GCPC 2023 Presentation of Solutions

June 17, 2023

GCPC 2023 Jury

- **Paul Jungeblut**
  Karlsruhe Institute of Technology
- **Felicia Lucke**
  Fribourg University CH, CPUlm
- **Jannik Olbrich**
  Ulm University, CPUlm
- **Christopher Weyand**
  Karlsruhe Institute of Technology
- **Marcel Wienöbst**
  University of Lübeck, CPUlm
- **Paul Wild**
  Friedrich-Alexander University
  Erlangen-Nürnberg, CPUlm
- **Wendy Yi**
  Karlsruhe Institute of Technology
- **Michael Zündorf**
  Karlsruhe Institute of Technology, CPUlm

GCPC 2023 Test Solvers

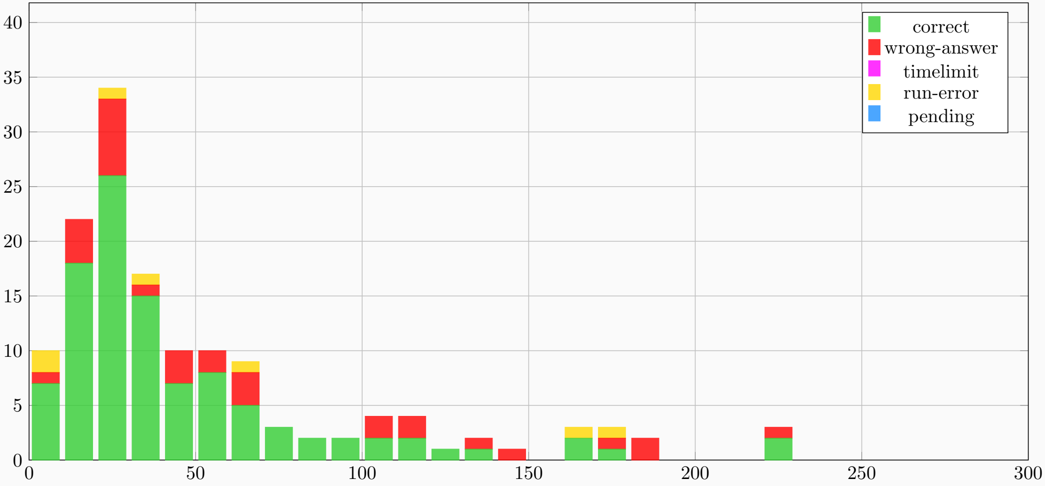
- **Rudolf Fleischer**
  Heinrich-Heine-University Düsseldorf, CPUlm
- **Florian Marwitz**
  University of Lübeck
- **Michael Ruderer**
  Augsburg University, CPUlm
- **Erik Sünderhauf**
  Technical University of Munich

GCPC 2023 Technical Team

- **Nathan Maier**
  CPUlm
- **Alexander Schmid**
  CPUlm

E: Eszett

Problem Author: Paul Wild



E: Eszett

Problem Author: Paul Wild

Problem

Given an uppercase string, find all of its transformations into lowercase, where each `SS` may transform to either `ss` or `B` (approximating the German 'ß'). The string contains at most three `S`.

E: Eszett

Problem Author: Paul Wild

Problem

Given an uppercase string, find all of its transformations into lowercase, where each `SS` may transform to either `ss` or `B` (approximating the German 'ß'). The string contains at most three `S`.

Solution

- If the string contains `SSS`, there are three solutions, replacing `SSS` with `sss`, `sB` and `Bs` respectively.

E: Eszett

Problem Author: Paul Wild

Problem

Given an uppercase string, find all of its transformations into lowercase, where each `SS` may transform to either `ss` or `B` (approximating the German 'ß'). The string contains at most three `S`.

Solution

- If the string contains `SSS`, there are three solutions, replacing `SSS` with `sss`, `sB` and `Bs` respectively.
- Otherwise, if the string contains `SS`, there are exactly two solutions, one with `ss` and one with `B`.

E: Eszett

Problem Author: Paul Wild

Problem

Given an uppercase string, find all of its transformations into lowercase, where each `SS` may transform to either `ss` or `B` (approximating the German ‘ß’). The string contains at most three `S`.

Solution

- If the string contains `SSS`, there are three solutions, replacing `SSS` with `sss`, `sB` and `Bs` respectively.
- Otherwise, if the string contains `SS`, there are exactly two solutions, one with `ss` and one with `B`.
- Otherwise, the only solution is the lowercase version of the string.

E: Eszett

Problem Author: Paul Wild

Problem

Given an uppercase string, find all of its transformations into lowercase, where each `SS` may transform to either `ss` or `B` (approximating the German 'ß'). The string contains at most three `S`.

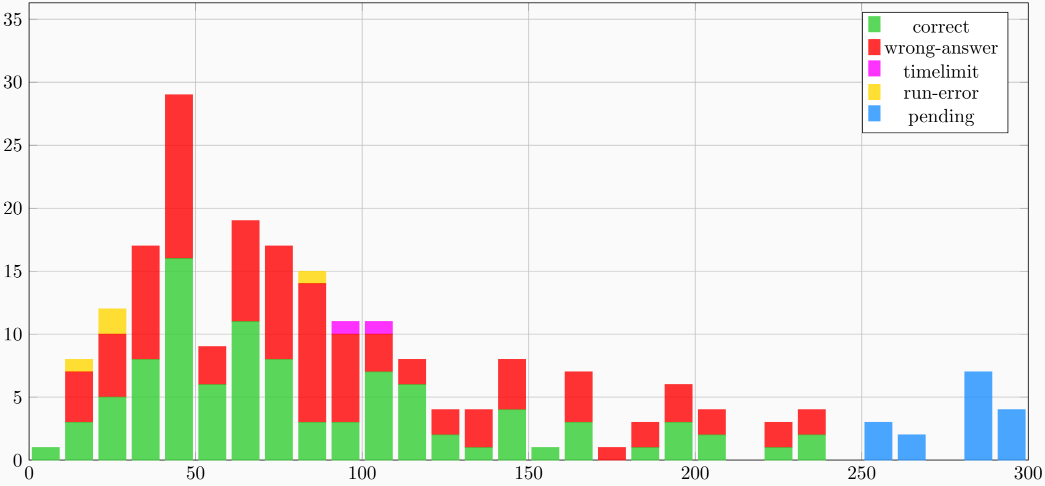
Solution

- If the string contains `SSS`, there are three solutions, replacing `SSS` with `sss`, `sB` and `Bs` respectively.
- Otherwise, if the string contains `SS`, there are exactly two solutions, one with `ss` and one with `B`.
- Otherwise, the only solution is the lowercase version of the string.
- Sample Implementation in Python:

```python
a = input().lower()
if a.find('sss') != -1:
  print(a.replace('sss', 'sB'))
  print(a.replace('sss', 'Bs'))
elif a.find('ss') != -1:
  print(a.replace('ss', 'B'))
print(a)
```

G: German Conference for Public Counting

Problem Author: Paul Wild



G: German Conference for Public Counting

Problem Author: Paul Wild

Problem

Count the number of signs with digits needed to display all numbers from 0 to n.

Example ($n = 15$)

$\boxed{1}\boxed{5}$, $\boxed{1}\boxed{4}$, $\boxed{1}\boxed{3}$, $\boxed{1}\boxed{2}$, $\boxed{1}\boxed{1}$, $\boxed{1}\boxed{0}$, $\boxed{9}$, $\boxed{8}$, $\boxed{7}$, $\boxed{6}$, $\boxed{5}$, $\boxed{4}$, $\boxed{3}$, $\boxed{2}$, $\boxed{1}$, $\boxed{0}$

We need 11 signs: $\boxed{0}$, $\boxed{1}$, $\boxed{1}$, $\boxed{2}$, $\boxed{3}$, $\boxed{4}$, $\boxed{5}$, $\boxed{6}$, $\boxed{7}$, $\boxed{8}$, $\boxed{9}$

G: German Conference for Public Counting

Problem Author: Paul Wild

Problem

Count the number of signs with digits needed to display all numbers from 0 to n.

Example ($n = 15$)

$\boxed{1}\boxed{5}, \boxed{1}\boxed{4}, \boxed{1}\boxed{3}, \boxed{1}\boxed{2}, \boxed{1}\boxed{1}, \boxed{1}\boxed{0}, \boxed{9}, \boxed{8}, \boxed{7}, \boxed{6}, \boxed{5}, \boxed{4}, \boxed{3}, \boxed{2}, \boxed{1}, \boxed{0}$

We need 11 signs: $\boxed{0}, \boxed{1}, \boxed{1}, \boxed{2}, \boxed{3}, \boxed{4}, \boxed{5}, \boxed{6}, \boxed{7}, \boxed{8}, \boxed{9}$

Solution

- For each digit, find the number in the range that uses the most copies of that digit.

G: German Conference for Public Counting

Problem Author: Paul Wild

Problem

Count the number of signs with digits needed to display all numbers from 0 to n.

Example ($n = 15$)

$\boxed{1}\boxed{5}, \boxed{1}\boxed{4}, \boxed{1}\boxed{3}, \boxed{1}\boxed{2}, \boxed{1}\boxed{1}, \boxed{1}\boxed{0}, \boxed{9}, \boxed{8}, \boxed{7}, \boxed{6}, \boxed{5}, \boxed{4}, \boxed{3}, \boxed{2}, \boxed{1}, \boxed{0}$

We need 11 signs: $\boxed{0}, \boxed{1}, \boxed{1}, \boxed{2}, \boxed{3}, \boxed{4}, \boxed{5}, \boxed{6}, \boxed{7}, \boxed{8}, \boxed{9}$

Solution

- For each digit, find the number in the range that uses the most copies of that digit.
- For each digit from 1 to 9:
  - Find the longest *repdigit* (number made up only of that digit) not exceeding n.
  - Add its length to the result.

G: German Conference for Public Counting

Problem Author: Paul Wild

Problem

Count the number of signs with digits needed to display all numbers from 0 to n.

