Winter Contest 2025 Presentation of Solutions

The Winter Contest Jury

February 6, 2025

Winter Contest 2025 Jury

- **Lucas Alber**
  Karlsruhe Institute of Technology
- **Felicia Lucke**
  Durham University UK, CPUlm
- **Niyaz Nigmatullin**
  JetBrains
- **Jannik Olbrich**
  Ulm University, CPUlm
- **David Stangl**
  MOIA GmbH
- **Christopher Weyand**
  MOIA GmbH, CPUlm
- **Paul Wild**
  Friedrich-Alexander University
  Erlangen-Nürnberg, CPUlm
- **Wendy Yi**
  Karlsruhe Institute of Technology, CPUlm
- **Michael Zündorf**
  Karlsruhe Institute of Technology, CPUlm

Winter Contest 2025 Test Solvers

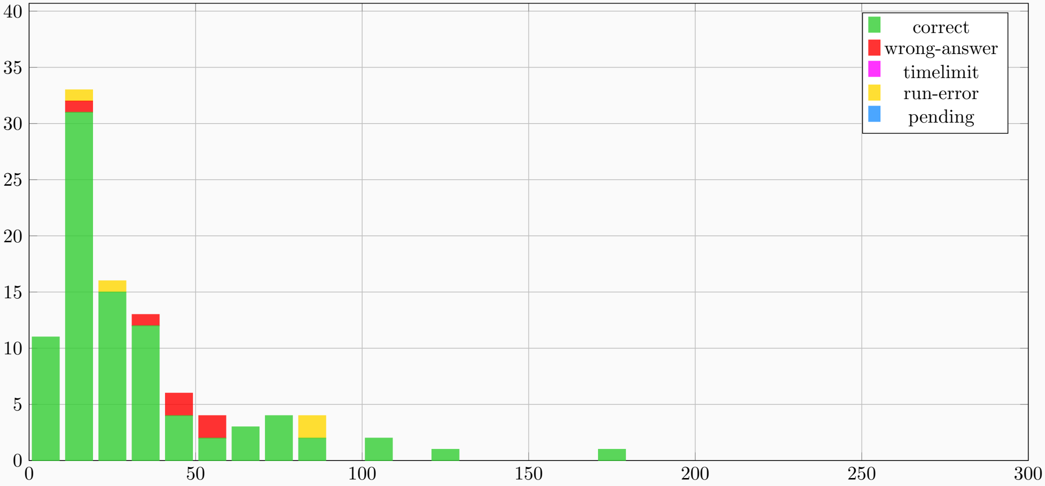
- **Brutenis Gliwa**
  Planet AI GmbH, CPUlm
- **Andreas Grigorjew**
  University of Helsinki FI, CPUlm

Winter Contest 2025 Technical Team

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

Jan 25: Winter Contest

Problem author: Paul Wild



Jan 25: Winter Contest

Problem author: Paul Wild

Problem

Given a range of years, find the number of dates falling into that range where each of year, month and day is a square number.

Jan 25: Winter Contest

Problem author: Paul Wild

Problem

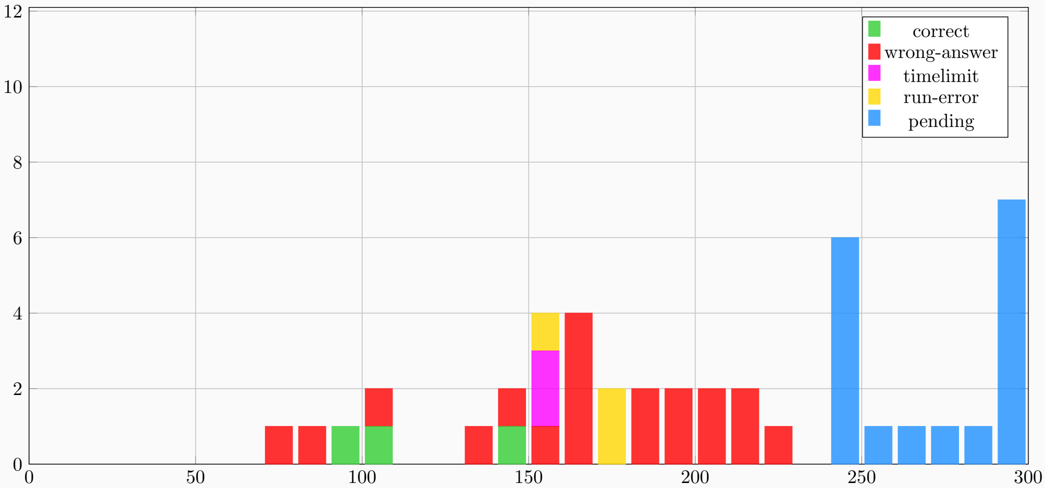
Given a range of years, find the number of dates falling into that range where each of year, month and day is a square number.

Solution

- Enumerating all dates in the range is kind of annoying but doable.
- We can make things easier for ourselves:
  - We only care about months 1, 4 and 9 and days 1, 4, 9, 16, 25.
  - So there are always 15 possible days for each square year.
- To find square years, loop over $45 \leq x \leq 99$ and check if x^2 is in the given range.

Jan 29: Chinese New Year

Problem author: Michael Zündorf

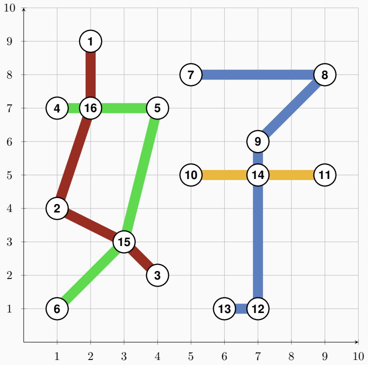


Jan 29: Chinese New Year

Problem author: Michael Zündorf

Problem

Given a planar graph G, decompose it into the minimal number of – not necessarily simple – paths.



Observation 1

- A single path only contains two odd degree vertices.

$\Rightarrow$ We need at least $\frac{\#oddvertices}{2}$ paths.

Jan 29: Chinese New Year

Problem author: Michael Zündorf

Observation 1

- A single path only contains two odd degree vertices.

$\Rightarrow$ We need at least $\frac{\#oddvertices}{2}$ paths.

Observation 2

- Even if there is no odd vertex, each component needs at least one path.

Jan 29: Chinese New Year

Problem author: Michael Zündorf

Observation 1

- A single path only contains two odd degree vertices.

$\Rightarrow$ We need at least $\frac{\#oddvertices}{2}$ paths.

Observation 2

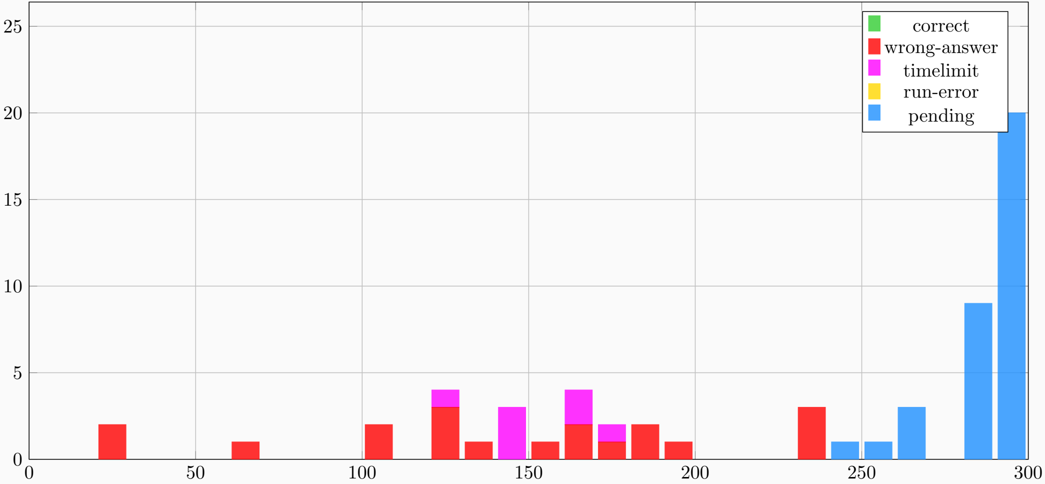
- Even if there is no odd vertex, each component needs at least one path.

