Freshmen Programming Contests 2026

Solutions presentation

By the Freshmen Programming Contests 2026 jury for:

- AAPJE in Amsterdam
- FPC in Delft
- FPC in Eindhoven
- GAPC in Groningen
- Contest in Mons

April 25, 2026

Please do not post the problems online

Other universities will have their contests in the coming weeks.

Please, do not post/discuss the problems online before

Saturday 9 May 2026 at 17:00

A: Animal Attire

Problem author: Leon van der Waal

- **Problem:** Pick how many items in each of the k categories such that $\prod_k i \geq n$ while minimizing the total items

A: Animal Attire

Problem author: Leon van der Waal

- **Problem:** Pick how many items in each of the k categories such that $\prod_k i \geq n$ while minimizing the total items
- Naive solution: sum $= k \cdot \lceil n^{1/k} \rceil$

A: Animal Attire

Problem author: Leon van der Waal

- **Problem:** Pick how many items in each of the k categories such that $\prod_k i \geq n$ while minimizing the total items
- Naive solution: sum $= k \cdot \lceil n^{1/k} \rceil$
- TLE: increasing item by one, computing product, repeat until product $\geq n$

A: Animal Attire

Problem author: Leon van der Waal

- **Problem:** Pick how many items in each of the k categories such that $\prod_k i \geq n$ while minimizing the total items
- Naive solution: sum $= k \cdot \lceil n^{1/k} \rceil$
- TLE: increasing item by one, computing product, repeat until product $\geq n$
- Instead, $x = \lfloor n^{1/k} \rfloor$, initialize all elements to x and increase categories by one until product is $\geq n$. Time complexity: $O(1)$ (since $k \leq 100$)

A: Animal Attire

Problem author: Leon van der Waal

- **Problem:** Pick how many items in each of the k categories such that $\prod_k i \geq n$ while minimizing the total items
- Naive solution: sum $= k \cdot \lceil n^{1/k} \rceil$
- TLE: increasing item by one, computing product, repeat until product $\geq n$
- Instead, $x = \lfloor n^{1/k} \rfloor$, initialize all elements to x and increase categories by one until product is $\geq n$. Time complexity: $O(1)$ (since $k \leq 100$)

Statistics: ... submissions, ... accepted, ... unknown

B: Building Beaver

Problem author: Jeroen Op de Beek

- **Problem:** Print $4\sqrt{n}$ with sufficiently many digits.

B: Building Beaver

Problem author: Jeroen Op de Beek

- **Problem:** Print $4\sqrt{n}$ with sufficiently many digits.
- Naive solution: do something stupid. $\mathcal{O}(2^n)$ is too slow!

B: Building Beaver

Problem author: Jeroen Op de Beek

- **Problem:** Print $4\sqrt{n}$ with sufficiently many digits.
- Naive solution: do something stupid. $\mathcal{O}(2^n)$ is too slow!
- Instead, just compute the answer. $\mathcal{O}(1)$.

B: Building Beaver

Problem author: Jeroen Op de Beek

- **Problem:** Print $4\sqrt{n}$ with sufficiently many digits.
- Naive solution: do something stupid. $\mathcal{O}(2^n)$ is too slow!
- Instead, just compute the answer. $\mathcal{O}(1)$.

Statistics: ... submissions, ... accepted, ... unknown

C: Coven Complications

Problem author: Andrei Voicu

- **Problem:** Minimize total witch losses while sealing one community per day.

C: Coven Complications

Problem author: Andrei Voicu

- **Problem:** Minimize total witch losses while sealing one community per day.
- Precompute daily risk r_i for each owned community.

C: Coven Complications

Problem author: Andrei Voicu

- **Problem:** Minimize total witch losses while sealing one community per day.
- Precompute daily risk r_i for each owned community.
- Total loss $= \sum_k$(remaining total risk after sealing k-th community).

C: Coven Complications

Problem author: Andrei Voicu

- **Problem:** Minimize total witch losses while sealing one community per day.
- Precompute daily risk r_i for each owned community.
- Total loss $= \sum_k$(remaining total risk after sealing k-th community).
- Greedy: seal highest-risk first. Sort descending, accumulate using suffix sums. $\mathcal{O}(n \log n)$.