Example ($n = 15$)

$\boxed{1}\boxed{5}, \boxed{1}\boxed{4}, \boxed{1}\boxed{3}, \boxed{1}\boxed{2}, \boxed{1}\boxed{1}, \boxed{1}\boxed{0}, \boxed{9}, \boxed{8}, \boxed{7}, \boxed{6}, \boxed{5}, \boxed{4}, \boxed{3}, \boxed{2}, \boxed{1}, \boxed{0}$

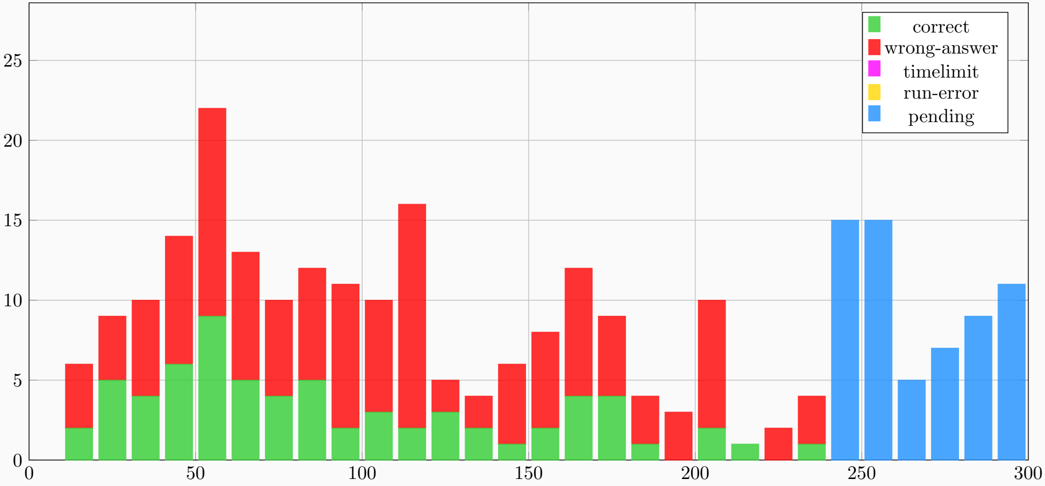
We need 11 signs: $\boxed{0}, \boxed{1}, \boxed{1}, \boxed{2}, \boxed{3}, \boxed{4}, \boxed{5}, \boxed{6}, \boxed{7}, \boxed{8}, \boxed{9}$

Solution

- For each digit, find the number in the range that uses the most copies of that digit.
- For each digit from 1 to 9:
  - Find the longest *repdigit* (number made up only of that digit) not exceeding n.
  - Add its length to the result.
- For the digit 0:
  - We always need at least one sign for the end of the countdown.
  - The smallest number to use two signs is 100, the smallest to use three signs is 1000, ...
  - Find the largest power of 10 not exceeding n and add the appropriate number of 0 signs.

M: Mischievous Math

Problem Author: Paul Wild



M: Mischievous Math

Problem Author: Paul Wild

Problem

Given an integer d, find integers a, b and c such that it is impossible to write d as the result of a mathematical expression involving a, b and c and using the four basic operations $+$, $-$, $\times$, and $\div$. All numbers must be distinct and from the range $\{1, \ldots, 100\}$.

M: Mischievous Math

Problem Author: Paul Wild

Problem

Given an integer d, find integers a, b and c such that it is impossible to write d as the result of a mathematical expression involving a, b and c and using the four basic operations $+$, $-$, $\times$, and $\div$. All numbers must be distinct and from the range $\{1, \ldots, 100\}$.

Solution

- If we put $a = 1$, $b = 2$ and $c = 3$, then the largest representable number is $9 = (1 + 2) \times 3$.

M: Mischievous Math

Problem Author: Paul Wild

Problem

Given an integer d, find integers a, b and c such that it is impossible to write d as the result of a mathematical expression involving a, b and c and using the four basic operations $+$, $-$, $\times$, and $\div$. All numbers must be distinct and from the range $\{1, \ldots, 100\}$.

Solution

- If we put $a = 1$, $b = 2$ and $c = 3$, then the largest representable number is $9 = (1 + 2) \times 3$.
- Therefore, if $d \geq 10$ we simply output `1 2 3`.

M: Mischievous Math

Problem Author: Paul Wild

Problem

Given an integer d, find integers a, b and c such that it is impossible to write d as the result of a mathematical expression involving a, b and c and using the four basic operations $+$, $-$, $\times$, and $\div$. All numbers must be distinct and from the range $\{1, \ldots, 100\}$.

Solution

- If we put $a = 1$, $b = 2$ and $c = 3$, then the largest representable number is $9 = (1 + 2) \times 3$.
- Therefore, if $d \geq 10$ we simply output `1 2 3`.
- Similarly, we can find a triple of numbers that works for all $d \leq 9$, for example:

`79 90 100`   `13 57 100`   `10 21 43`

M: Mischievous Math

Problem Author: Paul Wild

Problem

Given an integer d, find integers a, b and c such that it is impossible to write d as the result of a mathematical expression involving a, b and c and using the four basic operations $+$, $-$, $\times$, and $\div$. All numbers must be distinct and from the range $\{1, \ldots, 100\}$.

Solution

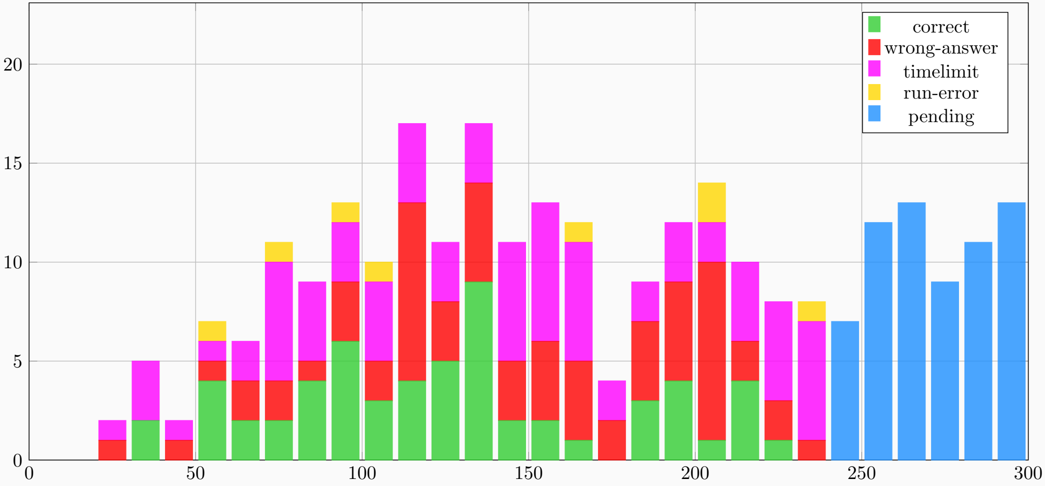
- If we put $a = 1$, $b = 2$ and $c = 3$, then the largest representable number is $9 = (1 + 2) \times 3$.
- Therefore, if $d \geq 10$ we simply output `1 2 3`.
- Similarly, we can find a triple of numbers that works for all $d \leq 9$, for example:

  `79 90 100`   `13 57 100`   `10 21 43`

- In total, exactly 29 486 out of the $\binom{100}{3} = 161\,700$ possible triples avoid all $d \leq 9$.

L: Loop Invariant

Problem Author: Michael Zündorf



L: Loop Invariant

Problem Author: Michael Zündorf

Problem

Given a valid balanced parentheses sequence (BPS) s, find a rotation of s that is a different valid BPS (if it exists).

Example

$()(())() \rightarrow (())()()$

L: Loop Invariant

Problem Author: Michael Zündorf

Problem

Given a valid balanced parentheses sequence (BPS) s, find a rotation of s that is a different valid BPS (if it exists).

Example

()(())() $\rightarrow$ (())()()

Solution

- Insight 1: We need to break at a point where the part on the left is a valid BPS. Otherwise, there will be an unmatched closing parenthesis to the right of the break: ()(())() $\rightarrow$ ())()()(

L: Loop Invariant

Problem Author: Michael Zündorf

Problem

Given a valid balanced parentheses sequence (BPS) s, find a rotation of s that is a different valid BPS (if it exists).

Example

()(())() → (())()()

Solution

- Insight 1: We need to break at a point where the part on the left is a valid BPS. Otherwise, there will be an unmatched closing parenthesis to the right of the break: ()(())() → ())()()(
- Split s at each possible break point into *words*: ()(())() → () (()) ()

L: Loop Invariant

Problem Author: Michael Zündorf

Problem

Given a valid balanced parentheses sequence (BPS) s, find a rotation of s that is a different valid BPS (if it exists).

Example

`()(())()` → `(())()()`

Solution

- Insight 1: We need to break at a point where the part on the left is a valid BPS. Otherwise, there will be an unmatched closing parenthesis to the right of the break: `()(())()` → `())()()(`
- Split s at each possible break point into *words*: `()(())()` → `()` `(())` `()`
- Insight 2: There is a solution if and only if there are (at least) two different words.

L: Loop Invariant

Problem Author: Michael Zündorf

Problem

Given a valid balanced parentheses sequence (BPS) s, find a rotation of s that is a different valid BPS (if it exists).

Example

`()(())()` $\rightarrow$ `(())()()`

Solution

- Insight 1: We need to break at a point where the part on the left is a valid BPS. Otherwise, there will be an unmatched closing parenthesis to the right of the break: `()(())()` $\rightarrow$ `())()()(`
- Split s at each possible break point into *words*: `()(())()` $\rightarrow$ `()` `(())` `()`
- Insight 2: There is a solution if and only if there are (at least) two different words.
- Thus, breaking at the earliest possible point gives a valid solution (if there exists one).

L: Loop Invariant

Problem Author: Michael Zündorf

Problem

Given a valid balanced parentheses sequence (BPS) s, find a rotation of s that is a different valid BPS (if it exists).

Example

`()(())()` $\rightarrow$ `(())()()`

Solution

- Insight 1: We need to break at a point where the part on the left is a valid BPS. Otherwise, there will be an unmatched closing parenthesis to the right of the break: `()(())()` $\rightarrow$ `())()()(`
- Split s at each possible break point into *words*: `()(())()` $\rightarrow$ `()` `(())` `()`
- Insight 2: There is a solution if and only if there are (at least) two different words.
- Thus, breaking at the earliest possible point gives a valid solution (if there exists one).
- Break at the earliest possible point and check if the result s' is different from s. Output s' if yes and "`unique`" otherwise.

L: Loop Invariant

Problem Author: Michael Zündorf

Problem

Given a valid balanced parentheses sequence (BPS) s, find a rotation of s that is a different valid BPS (if it exists).

Example

()(())() → (())()()

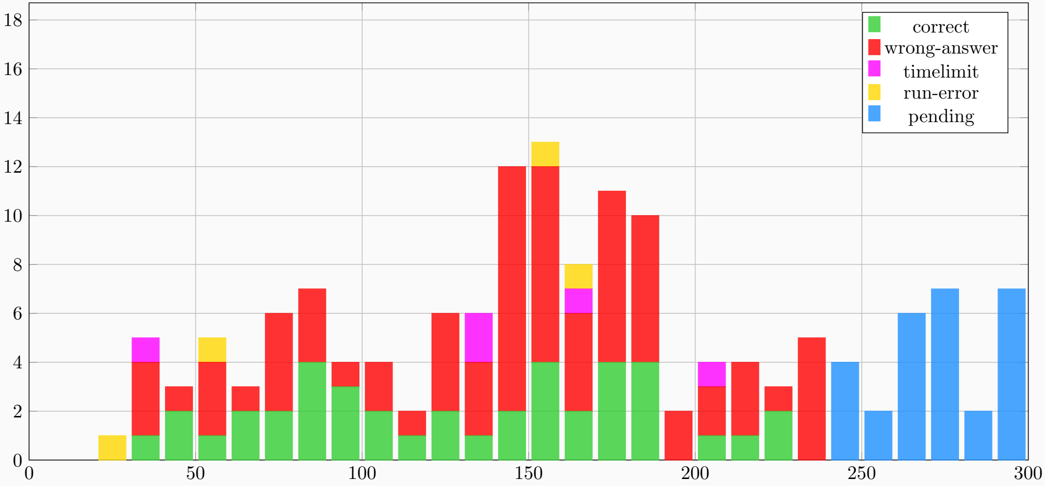
Solution

- Insight 1: We need to break at a point where the part on the left is a valid BPS. Otherwise, there will be an unmatched closing parenthesis to the right of the break: ()(())() → ())()()(
- Split s at each possible break point into *words*: ()(())() → () (()) ()
- Insight 2: There is a solution if and only if there are (at least) two different words.
- Thus, breaking at the earliest possible point gives a valid solution (if there exists one).
- Break at the earliest possible point and check if the result s' is different from s. Output s' if yes and "`unique`" otherwise.