Solution

- Pair up the odd degree vertices in any way and add an edge.
- The resulting graph contains an euler tour.
- Split this tour along the added edges.

Mar 3: Carnival

Problem author: Felicia Lucke



Mar 3: Carnival

Problem author: Felicia Lucke

Problem

Given a graph G of radius 2, colour the vertices with two colours R and B such that every vertex is adjacent to at most one vertex of the other colour.

Mar 3: Carnival

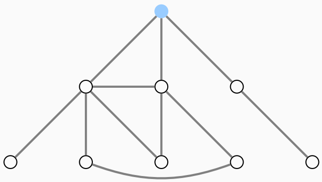
Problem author: Felicia Lucke

Problem

Given a graph G of radius 2, colour the vertices with two colours R and B such that every vertex is adjacent to at most one vertex of the other colour.

Solution

- Find v at distance at most 2 to every other vertex.
- Colour v blue.



Mar 3: Carnival

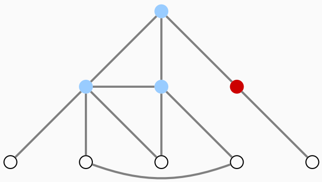
Problem author: Felicia Lucke

Problem

Given a graph G of radius 2, colour the vertices with two colours R and B such that every vertex is adjacent to at most one vertex of the other colour.

Solution

- Find v at distance at most 2 to every other vertex.
- Colour v blue.
- At most one neighbour of v is red.
- Try all options to colour the neighbours of v.
- In the case where all are blue, try all options to colour one other vertex of G red.



Mar 3: Carnival

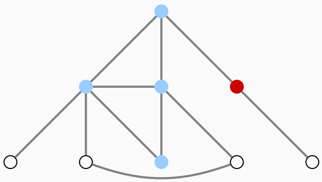
Problem author: Felicia Lucke

Problem

Given a graph G of radius 2, colour the vertices with two colours R and B such that every vertex is adjacent to at most one vertex of the other colour.

Solution

- Find v at distance at most 2 to every other vertex.
- Colour v blue.
- At most one neighbour of v is red.
- Try all options to colour the neighbours of v.
- In the case where all are blue, try all options to colour one other vertex of G red.
- Propagate the colouring:
  - A vertex with two red (blue) neighbours has to be red (blue).
  - If a red (blue) vertex has a blue (red) neighbour, all other neighbours are red (blue).



Mar 3: Carnival

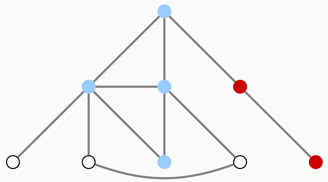
Problem author: Felicia Lucke

Problem

Given a graph G of radius 2, colour the vertices with two colours R and B such that every vertex is adjacent to at most one vertex of the other colour.

Solution

- Find v at distance at most 2 to every other vertex.
- Colour v blue.
- At most one neighbour of v is red.
- Try all options to colour the neighbours of v.
- In the case where all are blue, try all options to colour one other vertex of G red.
- Propagate the colouring:
  - A vertex with two red (blue) neighbours has to be red (blue).
  - If a red (blue) vertex has a blue (red) neighbour, all other neighbours are red (blue).

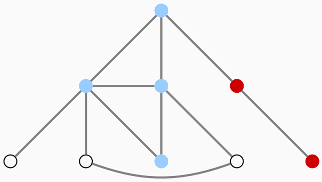


Mar 3: Carnival

Problem author: Felicia Lucke

Observation 1

Every uncoloured vertex has exactly one blue neighbour



Mar 3: Carnival

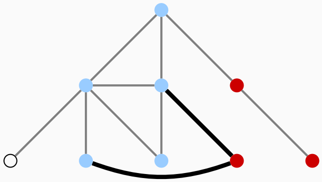
Problem author: Felicia Lucke

Observation 1

Every uncoloured vertex has exactly one blue neighbour

Observation 2

Every connected component in the graph of uncoloured vertices is either fully red or fully blue.



Mar 3: Carnival

Problem author: Felicia Lucke

Observation 1

Every uncoloured vertex has exactly one blue neighbour

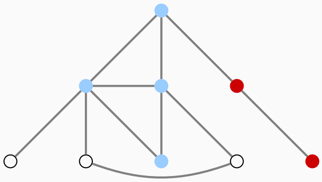
Observation 2

Every connected component in the graph of uncoloured vertices is either fully red or fully blue.

Solve by 2-SAT

- One variable per uncoloured component indicates whether red or blue, say true means blue.
- Clause (x) if a blue vertex has two neighbours in the same component
- Clause $(x \vee y)$ for any two components with a common blue neighbour

Total runtime $O(n * m)$



Mar 3: Carnival

Problem author: Felicia Lucke

Observation 1

Every uncoloured vertex has exactly one blue neighbour

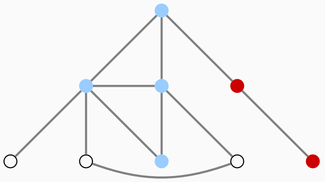
Observation 2

Every connected component in the graph of uncoloured vertices is either fully red or fully blue.

Solve by 2-SAT

- One variable per uncoloured component indicates whether red or blue, say true means blue.
- Clause (x) if a blue vertex has two neighbours in the same component
- Clause $(x \vee y)$ for any two components with a common blue neighbour

Total runtime $O(n * m)$

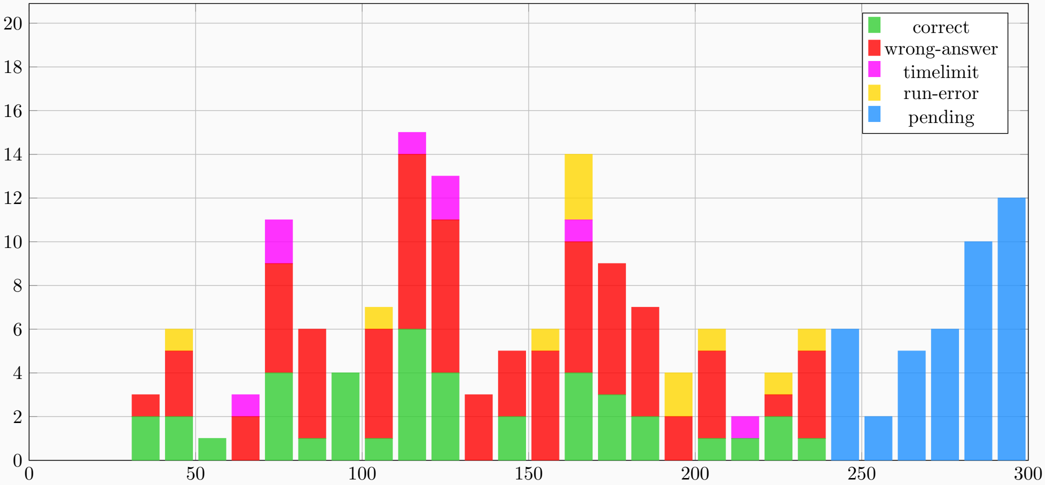


Note

For every 2-SAT formula, we can construct a graph where solving this problem corresponds to checking satisfiability.

Mar 14: Pi Day

Problem author: Paul Wild



Mar 14: Pi Day

Problem author: Paul Wild

Problem

- A calculator encrypts the digits 0–9 using the letters a–j.
- You can interactively add numbers to the calculator's display value.
- Your queries also use the letters a–j, so you don't know which numbers you add.
- Figure out enough parts of the cipher to print the current display number in plaintext.

Mar 14: Pi Day

Problem author: Paul Wild

Insight

Even though you don't have to, it's easiest to figure out the complete cipher.

Mar 14: Pi Day

Problem author: Paul Wild

Insight

Even though you don't have to, it's easiest to figure out the complete cipher.

Solution 1 – Adding single digit numbers

- Cycle through the letters `a`–`j` and them add one by one.
- Track the last digit to find out which letter corresponds to `0`.
- Track the second to last digit to find out the cyclic order of digits.
- After at most 30 queries you should know the full cipher.

Mar 14: Pi Day

Problem author: Paul Wild

Insight

Even though you don't have to, it's easiest to figure out the complete cipher.

