NWERC 2021 presentation of solutions

November 21, 2021

NWERC 2021 Jury

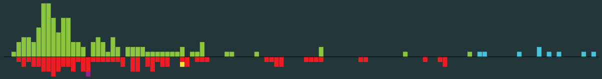
- **Per Austrin**
  KTH Royal Institute of Technology
- **Alexander Dietsch**
  e.solutions
- **Ragnar Groot Koerkamp**
  ETH Zurich
- **Antti Laaksonen**
  CSES
- **Bjarki Ágúst Guðmundsson**
  Google
- **Nils Gustafsson**
  KTH Royal Institute of Technology
- **Timon Knigge**
  ETH Zurich
- **Harry Smit**
  MPIM Bonn
- **Bergur Snorrason**
  University of Iceland
- **Pehr Söderman**
  Kattis
- **Jorke de Vlas**
  Utrecht University
- **Mees de Vries**
  IMC
- **Paul Wild**
  FAU Erlangen-Nürnberg
- **Michael Zündorf**
  Karlsruhe Institute of Technology

Big thanks to our test solvers

- **Bernhard Linn Hilmarsson**
  ETH Zurich
- **Robin Lee**
  Google
- **Ludo Pulles**
  Leiden University
- **Johan Sannemo**
  Kattis
- **Reinier Schmiermann**
  Utrecht University
- **Tobias Werth**
  Google

K: Knitpicking

Problem Author: Pehr Söderman

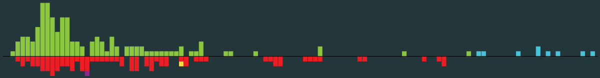


Problem

Given a drawer full of socks, compute how many you need to pick to be guaranteed to have a pair.

K: Knitpicking

Problem Author: Pehr Söderman



Problem

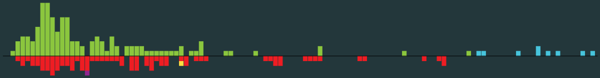
Given a drawer full of socks, compute how many you need to pick to be guaranteed to have a pair.

Solution

- Count the number of socks you can pick *without* a pair, then add 1 at the end.

K: Knitpicking

Problem Author: Pehr Söderman



Problem

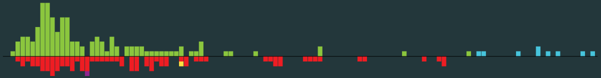
Given a drawer full of socks, compute how many you need to pick to be guaranteed to have a pair.

Solution

- Count the number of socks you can pick *without* a pair, then add 1 at the end.
- For every type of socks you can pick $\max(\texttt{left}, \texttt{right}, 1)$ socks: all socks of one side, or 1 any sock.

K: Knitpicking

Problem Author: Pehr Söderman



Problem

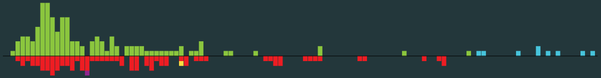
Given a drawer full of socks, compute how many you need to pick to be guaranteed to have a pair.

Solution

- Count the number of socks you can pick *without* a pair, then add 1 at the end.
- For every type of socks you can pick $\max(\texttt{left}, \texttt{right}, 1)$ socks: all socks of one side, or 1 any sock.
- Remember to output `impossible` when every sock type only has `left` socks, `right` socks, or a single `any` sock.

K: Knitpicking

Problem Author: Pehr Söderman



Problem

Given a drawer full of socks, compute how many you need to pick to be guaranteed to have a pair.

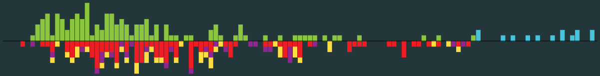
Solution

- Count the number of socks you can pick *without* a pair, then add 1 at the end.
- For every type of socks you can pick max(`left`, `right`, 1) socks: all socks of one side, or 1 any sock.
- Remember to output `impossible` when every sock type only has `left` socks, `right` socks, or a single `any` sock.

Statistics: 218 submissions, 126 accepted, 9 unknown

A: Access Denied

Problem Author: Pehr Söderman

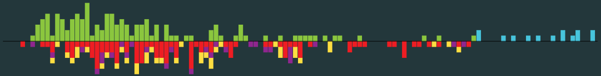


Problem

Use the timing of a password checker to guess a password.

A: Access Denied

Problem Author: Pehr Söderman



Problem

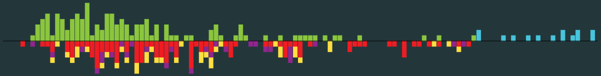
Use the timing of a password checker to guess a password.

Solution

- First find the length: guess passwords of lengths 1 to 20. The one with the longest timeout gives you the length.

A: Access Denied

Problem Author: Pehr Söderman



Problem

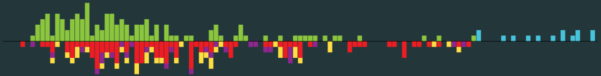
Use the timing of a password checker to guess a password.

Solution

- First find the length: guess passwords of lengths 1 to 20. The one with the longest timeout gives you the length.
- Once you have the length of the password, guess the letters one by one:

A: Access Denied

Problem Author: Pehr Söderman



Problem

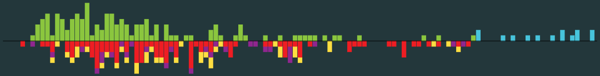
Use the timing of a password checker to guess a password.

Solution

- First find the length: guess passwords of lengths 1 to 20. The one with the longest timeout gives you the length.
- Once you have the length of the password, guess the letters one by one:
  - Iterate over all options (lowercase letters, uppercase letters, digits) until the timeout increases.

A: Access Denied

Problem Author: Pehr Söderman



Problem

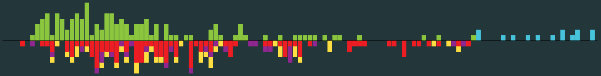
Use the timing of a password checker to guess a password.

Solution

- First find the length: guess passwords of lengths 1 to 20. The one with the longest timeout gives you the length.
- Once you have the length of the password, guess the letters one by one:
  - Iterate over all options (lowercase letters, uppercase letters, digits) until the timeout increases.
  - Then leave that letter and move on to the next one.

A: Access Denied

Problem Author: Pehr Söderman



Problem

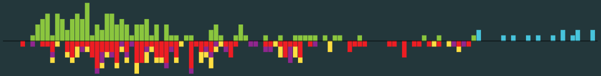
Use the timing of a password checker to guess a password.

Solution

- First find the length: guess passwords of lengths 1 to 20. The one with the longest timeout gives you the length.
- Once you have the length of the password, guess the letters one by one:
  - Iterate over all options (lowercase letters, uppercase letters, digits) until the timeout increases.
  - Then leave that letter and move on to the next one.
- This takes no more than $20 + 20 \times 62 = 1260$ queries.

A: Access Denied

Problem Author: Pehr Söderman



Problem

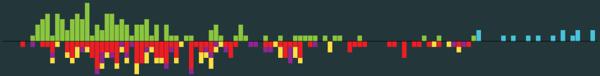
Use the timing of a password checker to guess a password.

Solution

- First find the length: guess passwords of lengths 1 to 20. The one with the longest timeout gives you the length.
- Once you have the length of the password, guess the letters one by one:
  - Iterate over all options (lowercase letters, uppercase letters, digits) until the timeout increases.
  - Then leave that letter and move on to the next one.
- This takes no more than $20 + 20 \times 62 = 1260$ queries.
- This is more or less your only option: the timeout doesn't give you any other information.

A: Access Denied

Problem Author: Pehr Söderman



Problem

Use the timing of a password checker to guess a password.

Solution

- First find the length: guess passwords of lengths 1 to 20. The one with the longest timeout gives you the length.
- Once you have the length of the password, guess the letters one by one:
  - Iterate over all options (lowercase letters, uppercase letters, digits) until the timeout increases.
  - Then leave that letter and move on to the next one.
- This takes no more than $20 + 20 \times 62 = 1260$ queries.
- This is more or less your only option: the timeout doesn't give you any other information.

Statistics: 300 submissions, 118 accepted, 14 unknown

J: Jet Set

Problem Author: Paul Wild

Problem

Given a list of stops on a trip, determine whether it passes through every meridian.

J: Jet Set

Problem Author: Paul Wild

Problem

Given a list of stops on a trip, determine whether it passes through every meridian.

Solution

- Observations:

J: Jet Set

Problem Author: Paul Wild

Problem

Given a list of stops on a trip, determine whether it passes through every meridian.

Solution

- Observations:
  - You can ignore the latitudes – they do not matter.