Time complexity: $\mathcal{O}(|s|)$

D: DnD Dice

Problem Author: Paul Wild



D: DnD Dice

Problem Author: Paul Wild

Problem

Given some dice with various numbers of sides, output the possible sums of dice rolls in order of probability.

D: DnD Dice

Problem Author: Paul Wild

Problem

Given some dice with various numbers of sides, output the possible sums of dice rolls in order of probability.

Solution

- Use dynamic programming, adding the dice one by one to the current probability distribution.
- If the current distribution is π and we add a k-sided die, then the new distribution π' is

$$\pi'(n) = \frac{1}{k}(\pi(n-1) + \cdots + \pi(n-k)).$$

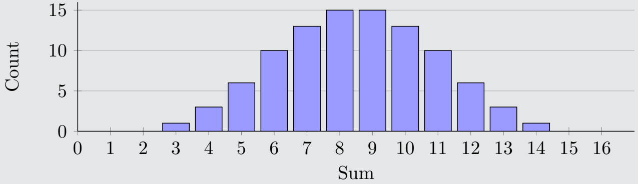
- *Pitfall*: beware of integer overflow when using counts instead of probabilities.

D: DnD Dice

Problem Author: Paul Wild

Easier Solution

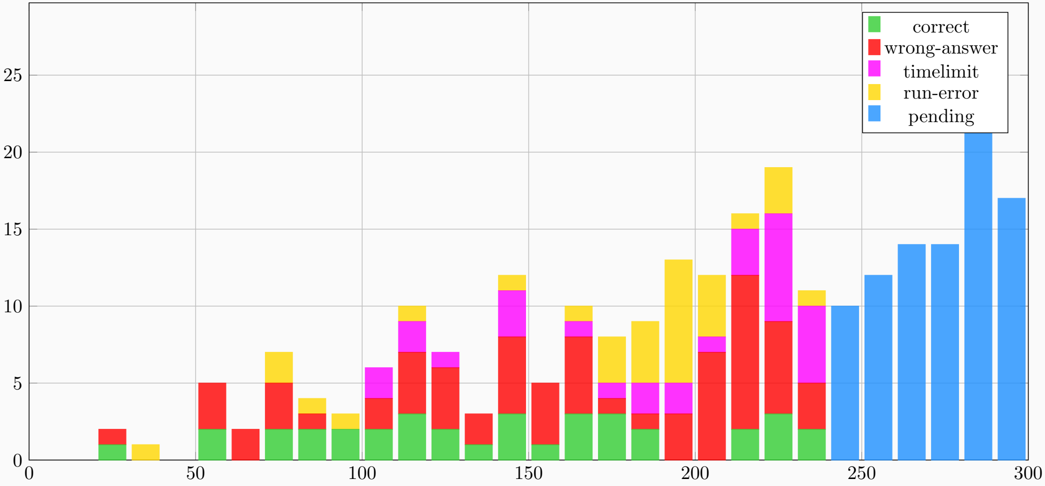
- Notice that the probability distribution is always symmetrical, e.g. for two d4 and one d6:



- This means we can find the final order without computing any probabilities!
- The solution is easiest to construct by starting at the extremes and taking turns moving inwards.

I: Investigating Frog Behaviour on Lily Pad Patterns

Problem Author: Michael Zündorf

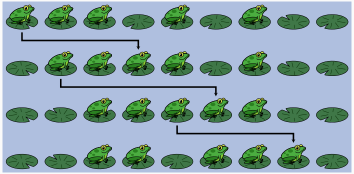


I: Investigating Frog Behaviour on Lily Pad Patterns

Problem Author: Michael Zündorf

Problem

Given at most $2 \cdot 10^5$ frogs with integral positions, and a list of at most 10^6 frog-IDs (events). For each ID in the list, move the corresponding frog to the next free position.

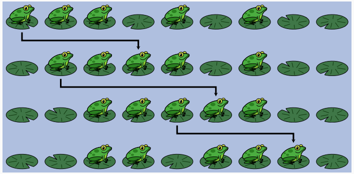


I: Investigating Frog Behaviour on Lily Pad Patterns

Problem Author: Michael Zündorf

Problem

Given at most $2 \cdot 10^5$ frogs with integral positions, and a list of at most 10^6 frog-IDs (events). For each ID in the list, move the corresponding frog to the next free position.



Solution

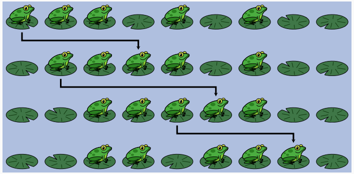
- Store for each position whether it is occupied or not in an array. Only positions up to $1.2 \cdot 10^6$ are relevant.

I: Investigating Frog Behaviour on Lily Pad Patterns

Problem Author: Michael Zündorf

Problem

Given at most $2 \cdot 10^5$ frogs with integral positions, and a list of at most 10^6 frog-IDs (events). For each ID in the list, move the corresponding frog to the next free position.



Solution

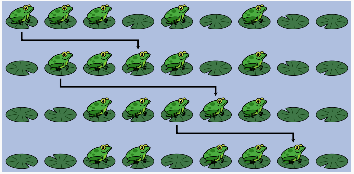
- Store for each position whether it is occupied or not in an array. Only positions up to $1.2 \cdot 10^6$ are relevant.
- Now, simulate the events: For a jump of frog i, currently at position p, scan the "occupied"-array starting at p and seek the next free position.

I: Investigating Frog Behaviour on Lily Pad Patterns

Problem Author: Michael Zündorf

Problem

Given at most $2 \cdot 10^5$ frogs with integral positions, and a list of at most 10^6 frog-IDs (events). For each ID in the list, move the corresponding frog to the next free position.



Solution

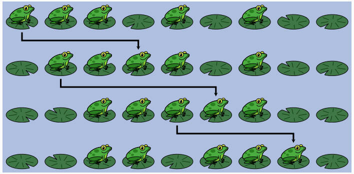
- Store for each position whether it is occupied or not in an array. Only positions up to $1.2 \cdot 10^6$ are relevant.
- Now, simulate the events: For a jump of frog i, currently at position p, scan the “occupied”-array starting at p and seek the next free position.
- Time complexity?

I: Investigating Frog Behaviour on Lily Pad Patterns

Problem Author: Michael Zündorf

Problem

Given at most $2 \cdot 10^5$ frogs with integral positions, and a list of at most 10^6 frog-IDs (events). For each ID in the list, move the corresponding frog to the next free position.



Solution

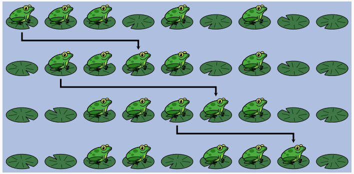
- Store for each position whether it is occupied or not in an array. Only positions up to $1.2 \cdot 10^6$ are relevant.
- Now, simulate the events: For a jump of frog i, currently at position p, scan the "occupied"-array starting at p and seek the next free position.
- Time complexity? $\mathcal{O}(n^2)$ This is too slow (unless very very very optimised...)

I: Investigating Frog Behaviour on Lily Pad Patterns

Problem Author: Michael Zündorf

Problem

Given at most $2 \cdot 10^5$ frogs with integral positions, and a list of at most 10^6 frog-IDs (events). For each ID in the list, move the corresponding frog to the next free position.



Solution

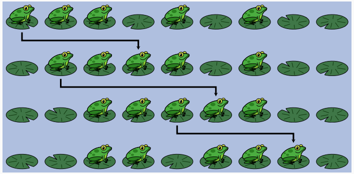
- Maintain the current positions of each frog and an ordered set S of currently free positions.

I: Investigating Frog Behaviour on Lily Pad Patterns

Problem Author: Michael Zündorf

Problem

Given at most $2 \cdot 10^5$ frogs with integral positions, and a list of at most 10^6 frog-IDs (events). For each ID in the list, move the corresponding frog to the next free position.



Solution

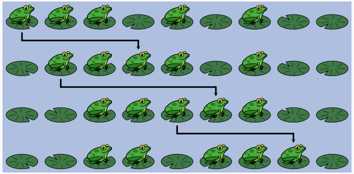
- Maintain the current positions of each frog and an ordered set S of currently free positions.
- Now the events can be simulated quickly. For a jump of frog i, currently at position p:
  - find $\min\{p' \in S : p < p'\}$ in $\mathcal{O}(\log |S|)$ using operations from the standard library.
  - update S and the position of frog i

I: Investigating Frog Behaviour on Lily Pad Patterns

Problem Author: Michael Zündorf

Problem

Given at most $2 \cdot 10^5$ frogs with integral positions, and a list of at most 10^6 frog-IDs (events). For each ID in the list, move the corresponding frog to the next free position.

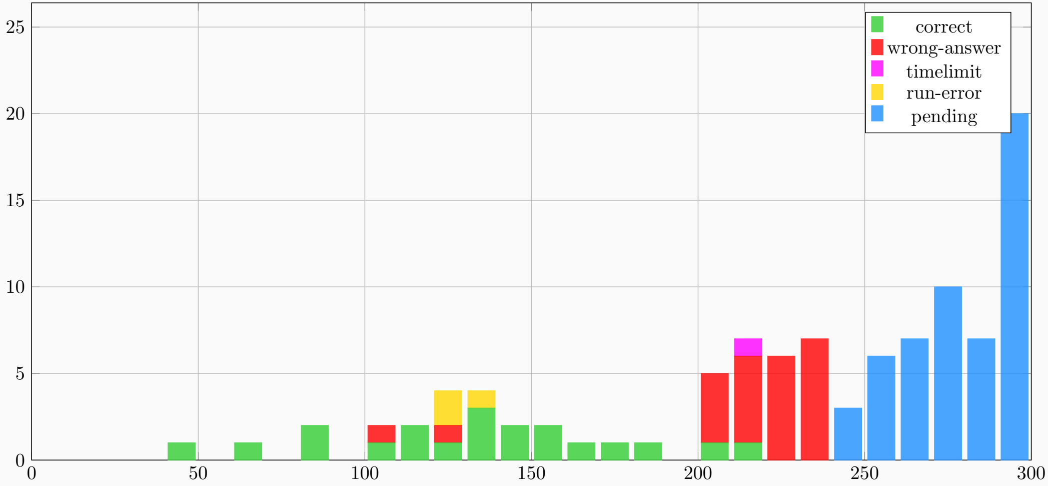


Solution

- Maintain the current positions of each frog and an ordered set S of currently free positions.
- Now the events can be simulated quickly. For a jump of frog i, currently at position p:
  - find $\min\{p' \in S : p < p'\}$ in $\mathcal{O}(\log |S|)$ using operations from the standard library.
  - update S and the position of frog i
- Total time complexity: $\mathcal{O}(n \log |S|)$

C: Cosmic Commute

Problem Author: Wendy Yi



C: Cosmic Commute

Problem Author: Wendy Yi

Problem

Given an undirected, unweighted graph with $n \leq 2 \cdot 10^5$ vertices and a set W of k wormholes, what is the length of the shortest expected path from s to t?