C: Coven Complications

Problem author: Andrei Voicu

- **Problem:** Minimize total witch losses while sealing one community per day.
- Precompute daily risk r_i for each owned community.
- Total loss $= \sum_k$(remaining total risk after sealing k-th community).
- Greedy: seal highest-risk first. Sort descending, accumulate using suffix sums. $\mathcal{O}(n \log n)$.

Statistics: ... submissions, ... accepted, ... unknown

D: Digit Display

Problem author: Moham Balfakeih

- **Problem:** Given n segments, display the largest possible number on 7-segment digits.

D: Digit Display

Problem author: Moham Balfakeih

- **Problem:** Given n segments, display the largest possible number on 7-segment digits.
- If $n < 2$: no digit can be displayed, output `impossible`.

D: Digit Display

Problem author: Moham Balfakeih

- **Problem:** Given n segments, display the largest possible number on 7-segment digits.
- If $n < 2$: no digit can be displayed, output `impossible`.
- **Key insight:** A larger number of digits always beats fewer digits (e.g. $11 > 9$).

D: Digit Display

Problem author: Moham Balfakeih

- **Problem:** Given n segments, display the largest possible number on 7-segment digits.
- If $n < 2$: no digit can be displayed, output `impossible`.
- **Key insight:** A larger number of digits always beats fewer digits (e.g. $11 > 9$).
- So we want to *maximize the number of digits* first.

D: Digit Display

Problem author: Moham Balfakeih

- **Problem:** Given n segments, display the largest possible number on 7-segment digits.
- If $n < 2$: no digit can be displayed, output `impossible`.
- **Key insight:** A larger number of digits always beats fewer digits (e.g. $11 > 9$).
- So we want to *maximize the number of digits* first.
- The cheapest digit is 1 at 2 segments. So $\lfloor n/2 \rfloor$ digits is optimal.

D: Digit Display

Problem author: Moham Balfakeih

- **Problem:** Given n segments, display the largest possible number on 7-segment digits.
- If $n < 2$: no digit can be displayed, output `impossible`.
- **Key insight:** A larger number of digits always beats fewer digits (e.g. $11 > 9$).
- So we want to *maximize the number of digits* first.
- The cheapest digit is `1` at 2 segments. So $\lfloor n/2 \rfloor$ digits is optimal.
- If n is even: all `1`s $\Rightarrow$ `111...1` ($n/2$ digits).

D: Digit Display

Problem author: Moham Balfakeih

- **Problem:** Given n segments, display the largest possible number on 7-segment digits.
- If $n < 2$: no digit can be displayed, output `impossible`.
- **Key insight:** A larger number of digits always beats fewer digits (e.g. $11 > 9$).
- So we want to *maximize the number of digits* first.
- The cheapest digit is `1` at 2 segments. So $\lfloor n/2 \rfloor$ digits is optimal.
- If n is even: all `1`s $\Rightarrow$ `111...1` ($n/2$ digits).
- If n is odd: one leftover segment. Replace the leading `1` (2 segments) with `7` (3 segments) $\Rightarrow$ `711...1`.

D: Digit Display

Problem author: Moham Balfakeih

- **Problem:** Given n segments, display the largest possible number on 7-segment digits.
- If $n < 2$: no digit can be displayed, output `impossible`.
- **Key insight:** A larger number of digits always beats fewer digits (e.g. $11 > 9$).
- So we want to *maximize the number of digits* first.
- The cheapest digit is 1 at 2 segments. So $\lfloor n/2 \rfloor$ digits is optimal.
- If n is even: all 1s $\Rightarrow$ `111...1` ($n/2$ digits).
- If n is odd: one leftover segment. Replace the leading 1 (2 segments) with 7 (3 segments) $\Rightarrow$ `711...1`.
- This is optimal because 7 is the largest single digit costing exactly 3 segments, and placing it first maximizes the value.

D: Digit Display

Problem author: Moham Balfakeih

- **Problem:** Given n segments, display the largest possible number on 7-segment digits.
- If $n < 2$: no digit can be displayed, output `impossible`.
- **Key insight:** A larger number of digits always beats fewer digits (e.g. $11 > 9$).
- So we want to *maximize the number of digits* first.
- The cheapest digit is 1 at 2 segments. So $\lfloor n/2 \rfloor$ digits is optimal.
- If n is even: all 1s $\Rightarrow$ `111...1` ($n/2$ digits).
- If n is odd: one leftover segment. Replace the leading 1 (2 segments) with 7 (3 segments) $\Rightarrow$ `711...1`.
- This is optimal because 7 is the largest single digit costing exactly 3 segments, and placing it first maximizes the value.
- $\mathcal{O}(n)$ to print the output (or $\mathcal{O}(1)$ to compute what to print).