Solution 1 – Adding single digit numbers

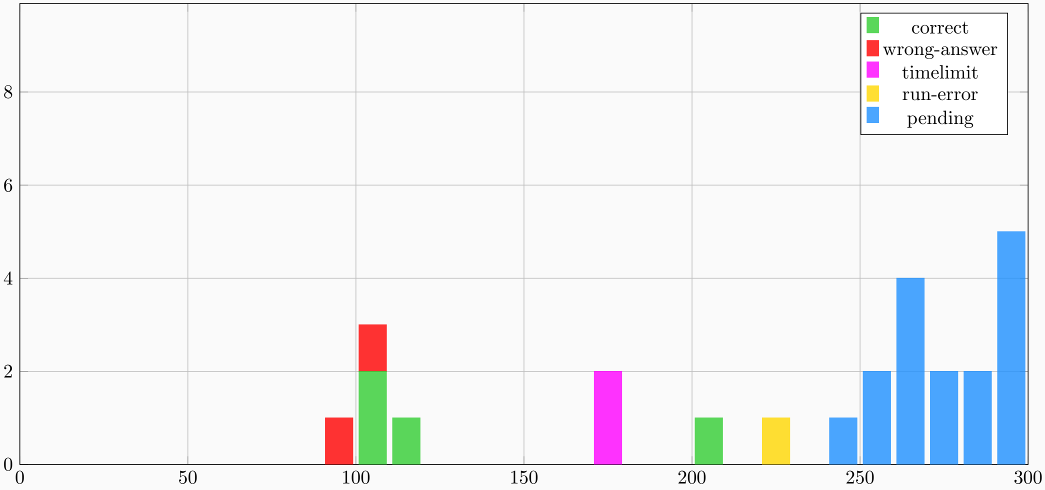
- Cycle through the letters `a-j` and them add one by one.
- Track the last digit to find out which letter corresponds to `0`.
- Track the second to last digit to find out the cyclic order of digits.
- After at most 30 queries you should know the full cipher.

Solution 2 – Adding large numbers

- Add a few random large numbers and note down the results.
- Solve the *cryptarithm* puzzle by trying out all 10! permutations of `a-j`.
- Given enough numbers (5 is plenty), the puzzle has a unique solution.

Mar 21: Colour Day

Problem author: Jannik Olbrich



Mar 21: Colour Day

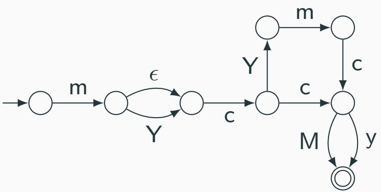
Problem author: Jannik Olbrich

Problem

Given two regular expressions α_1 and α_2 without nesting or Kleene-star, find a string in the languages' intersection (or determine that there is none)

- Build DAGs g_1 and g_2 representing all strings in the languages

$$(m)(Y \mid \epsilon)(c)(Ymc \mid c)(M \mid y)$$



Mar 21: Colour Day

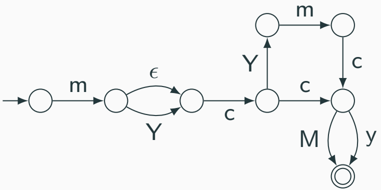
Problem author: Jannik Olbrich

Problem

Given two regular expressions α_1 and α_2 without nesting or Kleene-star, find a string in the languages' intersection (or determine that there is none)

- Build DAGs g_1 and g_2 representing all strings in the languages
- Use dynamic programming: Let $D(i,j)$ indicate whether there is a common string starting at i in g_1 and j in g_2

$(m)(Y \mid \epsilon)(c)(Ymc \mid c)(M \mid y)$



Mar 21: Colour Day

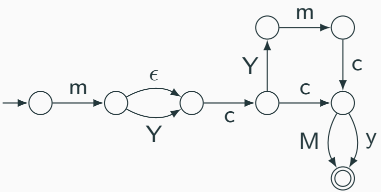
Problem author: Jannik Olbrich

Problem

Given two regular expressions α_1 and α_2 without nesting or Kleene-star, find a string in the languages' intersection (or determine that there is none)

- Build DAGs g_1 and g_2 representing all strings in the languages
- Use dynamic programming: Let $D(i, j)$ indicate whether there is a common string starting at i in g_1 and j in g_2
- Note that the average degree in the graphs is constant

$(m)(Y \mid \epsilon)(c)(Ymc \mid c)(M \mid y)$



Mar 21: Colour Day

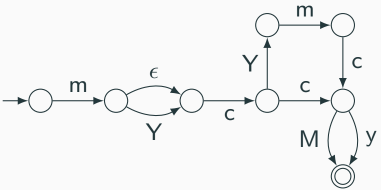
Problem author: Jannik Olbrich

Problem

Given two regular expressions α_1 and α_2 without nesting or Kleene-star, find a string in the languages' intersection (or determine that there is none)

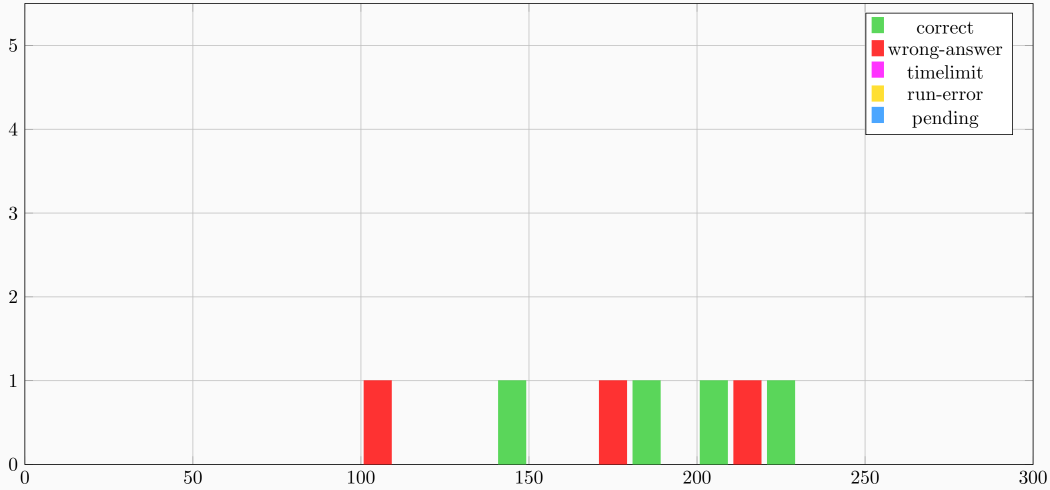
- Build DAGs g_1 and g_2 representing all strings in the languages
- Use dynamic programming: Let $D(i, j)$ indicate whether there is a common string starting at i in g_1 and j in g_2
- Note that the average degree in the graphs is constant
- Therefore, the time complexity is $\mathcal{O}(N \cdot M)$, where N and M are the number of nodes in the DAGs

$$(m)(Y \mid \epsilon)(c)(Ymc \mid c)(M \mid y)$$



Apr 20: Easter

Problem author: David Stangl



Apr 20: Easter

Problem author: David Stangl

Problem

You are given a route consisting of n stops in a fixed order, each with a travel time d from the previous stop and a time window $[s, e]$ during which you have to visit it. Additionally, there are m constraints indicating that the time between visiting stop a and stop b must be at most l. Find valid times to visit the stops, or determine that it is impossible.

Apr 20: Easter

Problem author: David Stangl

Problem

You are given a route consisting of n stops in a fixed order, each with a travel time d from the previous stop and a time window $[s, e]$ during which you have to visit it. Additionally, there are m constraints indicating that the time between visiting stop a and stop b must be at most l. Find valid times to visit the stops, or determine that it is impossible.

Solution

- Start with the solution that visits every stop as early as possible.

Apr 20: Easter

Problem author: David Stangl

Problem

You are given a route consisting of n stops in a fixed order, each with a travel time d from the previous stop and a time window $[s, e]$ during which you have to visit it. Additionally, there are m constraints indicating that the time between visiting stop a and stop b must be at most l. Find valid times to visit the stops, or determine that it is impossible.

Solution

- Start with the solution that visits every stop as early as possible.
- For each stop from 1 to n, select a fixed time by delaying it as much as possible while keeping within the time windows of it and later stops.