J: Jet Set

Problem Author: Paul Wild

Problem

Given a list of stops on a trip, determine whether it passes through every meridian.

Solution

- Observations:
  - You can ignore the latitudes – they do not matter.
  - If the longitude ever changes by 180 in a single flight, the trip goes over one of the poles, so the answer is yes.

J: Jet Set

Problem Author: Paul Wild

Problem

Given a list of stops on a trip, determine whether it passes through every meridian.

Solution

- Observations:
  - You can ignore the latitudes – they do not matter.
  - If the longitude ever changes by 180 in a single flight, the trip goes over one of the poles, so the answer is yes.
- Naïve solution:

J: Jet Set

Problem Author: Paul Wild

Problem

Given a list of stops on a trip, determine whether it passes through every meridian.

Solution

- Observations:
  - You can ignore the latitudes – they do not matter.
  - If the longitude ever changes by 180 in a single flight, the trip goes over one of the poles, so the answer is yes.
- Naïve solution:
  - Keep an array of 720 booleans, one for each meridian and half-meridian.

J: Jet Set

Problem Author: Paul Wild

Problem

Given a list of stops on a trip, determine whether it passes through every meridian.

Solution

- Observations:
  - You can ignore the latitudes – they do not matter.
  - If the longitude ever changes by 180 in a single flight, the trip goes over one of the poles, so the answer is yes.
- Naïve solution:
  - Keep an array of 720 booleans, one for each meridian and half-meridian.
  - When travelling to a new longitude, loop over the array and set the visited longitudes to true.

J: Jet Set

Problem Author: Paul Wild

Problem

Given a list of stops on a trip, determine whether it passes through every meridian.

Solution

- Observations:
  - You can ignore the latitudes – they do not matter.
  - If the longitude ever changes by 180 in a single flight, the trip goes over one of the poles, so the answer is `yes`.
- Naïve solution:
  - Keep an array of 720 booleans, one for each meridian and half-meridian.
  - When travelling to a new longitude, loop over the array and set the visited longitudes to true.
  - Finally, output `yes` if every element of the array is `true`, and `no` otherwise.

J: Jet Set

Problem Author: Paul Wild

Problem

Given a list of stops on a trip, determine whether it passes through every meridian.

Solution

- Observations:
  - You can ignore the latitudes – they do not matter.
  - If the longitude ever changes by 180 in a single flight, the trip goes over one of the poles, so the answer is `yes`.
- Naïve solution:
  - Keep an array of 720 booleans, one for each meridian and half-meridian.
  - When travelling to a new longitude, loop over the array and set the visited longitudes to true.
  - Finally, output `yes` if every element of the array is `true`, and `no` otherwise.
- This naïve solution is correct!

Problem

Given a list of stops on a trip, determine whether it passes through every meridian.

Solution

- Observations:
  - You can ignore the latitudes – they do not matter.
  - If the longitude ever changes by 180 in a single flight, the trip goes over one of the poles, so the answer is `yes`.
- Naïve solution:
  - Keep an array of 720 booleans, one for each meridian and half-meridian.
  - When travelling to a new longitude, loop over the array and set the visited longitudes to true.
  - Finally, output `yes` if every element of the array is `true`, and `no` otherwise.
- This naïve solution is correct!
- Pitfalls: be careful to correctly operate on the circular array.

J: Jet Set

Problem Author: Paul Wild

Problem

Given a list of stops on a trip, determine whether it passes through every meridian.

Solution

- Observations:
  - You can ignore the latitudes – they do not matter.
  - If the longitude ever changes by 180 in a single flight, the trip goes over one of the poles, so the answer is `yes`.
- Naïve solution:
  - Keep an array of 720 booleans, one for each meridian and half-meridian.
  - When travelling to a new longitude, loop over the array and set the visited longitudes to true.
  - Finally, output `yes` if every element of the array is `true`, and `no` otherwise.
- This naïve solution is correct!
- Pitfalls: be careful to correctly operate on the circular array.

Statistics: 342 submissions, 81 accepted, 74 unknown

J: Jet Set

Problem Author: Paul Wild

Edge case

Don't forget the edge case of going around for 359° degrees and then turning around!

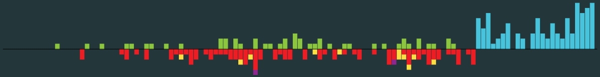
Edge case

```
--- Original
+++ New
@@ @@
         cout <<setprecision(1) << fixed;
         double dres = res/2.0;
         double unfix = dres >= M/2 ? dres -M : dres;
-        cout << unfix << "\n";
+        cout << "no " <<  unfix << "\n";
     }
 }
```

Please read the output section carefully.

D: Dyson Circle

Problem Author: Mees de Vries



Problem

Given some stars on a grid, encircle these with as few other grid points as possible.

D: Dyson Circle

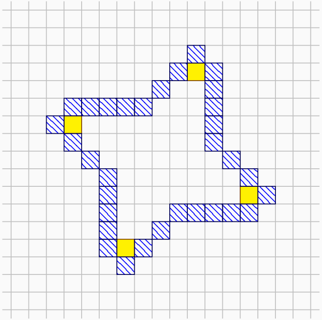
Problem Author: Mees de Vries

Problem

Given some stars on a grid, encircle these with as few other grid points as possible.

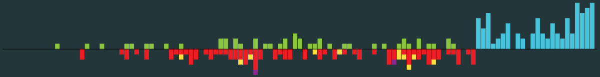
Solution

- Let's look at the first sample.



D: Dyson Circle

Problem Author: Mees de Vries

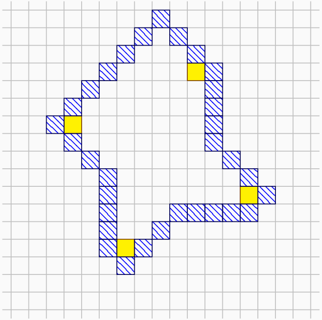


Problem

Given some stars on a grid, encircle these with as few other grid points as possible.

Solution

- Let's look at the first sample.
- We might as well remove a "dent" in our Dyson circle.



D: Dyson Circle

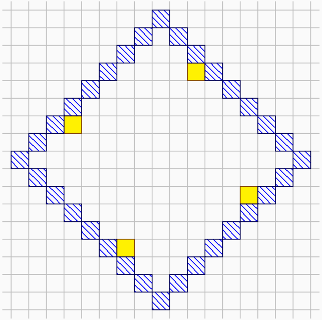
Problem Author: Mees de Vries

Problem

Given some stars on a grid, encircle these with as few other grid points as possible.

Solution

- Let's look at the first sample.
- We might as well remove a "dent" in our Dyson circle.
- In fact, we can do this with all dents.



D: Dyson Circle

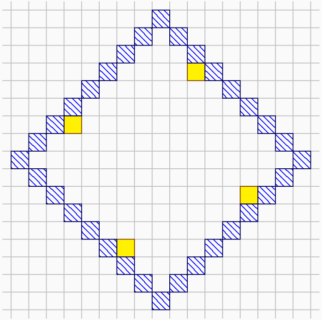
Problem Author: Mees de Vries

Problem

Given some stars on a grid, encircle these with as few other grid points as possible.

Solution

- Let's look at the first sample.
- We might as well remove a "dent" in our Dyson circle.
- In fact, we can do this with all dents.
- In general, a rectangle with diagonal edges is *always* an optimal solution.



D: Dyson Circle

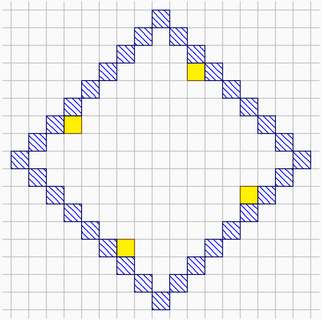
Problem Author: Mees de Vries

Problem

Given some stars on a grid, encircle these with as few other grid points as possible.

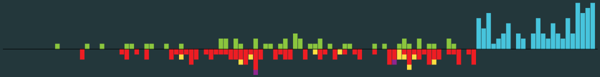
Solution

- The only suns that matter are the four suns that touch the edges of the rectangle: the ones that maximize $x + y$, $x - y$, $-x + y$, $-x - y$.



D: Dyson Circle

Problem Author: Mees de Vries



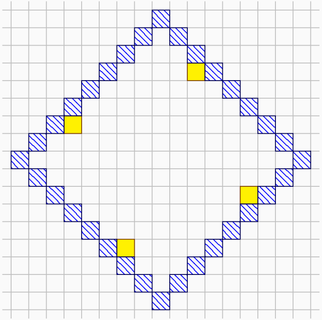
Problem

Given some stars on a grid, encircle these with as few other grid points as possible.