- By entering a wormhole, you are teleported to another wormhole chosen uniformly at random.
- You can use teleportation at most once.

C: Cosmic Commute

Problem Author: Wendy Yi

Problem

Given an undirected, unweighted graph with $n \leq 2 \cdot 10^5$ vertices and a set W of k wormholes, what is the length of the shortest expected path from s to t?

- By entering a wormhole, you are teleported to another wormhole chosen uniformly at random.
- You can use teleportation at most once.

Solution

- If you enter wormhole w, the expected distance from s to t is

$$\mathrm{dist}_w = \mathrm{dist}(s, w) + \frac{1}{k-1} \sum_{w' \in W \setminus \{w\}} \mathrm{dist}(w', t)$$

C: Cosmic Commute

Problem Author: Wendy Yi

Problem

Given an undirected, unweighted graph with $n \leq 2 \cdot 10^5$ vertices and a set W of k wormholes, what is the length of the shortest expected path from s to t?

- By entering a wormhole, you are teleported to another wormhole chosen uniformly at random.
- You can use teleportation at most once.

Solution

- If you enter wormhole w, the expected distance from s to t is

$$\mathrm{dist}_w = \mathrm{dist}(s, w) + \frac{1}{k-1} \sum_{w' \in W \setminus \{w\}} \mathrm{dist}(w', t) = \mathrm{dist}(s, w) + \frac{1}{k-1}(S - \mathrm{dist}(w, t))$$

with $S = \sum_{w' \in W} \mathrm{dist}(w', t)$

C: Cosmic Commute

Problem Author: Wendy Yi

Problem

Given an undirected, unweighted graph with $n \leq 2 \cdot 10^5$ vertices and a set W of k wormholes, what is the length of the shortest expected path from s to t?

- By entering a wormhole, you are teleported to another wormhole chosen uniformly at random.
- You can use teleportation at most once.

Solution

- If you enter wormhole w, the expected distance from s to t is

$$\mathrm{dist}_w = \mathrm{dist}(s, w) + \frac{1}{k-1} \sum_{w' \in W \setminus \{w\}} \mathrm{dist}(w', t) = \mathrm{dist}(s, w) + \frac{1}{k-1}(S - \mathrm{dist}(w, t))$$

  with $S = \sum_{w' \in W} \mathrm{dist}(w', t)$
- Compute $\mathrm{dist}(s, w)$ and $\mathrm{dist}(w, t)$ for each wormhole w and sum S using two BFS from s and t.

C: Cosmic Commute

Problem Author: Wendy Yi

Problem

Given an undirected, unweighted graph with $n \leq 2 \cdot 10^5$ vertices and a set W of k wormholes, what is the length of the shortest expected path from s to t?

- By entering a wormhole, you are teleported to another wormhole chosen uniformly at random.
- You can use teleportation at most once.

Solution

- If you enter wormhole w, the expected distance from s to t is

$$\mathrm{dist}_w = \mathrm{dist}(s,w) + \frac{1}{k-1} \sum_{w' \in W \setminus \{w\}} \mathrm{dist}(w',t) = \mathrm{dist}(s,w) + \frac{1}{k-1}(S - \mathrm{dist}(w,t))$$

  with $S = \sum_{w' \in W} \mathrm{dist}(w',t)$
- Compute $\mathrm{dist}(s,w)$ and $\mathrm{dist}(w,t)$ for each wormhole w and sum S using two BFS from s and t.
- Determine the wormhole you should enter to minimize the expected distance.

C: Cosmic Commute

Problem Author: Wendy Yi

Problem

Given an undirected, unweighted graph with $n \leq 2 \cdot 10^5$ vertices and a set W of k wormholes, what is the length of the shortest expected path from s to t?

- By entering a wormhole, you are teleported to another wormhole chosen uniformly at random.
- You can use teleportation at most once.

Solution

- It might be better to directly go from s to t without using any wormhole.

C: Cosmic Commute

Problem Author: Wendy Yi

Problem

Given an undirected, unweighted graph with $n \leq 2 \cdot 10^5$ vertices and a set W of k wormholes, what is the length of the shortest expected path from s to t?

- By entering a wormhole, you are teleported to another wormhole chosen uniformly at random.
- You can use teleportation at most once.

Solution

- It might be better to directly go from s to t without using any wormhole.
- Output the minimum $\min\{\mathrm{dist}(s, t), \min_{w \in W}\{\mathrm{dist}_w\}\}$.

C: Cosmic Commute

Problem Author: Wendy Yi

Problem

Given an undirected, unweighted graph with $n \leq 2 \cdot 10^5$ vertices and a set W of k wormholes, what is the length of the shortest expected path from s to t?

- By entering a wormhole, you are teleported to another wormhole chosen uniformly at random.
- You can use teleportation at most once.

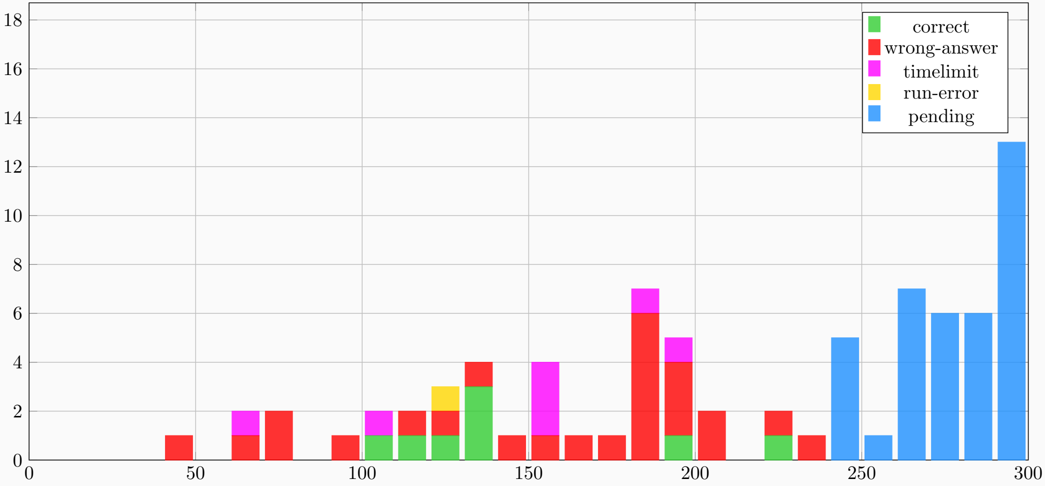
Solution

- It might be better to directly go from s to t without using any wormhole.
- Output the minimum $\min\{\mathrm{dist}(s,t), \min_{w \in W}\{\mathrm{dist}_w\}\}$.

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

B: Balloon Darts

Problem Author: Paul Jungeblut



B: Balloon Darts

Problem Author: Paul Jungeblut

Problem

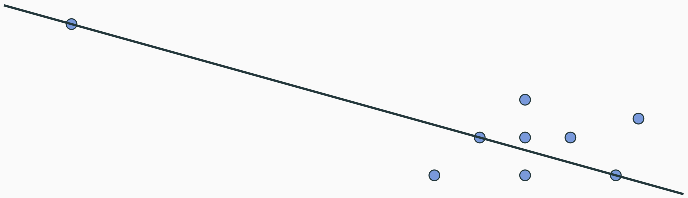
Given n points in the plane, determine if $k = 3$ lines are sufficient to cover all points.

B: Balloon Darts

Problem Author: Paul Jungeblut

Problem

Given n points in the plane, determine if $k = 3$ lines are sufficient to cover all points.

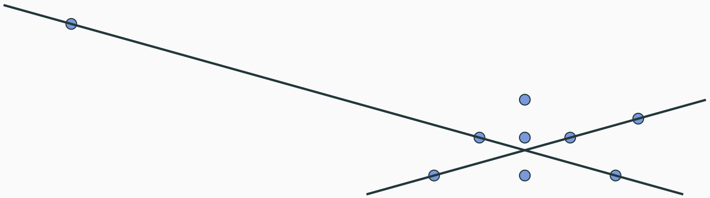


B: Balloon Darts

Problem Author: Paul Jungeblut

Problem

Given n points in the plane, determine if $k = 3$ lines are sufficient to cover all points.

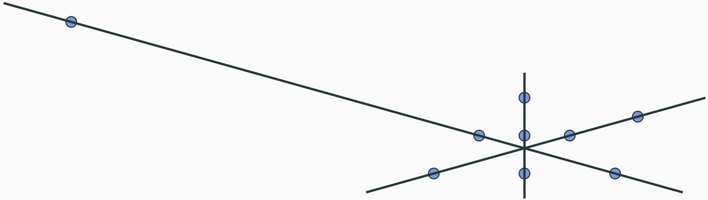


B: Balloon Darts

Problem Author: Paul Jungeblut

Problem

Given n points in the plane, determine if $k = 3$ lines are sufficient to cover all points.

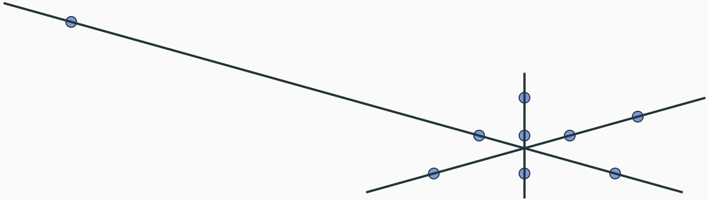


B: Balloon Darts

Problem Author: Paul Jungeblut

Problem

Given n points in the plane, determine if $k = 3$ lines are sufficient to cover all points.



Solution

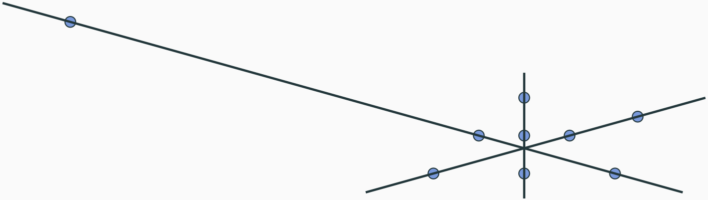
- If we have at most k points the answer is obviously Yes.
- If we select $k + 1$ points, one line has to go through two of those points.

B: Balloon Darts

Problem Author: Paul Jungeblut

Problem

Given n points in the plane, determine if $k = 3$ lines are sufficient to cover all points.



