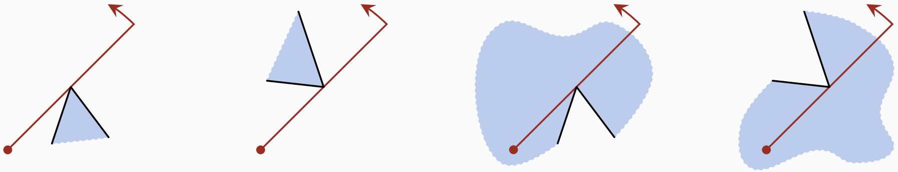


F: Fair Fruitcake Fragmenting

Problem author: Jannik Olbrich

Sweepline

- Number of intersections can only change at corners



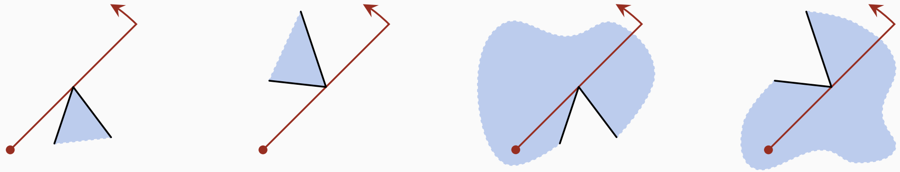
a: -2 b: $+2$ c: -2 d: $+2$

F: Fair Fruitcake Fragmenting

Problem author: Jannik Olbrich

Sweepline

- Number of intersections can only change at corners
- $+$ events come before $-$ events



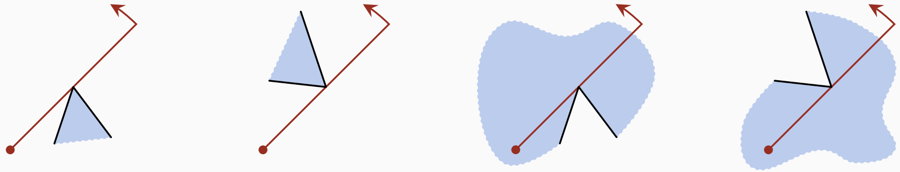
a: -2 b: $+2$ c: -2 d: $+2$

F: Fair Fruitcake Fragmenting

Problem author: Jannik Olbrich

Sweepline

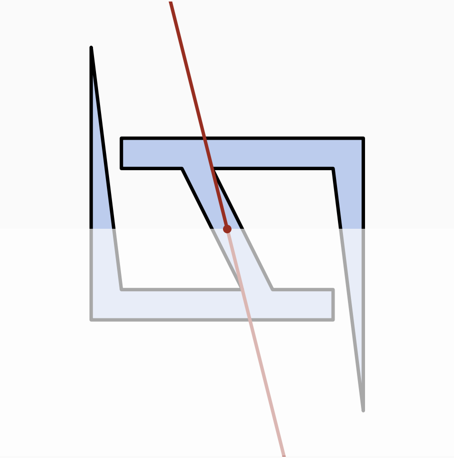
- Number of intersections can only change at corners
- $+$ events come before $-$ events
- If after a type a event the sweepline has size 1, the line is ok
- Type c events are only ok if we can rotate an ε further
  (add a dummy event halfway to the next actual event)