Solution

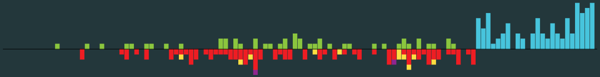
- The only suns that matter are the four suns that touch the edges of the rectangle: the ones that maximize $x+y$, $x-y$, $-x+y$, $-x-y$.
- So the general answer is

$$4 + \max_i(x_i + y_i) + \max_i(x_i - y_i) + \max_i(-x_i + y_i) + \max_i(-x_i - y_i).$$



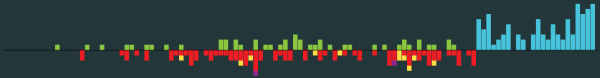
D: Dyson Circle

Problem Author: Mees de Vries



Problem

Given some stars on a grid, encircle these with as few other grid points as possible.



D: Dyson Circle

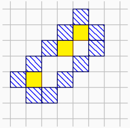
Problem Author: Mees de Vries

Problem

Given some stars on a grid, encircle these with as few other grid points as possible.

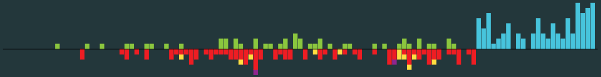
Gotchas

- If all of the suns are on a diagonal, you need one additional square to make the inside a contiguous region.



D: Dyson Circle

Problem Author: Mees de Vries

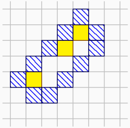


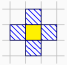
Problem

Given some stars on a grid, encircle these with as few other grid points as possible.

Gotchas

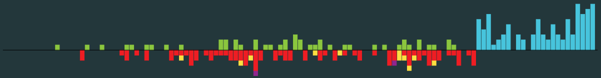
- If all of the suns are on a diagonal, you need one additional square to make the inside a contiguous region.
- However, if there is only one sun you do not need the additional square.





D: Dyson Circle

Problem Author: Mees de Vries

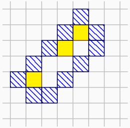


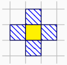
Problem

Given some stars on a grid, encircle these with as few other grid points as possible.

Gotchas

- If all of the suns are on a diagonal, you need one additional square to make the inside a contiguous region.
- However, if there is only one sun you do not need the additional square.





Statistics: 248 submissions, 48 accepted, 99 unknown

G: Glossary Arrangement

Problem Author: Jorke de Vlas

Problem

Given an alphabetical list of n words, split the list up into multiple columns so that the layout is at most w characters wide and the height is minimised.

```
user@pc ~/glossary $ ls
algorithm programming
contest   regional
eindhoven reykjavik
icpc      ru
nwerc
```

```
user@pc ~/glossary $ ls--
algorithm icpc  programming ru
contest   nwerc regional
eindhoven       reykjavik
```

G: Glossary Arrangement

Problem Author: Jorke de Vlas

Solution

- The answer can be found using binary search.

G: Glossary Arrangement

Problem Author: Jorke de Vlas

Solution

- The answer can be found using binary search.
- New problem: Is there a layout of height at most h?

Solution

- The answer can be found using binary search.
- New problem: Is there a layout of height at most h?
- Given h, solve the new problem using dynamic programming:

$$f(i) = \text{minimal width needed to split the first } i \text{ words into columns}$$

G: Glossary Arrangement

Problem Author: Jorke de Vlas

Solution

- The answer can be found using binary search.
- New problem: Is there a layout of height at most h?
- Given h, solve the new problem using dynamic programming:

$$f(i) = \text{minimal width needed to split the first } i \text{ words into columns}$$

- Number of states is n, and there are at most h transitions from each state.

G: Glossary Arrangement

Problem Author: Jorke de Vlas

Solution

- The answer can be found using binary search.
- New problem: Is there a layout of height at most h?
- Given h, solve the new problem using dynamic programming:

$$f(i) = \text{minimal width needed to split the first } i \text{ words into columns}$$

- Number of states is n, and there are at most h transitions from each state.
- Time complexity: $\mathcal{O}(n^2 \log(n))$.

G: Glossary Arrangement

Problem Author: Jorke de Vlas

Solution

- The answer can be found using binary search.
- New problem: Is there a layout of height at most h?
- Given h, solve the new problem using dynamic programming:

$$f(i) = \text{minimal width needed to split the first } i \text{ words into columns}$$

- Number of states is n, and there are at most h transitions from each state.
- Time complexity: $\mathcal{O}(n^2 \log(n))$.
- Can also speed up DP for an $\mathcal{O}(n \log^2(n))$ solution.

G: Glossary Arrangement

Problem Author: Jorke de Vlas

Solution

- The answer can be found using binary search.
- New problem: Is there a layout of height at most h?
- Given h, solve the new problem using dynamic programming:

$$f(i) = \text{minimal width needed to split the first } i \text{ words into columns}$$

- Number of states is n, and there are at most h transitions from each state.
- Time complexity: $\mathcal{O}(n^2 \log(n))$.
- Can also speed up DP for an $\mathcal{O}(n \log^2(n))$ solution.

Statistics: 102 submissions, 35 accepted, 31 unknown

H: Heating Up

Problem Author: Alexander Dietsch

Problem

Given a pizza with many slices, each having its own spiciness level. Eating a slice with a certain spiciness is only possible if you have enough tolerance, and it increases this tolerance by the spiciness level of the slice.

You are allowed to start at any slice but after every slice, you must continue with one of the neighbouring slices. Which initial minimal tolerance is needed to finish the pizza.

H: Heating Up

Problem Author: Alexander Dietsch

Solution

- Problem can be solved with binary search. (If tolerance x is enough, $x+1$ works as well)

H: Heating Up

Problem Author: Alexander Dietsch

Solution

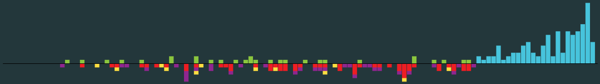
- Problem can be solved with binary search. (If tolerance x is enough, $x + 1$ works as well)
- New problem: Does tolerance x suffice to eat the whole pizza?

Solution

- Problem can be solved with binary search. (If tolerance x is enough, $x + 1$ works as well)
- New problem: Does tolerance x suffice to eat the whole pizza?
- Use a cyclic linked list, each element holds the spiciness level to finish the element and the increase in tolerance it gives. Initially 1 slice $=$ 1 element.

Solution

- Problem can be solved with binary search. (If tolerance x is enough, $x + 1$ works as well)
- New problem: Does tolerance x suffice to eat the whole pizza?
- Use a cyclic linked list, each element holds the spiciness level to finish the element and the increase in tolerance it gives. Initially 1 slice = 1 element.
- Visit all elements; on a visit:

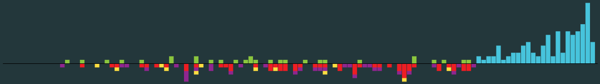


Solution

- Problem can be solved with binary search. (If tolerance x is enough, $x + 1$ works as well)
- New problem: Does tolerance x suffice to eat the whole pizza?
- Use a cyclic linked list, each element holds the spiciness level to finish the element and the increase in tolerance it gives. Initially 1 slice $=$ 1 element.
- Visit all elements; on a visit:
  - Check if the initial tolerance is high enough to finish the element.

H: Heating Up

Problem Author: Alexander Dietsch



Solution

- Problem can be solved with binary search. (If tolerance x is enough, $x + 1$ works as well)
- New problem: Does tolerance x suffice to eat the whole pizza?
- Use a cyclic linked list, each element holds the spiciness level to finish the element and the increase in tolerance it gives. Initially 1 slice $=$ 1 element.
- Visit all elements; on a visit:
  - Check if the initial tolerance is high enough to finish the element.
  - If so, check if the resulting tolerance is enough to finish a neighbouring element.

H: Heating Up

Problem Author: Alexander Dietsch

Solution

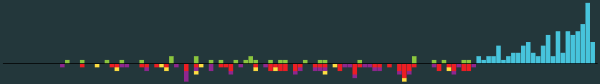
- Problem can be solved with binary search. (If tolerance x is enough, $x + 1$ works as well)
- New problem: Does tolerance x suffice to eat the whole pizza?
- Use a cyclic linked list, each element holds the spiciness level to finish the element and the increase in tolerance it gives. Initially 1 slice = 1 element.
- Visit all elements; on a visit:
  - Check if the initial tolerance is high enough to finish the element.
  - If so, check if the resulting tolerance is enough to finish a neighbouring element.
  - If that is the case, merge the elements. The spiciness level to finish the new element is the minimum, the increase in tolerance is the sum of both elements.