D: Digit Display

Problem author: Moham Balfakeih

- **Problem:** Given n segments, display the largest possible number on 7-segment digits.
- If $n < 2$: no digit can be displayed, output `impossible`.
- **Key insight:** A larger number of digits always beats fewer digits (e.g. $11 > 9$).
- So we want to *maximize the number of digits* first.
- The cheapest digit is 1 at 2 segments. So $\lfloor n/2 \rfloor$ digits is optimal.
- If n is even: all 1s $\Rightarrow$ `111...1` ($n/2$ digits).
- If n is odd: one leftover segment. Replace the leading 1 (2 segments) with 7 (3 segments) $\Rightarrow$ `711...1`.
- This is optimal because 7 is the largest single digit costing exactly 3 segments, and placing it first maximizes the value.
- $\mathcal{O}(n)$ to print the output (or $\mathcal{O}(1)$ to compute what to print).

Statistics: ... submissions, ... accepted, ... unknown

E: Ernest's Endeavour

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Given an undirected graph with n labeled nodes and m edges, check if any two nodes with the same label are connected.

E: Ernest's Endeavour

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Given an undirected graph with n labeled nodes and m edges, check if any two nodes with the same label are connected.
- Naive solution: For each node, traverse the graph until you reach another node with the same label.

E: Ernest's Endeavour

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Given an undirected graph with n labeled nodes and m edges, check if any two nodes with the same label are connected.
- Naive solution: For each node, traverse the graph until you reach another node with the same label.
- Time complexity: $\mathcal{O}(n \cdot (n + m))$ - too slow!

E: Ernest's Endeavour

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Given an undirected graph with n labeled nodes and m edges, check if any two nodes with the same label are connected.
- Naive solution: For each node, traverse the graph until you reach another node with the same label.
- Time complexity: $\mathcal{O}(n \cdot (n + m))$ - too slow!
- **Key insight:** Two nodes are connected iff they are part of the same connected component.

E: Ernest's Endeavour

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Given an undirected graph with n labeled nodes and m edges, check if any two nodes with the same label are connected.
- Naive solution: For each node, traverse the graph until you reach another node with the same label.
- Time complexity: $\mathcal{O}(n \cdot (n + m))$ - too slow!
- **Key insight:** Two nodes are connected iff they are part of the same connected component.
- Therefore, we only need to check if a connected component of the graph has two nodes with identical labels.

E: Ernest's Endeavour

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Given an undirected graph with n labeled nodes and m edges, check if any two nodes with the same label are connected.
- Naive solution: For each node, traverse the graph until you reach another node with the same label.
- Time complexity: $\mathcal{O}(n \cdot (n + m))$ - too slow!
- **Key insight:** Two nodes are connected iff they are part of the same connected component.
- Therefore, we only need to check if a connected component of the graph has two nodes with identical labels.
- For each connected component, mark the labels that it contains by traversing it once and save the indices of the associated nodes.

E: Ernest's Endeavour

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Given an undirected graph with n labeled nodes and m edges, check if any two nodes with the same label are connected.
- Naive solution: For each node, traverse the graph until you reach another node with the same label.
- Time complexity: $\mathcal{O}(n \cdot (n + m))$ - too slow!
- **Key insight:** Two nodes are connected iff they are part of the same connected component.
- Therefore, we only need to check if a connected component of the graph has two nodes with identical labels.
- For each connected component, mark the labels that it contains by traversing it once and save the indices of the associated nodes.
- If, while traversing, a node with an already marked label is found, the problem is done: output the index of the node with the previously marked label and the index of the newly found node.