Solution

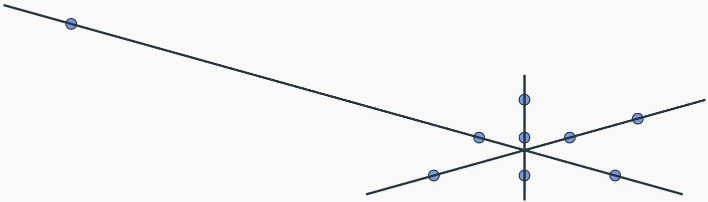
- If we have at most k points the answer is obviously Yes.
- If we select $k + 1$ points, one line has to go through two of those points.
- Given k and $n > k$ points solve the problem recursively:
  - Select $k + 1$ points and try all lines through two points.
  - For each line remove all covered points.
  - Check recursively with $k - 1$ and the remaining points.

B: Balloon Darts

Problem Author: Paul Jungeblut

Problem

Given n points in the plane, determine if $k = 3$ lines are sufficient to cover all points.



Solution

- If we have at most k points the answer is obviously Yes.
- If we select $k + 1$ points, one line has to go through two of those points.
- Given k and $n > k$ points solve the problem recursively:
  - Select $k + 1$ points and try all lines through two points.
  - For each line remove all covered points.
  - Check recursively with $k - 1$ and the remaining points.
- Time complexity for $(k = 3)$: $n \cdot \prod_{i=1}^{k} \binom{i+1}{2} = 18 \cdot n$

More Observations

- There must be a line which covers at least a third of all points.

More Observations

- There must be a line which covers at least a third of all points.
- There must be a line which covers at least half of all remaining points.

B: Balloon Darts

Problem Author: Paul Jungeblut

More Observations

- There must be a line which covers at least a third of all points.
- There must be a line which covers at least half of all remaining points.
- There must be a line which covers all remaining points.

More Observations

- There must be a line which covers at least a third of all points.
- There must be a line which covers at least half of all remaining points.
- There must be a line which covers all remaining points.

Solution 2

- Recursively select a random line through two points.
- At step k check if the chosen line covers $\frac{1}{k}$ of all points.

B: Balloon Darts

Problem Author: Paul Jungeblut

More Observations

- There must be a line which covers at least a third of all points.
- There must be a line which covers at least half of all remaining points.
- There must be a line which covers all remaining points.

Solution 2

- Recursively select a random line through two points.
- At step k check if the chosen line covers $\frac{1}{k}$ of all points.

  Yes: recursively continue with $k-1$ and the remaining points.

B: Balloon Darts

Problem Author: Paul Jungeblut

More Observations

- There must be a line which covers at least a third of all points.
- There must be a line which covers at least half of all remaining points.
- There must be a line which covers all remaining points.

Solution 2

- Recursively select a random line through two points.
- At step k check if the chosen line covers $\frac{1}{k}$ of all points.

  Yes: recursively continue with $k - 1$ and the remaining points.

  No: try another line or abort after sufficient many tries $(\sim 5 \cdot k)$.

B: Balloon Darts

Problem Author: Paul Jungeblut

More Observations

- There must be a line which covers at least a third of all points.
- There must be a line which covers at least half of all remaining points.
- There must be a line which covers all remaining points.

Solution 2

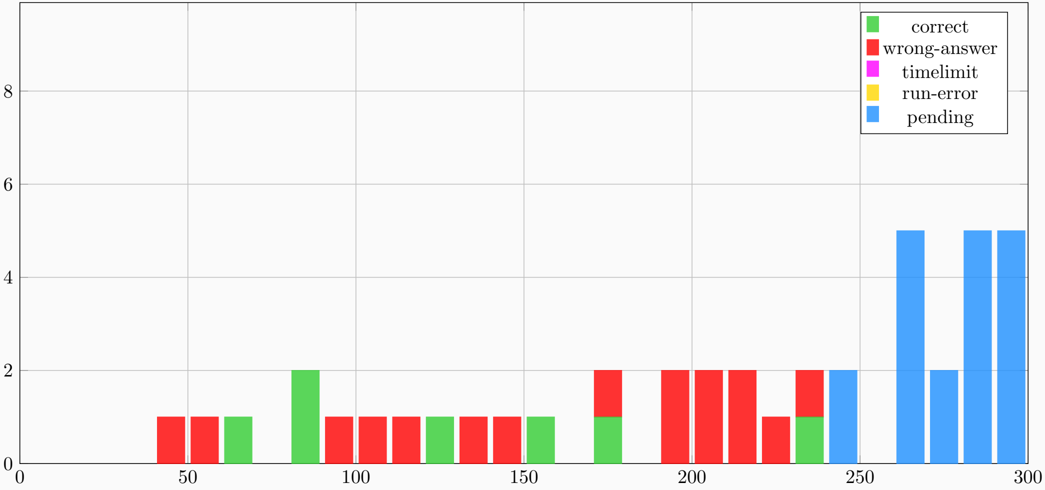
- Recursively select a random line through two points.
- At step k check if the chosen line covers $\frac{1}{k}$ of all points.

  Yes: recursively continue with $k - 1$ and the remaining points.

  No: try another line or abort after sufficient many tries $(\sim 5 \cdot k)$.
- Time complexity for $(k = 3)$: $n \cdot 5 \cdot k! = 30 \cdot n$

F: Freestyle Masonry

Problem Author: Michael Zündorf

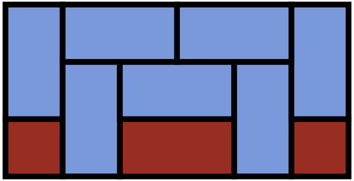


F: Freestyle Masonry

Problem Author: Michael Zündorf

Problem

Given the height field representing a wall, decide if you can add 2×1 blocks to create a wall of width exactly w and height exactly h.

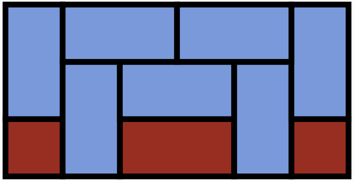


F: Freestyle Masonry

Problem Author: Michael Zündorf

Problem

Given the height field representing a wall, decide if you can add 2×1 blocks to create a wall of width exactly w and height exactly h.



Solution

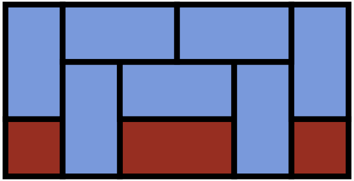
- Given a subgraph of a $w \times h$ grid graph, decide if it has a perfect matching
- Since the graph is a grid i.e. bipartite this can be done in $w \cdot h \cdot \sqrt{w \cdot h}$

F: Freestyle Masonry

Problem Author: Michael Zündorf

Problem

Given the height field representing a wall, decide if you can add 2×1 blocks to create a wall of width exactly w and height exactly h.



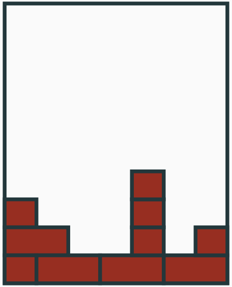
Solution?

- Given a subgraph of a $w \times h$ grid graph, decide if it has a perfect matching
- Since the graph is a grid i.e. bipartite this can be done in $w \cdot h \cdot \sqrt{w \cdot h}$

$\Rightarrow$ This is much too slow

F: Freestyle Masonry

Problem Author: Michael Zündorf

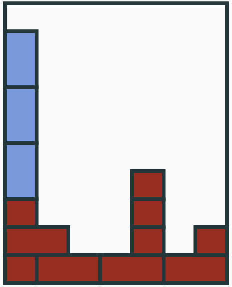


Solution

- There is a greedy strategy which finds a perfect matching if one exists

F: Freestyle Masonry

Problem Author: Michael Zündorf

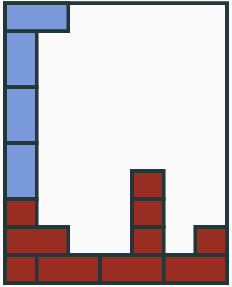


Solution

- There is a greedy strategy which finds a perfect matching if one exists
  - Go from left to right
  - Place as many 1×2 blocks from the bottom to the top as fit in the current column
  - If needed place 2×1 blocks on the top which go into the next column

F: Freestyle Masonry

Problem Author: Michael Zündorf

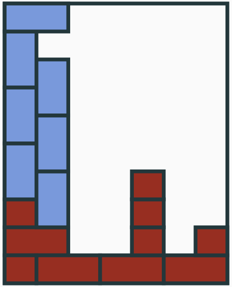


Solution

- There is a greedy strategy which finds a perfect matching if one exists
  - Go from left to right
  - Place as many 1×2 blocks from the bottom to the top as fit in the current column
  - If needed place 2×1 blocks on the top which go into the next column

F: Freestyle Masonry

Problem Author: Michael Zündorf

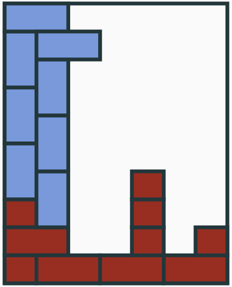


Solution

- There is a greedy strategy which finds a perfect matching if one exists
  - Go from left to right
  - Place as many 1×2 blocks from the bottom to the top as fit in the current column
  - If needed place 2×1 blocks on the top which go into the next column

F: Freestyle Masonry

Problem Author: Michael Zündorf

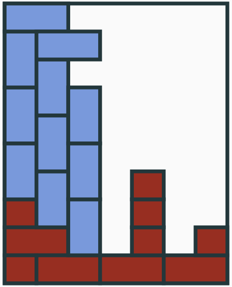


Solution

- There is a greedy strategy which finds a perfect matching if one exists
  - Go from left to right
  - Place as many 1×2 blocks from the bottom to the top as fit in the current column
  - If needed place 2×1 blocks on the top which go into the next column

F: Freestyle Masonry

Problem Author: Michael Zündorf

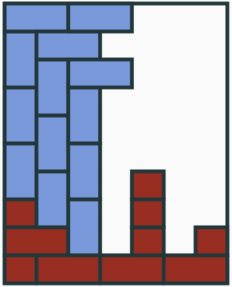


Solution

- There is a greedy strategy which finds a perfect matching if one exists
  - Go from left to right
  - Place as many 1×2 blocks from the bottom to the top as fit in the current column
  - If needed place 2×1 blocks on the top which go into the next column

F: Freestyle Masonry

Problem Author: Michael Zündorf

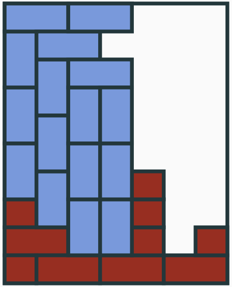


Solution

- There is a greedy strategy which finds a perfect matching if one exists
  - Go from left to right
  - Place as many 1×2 blocks from the bottom to the top as fit in the current column
  - If needed place 2×1 blocks on the top which go into the next column