a: -2 b: $+2$ c: -2 d: $+2$

Edgecase

There might be no valid cut line that goes through any corner

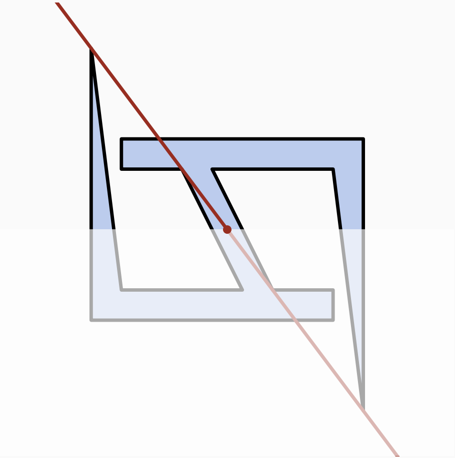


F: Fair Fruitcake Fragmenting

Problem author: Jannik Olbrich

Edgecase

There might be no valid cut line that goes through any corner

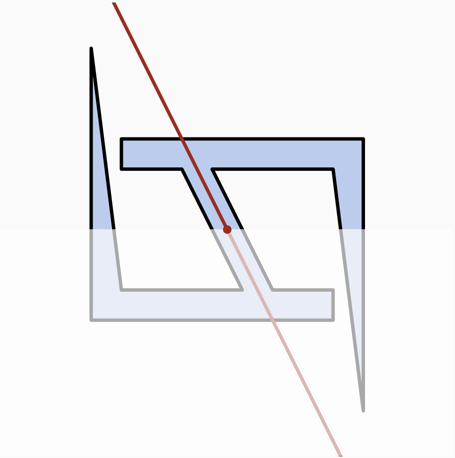


F: Fair Fruitcake Fragmenting

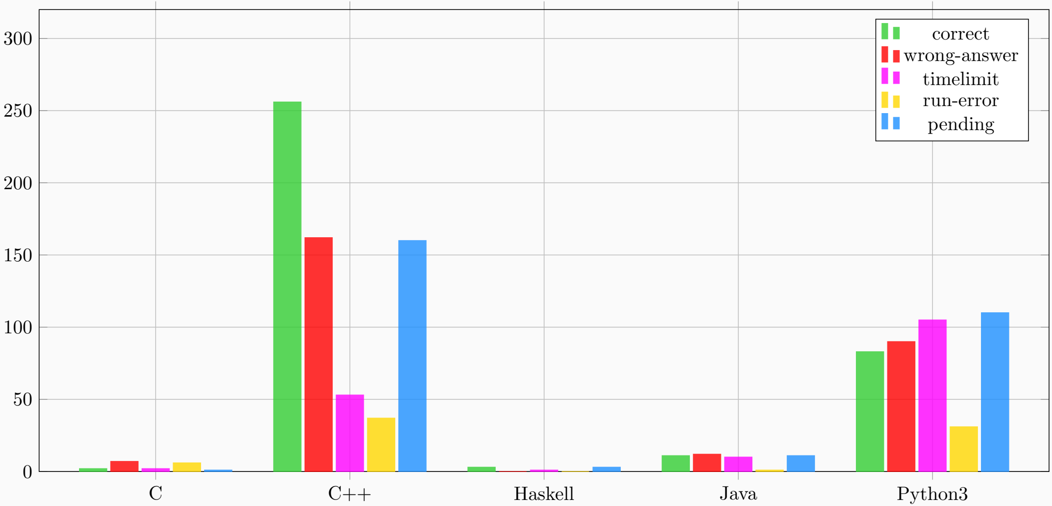
Problem author: Jannik Olbrich

Edgecase

There might be no valid cut line that goes through any corner



Language stats



Random facts

Jury work

- 583 secret test cases ($\approx$ 45 per problem)

Random facts

Jury work

- 583 secret test cases ($\approx$ 45 per problem)
- 149 jury solutions

Random facts

Jury work

- 583 secret test cases (≈ 45 per problem)
- 149 jury solutions
- The minimum number of lines the jury needed to solve all problems is

$$8 + 3 + 21 + 43 + 32 + 53 + 23 + 46 + 16 + 38 + 6 + 18 + 6 = 313$$

  On average 24.1 lines per problem

Random facts

Jury work

- 583 secret test cases ($\approx$ 45 per problem)
- 149 jury solutions
- The minimum number of lines the jury needed to solve all problems is

  $$8 + 3 + 21 + 43 + 32 + 53 + 23 + 46 + 16 + 38 + 6 + 18 + 6 = 313$$

  On average 24.1 lines per problem
- The minimum number of characters the jury needed to solve all problems is

  $$231 + 196 + 495 + 828 + 674 + 1109 + 818 + 1407 + 393 + 952 + 254 + 615 + 231$$

  On average 631 characters per problem