Apr 20: Easter

Problem author: David Stangl

Problem

You are given a route consisting of n stops in a fixed order, each with a travel time d from the previous stop and a time window $[s, e]$ during which you have to visit it. Additionally, there are m constraints indicating that the time between visiting stop a and stop b must be at most l. Find valid times to visit the stops, or determine that it is impossible.

Solution

- Start with the solution that visits every stop as early as possible.
- For each stop from 1 to n, select a fixed time by delaying it as much as possible while keeping within the time windows of it and later stops.
- After fixing the time for a stop a, all constraints that limit the time taken between a and a later stop b can be translated into adjustments of the time window of b.

Apr 20: Easter

Problem author: David Stangl

Problem

You are given a route consisting of n stops in a fixed order, each with a travel time d from the previous stop and a time window $[s, e]$ during which you have to visit it. Additionally, there are m constraints indicating that the time between visiting stop a and stop b must be at most l. Find valid times to visit the stops, or determine that it is impossible.

Solution

- Start with the solution that visits every stop as early as possible.
- For each stop from 1 to n, select a fixed time by delaying it as much as possible while keeping within the time windows of it and later stops.
- After fixing the time for a stop a, all constraints that limit the time taken between a and a later stop b can be translated into adjustments of the time window of b.
- If any time ends up outside the stops time window then there is no solution.

Apr 20: Easter

Problem author: David Stangl

Problem

You are given a route consisting of n stops in a fixed order, each with a travel time d from the previous stop and a time window $[s, e]$ during which you have to visit it. Additionally, there are m constraints indicating that the time between visiting stop a and stop b must be at most l. Find valid times to visit the stops, or determine that it is impossible.

Implementation in $\mathcal{O}((n + m) \log n)$

- For each stop, keep the maximum total delay before reaching it in a data structure (segment tree, `std::set`, etc.).

Apr 20: Easter

Problem author: David Stangl

Problem

You are given a route consisting of n stops in a fixed order, each with a travel time d from the previous stop and a time window $[s, e]$ during which you have to visit it. Additionally, there are m constraints indicating that the time between visiting stop a and stop b must be at most l. Find valid times to visit the stops, or determine that it is impossible.

Implementation in $\mathcal{O}((n + m) \log n)$

- For each stop, keep the maximum total delay before reaching it in a data structure (segment tree, `std::set`, etc.).
- When fixing a stop a

Apr 20: Easter

Problem author: David Stangl

Problem

You are given a route consisting of n stops in a fixed order, each with a travel time d from the previous stop and a time window $[s, e]$ during which you have to visit it. Additionally, there are m constraints indicating that the time between visiting stop a and stop b must be at most l. Find valid times to visit the stops, or determine that it is impossible.

Implementation in $\mathcal{O}((n+m)\log n)$

- For each stop, keep the maximum total delay before reaching it in a data structure (segment tree, `std::set`, etc.).
- When fixing a stop a
    - Select the minimum total delay from the data structure to determine how much to delay visiting a.

Apr 20: Easter

Problem author: David Stangl

Problem

You are given a route consisting of n stops in a fixed order, each with a travel time d from the previous stop and a time window $[s, e]$ during which you have to visit it. Additionally, there are m constraints indicating that the time between visiting stop a and stop b must be at most l. Find valid times to visit the stops, or determine that it is impossible.

Implementation in $\mathcal{O}((n+m)\log n)$

- For each stop, keep the maximum total delay before reaching it in a data structure (segment tree, `std::set`, etc.).
- When fixing a stop a
  - Select the minimum total delay from the data structure to determine how much to delay visiting a.
  - Remove the entry for a from the data structure.

Apr 20: Easter

Problem author: David Stangl

Problem

You are given a route consisting of n stops in a fixed order, each with a travel time d from the previous stop and a time window $[s, e]$ during which you have to visit it. Additionally, there are m constraints indicating that the time between visiting stop a and stop b must be at most l. Find valid times to visit the stops, or determine that it is impossible.

Implementation in $\mathcal{O}((n + m) \log n)$

- For each stop, keep the maximum total delay before reaching it in a data structure (segment tree, `std::set`, etc.).
- When fixing a stop a
  - Select the minimum total delay from the data structure to determine how much to delay visiting a.
  - Remove the entry for a from the data structure.
- When incorporating constraints into time windows of stops, update the stops entries in the data structure accordingly.

Apr 20: Easter

Problem author: David Stangl

Problem

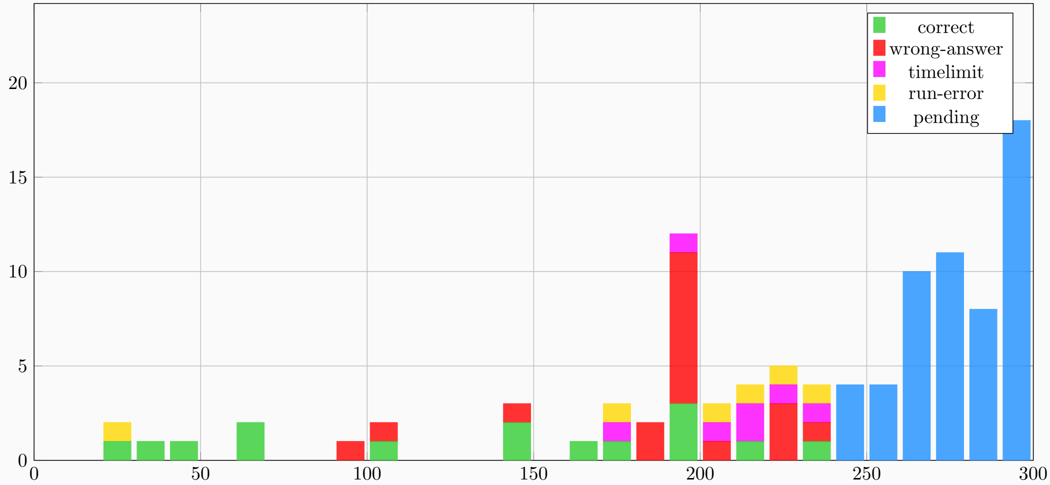
You are given a route consisting of n stops in a fixed order, each with a travel time d from the previous stop and a time window $[s, e]$ during which you have to visit it. Additionally, there are m constraints indicating that the time between visiting stop a and stop b must be at most l. Find valid times to visit the stops, or determine that it is impossible.

Alternative Solution

The problem can also be translated into a shortest path problem solvable with Dijkstra's algorithm. Due to the nature of the graph it can even be solved in $\mathcal{O}(n + m)$.

Jul 5: Dependence Day

Problem author: Christopher Weyand



Jul 5: Dependence Day

Problem author: Christopher Weyand

Problem

Given n intervals that represent table reservations and m intervals that represent waiter shifts. At each point in time, when there are a active reservations and b active shifts, then each of those b waiters is responsible for at most $\lceil a/b \rceil$ tables. For each shift, output the maximum number of tables that the waiter is responsible for at some point in time during their shift.

Jul 5: Dependence Day

Problem author: Christopher Weyand

Problem

Given n intervals that represent table reservations and m intervals that represent waiter shifts. At each point in time, when there are a active reservations and b active shifts, then each of those b waiters is responsible for at most $\lceil a/b \rceil$ tables. For each shift, output the maximum number of tables that the waiter is responsible for at some point in time during their shift.

Solution 1

- Process times when a reservation or shift starts or ends in sorted order while maintaining the current number of active shifts and occupied tables.

Jul 5: Dependence Day

Problem author: Christopher Weyand

Problem

Given n intervals that represent table reservations and m intervals that represent waiter shifts. At each point in time, when there are a active reservations and b active shifts, then each of those b waiters is responsible for at most $\lceil a/b \rceil$ tables. For each shift, output the maximum number of tables that the waiter is responsible for at some point in time during their shift.

Solution 1

- Process times when a reservation or shift starts or ends in sorted order while maintaining the current number of active shifts and occupied tables.
- Compute tables per waiter for each point in time. We call this *load*.

Jul 5: Dependence Day

Problem author: Christopher Weyand

Problem

Given n intervals that represent table reservations and m intervals that represent waiter shifts. At each point in time, when there are a active reservations and b active shifts, then each of those b waiters is responsible for at most $\lceil a/b \rceil$ tables. For each shift, output the maximum number of tables that the waiter is responsible for at some point in time during their shift.