F: Freestyle Masonry

Problem Author: Michael Zündorf

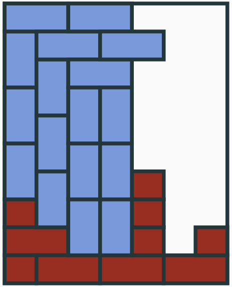


Solution

- There is a greedy strategy which finds a perfect matching if one exists
  - Go from left to right
  - Place as many 1×2 blocks from the bottom to the top as fit in the current column
  - If needed place 2×1 blocks on the top which go into the next column

F: Freestyle Masonry

Problem Author: Michael Zündorf

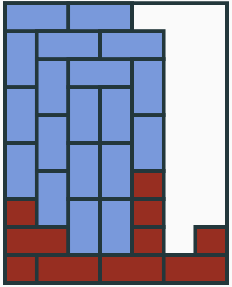


Solution

- There is a greedy strategy which finds a perfect matching if one exists
  - Go from left to right
  - Place as many 1×2 blocks from the bottom to the top as fit in the current column
  - If needed place 2×1 blocks on the top which go into the next column

F: Freestyle Masonry

Problem Author: Michael Zündorf

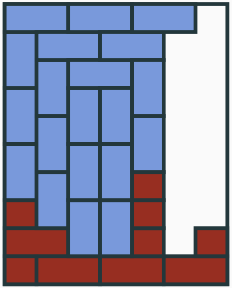


Solution

- There is a greedy strategy which finds a perfect matching if one exists
  - Go from left to right
  - Place as many 1×2 blocks from the bottom to the top as fit in the current column
  - If needed place 2×1 blocks on the top which go into the next column

F: Freestyle Masonry

Problem Author: Michael Zündorf

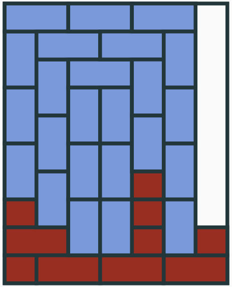


Solution

- There is a greedy strategy which finds a perfect matching if one exists
  - Go from left to right
  - Place as many 1×2 blocks from the bottom to the top as fit in the current column
  - If needed place 2×1 blocks on the top which go into the next column

F: Freestyle Masonry

Problem Author: Michael Zündorf

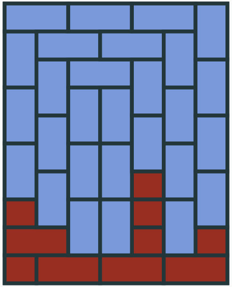


Solution

- There is a greedy strategy which finds a perfect matching if one exists
  - Go from left to right
  - Place as many 1×2 blocks from the bottom to the top as fit in the current column
  - If needed place 2×1 blocks on the top which go into the next column

F: Freestyle Masonry

Problem Author: Michael Zündorf

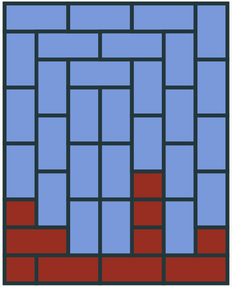


Solution

- There is a greedy strategy which finds a perfect matching if one exists
  - Go from left to right
  - Place as many 1×2 blocks from the bottom to the top as fit in the current column
  - If needed place 2×1 blocks on the top which go into the next column

F: Freestyle Masonry

Problem Author: Michael Zündorf

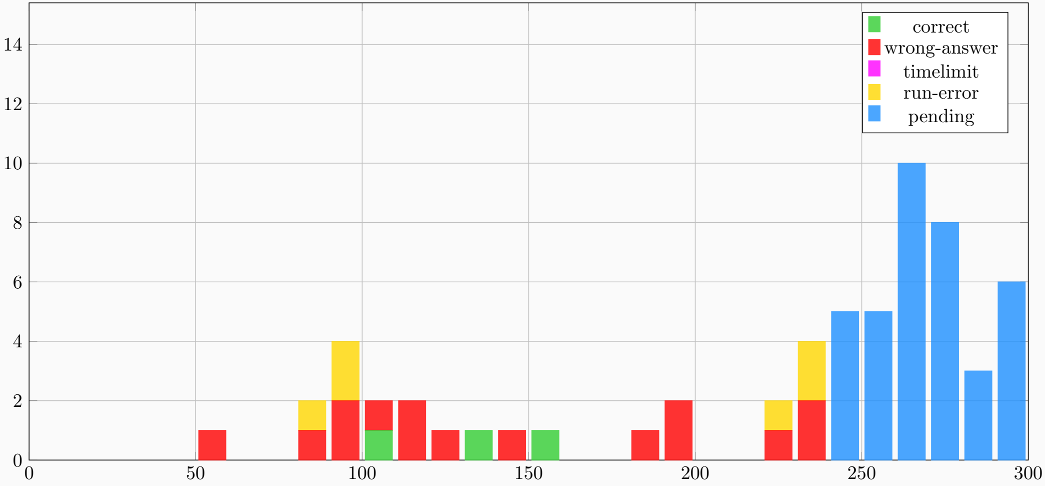


Solution

- There is a greedy strategy which finds a perfect matching if one exists
  - Go from left to right
  - Place as many 1×2 blocks from the bottom to the top as fit in the current column
  - If needed place 2×1 blocks on the top which go into the next column
- To simulate this efficiently, only store the height of the lowest brick coming from the left
- This value either increases or decreases by 1 if we go to the next column

K: Kaldorian Knights

Problem Author: Marcel Wienöbst



K: Kaldorian Knights

Problem Author: Marcel Wienöbst

Problem

Given $a_1, \ldots, a_k$, compute how many permutations of $(1, \ldots, n)$ do not have $1, 2, \ldots, \sum_{i=1}^{l} a_i$ (in some order) in the first $\sum_{i=1}^{l} a_i$ places (for some $l = 1, \ldots k$).

K: Kaldorian Knights

Problem Author: Marcel Wienöbst

Problem

Given $a_1, \ldots, a_k$, compute how many permutations of $(1, \ldots, n)$ do not have $1, 2, \ldots, \sum_{i=1}^{l} a_i$ (in some order) in the first $\sum_{i=1}^{l} a_i$ places (for some $l = 1, \ldots k$).

Solution

- Denote the number of such permutations by $p(n, k)$ and let $A(i) = \sum_{j=1}^{i} a_j$.

K: Kaldorian Knights

Problem Author: Marcel Wienöbst

Problem

Given $a_1, \dots, a_k$, compute how many permutations of $(1, \dots, n)$ do not have $1, 2, \dots, \sum_{i=1}^{l} a_i$ (in some order) in the first $\sum_{i=1}^{l} a_i$ places (for some $l = 1, \dots k$).

Solution

- Denote the number of such permutations by $p(n, k)$ and let $A(i) = \sum_{j=1}^{i} a_j$.
- Following the definition, we can count $p(n, k)$ as the number of *all* permutations minus the forbidden ones.

K: Kaldorian Knights

Problem Author: Marcel Wienöbst

Problem

Given $a_1, \dots, a_k$, compute how many permutations of $(1, \dots, n)$ do not have $1, 2, \dots, \sum_{i=1}^{l} a_i$ (in some order) in the first $\sum_{i=1}^{l} a_i$ places (for some $l = 1, \dots k$).

Solution

- Denote the number of such permutations by $p(n, k)$ and let $A(i) = \sum_{j=1}^{i} a_j$.
- Following the definition, we can count $p(n, k)$ as the number of *all* permutations minus the forbidden ones.
- To avoid subtracting forbidden permutations more than once, we use a recursive formulation:

$$p(n, k) = n! - \sum_{i=1}^{k} (n - A[i])! \times p(A[i], i - 1)$$

K: Kaldorian Knights

Problem Author: Marcel Wienöbst

Problem

Given $a_1, \ldots, a_k$, compute how many permutations of $(1, \ldots, n)$ do not have $1, 2, \ldots, \sum_{i=1}^{l} a_i$ (in some order) in the first $\sum_{i=1}^{l} a_i$ places (for some $l = 1, \ldots k$).

Solution

- Denote the number of such permutations by $p(n, k)$ and let $A(i) = \sum_{j=1}^{i} a_j$.
- Following the definition, we can count $p(n, k)$ as the number of *all* permutations minus the forbidden ones.
- To avoid subtracting forbidden permutations more than once, we use a recursive formulation:

$$p(n, k) = n! - \sum_{i=1}^{k} (n - A[i])! \times p(A[i], i - 1)$$

- $p(A[i], i - 1)$ counts the forbidden prefixes of length $A[i]$, which do not themselves contain a shorter forbidden prefix.

K: Kaldorian Knights

Problem Author: Marcel Wienöbst

Problem

Given $a_1, \ldots, a_k$, compute how many permutations of $(1, \ldots, n)$ do not have $1, 2, \ldots, \sum_{i=1}^{l} a_i$ (in some order) in the first $\sum_{i=1}^{l} a_i$ places (for some $l = 1, \ldots k$).

Solution

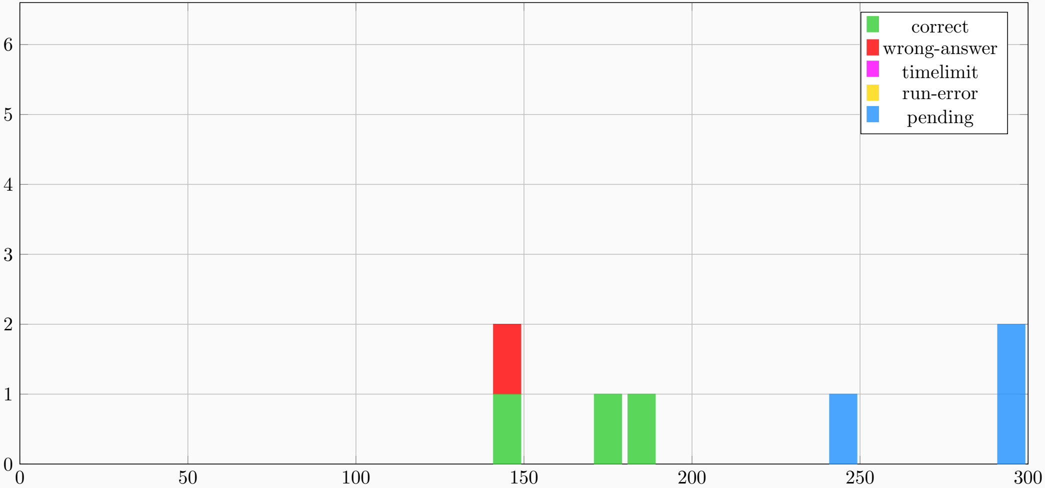
- Denote the number of such permutations by $p(n, k)$ and let $A(i) = \sum_{j=1}^{i} a_j$.
- Following the definition, we can count $p(n, k)$ as the number of *all* permutations minus the forbidden ones.
- To avoid subtracting forbidden permutations more than once, we use a recursive formulation:

$$p(n, k) = n! - \sum_{i=1}^{k} (n - A[i])! \times p(A[i], i - 1)$$

- $p(A[i], i - 1)$ counts the forbidden prefixes of length $A[i]$, which do not themselves contain a shorter forbidden prefix.
- The recursion can be evaluated using dynamic programming in time $\mathcal{O}(k^2)$.