E: Ernest's Endeavour

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Given an undirected graph with n labeled nodes and m edges, check if any two nodes with the same label are connected.
- Naive solution: For each node, traverse the graph until you reach another node with the same label.
- Time complexity: $\mathcal{O}(n \cdot (n + m))$ - too slow!
- **Key insight:** Two nodes are connected iff they are part of the same connected component.
- Therefore, we only need to check if a connected component of the graph has two nodes with identical labels.
- For each connected component, mark the labels that it contains by traversing it once and save the indices of the associated nodes.
- If, while traversing, a node with an already marked label is found, the problem is done: output the index of the node with the previously marked label and the index of the newly found node.
- If all connected components have been checked and no matching labels were found, output 'safe'.

E: Ernest's Endeavour

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Given an undirected graph with n labeled nodes and m edges, check if any two nodes with the same label are connected.
- Naive solution: For each node, traverse the graph until you reach another node with the same label.
- Time complexity: $\mathcal{O}(n \cdot (n + m))$ - too slow!
- **Key insight:** Two nodes are connected iff they are part of the same connected component.
- Therefore, we only need to check if a connected component of the graph has two nodes with identical labels.
- For each connected component, mark the labels that it contains by traversing it once and save the indices of the associated nodes.
- If, while traversing, a node with an already marked label is found, the problem is done: output the index of the node with the previously marked label and the index of the newly found node.
- If all connected components have been checked and no matching labels were found, output 'safe'.
- Time complexity: $\mathcal{O}(n + m)$.

E: Ernest's Endeavour

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Given an undirected graph with n labeled nodes and m edges, check if any two nodes with the same label are connected.
- Naive solution: For each node, traverse the graph until you reach another node with the same label.
- Time complexity: $\mathcal{O}(n \cdot (n + m))$ - too slow!
- **Key insight:** Two nodes are connected iff they are part of the same connected component.
- Therefore, we only need to check if a connected component of the graph has two nodes with identical labels.
- For each connected component, mark the labels that it contains by traversing it once and save the indices of the associated nodes.
- If, while traversing, a node with an already marked label is found, the problem is done: output the index of the node with the previously marked label and the index of the newly found node.
- If all connected components have been checked and no matching labels were found, output 'safe'.
- Time complexity: $\mathcal{O}(n + m)$.

Statistics: ... submissions, ... accepted, ... unknown

F: Ford's Funny Fields

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Choose a subset of fields such that their total length is at least s, minimizing the total cost.

F: Ford's Funny Fields

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Choose a subset of fields such that their total length is at least s, minimizing the total cost.
- This is a variation of the classic 0/1 Knapsack Problem.

F: Ford's Funny Fields

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Choose a subset of fields such that their total length is at least s, minimizing the total cost.
- This is a variation of the classic 0/1 Knapsack Problem.
- Iterate over all individual fields. Let $DP[j]$ be the minimum cost to reach a covered length j. We cap j at s.

F: Ford's Funny Fields

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Choose a subset of fields such that their total length is at least s, minimizing the total cost.
- This is a variation of the classic $0/1$ Knapsack Problem.
- Iterate over all individual fields. Let $DP[j]$ be the minimum cost to reach a covered length j. We cap j at s.
- Since there are at most $n \cdot x \leq 2000$ fields and $s \leq 2000$, DP updates take $\mathcal{O}(\sum x \cdot s)$ time, which securely passes bounds.

F: Ford's Funny Fields

Problem author: David Ghiberdic, Rares Rauta

- **Problem:** Choose a subset of fields such that their total length is at least s, minimizing the total cost.
- This is a variation of the classic 0/1 Knapsack Problem.
- Iterate over all individual fields. Let $DP[j]$ be the minimum cost to reach a covered length j. We cap j at s.
- Since there are at most $n \cdot x \leq 2000$ fields and $s \leq 2000$, DP updates take $\mathcal{O}(\sum x \cdot s)$ time, which securely passes bounds.

Statistics: ... submissions, ... accepted, ... unknown

G: Game Night Groups

Problem author: Arnoud van der Leer

- **Problem:** Print $4\sqrt{n}$ with sufficiently many digits.

G: Game Night Groups

Problem author: Arnoud van der Leer

- **Problem:** Print $4\sqrt{n}$ with sufficiently many digits.
- Naive solution: do something stupid. $\mathcal{O}(2^n)$ is too slow!

G: Game Night Groups

Problem author: Arnoud van der Leer

- **Problem:** Print $4\sqrt{n}$ with sufficiently many digits.
- Naive solution: do something stupid. $\mathcal{O}(2^n)$ is too slow!
- Instead, just compute the answer. $\mathcal{O}(1)$.