Solution 1

- Process times when a reservation or shift starts or ends in sorted order while maintaining the current number of active shifts and occupied tables.
- Compute tables per waiter for each point in time. We call this *load*.
- Maintain the previous loads in a stack (the most recent on top).

Jul 5: Dependence Day

Problem author: Christopher Weyand

Problem

Given n intervals that represent table reservations and m intervals that represent waiter shifts. At each point in time, when there are a active reservations and b active shifts, then each of those b waiters is responsible for at most $\lceil a/b \rceil$ tables. For each shift, output the maximum number of tables that the waiter is responsible for at some point in time during their shift.

Solution 1

- Process times when a reservation or shift starts or ends in sorted order while maintaining the current number of active shifts and occupied tables.
- Compute tables per waiter for each point in time. We call this *load*.
- Maintain the previous loads in a stack (the most recent on top).
- Before the current load is pushed onto the stack, pop smaller loads from the top.

Jul 5: Dependence Day

Problem author: Christopher Weyand

Problem

Given n intervals that represent table reservations and m intervals that represent waiter shifts. At each point in time, when there are a active reservations and b active shifts, then each of those b waiters is responsible for at most $\lceil a/b \rceil$ tables. For each shift, output the maximum number of tables that the waiter is responsible for at some point in time during their shift.

Solution 1

- Process times when a reservation or shift starts or ends in sorted order while maintaining the current number of active shifts and occupied tables.
- Compute tables per waiter for each point in time. We call this *load*.
- Maintain the previous loads in a stack (the most recent on top).
- Before the current load is pushed onto the stack, pop smaller loads from the top.
- Thus the stack will be ordered ascending by load (from top to bottom).

Jul 5: Dependence Day

Problem author: Christopher Weyand

Problem

Given n intervals that represent table reservations and m intervals that represent waiter shifts. At each point in time, when there are a active reservations and b active shifts, then each of those b waiters is responsible for at most $\lceil a/b \rceil$ tables. For each shift, output the maximum number of tables that the waiter is responsible for at some point in time during their shift.

Solution 1

- Process times when a reservation or shift starts or ends in sorted order while maintaining the current number of active shifts and occupied tables.
- Compute tables per waiter for each point in time. We call this *load*.
- Maintain the previous loads in a stack (the most recent on top).
- Before the current load is pushed onto the stack, pop smaller loads from the top.
- Thus the stack will be ordered ascending by load (from top to bottom).
- If a shift ends, query the maximum load since the start of the shift via binary search on the stack.

Jul 5: Dependence Day

Problem author: Christopher Weyand

Problem

Given n intervals that represent table reservations and m intervals that represent waiter shifts. At each point in time, when there are a active reservations and b active shifts, then each of those b waiters is responsible for at most $\lceil a/b \rceil$ tables. For each shift, output the maximum number of tables that the waiter is responsible for at some point in time during their shift.

Solution 1

- Process times when a reservation or shift starts or ends in sorted order while maintaining the current number of active shifts and occupied tables.
- Compute tables per waiter for each point in time. We call this *load*.
- Maintain the previous loads in a stack (the most recent on top).
- Before the current load is pushed onto the stack, pop smaller loads from the top.
- Thus the stack will be ordered ascending by load (from top to bottom).
- If a shift ends, query the maximum load since the start of the shift via binary search on the stack.
- Runtime $\mathcal{O}((n+m)\log(n+m))$.

Jul 5: Dependence Day

Problem author: Christopher Weyand

Problem

Given n intervals that represent table reservations and m intervals that represent waiter shifts. At each point in time, when there are a active reservations and b active shifts, then each of those b waiters is responsible for at most $\lceil a/b \rceil$ tables. For each shift, output the maximum number of tables that the waiter is responsible for at some point in time during their shift.

Solution 2

- Do coordinate compression on all time points.

Jul 5: Dependence Day

Problem author: Christopher Weyand

Problem

Given n intervals that represent table reservations and m intervals that represent waiter shifts. At each point in time, when there are a active reservations and b active shifts, then each of those b waiters is responsible for at most $\lceil a/b \rceil$ tables. For each shift, output the maximum number of tables that the waiter is responsible for at some point in time during their shift.

Solution 2

- Do coordinate compression on all time points.
- Compute the load at each time like in Solution 1.

Jul 5: Dependence Day

Problem author: Christopher Weyand

Problem

Given n intervals that represent table reservations and m intervals that represent waiter shifts. At each point in time, when there are a active reservations and b active shifts, then each of those b waiters is responsible for at most $\lceil a/b \rceil$ tables. For each shift, output the maximum number of tables that the waiter is responsible for at some point in time during their shift.

Solution 2

- Do coordinate compression on all time points.
- Compute the load at each time like in Solution 1.
- From the loads, build any data structure that supports fast range-maximum queries. E.g. Segment Tree or Sparse Table.

Jul 5: Dependence Day

Problem author: Christopher Weyand

Problem

Given n intervals that represent table reservations and m intervals that represent waiter shifts. At each point in time, when there are a active reservations and b active shifts, then each of those b waiters is responsible for at most $\lceil a/b \rceil$ tables. For each shift, output the maximum number of tables that the waiter is responsible for at some point in time during their shift.

Solution 2

- Do coordinate compression on all time points.
- Compute the load at each time like in Solution 1.
- From the loads, build any data structure that supports fast range-maximum queries. E.g. Segment Tree or Sparse Table.
- Compute the result for each shift by querying the data structure.

Jul 5: Dependence Day

Problem author: Christopher Weyand

Problem

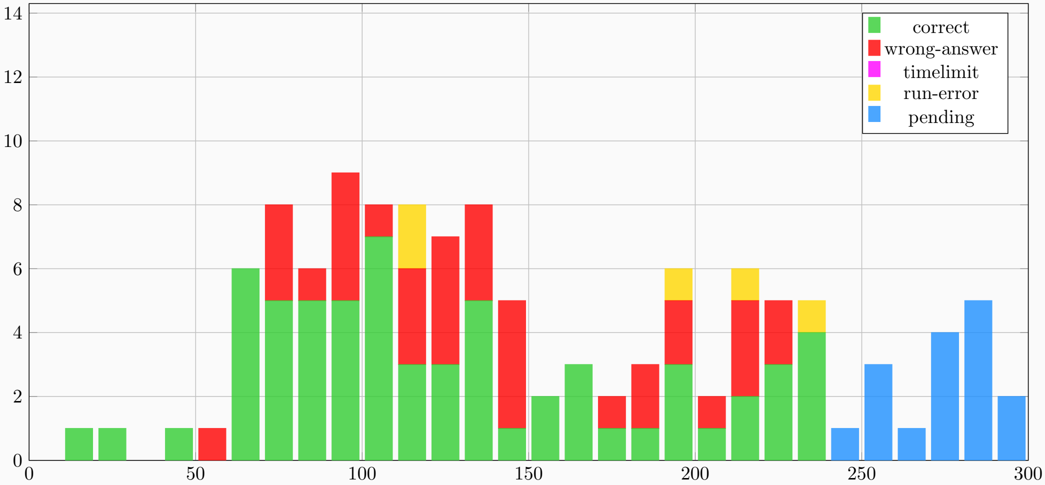
Given n intervals that represent table reservations and m intervals that represent waiter shifts. At each point in time, when there are a active reservations and b active shifts, then each of those b waiters is responsible for at most $\lceil a/b \rceil$ tables. For each shift, output the maximum number of tables that the waiter is responsible for at some point in time during their shift.

Solution 2

- Do coordinate compression on all time points.
- Compute the load at each time like in Solution 1.
- From the loads, build any data structure that supports fast range-maximum queries. E.g. Segment Tree or Sparse Table.
- Compute the result for each shift by querying the data structure.
- Runtime $\mathcal{O}((n+m)\log(n+m))$.

Jul 25: Sysadmin Day

Problem author: Paul Wild



Jul 25: Sysadmin Day

Problem author: Paul Wild

Problem

- Print an image consisting of pixels in up to eight colours using a CMYK printer.
- Toners are available in cyan, magenta, yellow and black.
- White, red, green and blue pixels can be achieved using subtractive colour mixing.
- Given the amount and per-pixel cost of each toner, minimize the total cost.