Solution

- Problem can be solved with binary search. (If tolerance x is enough, $x+1$ works as well)
- New problem: Does tolerance x suffice to eat the whole pizza?
- Use a cyclic linked list, each element holds the spiciness level to finish the element and the increase in tolerance it gives. Initially 1 slice = 1 element.
- Visit all elements; on a visit:
  - Check if the initial tolerance is high enough to finish the element.
  - If so, check if the resulting tolerance is enough to finish a neighbouring element.
  - If that is the case, merge the elements. The spiciness level to finish the new element is the minimum, the increase in tolerance is the sum of both elements.
- If the linked list can be merged into a single element, the initial tolerance is enough to finish the pizza.

H: Heating Up

Problem Author: Alexander Dietsch



Solution

- Problem can be solved with binary search. (If tolerance x is enough, $x + 1$ works as well)
- New problem: Does tolerance x suffice to eat the whole pizza?
- Use a cyclic linked list, each element holds the spiciness level to finish the element and the increase in tolerance it gives. Initially 1 slice $=$ 1 element.
- Visit all elements; on a visit:
  - Check if the initial tolerance is high enough to finish the element.
  - If so, check if the resulting tolerance is enough to finish a neighbouring element.
  - If that is the case, merge the elements. The spiciness level to finish the new element is the minimum, the increase in tolerance is the sum of both elements.
- If the linked list can be merged into a single element, the initial tolerance is enough to finish the pizza.

Statistics: 252 submissions, 29 accepted, 124 unknown

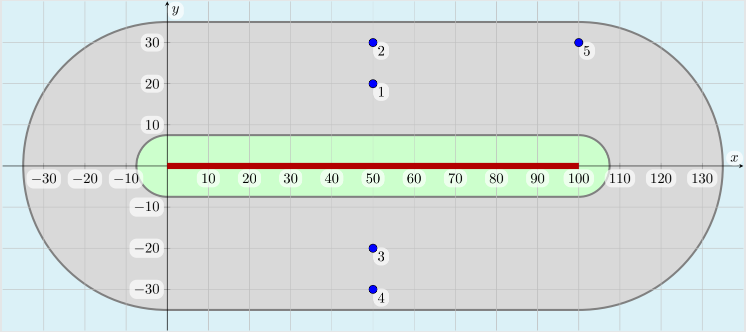
F: Flatland Olympics

Problem Author: Harry Smit

Problem

Given a line segment s and a set of n points $p_1, \ldots, p_n$. Find the number of pairs of points p_i, p_j $(i < j)$ such that both points lie on the same side of s and the line through p_i and p_j intersects s.

Example



F: Flatland Olympics

Problem Author: Harry Smit

observation

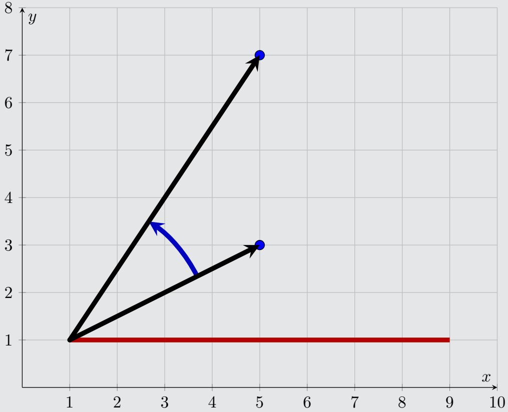
- Observe how the relation of two points changes while moving from one end to the other of the line segment s:

F: Flatland Olympics

Problem Author: Harry Smit

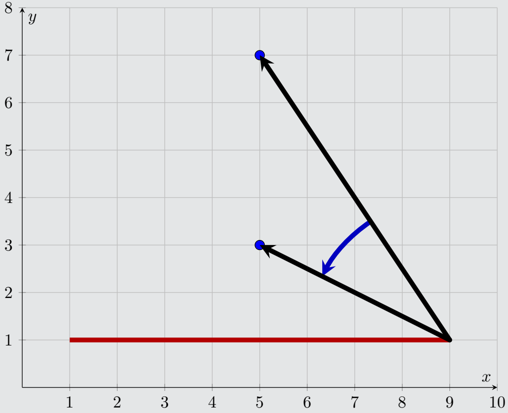
observation

- Observe how the relation of two points changes while moving from one end to the other of the line segment *s*:



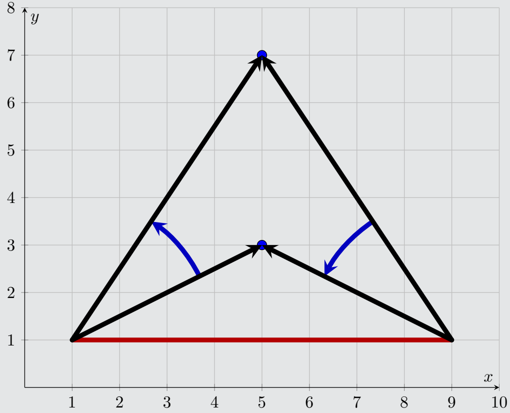
observation

- Observe how the relation of two points changes while moving from one end to the other of the line segment s:



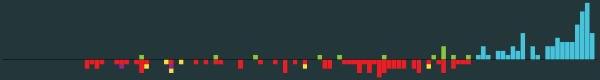
observation

- Observe how the relation of two points changes while moving from one end to the other of the line segment s:



F: Flatland Olympics

Problem Author: Harry Smit

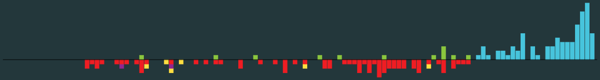


Solution

- Separate the points above and below s in two different sets.

F: Flatland Olympics

Problem Author: Harry Smit



Solution

- Separate the points above and below s in two different sets.
- For each set:
  - Sort the points around the *start* of s.
  - Sort the points around the *end* of s.
  - A pair of points has to be counted if their order in these two sequences differ.

Solution

- Separate the points above and below s in two different sets.
- For each set:
  - Sort the points around the *start* of s.
  - Sort the points around the *end* of s.
  - A pair of points has to be counted if their order in these two sequences differ.
- We need to find the number of *inversions* between two permutations.
- This can be done in $\mathcal{O}(n \log(n))$.

F: Flatland Olympics

Problem Author: Harry Smit

Solution

- Separate the points above and below s in two different sets.
- For each set:
  - Sort the points around the *start* of s.
  - Sort the points around the *end* of s.
  - A pair of points has to be counted if their order in these two sequences differ.
- We need to find the number of *inversions* between two permutations.
- This can be done in $\mathcal{O}(n \log(n))$.

Gotcha

- Points lying along the line through s.
- Multiple points collinear with the start or the end of s.

Statistics: 179 submissions, 12 accepted, 86 unknown

E: Exchange Students

Problem Author: Nils Gustafsson

Problem

Given two permutations g and h of size $n \leq 300\,000$, turn g into h by swapping pairs of elements with only smaller elements in between them. How many moves are needed and find the first up to 200 000 moves.

E: Exchange Students

Problem Author: Nils Gustafsson

Problem

Given two permutations g and h of size $n \leq 300\,000$, turn g into h by swapping pairs of elements with only smaller elements in between them. How many moves are needed and find the first up to 200 000 moves.

Solution

- Observation: in an optimal solution, you can reorder the swaps to first do all swaps involving the shortest students.

E: Exchange Students

Problem Author: Nils Gustafsson

Problem

Given two permutations g and h of size $n \leq 300\,000$, turn g into h by swapping pairs of elements with only smaller elements in between them. How many moves are needed and find the first up to 200 000 moves.

Solution

- Observation: in an optimal solution, you can reorder the swaps to first do all swaps involving the shortest students.
- When doing swaps involving the shortest students, they always move one step at a time.

E: Exchange Students

Problem Author: Nils Gustafsson

Problem

Given two permutations g and h of size $n \leq 300\,000$, turn g into h by swapping pairs of elements with only smaller elements in between them. How many moves are needed and find the first up to 200 000 moves.

Solution

- Observation: in an optimal solution, you can reorder the swaps to first do all swaps involving the shortest students.
- When doing swaps involving the shortest students, they always move one step at a time.
- After the shortest students are in place, they do not affect any of the other swaps, and you can remove them from the sequence.