G: Game Night Groups

Problem author: Arnoud van der Leer

- **Problem:** Print $4\sqrt{n}$ with sufficiently many digits.
- Naive solution: do something stupid. $\mathcal{O}(2^n)$ is too slow!
- Instead, just compute the answer. $\mathcal{O}(1)$.

Statistics: ... submissions, ... accepted, ... unknown

H: Hasty Hiker

Problem author: Jeroen Op de Beek

- **Problem:** Count the number of beautiful trails.

H: Hasty Hiker

Problem author: Jeroen Op de Beek

- **Problem:** Count the number of beautiful trails.
- **Naive solution:** Loop over all 4-tuples and check if they are beautiful trails.

H: Hasty Hiker

Problem author: Jeroen Op de Beek

- **Problem:** Count the number of beautiful trails.
- **Naive solution:** Loop over all 4-tuples and check if they are beautiful trails.
- **Complexity:** $\mathcal{O}\left(n^4\right)$, too slow!

H: Hasty Hiker

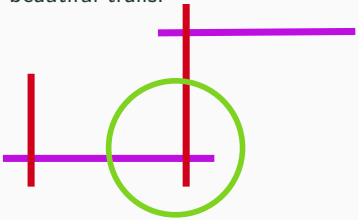
Problem author: Jeroen Op de Beek

- **Problem:** Count the number of beautiful trails.

H: Hasty Hiker

Problem author: Jeroen Op de Beek

- **Problem:** Count the number of beautiful trails.

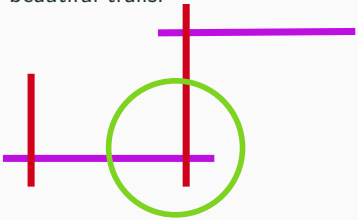


- **Faster solution:** Let's loop over the two roads in the middle of the trail.

H: Hasty Hiker

Problem author: Jeroen Op de Beek

- **Problem:** Count the number of beautiful trails.

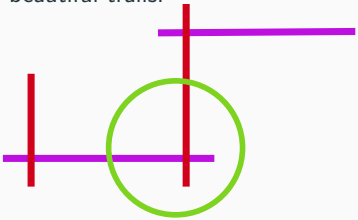


- **Faster solution:** Let's loop over the two roads in the middle of the trail.
- **Faster solution:** First check whether they intersect.

H: Hasty Hiker

Problem author: Jeroen Op de Beek

- **Problem:** Count the number of beautiful trails.

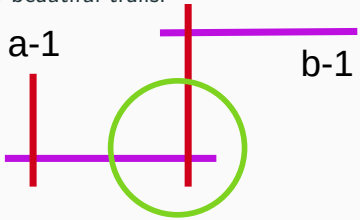


- **Faster solution:** Let's loop over the two roads in the middle of the trail.
- **Faster solution:** First check whether they intersect.
- **Faster solution:** Count how many roads intersect these two roads. Lets call these counts a and b, respectively.

H: Hasty Hiker

Problem author: Jeroen Op de Beek

- **Problem:** Count the number of beautiful trails.

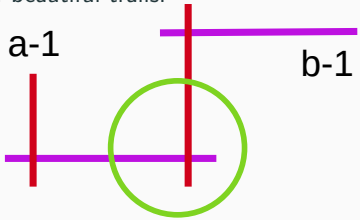


- **Faster solution:** Let's loop over the two roads in the middle of the trail.
- **Faster solution:** First check whether they intersect.
- **Faster solution:** Count how many roads intersect these two roads. Lets call these counts a and b, respectively.
- **Faster solution:** Then add $(a-1) \times (b-1)$ to the answer.

H: Hasty Hiker

Problem author: Jeroen Op de Beek

- **Problem:** Count the number of beautiful trails.



- **Faster solution:** Let's loop over the two roads in the middle of the trail.
- **Faster solution:** First check whether they intersect.
- **Faster solution:** Count how many roads intersect these two roads. Lets call these counts a and b, respectively.
- **Faster solution:** Then add $(a-1) \times (b-1)$ to the answer.
- **Complexity:** $\mathcal{O}(n)$ per intersection, so $\mathcal{O}\left(n^3\right)$ in total. Still too slow.