J: Japanese Lottery

Problem Author: Michael Zündorf

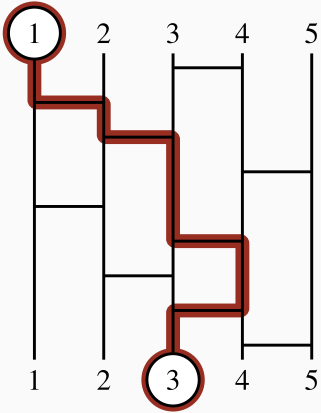


J: Japanese Lottery

Problem Author: Michael Zündorf

Problem

Given a game of Amida-kuji, i.e. k legs and some horizontal bars which change over time, decide how many horizontal bars you need to remove to connect the ith start to the ith end.



J: Japanese Lottery

Problem Author: Michael Zündorf

Solution

- The game state can be represented by a permutation.
- Adding/removing a bar always changes the number of cycles in the permutation by 1.
- We want to build the identity permutation, which has k cycles.

J: Japanese Lottery

Problem Author: Michael Zündorf

Solution

- The game state can be represented by a permutation.
- Adding/removing a bar always changes the number of cycles in the permutation by 1.
- We want to build the identity permutation, which has k cycles.
- There is always a bar whose addition/removal increases the number of cycles.

$\Rightarrow$ The answer is k minus the number of cycles.

J: Japanese Lottery

Problem Author: Michael Zündorf

Solution

- The game state can be represented by a permutation.
- Adding/removing a bar always changes the number of cycles in the permutation by 1.
- We want to build the identity permutation, which has k cycles.
- There is always a bar whose addition/removal increases the number of cycles.

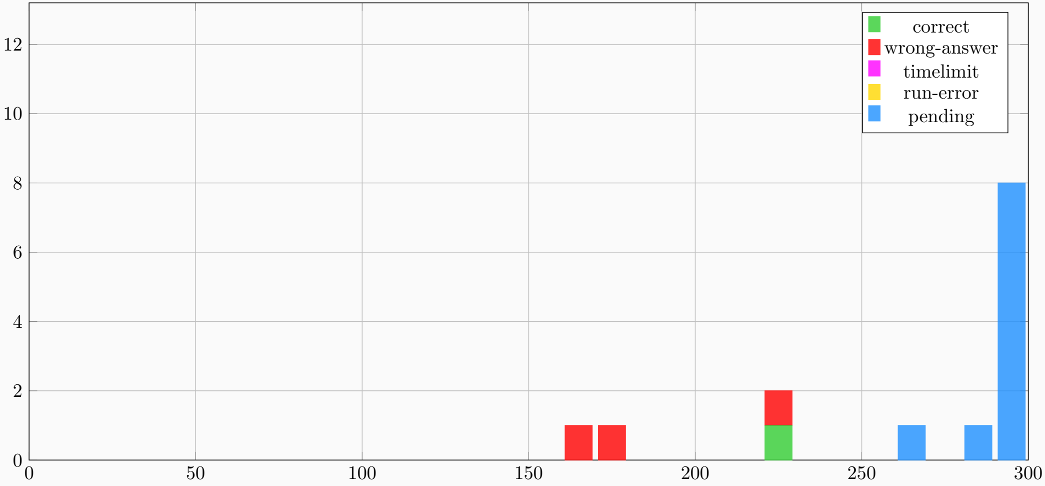
$\Rightarrow$ The answer is k minus the number of cycles.

- Notice that the actual layout of the bars is irrelevant.

$\Rightarrow$ We only need to maintain the current permutation (for example with a Segment Tree).

H: Highway Combinatorics

Problem Author: Michael Zündorf

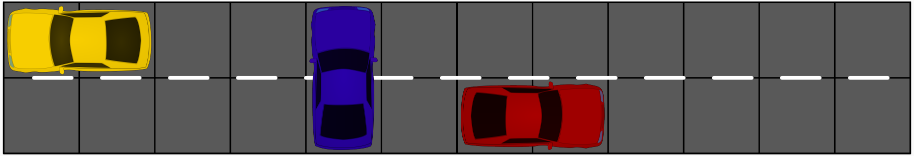


H: Highway Combinatorics

Problem Author: Michael Zündorf

Problem

Find a subgraph of a 2×200 grid which has exactly n perfect matchings modulo $10^9 + 7$.

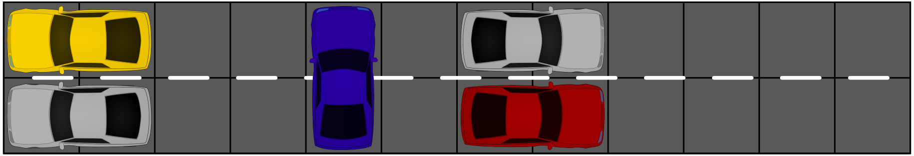


H: Highway Combinatorics

Problem Author: Michael Zündorf

Problem

Find a subgraph of a 2×200 grid which has exactly n perfect matchings modulo $10^9 + 7$.



Observations

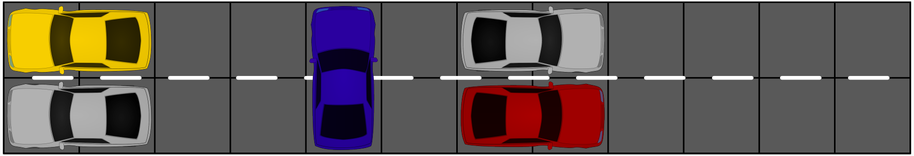
- Some edges are contained in every matching
- The remaining edges are matched in grids of the form $2 \times m_i$
- A $2 \times m$ grid has $\mathit{fibonacci}(m)$ perfect matchings

H: Highway Combinatorics

Problem Author: Michael Zündorf

Problem

Find a subgraph of a 2×200 grid which has exactly n perfect matchings modulo $10^9 + 7$.



Observations

- Some edges are contained in every matching
- The remaining edges are matched in grids of the form $2 \times m_i$
- A $2 \times m$ grid has $\mathit{fibonacci}(m)$ perfect matchings
- This is equivalent to: find k Fibonacci numbers,
  - whose sum is less than 200,
  - whose product is congruent to n modulo $10^9 + 7$.

H: Highway Combinatorics

Problem Author: Michael Zündorf

Problem

- Find a (multi)set of positive Integers S, such that
  - $\sum_{i \in S} i < 200$, and
  - $\prod_{i \in S} fib(i) \equiv n \mod 10^9 + 7$.

Solution

- Meet in the Middle

H: Highway Combinatorics

Problem Author: Michael Zündorf

Problem

- Find a (multi)set of positive Integers S, such that
  - $\sum_{i \in S} i < 200$, and
  - $\prod_{i \in S} fib(i) \equiv n \mod 10^9 + 7$.

Solution

- Meet in the Middle
  - Repeat a times: Randomly pick a multiset S_1 with $\sum_{i \in S_1} i < 100$ and store it indexed by $\prod_{i \in S_1} fib(i)$

H: Highway Combinatorics

Problem Author: Michael Zündorf

Problem

- Find a (multi)set of positive Integers S, such that
  - $\sum_{i \in S} i < 200$, and
  - $\prod_{i \in S} fib(i) \equiv n \mod 10^9 + 7$.

Solution

- Meet in the Middle
  - Repeat a times: Randomly pick a multiset S_1 with $\sum_{i \in S_1} i < 100$ and store it indexed by $\prod_{i \in S_1} fib(i)$
  - Repeat b times: Randomly pick a multiset S_2 with $\sum_{i \in S_2} i < 100$ and check if some S_1 with $\prod_{i \in S_1} fib(i) \equiv n \cdot \left(\prod_{i \in S_2} fib(i)\right)^{-1}$ has been stored
  - If yes, we found a solution because $\prod_{i \in S_1 \cup S_2} fib(i) \equiv n$ and $\sum_{i \in S_1 \cup S_2} i < 200$

H: Highway Combinatorics

Problem Author: Michael Zündorf

Problem

- Find a (multi)set of positive Integers S, such that
  - $\sum_{i \in S} i < 200$, and
  - $\prod_{i \in S} fib(i) \equiv n \mod 10^9 + 7$.

Solution

- Meet in the Middle
  - Repeat a times: Randomly pick a multiset S_1 with $\sum_{i \in S_1} i < 100$ and store it indexed by $\prod_{i \in S_1} fib(i)$
  - Repeat b times: Randomly pick a multiset S_2 with $\sum_{i \in S_2} i < 100$ and check if some S_1 with $\prod_{i \in S_1} fib(i) \equiv n \cdot \left(\prod_{i \in S_2} fib(i)\right)^{-1}$ has been stored
  - If yes, we found a solution because $\prod_{i \in S_1 \cup S_2} fib(i) \equiv n$ and $\sum_{i \in S_1 \cup S_2} i < 200$
- For $a = b = 10^6$ we test (up to) 10^{12} combinations, but there are only $10^9 + 7$ possible outcomes

$\Rightarrow$ We have to be really unlucky to not find a combination for some fixed n

H: Highway Combinatorics

Problem Author: Michael Zündorf

Problem

- Find a (multi)set of positive Integers S, such that
  - $\sum_{i \in S} i < 200$, and
  - $\prod_{i \in S} fib(i) \equiv n \mod 10^9 + 7$.

Solution

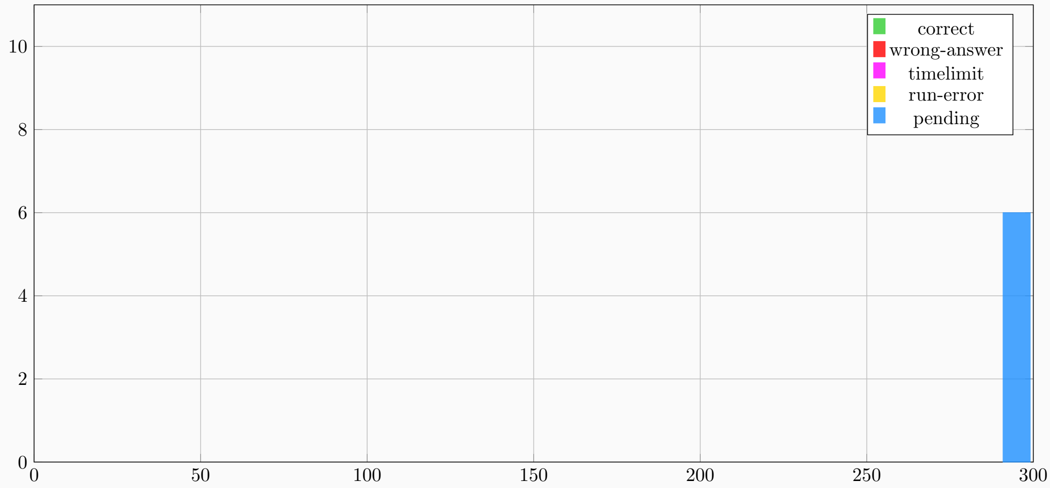
- Meet in the Middle
  - Repeat a times: Randomly pick a multiset S_1 with $\sum_{i \in S_1} i < 100$ and store it indexed by $\prod_{i \in S_1} fib(i)$
  - Repeat b times: Randomly pick a multiset S_2 with $\sum_{i \in S_2} i < 100$ and check if some S_1 with $\prod_{i \in S_1} fib(i) \equiv n \cdot \left(\prod_{i \in S_2} fib(i)\right)^{-1}$ has been stored
  - If yes, we found a solution because $\prod_{i \in S_1 \cup S_2} fib(i) \equiv n$ and $\sum_{i \in S_1 \cup S_2} i < 200$
- For $a = b = 10^6$ we test (up to) 10^{12} combinations, but there are only $10^9 + 7$ possible outcomes