E: Exchange Students

Problem Author: Nils Gustafsson

Problem

Given two permutations g and h of size $n \leq 300\,000$, turn g into h by swapping pairs of elements with only smaller elements in between them. How many moves are needed and find the first up to 200 000 moves.

Solution

- Observation: in an optimal solution, you can reorder the swaps to first do all swaps involving the shortest students.
- When doing swaps involving the shortest students, they always move one step at a time.
- After the shortest students are in place, they do not affect any of the other swaps, and you can remove them from the sequence.
- Now your sequence has one fewer height, and you can repeat.

E: Exchange Students

Problem Author: Nils Gustafsson

Solution

- If the shortest students are in locations $a_1, \ldots, a_k$ in g and $b_1, \ldots, b_k$ in h, then it takes

$$\sum_{i=1}^{k} |a_i - b_i|$$

steps to get them into the right location.

Solution

- If the shortest students are in locations $a_1, \ldots, a_k$ in g and $b_1, \ldots, b_k$ in h, then it takes

$$\sum_{i=1}^{k} |a_i - b_i|$$

steps to get them into the right location.

- When removing the shortest students, use a Segment or Fenwick tree to keep track of locations of the other students in the new sequence.

E: Exchange Students

Problem Author: Nils Gustafsson

Solution

- If the shortest students are in locations $a_1, \ldots, a_k$ in g and $b_1, \ldots, b_k$ in h, then it takes

$$\sum_{i=1}^{k} |a_i - b_i|$$

steps to get them into the right location.

- When removing the shortest students, use a Segment or Fenwick tree to keep track of locations of the other students in the new sequence.
- Do the reconstruction while you count the steps, as long as you have not reached the number of steps you have to output.

E: Exchange Students

Problem Author: Nils Gustafsson

Solution

- If the shortest students are in locations $a_1, \ldots, a_k$ in g and $b_1, \ldots, b_k$ in h, then it takes

$$\sum_{i=1}^{k} |a_i - b_i|$$

  steps to get them into the right location.
- When removing the shortest students, use a Segment or Fenwick tree to keep track of locations of the other students in the new sequence.
- Do the reconstruction while you count the steps, as long as you have not reached the number of steps you have to output.
- Take care to not swap with equal elements. From $1, 1, 2$ to $2, 1, 1$, the first 1 needs to go right, but that is only possible by swapping the 2 to the left.

E: Exchange Students

Problem Author: Nils Gustafsson

Solution

- Challenge: Can you do it in $O(n \lg n + moves)$?

E: Exchange Students

Problem Author: Nils Gustafsson

Solution

- Challenge: Can you do it in $O(n \lg n + moves)$?
- Challenge: Can you do it in $O(n \lg n + moves \lg n)$, but by processing all elements in random order?

E: Exchange Students

Problem Author: Nils Gustafsson

Solution

- Challenge: Can you do it in $O(n \lg n + moves)$?
- Challenge: Can you do it in $O(n \lg n + moves \lg n)$, but by processing all elements in random order?

Statistics: 24 submissions, 4 accepted, 13 unknown

Problem Author: Paul Wild

Problem

Given a specific number of each of the letters M, D, C, L, X, V, I, what is the least number of Roman numerals that can be formed while using exactly the required number of each letter?

I: `IXth` Problem

Problem Author: Paul Wild

Problem

Given a specific number of each of the letters `M`, `D`, `C`, `L`, `X`, `V`, `I`, what is the least number of Roman numerals that can be formed while using exactly the required number of each letter?

Insight

We can use binary search on the answer. New subproblem: Given an integer n, can we form at most n numerals using all the tiles?

Solution for subproblem

Start with n empty strings and add the digits in order from M to I.

$$\texttt{M} \times 4 \quad \texttt{D} \times 1 \quad \texttt{C} \times 7 \quad \texttt{L} \times 1 \quad \texttt{X} \times 3 \quad \texttt{V} \times 1 \quad \texttt{I} \times 3$$

1.
2.

Problem Author: Paul Wild

Solution for subproblem

Start with n empty strings and add the digits in order from M to I.

$$\mathtt{M} \times 0 \quad \mathtt{D} \times 1 \quad \mathtt{C} \times 7 \quad \mathtt{L} \times 1 \quad \mathtt{X} \times 3 \quad \mathtt{V} \times 1 \quad \mathtt{I} \times 3$$

1. MMM
2. M

- Distribute M, C, X and I in groups of three, and D, L and V on their own.

Solution for subproblem

Start with n empty strings and add the digits in order from M to I.

$$\mathtt{M} \times 0 \quad \mathtt{D} \times 0 \quad \mathtt{C} \times 7 \quad \mathtt{L} \times 1 \quad \mathtt{X} \times 3 \quad \mathtt{V} \times 1 \quad \mathtt{I} \times 3$$

1. MMMD
2. M

- Distribute M, C, X and I in groups of three, and D, L and V on their own.

Solution for subproblem

Start with n empty strings and add the digits in order from M to I.

$$\mathtt{M} \times 0 \quad \mathtt{D} \times 0 \quad \mathtt{C} \times 1 \quad \mathtt{L} \times 1 \quad \mathtt{X} \times 3 \quad \mathtt{V} \times 1 \quad \mathtt{I} \times 3$$

1. MMMDCCC
2. MCCC

- Distribute M, C, X and I in groups of three, and D, L and V on their own.

Solution for subproblem

Start with n empty strings and add the digits in order from `M` to `I`.

$$\texttt{M} \times 0 \quad \texttt{D} \times 0 \quad \texttt{C} \times 0 \quad \texttt{L} \times 1 \quad \texttt{X} \times 2 \quad \texttt{V} \times 1 \quad \texttt{I} \times 3$$

1. `MMMDCCCXC`
2. `MCCC`

- Distribute `M`, `C`, `X` and `I` in groups of three, and `D`, `L` and `V` on their own.
- If there is not enough room for all `M`, `C` or `X`, try filling up with copies of `CM`, `XC` or `IX`.

I: `IXth` Problem

Problem Author: Paul Wild

Solution for subproblem

Start with n empty strings and add the digits in order from `M` to `I`.

$$\texttt{M} \times 0 \quad \texttt{D} \times 0 \quad \texttt{C} \times 0 \quad \texttt{L} \times 0 \quad \texttt{X} \times 2 \quad \texttt{V} \times 1 \quad \texttt{I} \times 3$$

1. `MMMDCCCXC`
2. `MCCCL`

- Distribute `M`, `C`, `X` and `I` in groups of three, and `D`, `L` and `V` on their own.
- If there is not enough room for all `M`, `C` or `X`, try filling up with copies of `CM`, `XC` or `IX`.
- Fill up later letters from the first available slot.

Solution for subproblem

Start with n empty strings and add the digits in order from M to I.

$$\texttt{M} \times 0 \quad \texttt{D} \times 0 \quad \texttt{C} \times 0 \quad \texttt{L} \times 0 \quad \texttt{X} \times 0 \quad \texttt{V} \times 1 \quad \texttt{I} \times 3$$

1. MMMDCCCXC
2. MCCCLXX

- Distribute M, C, X and I in groups of three, and D, L and V on their own.
- If there is not enough room for all M, C or X, try filling up with copies of CM, XC or IX.
- Fill up later letters from the first available slot.

Problem Author: Paul Wild

Solution for subproblem

Start with n empty strings and add the digits in order from M to I.

$$\mathtt{M}\times 0 \quad \mathtt{D}\times 0 \quad \mathtt{C}\times 0 \quad \mathtt{L}\times 0 \quad \mathtt{X}\times 0 \quad \mathtt{V}\times 0 \quad \mathtt{I}\times 3$$

1. MMMDCCCXCV
2. MCCCLXX

- Distribute M, C, X and I in groups of three, and D, L and V on their own.
- If there is not enough room for all M, C or X, try filling up with copies of CM, XC or IX.
- Fill up later letters from the first available slot.

Solution for subproblem

Start with n empty strings and add the digits in order from M to I.

$$\texttt{M} \times 0 \quad \texttt{D} \times 0 \quad \texttt{C} \times 0 \quad \texttt{L} \times 0 \quad \texttt{X} \times 0 \quad \texttt{V} \times 0 \quad \texttt{I} \times 0$$

1. MMMDCCCXCVIII
2. MCCCLXX

- Distribute M, C, X and I in groups of three, and D, L and V on their own.
- If there is not enough room for all M, C or X, try filling up with copies of CM, XC or IX.
- Fill up later letters from the first available slot.