H: Hasty Hiker

Problem author: Jeroen Op de Beek

- **Problem:** Count the number of beautiful trails.

H: Hasty Hiker

Problem author: Jeroen Op de Beek

- **Problem:** Count the number of beautiful trails.
- **Solution:** We can optimize this, by precomputing the number of other roads each road intersects.

H: Hasty Hiker

Problem author: Jeroen Op de Beek

- **Problem:** Count the number of beautiful trails.
- **Solution:** We can optimize this, by precomputing the number of other roads each road intersects.
- **Complexity:** $\mathcal{O}\left(n^2\right)$, fast enough!

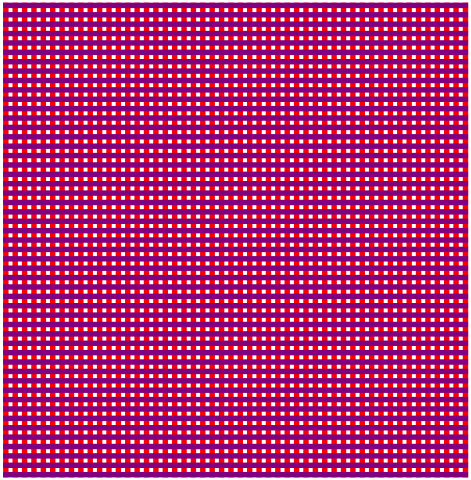
H: Hasty Hiker

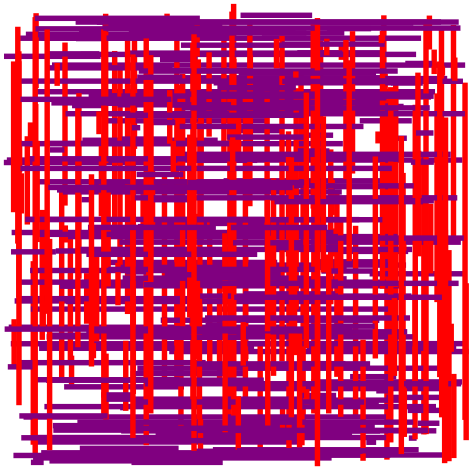
Problem author: Jeroen Op de Beek

- **Problem:** Count the number of beautiful trails.
- **Solution:** We can optimize this, by precomputing the number of other roads each road intersects.
- **Complexity:** $\mathcal{O}\left(n^2\right)$, fast enough!
- **Bonus:** It is possible to solve this problem in $\mathcal{O}\left(n \log(n)\right)$ using fenwick tree + sweepline.

H: Hasty Hiker

Problem author: Jeroen Op de Beek





Statistics: . . . submissions, . . . accepted, . . . unknown

I: Intricate Idioms

Problem author: Jeroen Op de Beek

- **Problem:** Print $4\sqrt{n}$ with sufficiently many digits.

I: Intricate Idioms

Problem author: Jeroen Op de Beek

- **Problem:** Print $4\sqrt{n}$ with sufficiently many digits.
- Naive solution: do something stupid. $\mathcal{O}(2^n)$ is too slow!

I: Intricate Idioms

Problem author: Jeroen Op de Beek

- **Problem:** Print $4\sqrt{n}$ with sufficiently many digits.
- Naive solution: do something stupid. $\mathcal{O}(2^n)$ is too slow!
- Instead, just compute the answer. $\mathcal{O}(1)$.

I: Intricate Idioms

Problem author: Jeroen Op de Beek

- **Problem:** Print $4\sqrt{n}$ with sufficiently many digits.
- Naive solution: do something stupid. $\mathcal{O}(2^n)$ is too slow!
- Instead, just compute the answer. $\mathcal{O}(1)$.

Statistics: ... submissions, ... accepted, ... unknown

J: Joker Juggling

Problem author: Jeroen Op de Beek

- **Problem:** Check whether a pattern containing letters and jokers ('*') matches a word.

J: Joker Juggling

Problem author: Jeroen Op de Beek

- **Problem:** Check whether a pattern containing letters and jokers ('*') matches a word.
- First check: can every joker represent the same number of letters?
  - For example, "a**" can never match "abcd" because a joker cannot represent $1\frac{1}{2}$ letters.