Jul 25: Sysadmin Day

Problem author: Paul Wild

Problem

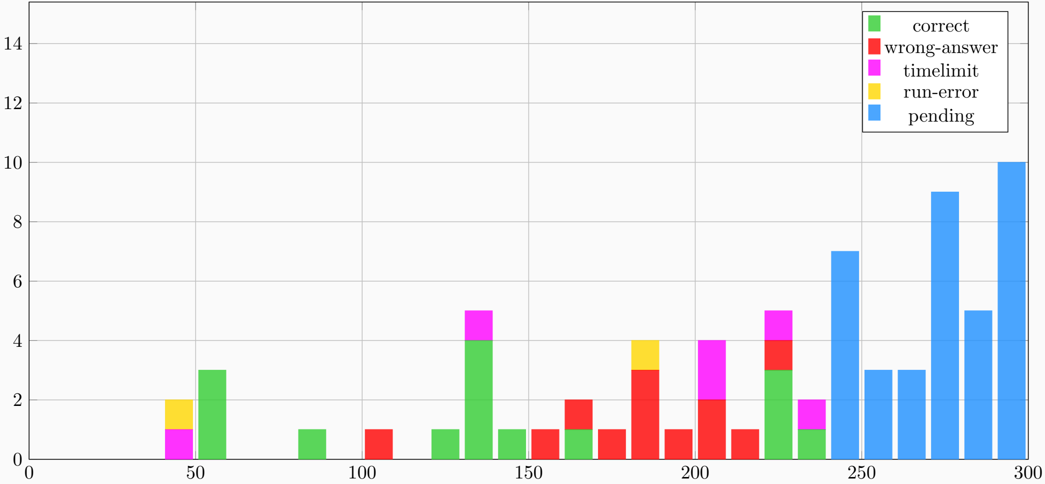
- Print an image consisting of pixels in up to eight colours using a CMYK printer.
- Toners are available in cyan, magenta, yellow and black.
- White, red, green and blue pixels can be achieved using subtractive colour mixing.
- Given the amount and per-pixel cost of each toner, minimize the total cost.

Solution

- Each colour except for black can be printed in a unique way.
- Print all of these in a first pass and record how much toner of each type is left.
- In a second pass, greedily choose the cheapest option for each black pixel.
- Running time: $\mathcal{O}(h \cdot w)$

Jul 30: Friendship Day

Problem author: Jannik Olbrich



Jul 30: Friendship Day

Problem author: Jannik Olbrich

Problem

Given a Graph, map each vertex to a line such that two lines intersect iff the corresponding vertices are neighbours

Solution

- Two lines intersect iff they are not parallel

Jul 30: Friendship Day

Problem author: Jannik Olbrich

Problem

Given a Graph, map each vertex to a line such that two lines intersect iff the corresponding vertices are neighbours

Solution

- Two lines intersect iff they are not parallel
- The line of a vertex v must be parallel to the line of every u that is not a neighbour of v

Jul 30: Friendship Day

Problem author: Jannik Olbrich

Problem

Given a Graph, map each vertex to a line such that two lines intersect iff the corresponding vertices are neighbours

Solution

- Two lines intersect iff they are not parallel
- The line of a vertex v must be parallel to the line of every u that is not a neighbour of v
- This is possible iff the complement of the graph is a disjoint union of cliques

Jul 30: Friendship Day

Problem author: Jannik Olbrich

Problem

Given a Graph, map each vertex to a line such that two lines intersect iff the corresponding vertices are neighbours

Solution

- Two lines intersect iff they are not parallel
- The line of a vertex v must be parallel to the line of every u that is not a neighbour of v
- This is possible iff the complement of the graph is a disjoint union of cliques
- But the complement graph has way too many edges...

Jul 30: Friendship Day

Problem author: Jannik Olbrich

Problem

Given a Graph, map each vertex to a line such that two lines intersect iff the corresponding vertices are neighbours

Solution

- Two lines intersect iff they are not parallel
- The line of a vertex v must be parallel to the line of every u that is not a neighbour of v
- This is possible iff the complement of the graph is a disjoint union of cliques
- But the complement graph has way too many edges...
- Use a DSU data structure to maintain the cliques: Loop over the vertices that are the representative element of their set, and union each with all sets that are not adjacent to this vertex.

Jul 30: Friendship Day

Problem author: Jannik Olbrich

Problem

Given a Graph, map each vertex to a line such that two lines intersect iff the corresponding vertices are neighbours

Solution

- Two lines intersect iff they are not parallel
- The line of a vertex v must be parallel to the line of every u that is not a neighbour of v
- This is possible iff the complement of the graph is a disjoint union of cliques
- But the complement graph has way too many edges...
- Use a DSU data structure to maintain the cliques: Loop over the vertices that are the representative element of their set, and union each with all sets that are not adjacent to this vertex.
- Now we just need to ensure that the set of edges exactly matches the set of edges between the DSU-sets

Jul 30: Friendship Day

Problem author: Jannik Olbrich

Problem

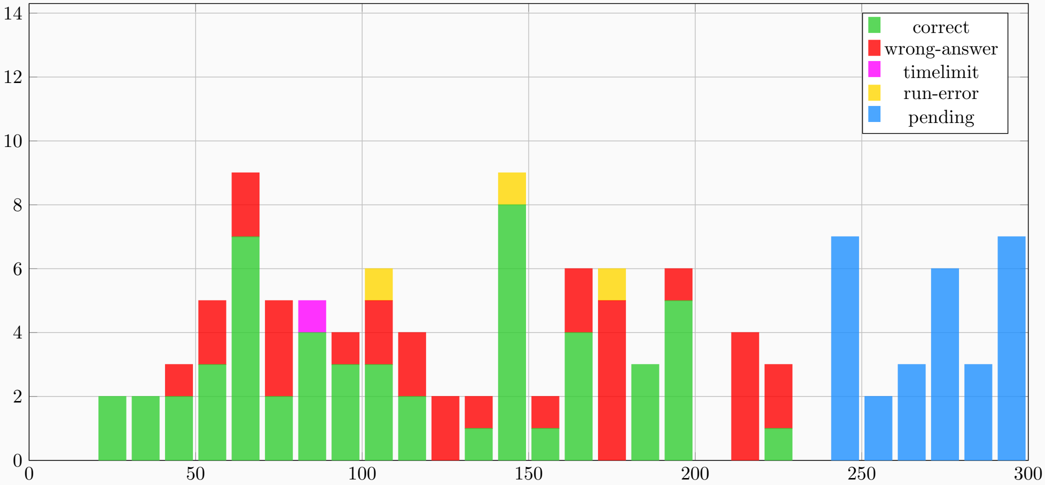
Given a Graph, map each vertex to a line such that two lines intersect iff the corresponding vertices are neighbours

Solution

- Two lines intersect iff they are not parallel
- The line of a vertex v must be parallel to the line of every u that is not a neighbour of v
- This is possible iff the complement of the graph is a disjoint union of cliques
- But the complement graph has way too many edges...
- Use a DSU data structure to maintain the cliques: Loop over the vertices that are the representative element of their set, and union each with all sets that are not adjacent to this vertex.
- Now we just need to ensure that the set of edges exactly matches the set of edges between the DSU-sets
- Time complexity: $\mathcal{O}(n + m \log^* n)$

Nov 14: Domino Day

Problem author: Lucas Alber



Nov 14: Domino Day

Problem author: Lucas Alber

Problem

Given 2^n colors, compute a sequence of colors, such that all permutations of $[1, \ldots, 2^n]$ that are obtainable by swapping at inner nodes of an implicit binary tree are contained as subsequence.

Nov 14: Domino Day

Problem author: Lucas Alber

Problem

Given 2^n colors, compute a sequence of colors, such that all permutations of $[1, \ldots, 2^n]$ that are obtainable by swapping at inner nodes of an implicit binary tree are contained as subsequence.

Observation

- $n \Rightarrow n+1$: The new colors never mix with the old colors using the described permutations.
- We can reuse the same solution and obtain a shortest sequence B for the new colors by adding 2^n to the current sequence A.
- To allow for the colors to appear in any order, the resulting sequence has to have the form ABA.