$\Rightarrow$ We have to be really unlucky to not find a combination for some fixed n

- Special case: $n = 0$, find a graph without a perfect matching

A: Adolescent Architecture 2

Problem Author: Paul Wild



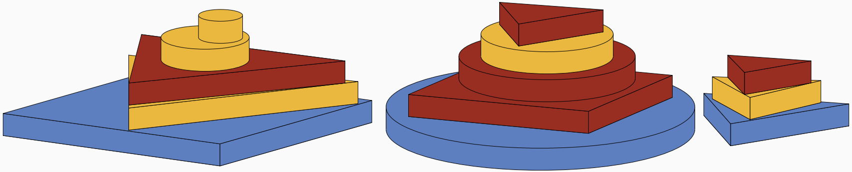
A: Adolescent Architecture 2

Problem Author: Paul Wild

Problem

Find the number of winning moves in a block stacking game:

- There are multiple stacks of blocks.
- Players alternate placing blocks on top of these.
- The first player unable to move loses.
- Each block must fit strictly within the one below it.
- There are three shapes with blocks of any integer size: circles, triangles, squares.



A: Adolescent Architecture 2

Problem Author: Paul Wild

Subproblem

Given two blocks, determine if one of them fits inside the other.

A: Adolescent Architecture 2

Problem Author: Paul Wild

Subproblem

Given two blocks, determine if one of them fits inside the other.

Solution to Subproblem

- Consider each pair of shapes ($\{\triangle, \square, \bigcirc\}$) separately.

A: Adolescent Architecture 2

Problem Author: Paul Wild

Subproblem

Given two blocks, determine if one of them fits inside the other.

Solution to Subproblem

- Consider each pair of shapes ($\{\triangle, \square, \bigcirc\}$) separately.
- For instance, if $\square_m$ is a square with side length m and $\bigcirc_n$ is a circle with radius n, then

$$\square_m \text{ fits inside } \bigcirc_n \iff m < \sqrt{2} \cdot n.$$

A: Adolescent Architecture 2

Problem Author: Paul Wild

Subproblem

Given two blocks, determine if one of them fits inside the other.

Solution to Subproblem

- Consider each pair of shapes ($\{\triangle, \square, \bigcirc\}$) separately.
- For instance, if $\square_m$ is a square with side length m and $\bigcirc_n$ is a circle with radius n, then

$$\square_m \text{ fits inside } \bigcirc_n \iff m < \sqrt{2} \cdot n.$$

- Similarly, for each $S, T \in \{\triangle, \square, \bigcirc\}$, there exists some $\alpha_{S,T}$ such that

$$S_m \text{ fits inside } T_n \iff m < \alpha_{S,T} \cdot n.$$

A: Adolescent Architecture 2

Problem Author: Paul Wild

Subproblem

Given two blocks, determine if one of them fits inside the other.

Solution to Subproblem

- Consider each pair of shapes ($\{\triangle, \square, \bigcirc\}$) separately.
- For instance, if $\square_m$ is a square with side length m and $\bigcirc_n$ is a circle with radius n, then

$$\square_m \text{ fits inside } \bigcirc_n \iff m < \sqrt{2} \cdot n.$$

- Similarly, for each $S, T \in \{\triangle, \square, \bigcirc\}$, there exists some $\alpha_{S,T}$ such that

$$S_m \text{ fits inside } T_n \iff m < \alpha_{S,T} \cdot n.$$

- These numbers can be found using high school geometry.

A: Adolescent Architecture 2

Problem Author: Paul Wild

Subproblem

Given two blocks, determine if one of them fits inside the other.

Solution to Subproblem

- Consider each pair of shapes ($\{\triangle, \square, \bigcirc\}$) separately.
- For instance, if $\square_m$ is a square with side length m and $\bigcirc_n$ is a circle with radius n, then

$$\square_m \text{ fits inside } \bigcirc_n \iff m < \sqrt{2} \cdot n.$$

- Similarly, for each $S, T \in \{\triangle, \square, \bigcirc\}$, there exists some $\alpha_{S,T}$ such that

$$S_m \text{ fits inside } T_n \iff m < \alpha_{S,T} \cdot n.$$

- These numbers can be found using high school geometry.
- *Pitfall*: Near misses are possible, so use extended precision (`long double`, `BigDecimal`, `Decimal`).

A: Adolescent Architecture 2

Problem Author: Paul Wild

Solution

- This is a combinatorial game where for each stack we only care about its topmost block.

A: Adolescent Architecture 2

Problem Author: Paul Wild

Solution

- This is a combinatorial game where for each stack we only care about its topmost block.
- Use the Sprague-Grundy theorem to assign each block B a Grundy value $G(B)$.

A: Adolescent Architecture 2

Problem Author: Paul Wild

Solution

- This is a combinatorial game where for each stack we only care about its topmost block.
- Use the Sprague-Grundy theorem to assign each block B a Grundy value $G(B)$.
- By careful analysis and/or dynamic programming we can find closed forms:

$$G(\triangle_n) = n - 1 \qquad G(\square_n) = \lfloor(\sqrt{6} - \sqrt{2})n\rfloor \qquad G(\bigcirc_n) = \begin{cases} 2, & \text{if } n = 1 \\ \lfloor\sqrt{3}n\rfloor, & \text{otherwise} \end{cases}$$

A: Adolescent Architecture 2

Problem Author: Paul Wild

Solution

- This is a combinatorial game where for each stack we only care about its topmost block.
- Use the Sprague-Grundy theorem to assign each block B a Grundy value $G(B)$.
- By careful analysis and/or dynamic programming we can find closed forms:

$$G(\triangle_n) = n - 1 \qquad G(\square_n) = \lfloor(\sqrt{6} - \sqrt{2})n\rfloor \qquad G(\bigcirc_n) = \begin{cases} 2, & \text{if } n = 1 \\ \lfloor\sqrt{3}n\rfloor, & \text{otherwise} \end{cases}$$

- A position is losing iff the bitwise XOR of Grundy values of the stacks is 0.

A: Adolescent Architecture 2

Problem Author: Paul Wild

Solution

- This is a combinatorial game where for each stack we only care about its topmost block.
- Use the Sprague-Grundy theorem to assign each block B a Grundy value $G(B)$.
- By careful analysis and/or dynamic programming we can find closed forms:

$$G(\triangle_n) = n - 1 \qquad G(\square_n) = \lfloor(\sqrt{6} - \sqrt{2})n\rfloor \qquad G(\bigcirc_n) = \begin{cases} 2, & \text{if } n = 1 \\ \lfloor\sqrt{3}n\rfloor, & \text{otherwise} \end{cases}$$

- A position is losing iff the bitwise XOR of Grundy values of the stacks is 0.
- For each stack, compute the Grundy value needed to create a losing position.

A: Adolescent Architecture 2

Problem Author: Paul Wild

Solution

- This is a combinatorial game where for each stack we only care about its topmost block.
- Use the Sprague-Grundy theorem to assign each block B a Grundy value $G(B)$.
- By careful analysis and/or dynamic programming we can find closed forms:

$$G(\triangle_n) = n - 1 \qquad G(\square_n) = \lfloor(\sqrt{6} - \sqrt{2})n\rfloor \qquad G(\bigcirc_n) = \begin{cases} 2, & \text{if } n = 1 \\ \lfloor\sqrt{3}n\rfloor, & \text{otherwise} \end{cases}$$

- A position is losing iff the bitwise XOR of Grundy values of the stacks is 0.
- For each stack, compute the Grundy value needed to create a losing position.
- For each shape, check whether a block with that Grundy value exists and constitutes a legal move.

A: Adolescent Architecture 2

Problem Author: Paul Wild

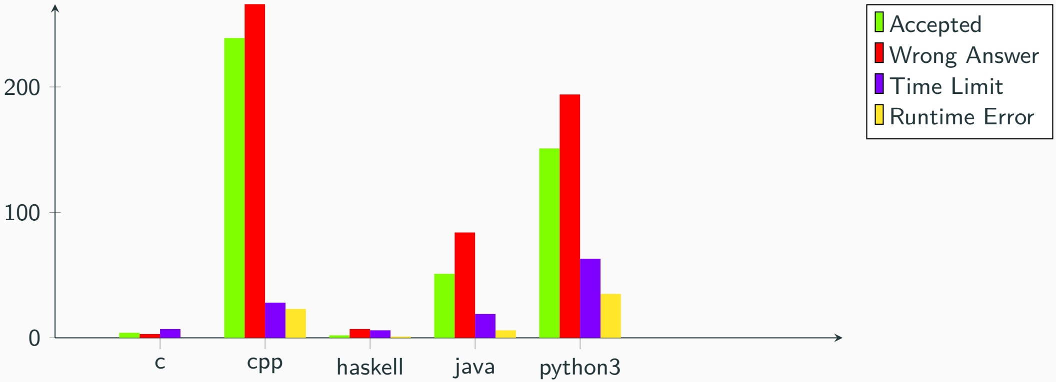
Solution

- This is a combinatorial game where for each stack we only care about its topmost block.
- Use the Sprague-Grundy theorem to assign each block B a Grundy value $G(B)$.
- By careful analysis and/or dynamic programming we can find closed forms:

$$G(\triangle_n) = n - 1 \qquad G(\square_n) = \lfloor(\sqrt{6} - \sqrt{2})n\rfloor \qquad G(\bigcirc_n) = \begin{cases} 2, & \text{if } n = 1 \\ \lfloor\sqrt{3}n\rfloor, & \text{otherwise} \end{cases}$$

- A position is losing iff the bitwise XOR of Grundy values of the stacks is 0.
- For each stack, compute the Grundy value needed to create a losing position.
- For each shape, check whether a block with that Grundy value exists and constitutes a legal move.
- Total runtime: $\mathcal{O}(n)$.

Language stats



Random facts

Jury work

- 275 commits

Random facts

Jury work

- 275 commits
- 773 secret test cases (≈ 59.5 per problem)

Random facts

Jury work

- 275 commits
- 773 secret test cases (≈ 59.5 per problem)
- 133 jury solutions

Random facts

Jury work

- 275 commits
- 773 secret test cases (≈ 59.5 per problem)
- 133 jury solutions
- The minimum number of lines the jury needed to solve all problems is

$$20 + 13 + 19 + 7 + 2 + 6 + 2 + 21 + 18 + 19 + 10 + 3 + 1 = 141$$

  On average 10.8 lines per problem