J: Joker Juggling

Problem author: Jeroen Op de Beek

- **Problem:** Check whether a pattern containing letters and jokers ('*') matches a word.
- First check: can every joker represent the same number of letters?
  - For example, "a**" can never match "abcd" because a joker cannot represent $1\frac{1}{2}$ letters.
- Knowing how many letters each joker represents, eagerly match the first joker.
  - For example, for "ba**" matching "banana", each joker represents 2 letters, so this must be "na".

J: Joker Juggling

Problem author: Jeroen Op de Beek

- **Problem:** Check whether a pattern containing letters and jokers ('*') matches a word.
- First check: can every joker represent the same number of letters?
  - For example, "a**" can never match "abcd" because a joker cannot represent $1\frac{1}{2}$ letters.
- Knowing how many letters each joker represents, eagerly match the first joker.
  - For example, for "ba**" matching "banana", each joker represents 2 letters, so this must be "na".
- Replace every joker with its replacement and compare it with the given word.
  - Beware of repeated string concatenation, avoid $\mathcal{O}(n^2)$ running time.

J: Joker Juggling

Problem author: Jeroen Op de Beek

- **Problem:** Check whether a pattern containing letters and jokers ('*') matches a word.
- First check: can every joker represent the same number of letters?
  - For example, "a**" can never match "abcd" because a joker cannot represent $1\frac{1}{2}$ letters.
- Knowing how many letters each joker represents, eagerly match the first joker.
  - For example, for "ba**" matching "banana", each joker represents 2 letters, so this must be "na".
- Replace every joker with its replacement and compare it with the given word.
  - Beware of repeated string concatenation, avoid $\mathcal{O}(n^2)$ running time.

Statistics: ... submissions, ... accepted, ... unknown

K: Kracked Enkoder

Problem author: Leon van der Waal

- **Multi-pass Problem:** Encode an integer $1 \leq x \leq 2^{30} - 2$ into 29 bits, and decode it back.

K: Kracked Enkoder

Problem author: Leon van der Waal

- **Multi-pass Problem:** Encode an integer $1 \leq x \leq 2^{30} - 2$ into 29 bits, and decode it back.
- Normally, you need 30 bits to encode integers up to 2^{30}.

K: Kracked Enkoder

Problem author: Leon van der Waal

- **Multi-pass Problem:** Encode an integer $1 \leq x \leq 2^{30} - 2$ into 29 bits, and decode it back.
- Normally, you need 30 bits to encode integers up to 2^{30}.
- **Solution 1:** We can (ab)use leading zeroes.
  For example, "00" could encode a different value than "0".

K: Kracked Enkoder

Problem author: Leon van der Waal

- **Multi-pass Problem:** Encode an integer $1 \leq x \leq 2^{30} - 2$ into 29 bits, and decode it back.
- Normally, you need 30 bits to encode integers up to 2^{30}.
- **Solution 1:** We can (ab)use leading zeroes.
  For example, "00" could encode a different value than "0".
- **Solution 2:** Treat the initial '1' as implicit (we do not need to encode 0).
  For example, encode 42 as "01010". When decoding, prepend the implicit '1' again.
  (This is also how the mantissa of IEEE-754 floating-point numbers are encoded.)

K: Kracked Enkoder

Problem author: Leon van der Waal

- **Multi-pass Problem:** Encode an integer $1 \leq x \leq 2^{30} - 2$ into 29 bits, and decode it back.
- Normally, you need 30 bits to encode integers up to 2^{30}.
- **Solution 1:** We can (ab)use leading zeroes.
  For example, "00" could encode a different value than "0".
- **Solution 2:** Treat the initial '1' as implicit (we do not need to encode 0).
  For example, encode 42 as "01010". When decoding, prepend the implicit '1' again.
  (This is also how the mantissa of IEEE-754 floating-point numbers are encoded.)
  - Edge-case: you cannot encode the integer 1 as the empty string ("").
    Either increment the values by one, or encode 1 as a string of 29 '1's.

K: Kracked Enkoder

Problem author: Leon van der Waal

- **Multi-pass Problem:** Encode an integer $1 \leq x \leq 2^{30} - 2$ into 29 bits, and decode it back.
- Normally, you need 30 bits to encode integers up to 2^{30}.
- **Solution 1:** We can (ab)use leading zeroes.
  For example, "00" could encode a different value than "0".
- **Solution 2:** Treat the initial '1' as implicit (we do not need to encode 0).
  For example, encode 42 as "01010". When decoding, prepend the implicit '1' again.
  (This is also how the mantissa of IEEE-754 floating-point numbers are encoded.)
  - Edge-case: you cannot encode the integer 1 as the empty string ("").
    Either increment the values by one, or encode 1 as a string of 29 '1's.