Solution for subproblem

Start with n empty strings and add the digits in order from M to I.

$$\mathtt{M} \times 0 \quad \mathtt{D} \times 0 \quad \mathtt{C} \times 0 \quad \mathtt{L} \times 0 \quad \mathtt{X} \times 0 \quad \mathtt{V} \times 0 \quad \mathtt{I} \times 0$$

1. MMMDCCCXCVIII
2. MCCCLXX

- Distribute M, C, X and I in groups of three, and D, L and V on their own.
- If there is not enough room for all M, C or X, try filling up with copies of CM, XC or IX.
- Fill up later letters from the first available slot.
- If you run out of letters or room at any point, abort.

Solution for subproblem

Start with n empty strings and add the digits in order from M to I.

$$\texttt{M}\times 0 \quad \texttt{D}\times 0 \quad \texttt{C}\times 0 \quad \texttt{L}\times 0 \quad \texttt{X}\times 0 \quad \texttt{V}\times 0 \quad \texttt{I}\times 0$$

1. MMMDCCCXCVIII
2. MCCCLXX

- Distribute M, C, X and I in groups of three, and D, L and V on their own.
- If there is not enough room for all M, C or X, try filling up with copies of CM, XC or IX.
- Fill up later letters from the first available slot.
- If you run out of letters or room at any point, abort.
- As the numbers are huge, speed it up by grouping equal numbers together.

I: IXth Problem

Problem Author: Paul Wild

Solution for subproblem

Start with n empty strings and add the digits in order from M to I.

$$\mathtt{M}\times 0 \quad \mathtt{D}\times 0 \quad \mathtt{C}\times 0 \quad \mathtt{L}\times 0 \quad \mathtt{X}\times 0 \quad \mathtt{V}\times 0 \quad \mathtt{I}\times 0$$

1. MMMDCCCXCVIII
2. MCCCLXX

- Distribute M, C, X and I in groups of three, and D, L and V on their own.
- If there is not enough room for all M, C or X, try filling up with copies of CM, XC or IX.
- Fill up later letters from the first available slot.
- If you run out of letters or room at any point, abort.
- As the numbers are huge, speed it up by grouping equal numbers together.

Statistics: 34 submissions, 2 accepted, 20 unknown

B: Boredom Buster

Problem Author: Nils Gustafsson

Problem

Play a single player version of the game Memory (aka Concentration), where the cards are randomly shuffled before and after reveal.

Problem Author: Nils Gustafsson

Problem

Play a single player version of the game Memory (aka Concentration), where the cards are randomly shuffled before and after reveal.

Solution

- **First attempt**: Revealing the cards with indices i and j will give you the card numbers x and y. If you now query (j, k) and you get result (x, z) for some different z, you can deduce that $c_i = y$.
- Repeating this logic $n - 1$ times, $n - 2$ cards will be known. We still have to take care of the last two, but this is too many queries.
- **Insight**: We have to exploit the fact that there are many duplicates in the deck.

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- **Attempt 2**: Query for $(1,2),(3,4),(5,6),\cdots$. This gives you $\frac{n}{2}$ tuples on the form (i,j,x,y) meaning that the cards on positions i and j have values x, y.

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- **Attempt 2**: Query for $(1, 2), (3, 4), (5, 6), \cdots$. This gives you $\frac{n}{2}$ tuples on the form (i, j, x, y) meaning that the cards on positions i and j have values x, y.
- Take two tuples on the form (i_1, j_1, x, y), (i_2, j_2, y, z) and query (i_1, i_2).

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- **Attempt 2**: Query for $(1,2),(3,4),(5,6),\cdots$. This gives you $\frac{n}{2}$ tuples on the form (i,j,x,y) meaning that the cards on positions i and j have values x, y.
- Take two tuples on the form (i_1,j_1,x,y), (i_2,j_2,y,z) and query (i_1,i_2).
  - With 75% probability the answer will be different numbers (e.g. (x,z)). This will give you two card positions and creates another tuple (i_1,i_2,x,z).

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- **Attempt 2**: Query for $(1,2), (3,4), (5,6), \cdots$. This gives you $\frac{n}{2}$ tuples on the form (i, j, x, y) meaning that the cards on positions i and j have values x, y.
- Take two tuples on the form (i_1, j_1, x, y), (i_2, j_2, y, z) and query (i_1, i_2).
  - With 75% probability the answer will be different numbers (e.g. (x, z)). This will give you two card positions and creates another tuple (i_1, i_2, x, z).
  - With 25% probability the answer will be (y, y) which gives you all four cards.

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- **Attempt 2**: Query for $(1,2),(3,4),(5,6),\cdots$. This gives you $\frac{n}{2}$ tuples on the form (i,j,x,y) meaning that the cards on positions i and j have values x, y.
- Take two tuples on the form (i_1,j_1,x,y), (i_2,j_2,y,z) and query (i_1,i_2).
  - With 75% probability the answer will be different numbers (e.g. (x,z)). This will give you two card positions and creates another tuple (i_1,i_2,x,z).
  - With 25% probability the answer will be (y,y) which gives you all four cards.
- By naively pairing up the tuples to get these collisions and executing the above strategy, you will solve the problem with around $\frac{15}{16}n$ queries. But this is still not enough!

Solution

- How to cause many collisions using the idea on the previous slide?

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- How to cause many collisions using the idea on the previous slide?
- Let's model the problem as a graph with $\frac{n}{2}$ vertices, one for each card number.

Solution

- How to cause many collisions using the idea on the previous slide?
- Let's model the problem as a graph with $\frac{n}{2}$ vertices, one for each card number.
- For every tuple (i, j, x, y) add an edge between x and y. This gives components which are either cycles or paths.

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

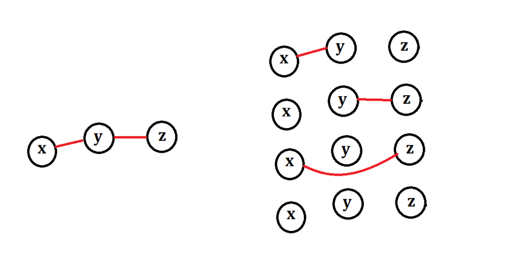
- How to cause many collisions using the idea on the previous slide?
- Let's model the problem as a graph with $\frac{n}{2}$ vertices, one for each card number.
- For every tuple (i, j, x, y) add an edge between x and y. This gives components which are either cycles or paths.
- Querying for pairs of adjacent edges has a chance of eliminating both edges.

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- How to cause many collisions using the idea on the previous slide?
- Let's model the problem as a graph with $\frac{n}{2}$ vertices, one for each card number.
- For every tuple (i, j, x, y) add an edge between x and y. This gives components which are either cycles or paths.
- Querying for pairs of adjacent edges has a chance of eliminating both edges.



B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- Make cycles into paths by querying two adjacent edges.

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- Make cycles into paths by querying two adjacent edges.
- Greedily query edges at the endpoints of paths, as long as they exist.

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- Make cycles into paths by querying two adjacent edges.
- Greedily query edges at the endpoints of paths, as long as they exist.
- Save components of size 2 for last.

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- Make cycles into paths by querying two adjacent edges.
- Greedily query edges at the endpoints of paths, as long as they exist.
- Save components of size 2 for last.
- This gives a solution with around $\frac{11}{12}n$ queries, which is good enough.

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- Make cycles into paths by querying two adjacent edges.
- Greedily query edges at the endpoints of paths, as long as they exist.
- Save components of size 2 for last.
- This gives a solution with around $\frac{11}{12}n$ queries, which is good enough.
- More details that need to be taken care of:

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- Make cycles into paths by querying two adjacent edges.
- Greedily query edges at the endpoints of paths, as long as they exist.
- Save components of size 2 for last.
- This gives a solution with around $\frac{11}{12}n$ queries, which is good enough.
- More details that need to be taken care of:
  - Cycles of length 2 are special, they have a 50% chance of being unaffected.

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- Make cycles into paths by querying two adjacent edges.
- Greedily query edges at the endpoints of paths, as long as they exist.
- Save components of size 2 for last.
- This gives a solution with around $\frac{11}{12}n$ queries, which is good enough.
- More details that need to be taken care of:
  - Cycles of length 2 are special, they have a 50% chance of being unaffected.
  - At the end one tuple (i, j, x, y) will remain. To take care of it, find another index k with value x, and randomly query pairs in $\{i, j, k\}$ until a collision happens.

B: Boredom Buster

Problem Author: Nils Gustafsson

Solution

- Make cycles into paths by querying two adjacent edges.
- Greedily query edges at the endpoints of paths, as long as they exist.
- Save components of size 2 for last.
- This gives a solution with around $\frac{11}{12}n$ queries, which is good enough.
- More details that need to be taken care of:
  - Cycles of length 2 are special, they have a 50% chance of being unaffected.
  - At the end one tuple (i, j, x, y) will remain. To take care of it, find another index k with value x, and randomly query pairs in $\{i, j, k\}$ until a collision happens.