Nov 14: Domino Day

Problem author: Lucas Alber

Problem

Given 2^n colors, compute a sequence of colors, such that all permutations of $[1, ..., 2^n]$ that are obtainable by swapping at inner nodes of an implicit binary tree are contained as subsequence.

Observation

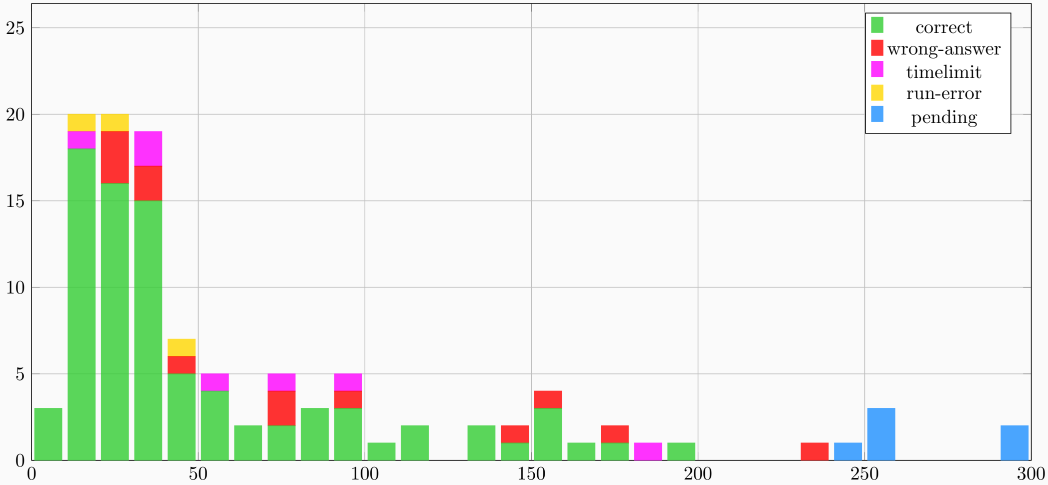
- $n \Rightarrow n+1$: The new colors never mix with the old colors using the described permutations.
- We can reuse the same solution and obtain a shortest sequence B for the new colors by adding 2^n to the current sequence A.
- To allow for the colors to appear in any order, the resulting sequence has to have the form ABA.

Solution

- Start with $i = 0, s = [1]$.
- Obtain the next sequence as $s * (s + 2^i) * s$ and set $i = i + 1$.
- Repeat until $i = n$.

Nov 27: Thanksgiving

Problem author: Christopher Weyand



Nov 27: Thanksgiving

Problem author: Christopher Weyand

Problem

Given a graph with one outgoing edge per node. How many nodes are reachable from node 1?

Nov 27: Thanksgiving

Problem author: Christopher Weyand

Problem

Given a graph with one outgoing edge per node. How many nodes are reachable from node 1?

Solution

- Follow the edges starting from node 1.

Nov 27: Thanksgiving

Problem author: Christopher Weyand

Problem

Given a graph with one outgoing edge per node. How many nodes are reachable from node 1?

Solution

- Follow the edges starting from node 1.
- Mark all visited nodes.

Nov 27: Thanksgiving

Problem author: Christopher Weyand

Problem

Given a graph with one outgoing edge per node. How many nodes are reachable from node 1?

Solution

- Follow the edges starting from node 1.
- Mark all visited nodes.
- Stop once you encounter a node that is already marked (cycle).

Nov 27: Thanksgiving

Problem author: Christopher Weyand

Problem

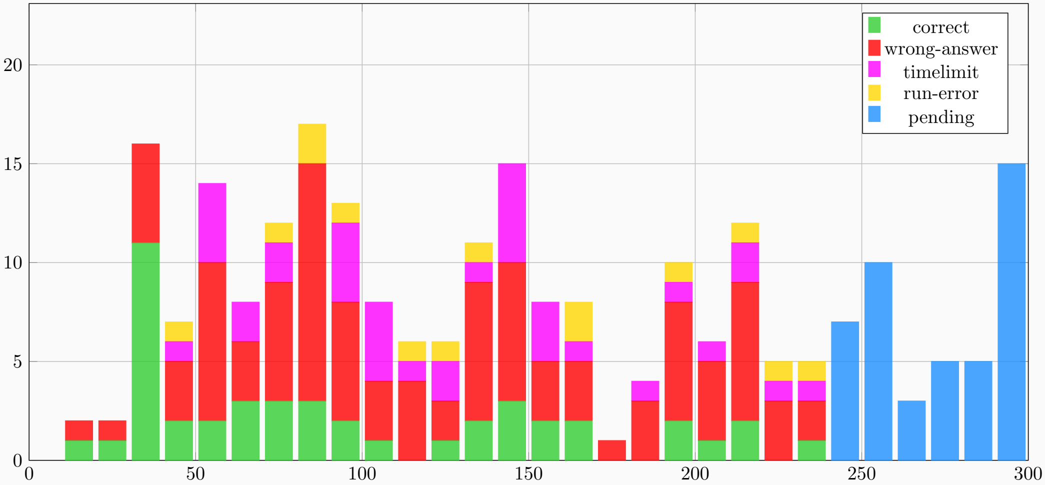
Given a graph with one outgoing edge per node. How many nodes are reachable from node 1?

Solution

- Follow the edges starting from node 1.
- Mark all visited nodes.
- Stop once you encounter a node that is already marked (cycle).
- Count how many nodes are marked in the end.

Dec 1: Advent

Problem author: Wendy Yi



Dec 1: Advent

Problem author: Wendy Yi

Problem

Given n chocolate pieces of k types, arrange them such that no two pieces of the same type are next to each other.

Dec 1: Advent

Problem author: Wendy Yi

Problem

Given n chocolate pieces of k types, arrange them such that no two pieces of the same type are next to each other.

Observation

- This is impossible if there are more than $\lceil \frac{n}{2} \rceil$ pieces of the same type, otherwise possible.

Dec 1: Advent

Problem author: Wendy Yi

Problem

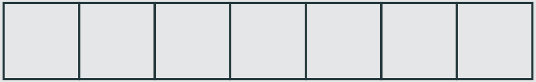
Given n chocolate pieces of k types, arrange them such that no two pieces of the same type are next to each other.

Observation

- This is impossible if there are more than $\lceil \frac{n}{2} \rceil$ pieces of the same type, otherwise possible.

Solution

- Sort the types in decreasing order.
- Starting with the most common type, fill every other day with a piece.
- After reaching the end, repeat the process, starting with the second day.



Dec 1: Advent

Problem author: Wendy Yi

Problem

Given n chocolate pieces of k types, arrange them such that no two pieces of the same type are next to each other.

Observation

- This is impossible if there are more than $\lceil \frac{n}{2} \rceil$ pieces of the same type, otherwise possible.

Solution

- Sort the types in decreasing order.
- Starting with the most common type, fill every other day with a piece.
- After reaching the end, repeat the process, starting with the second day.

2						

Dec 1: Advent

Problem author: Wendy Yi

Problem

Given n chocolate pieces of k types, arrange them such that no two pieces of the same type are next to each other.

Observation

- This is impossible if there are more than $\lceil \frac{n}{2} \rceil$ pieces of the same type, otherwise possible.

Solution

- Sort the types in decreasing order.
- Starting with the most common type, fill every other day with a piece.
- After reaching the end, repeat the process, starting with the second day.

2		2				

Dec 1: Advent

Problem author: Wendy Yi

Problem

Given n chocolate pieces of k types, arrange them such that no two pieces of the same type are next to each other.

Observation

- This is impossible if there are more than $\lceil \frac{n}{2} \rceil$ pieces of the same type, otherwise possible.

Solution

- Sort the types in decreasing order.
- Starting with the most common type, fill every other day with a piece.
- After reaching the end, repeat the process, starting with the second day.

2		2		2		

Dec 1: Advent

Problem author: Wendy Yi

Problem

Given n chocolate pieces of k types, arrange them such that no two pieces of the same type are next to each other.

Observation

- This is impossible if there are more than $\lceil \frac{n}{2} \rceil$ pieces of the same type, otherwise possible.