Statistics: ... submissions, ... accepted, ... unknown

L: Labyrinth Obstacle Layout

Problem author: Jeroen Op de Beek

- **Problem:** Build a grid with exactly two paths from top-left to bottom-right.

L: Labyrinth Obstacle Layout

Problem author: Jeroen Op de Beek

- **Problem:** Build a grid with exactly two paths from top-left to bottom-right.
- From the top-left, you have two options: walk to the right and to the bottom.

L: Labyrinth Obstacle Layout

Problem author: Jeroen Op de Beek

- **Problem:** Build a grid with exactly two paths from top-left to bottom-right.
- From the top-left, you have two options: walk to the right and to the bottom.
- **Simplest solution:** One path walks all the way to the top-right and then to the bottom-right, the other path all the way to the bottom-left and then to the bottom-right.

L: Labyrinth Obstacle Layout

Problem author: Jeroen Op de Beek

- **Problem:** Build a grid with exactly two paths from top-left to bottom-right.
- From the top-left, you have two options: walk to the right and to the bottom.
- **Simplest solution:** One path walks all the way to the top-right and then to the bottom-right, the other path all the way to the bottom-left and then to the bottom-right.
- Of course, there are many other solutions possible.

L: Labyrinth Obstacle Layout

Problem author: Jeroen Op de Beek

- **Problem:** Build a grid with exactly two paths from top-left to bottom-right.
- From the top-left, you have two options: walk to the right and to the bottom.
- **Simplest solution:** One path walks all the way to the top-right and then to the bottom-right, the other path all the way to the bottom-left and then to the bottom-right.
- Of course, there are many other solutions possible.

Statistics: ... submissions, ... accepted, ... unknown

Random facts

Jury work

- ??? commits (last year: 426)

Random facts

Jury work

- ??? commits (last year: 426)
- ??? secret test cases (last year: 625)

Random facts

Jury work

- ??? commits (last year: 426)
- ??? secret test cases (last year: 625)
- ??? accepted jury/proofreader solutions (last year: 160)

Random facts

Jury work

- ??? commits (last year: 426)
- ??? secret test cases (last year: 625)
- ??? accepted jury/proofreader solutions (last year: 160)
- The minimum[1] number of lines the jury needed to solve all problems is

$$?+?+?+?+?+?+?+?+?+?+?+? = ??$$

  On average ?.? lines per problem, up from 3.5 last year

[1] *After* codegolfing

Thanks to the proofreaders:

- Arnoud van der Leer (Delft / Leiden)
- Bartjan Henkemans (TU Eindhoven)
- Davina van Meer (Delft)
- Christophe Grandmont (UMONS, 🎈)
- Kévin Dubrulle (UMONS, 🎈)
- Moham Balfakeih (TU Delft)
- Thomas Verwoerd (TU Delft, Kotlin Hero 🎈)
- Thore Husfeldt (ITU Copenhagen)
- Wietze Koops (Lund Uni.../...versity of Copenhagen, 🎈)

🎈 = solved most problems

Thanks to the Jury for the Freshmen Programming Contests:

- Andrei Voicu (TU Delft)
- David Ghiberdic (TU Eindhoven)
- Gianmaria Piergianni (TU Delft)
- Jeroen Op de Beek (TU Delft)
- Leon van der Waal (TU Delft)
- Liudas Staniulis (VU Amsterdam)
- Maarten Sijm (TU Delft)
- Mattia Marziali (RU Groningen)
- Rares Rauta (TU Eindhoven)

UMONS
Université de Mons

Open online contest

Want to solve the problems you could not finish?
Or have friends that like to solve algorithmic problems?

`https://fpcs2026.bapc.eu/`

Saturday 9 May 2026 13:00–17:00

Please, do not post/discuss the problems online before this time!

Excited to participate in the next contest?

Register for the BAPC Preliminaries in September!

Want to organize these contests?

Join the organizing committee!

Want to create programming problems for FPCs next year?

Either join the committee, or contact Maarten Sijm