Statistics: 23 submissions, 1 accepted, 8 unknown

L: Lucky Shirt

Problem Author: Ragnar Groot Koerkamp

Problem

Given is a list of n shirts. We choose k integers $l_1, \ldots, l_k$ uniformly at random and then randomly permute the first l_j shirts for $j \in \{1, \ldots, k\}$. What is the expected position of the shirt that started at position i (1-based)?

Problem

Given is a list of n shirts. We choose k integers $l_1, \ldots, l_k$ uniformly at random and then randomly permute the first l_j shirts for $j \in \{1, \ldots, k\}$. What is the expected position of the shirt that started at position i (1-based)?

First idea

- Calculate the probability p_a that your lucky shirt ends up at position a for all $a \in \{1, \ldots, n\}$.

L: Lucky Shirt

Problem Author: Ragnar Groot Koerkamp

Problem

Given is a list of n shirts. We choose k integers $l_1, \ldots, l_k$ uniformly at random and then randomly permute the first l_j shirts for $j \in \{1, \ldots, k\}$. What is the expected position of the shirt that started at position i (1-based)?

First idea

- Calculate the probability p_a that your lucky shirt ends up at position a for all $a \in \{1, \ldots, n\}$.
- The answer is

$$\sum_{a=1}^{n} a \cdot p_a.$$

L: Lucky Shirt

Problem Author: Ragnar Groot Koerkamp

Problem

Given is a list of n shirts. We choose k integers $l_1, \ldots, l_k$ uniformly at random and then randomly permute the first l_j shirts for $j \in \{1, \ldots, k\}$. What is the expected position of the shirt that started at position i (1-based)?

First idea

- Calculate the probability p_a that your lucky shirt ends up at position a for all $a \in \{1, \ldots, n\}$.
- The answer is

$$\sum_{a=1}^{n} a \cdot p_a.$$

- However, p_a does not have a nice formula.

L: Lucky Shirt

Problem Author: Ragnar Groot Koerkamp

Solution (1/2)

- Key observation: once the lucky shirt is shuffled, its location is uniform among the shuffled shirts.

L: Lucky Shirt

Problem Author: Ragnar Groot Koerkamp

Solution (1/2)

- Key observation: once the lucky shirt is shuffled, its location is uniform among the shuffled shirts.
- Only $M := \max_j l_j$ is relevant! We distinguish two simple cases.

L: Lucky Shirt

Problem Author: Ragnar Groot Koerkamp

Solution (1/2)

- Key observation: once the lucky shirt is shuffled, its location is uniform among the shuffled shirts.
- Only $M := \max_j l_j$ is relevant! We distinguish two simple cases.
- Case 1: the shirt never moves during the process.

L: Lucky Shirt

Problem Author: Ragnar Groot Koerkamp

Solution (1/2)

- Key observation: once the lucky shirt is shuffled, its location is uniform among the shuffled shirts.
- Only $M := \max_j l_j$ is relevant! We distinguish two simple cases.
- Case 1: the shirt never moves during the process.
  - This happens exactly when $M < i$.

Solution (1/2)

- Key observation: once the lucky shirt is shuffled, its location is uniform among the shuffled shirts.
- Only $M := \max_j l_j$ is relevant! We distinguish two simple cases.
- Case 1: the shirt never moves during the process.
  - This happens exactly when $M < i$.
  - The (expected) position of the shirt is i.

L: Lucky Shirt

Problem Author: Ragnar Groot Koerkamp

Solution (1/2)

- Key observation: once the lucky shirt is shuffled, its location is uniform among the shuffled shirts.
- Only $M := \max_j l_j$ is relevant! We distinguish two simple cases.
- Case 1: the shirt never moves during the process.
  - This happens exactly when $M < i$.
  - The (expected) position of the shirt is i.
- Case 2: the shirt is shuffled at least once.

L: Lucky Shirt

Problem Author: Ragnar Groot Koerkamp

Solution (1/2)

- Key observation: once the lucky shirt is shuffled, its location is uniform among the shuffled shirts.
- Only $M := \max_j l_j$ is relevant! We distinguish two simple cases.
- Case 1: the shirt never moves during the process.
  - This happens exactly when $M < i$.
  - The (expected) position of the shirt is i.
- Case 2: the shirt is shuffled at least once.
  - This happens exactly when $M \geq i$.

Solution (1/2)

- Key observation: once the lucky shirt is shuffled, its location is uniform among the shuffled shirts.
- Only $M := \max_j l_j$ is relevant! We distinguish two simple cases.
- Case 1: the shirt never moves during the process.
  - This happens exactly when $M < i$.
  - The (expected) position of the shirt is i.
- Case 2: the shirt is shuffled at least once.
  - This happens exactly when $M \geq i$.
  - You cannot distinguish the lucky shirt from any of the other first M shirts

L: Lucky Shirt

Problem Author: Ragnar Groot Koerkamp

Solution (1/2)

- Key observation: once the lucky shirt is shuffled, its location is uniform among the shuffled shirts.
- Only $M := \max_j l_j$ is relevant! We distinguish two simple cases.
- Case 1: the shirt never moves during the process.
  - This happens exactly when $M < i$.
  - The (expected) position of the shirt is i.
- Case 2: the shirt is shuffled at least once.
  - This happens exactly when $M \geq i$.
  - You cannot distinguish the lucky shirt from any of the other first M shirts
  - The (expected) position of the shirt is $(M+1)/2$.

Solution (2/2)

- Thus the answer equals

$$i \cdot \mathbb{P}(M < i) + \sum_{a=i}^{n} \frac{a+1}{2} \cdot \mathbb{P}(M = a).$$

Solution (2/2)

- Thus the answer equals

$$i \cdot \mathbb{P}(M < i) + \sum_{a=i}^{n} \frac{a+1}{2} \cdot \mathbb{P}(M = a).$$

- As the l_j are chosen uniformly at random (and independent of one another),

Solution (2/2)

- Thus the answer equals
$$i \cdot \mathbb{P}(M < i) + \sum_{a=i}^{n} \frac{a+1}{2} \cdot \mathbb{P}(M = a).$$
- As the l_j are chosen uniformly at random (and independent of one another),

Solution (2/2)

- Thus the answer equals

$$i \cdot \mathbb{P}(M < i) + \sum_{a=i}^{n} \frac{a+1}{2} \cdot \mathbb{P}(M = a).$$

- As the l_j are chosen uniformly at random (and independent of one another),

$$\mathbb{P}(M < i) = \left(\frac{i-1}{n}\right)^k, \text{ and}$$

$$\mathbb{P}(M = a) = \mathbb{P}(M < a+1) - \mathbb{P}(M < a) = \left(\frac{a}{n}\right)^k - \left(\frac{a-1}{n}\right)^k.$$

Statistics: 30 submissions, 1 accepted, 25 unknown

C: Cutting Edge

Problem Author: Paul Wild

Problem

Given a desired volume $v/6$, find a set of integer-valued points whose convex hull has this volume.

C: Cutting Edge

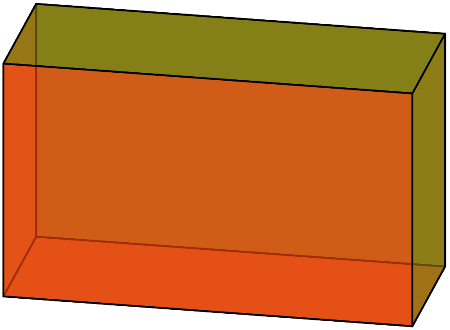
Problem Author: Paul Wild

Problem

Given a desired volume $v/6$, find a set of integer-valued points whose convex hull has this volume.

General idea

- Start with a cuboid and cut away tetrahedra from four of the corners.



C: Cutting Edge

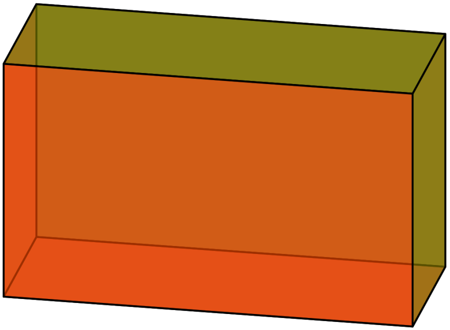
Problem Author: Paul Wild

Problem

Given a desired volume $v/6$, find a set of integer-valued points whose convex hull has this volume.