Solution

- Sort the types in decreasing order.
- Starting with the most common type, fill every other day with a piece.
- After reaching the end, repeat the process, starting with the second day.

2		2		2		1

Dec 1: Advent

Problem author: Wendy Yi

Problem

Given n chocolate pieces of k types, arrange them such that no two pieces of the same type are next to each other.

Observation

- This is impossible if there are more than $\lceil \frac{n}{2} \rceil$ pieces of the same type, otherwise possible.

Solution

- Sort the types in decreasing order.
- Starting with the most common type, fill every other day with a piece.
- After reaching the end, repeat the process, starting with the second day.

2	1	2		2		1

Dec 1: Advent

Problem author: Wendy Yi

Problem

Given n chocolate pieces of k types, arrange them such that no two pieces of the same type are next to each other.

Observation

- This is impossible if there are more than $\lceil \frac{n}{2} \rceil$ pieces of the same type, otherwise possible.

Solution

- Sort the types in decreasing order.
- Starting with the most common type, fill every other day with a piece.
- After reaching the end, repeat the process, starting with the second day.

2	1	2	3	2		1

Dec 1: Advent

Problem author: Wendy Yi

Problem

Given n chocolate pieces of k types, arrange them such that no two pieces of the same type are next to each other.

Observation

- This is impossible if there are more than $\lceil \frac{n}{2} \rceil$ pieces of the same type, otherwise possible.

Solution

- Sort the types in decreasing order.
- Starting with the most common type, fill every other day with a piece.
- After reaching the end, repeat the process, starting with the second day.

2	1	2	3	2	4	1

Dec 1: Advent

Problem author: Wendy Yi

Problem

Given n chocolate pieces of k types, arrange them such that no two pieces of the same type are next to each other.

Observation

- This is impossible if there are more than $\lceil \frac{n}{2} \rceil$ pieces of the same type, otherwise possible.

Solution

- Sort the types in decreasing order.
- Starting with the most common type, fill every other day with a piece.
- After reaching the end, repeat the process, starting with the second day.

2	1	2	3	2	4	1

Running time: $O(n \log(n))$

Dec 1: Advent

Problem author: Wendy Yi

Problem

Given n chocolate pieces of k types, arrange them such that no two pieces of the same type are next to each other.

Observation

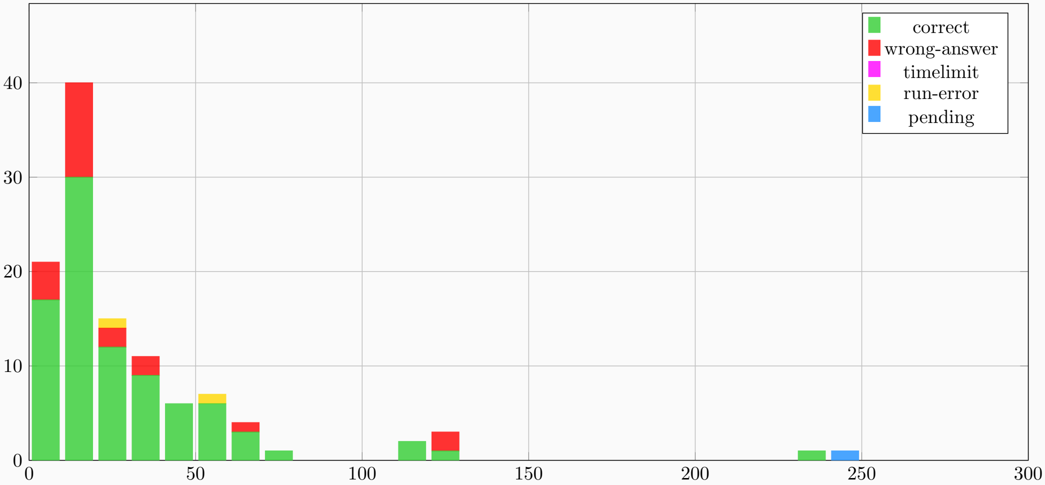
- This is impossible if there are more than $\lceil \frac{n}{2} \rceil$ pieces of the same type, otherwise possible.

Alternative solution

- Fill each day with a piece of currently most common type that is allowed to use (i.e, not used yesterday).
- Keep track of number of pieces per type and current maximum using a priority queue.
- Running time: $O(n \log(n))$

Dec 31: New Year's Eve

Problem author: Christopher Weyand



Dec 31: New Year’s Eve

Problem author: Christopher Weyand

Problem

You are planning n trips that each cost 100 Euro. Bahncard X gives X% discount on all trips. You can buy Bahncard 25, 50, or 100 for a, b, c Euro, respectively. Should you buy a Bahncard, and if so, which one?

Dec 31: New Year's Eve

Problem author: Christopher Weyand

Problem

You are planning n trips that each cost 100 Euro. Bahncard X gives X% discount on all trips. You can buy Bahncard 25, 50, or 100 for a, b, c Euro, respectively. Should you buy a Bahncard, and if so, which one?

Total Costs

- No Bahncard: $100n$

Dec 31: New Year's Eve

Problem author: Christopher Weyand

Problem

You are planning n trips that each cost 100 Euro. Bahncard X gives X% discount on all trips. You can buy Bahncard 25, 50, or 100 for a, b, c Euro, respectively. Should you buy a Bahncard, and if so, which one?

Total Costs

- No Bahncard: $100n$
- Bahncard 25: $75n + a$

Dec 31: New Year's Eve

Problem author: Christopher Weyand

Problem

You are planning n trips that each cost 100 Euro. Bahncard X gives X% discount on all trips. You can buy Bahncard 25, 50, or 100 for a, b, c Euro, respectively. Should you buy a Bahncard, and if so, which one?

Total Costs

- No Bahncard: $100n$
- Bahncard 25: $75n + a$
- Bahncard 50: $50n + b$

Dec 31: New Year's Eve

Problem author: Christopher Weyand

Problem

You are planning n trips that each cost 100 Euro. Bahncard X gives X% discount on all trips. You can buy Bahncard 25, 50, or 100 for a, b, c Euro, respectively. Should you buy a Bahncard, and if so, which one?

Total Costs

- No Bahncard: $100n$
- Bahncard 25: $75n + a$
- Bahncard 50: $50n + b$
- Bahncard 100: c

Dec 31: New Year's Eve

Problem author: Christopher Weyand

Problem

You are planning n trips that each cost 100 Euro. Bahncard X gives X% discount on all trips. You can buy Bahncard 25, 50, or 100 for a, b, c Euro, respectively. Should you buy a Bahncard, and if so, which one?

Total Costs

- No Bahncard: $100n$
- Bahncard 25: $75n + a$
- Bahncard 50: $50n + b$
- Bahncard 100: c

Dec 31: New Year's Eve

Problem author: Christopher Weyand

Problem

You are planning n trips that each cost 100 Euro. Bahncard X gives X% discount on all trips. You can buy Bahncard 25, 50, or 100 for a, b, c Euro, respectively. Should you buy a Bahncard, and if so, which one?

Total Costs

- No Bahncard: $100n$
- Bahncard 25: $75n + a$
- Bahncard 50: $50n + b$
- Bahncard 100: c

Solution

Compute the cost for each option and print the cheapest.

Random facts

Jury work

- 652 secret test cases (≈ 50 per problem)

Random facts

Jury work

- 652 secret test cases ($\approx$ 50 per problem)
- 119 jury solutions

Random facts

Jury work

- 652 secret test cases (≈ 50 per problem)
- 119 jury solutions
- The minimum number of lines the jury needed to solve all problems is

$$1 + 41 + 110 + 14 + 48 + 39 + 37 + 6 + 13 + 2 + 2 + 5 + 2 = 320$$

  On average 24.6 lines per problem

Random facts

Jury work

- 652 secret test cases (≈ 50 per problem)
- 119 jury solutions
- The minimum number of lines the jury needed to solve all problems is

$$1 + 41 + 110 + 14 + 48 + 39 + 37 + 6 + 13 + 2 + 2 + 5 + 2 = 320$$

  On average 24.6 lines per problem
- The minimum number of characters the jury needed to solve all problems is

$$102 + 1421 + 4397 + 336 + 1191 + 1259 + 929 + 431 + 692 + 119 + 124 + 207 + 166$$

  On average 875 characters per problem