General idea

- Start with a cuboid and cut away tetrahedra from four of the corners.
- Take a cuboid with size $a \times b \times c$ (assume wlog $a \leq b \leq c$), where $ab(c-1) \leq v/6 \leq abc$.



C: Cutting Edge

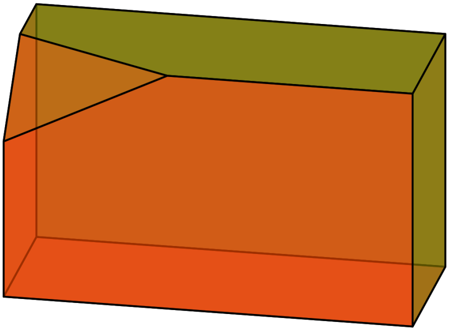
Problem Author: Paul Wild

Problem

Given a desired volume $v/6$, find a set of integer-valued points whose convex hull has this volume.

General idea

- Start with a cuboid and cut away tetrahedra from four of the corners.
- Take a cuboid with size $a \times b \times c$ (assume wlog $a \le b \le c$), where $ab(c-1) \le v/6 \le abc$.
- A tetrahedron with edge sizes u, v and w has volume $uvw/6$, and we can cut off four tetrahedra that don't interfere with one another.



C: Cutting Edge

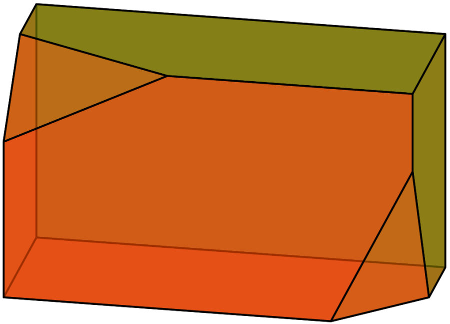
Problem Author: Paul Wild

Problem

Given a desired volume $v/6$, find a set of integer-valued points whose convex hull has this volume.

General idea

- Start with a cuboid and cut away tetrahedra from four of the corners.
- Take a cuboid with size $a \times b \times c$ (assume wlog $a \leq b \leq c$), where $ab(c-1) \leq v/6 \leq abc$.
- A tetrahedron with edge sizes u, v and w has volume $uvw/6$, and we can cut off four tetrahedra that don't interfere with one another.



Finding the right tetrahedra

- We need to cut off a total volume of $abc - v/6$ from the cuboid. Let $r := 6abc - v$. Note $0 \leq r \leq 6ab$.

Finding the right tetrahedra

- We need to cut off a total volume of $abc - v/6$ from the cuboid. Let $r := 6abc - v$. Note $0 \leq r \leq 6ab$.
- Can we find four tetrahedra with the desired volume, that is, does the set

$$S := \{uvw \mid 0 \leq u \leq a,\, 0 \leq v \leq b,\, 0 \leq w \leq c\}$$

contain four elements that sum to r?

Finding the right tetrahedra

- We need to cut off a total volume of $abc - v/6$ from the cuboid. Let $r := 6abc - v$. Note $0 \leq r \leq 6ab$.
- Can we find four tetrahedra with the desired volume, that is, does the set

$$S := \{uvw \mid 0 \leq u \leq a,\, 0 \leq v \leq b,\, 0 \leq w \leq c\}$$

  contain four elements that sum to r?
- If $c \geq 6$, this is easily done with (at most) three polyhedra:

Finding the right tetrahedra

- We need to cut off a total volume of $abc - v/6$ from the cuboid. Let $r := 6abc - v$. Note $0 \leq r \leq 6ab$.
- Can we find four tetrahedra with the desired volume, that is, does the set

$$S := \{uvw \mid 0 \leq u \leq a,\ 0 \leq v \leq b,\ 0 \leq w \leq c\}$$

  contain four elements that sum to r?
- If $c \geq 6$, this is easily done with (at most) three polyhedra:
  - For the first one, take $u_0 = a$, $v_0 = b$, $w_0 = \lfloor \frac{r}{ab} \rfloor$

Problem Author: Paul Wild

Finding the right tetrahedra

- We need to cut off a total volume of $abc - v/6$ from the cuboid. Let $r := 6abc - v$. Note $0 \leq r \leq 6ab$.
- Can we find four tetrahedra with the desired volume, that is, does the set

$$S := \{uvw \mid 0 \leq u \leq a,\ 0 \leq v \leq b,\ 0 \leq w \leq c\}$$

  contain four elements that sum to r?
- If $c \geq 6$, this is easily done with (at most) three polyhedra:
  - For the first one, take $u_0 = a$, $v_0 = b$, $w_0 = \lfloor \frac{r}{ab} \rfloor$
  - For the second one, take $u_1 = a$, $v_1 = \lfloor \frac{r - u_0 v_0 w_0}{a} \rfloor$, $w_1 = 1$

C: Cutting Edge

Problem Author: Paul Wild

Finding the right tetrahedra

- We need to cut off a total volume of $abc - v/6$ from the cuboid. Let $r := 6abc - v$. Note $0 \leq r \leq 6ab$.
- Can we find four tetrahedra with the desired volume, that is, does the set

$$S := \{uvw \mid 0 \leq u \leq a,\, 0 \leq v \leq b,\, 0 \leq w \leq c\}$$

  contain four elements that sum to r?
- If $c \geq 6$, this is easily done with (at most) three polyhedra:
  - For the first one, take $u_0 = a$, $v_0 = b$, $w_0 = \lfloor \frac{r}{ab} \rfloor$
  - For the second one, take $u_1 = a$, $v_1 = \lfloor \frac{r - u_0 v_0 w_0}{a} \rfloor$, $w_1 = 1$
  - For the last one, take $u_2 = r - u_0 v_0 w_0 - u_1 v_1 w_1$, $v_2 = 1$, $w_2 = 1$.

Finding the right tetrahedra

- We need to cut off a total volume of $abc - v/6$ from the cuboid. Let $r := 6abc - v$. Note $0 \leq r \leq 6ab$.
- Can we find four tetrahedra with the desired volume, that is, does the set

$$S := \{uvw \mid 0 \leq u \leq a,\, 0 \leq v \leq b,\, 0 \leq w \leq c\}$$

  contain four elements that sum to r?
- If $c \geq 6$, this is easily done with (at most) three polyhedra:
  - For the first one, take $u_0 = a$, $v_0 = b$, $w_0 = \lfloor \frac{r}{ab} \rfloor$
  - For the second one, take $u_1 = a$, $v_1 = \lfloor \frac{r - u_0 v_0 w_0}{a} \rfloor$, $w_1 = 1$
  - For the last one, take $u_2 = r - u_0 v_0 w_0 - u_1 v_1 w_1$, $v_2 = 1$, $w_2 = 1$.
- If $c \leq 5$, then so are a and b, which implies that $|S|$ is small (at most 31).

C: Cutting Edge

Problem Author: Paul Wild

Finding the right tetrahedra

- We need to cut off a total volume of $abc - v/6$ from the cuboid. Let $r := 6abc - v$. Note $0 \leq r \leq 6ab$.
- Can we find four tetrahedra with the desired volume, that is, does the set

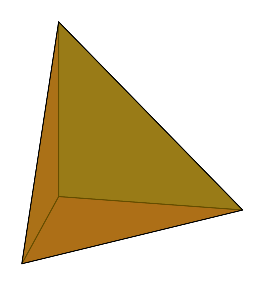
$$S := \{uvw \mid 0 \leq u \leq a,\, 0 \leq v \leq b,\, 0 \leq w \leq c\}$$

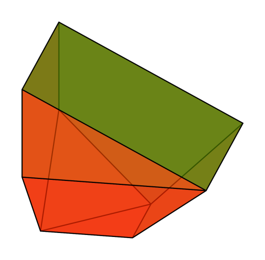
  contain four elements that sum to r?
- If $c \geq 6$, this is easily done with (at most) three polyhedra:
  - For the first one, take $u_0 = a$, $v_0 = b$, $w_0 = \lfloor \frac{r}{ab} \rfloor$
  - For the second one, take $u_1 = a$, $v_1 = \lfloor \frac{r - u_0 v_0 w_0}{a} \rfloor$, $w_1 = 1$
  - For the last one, take $u_2 = r - u_0 v_0 w_0 - u_1 v_1 w_1$, $v_2 = 1$, $w_2 = 1$.
- If $c \leq 5$, then so are a and b, which implies that $|S|$ is small (at most 31).
- Brute force all combinations to check if r can be written as a sum of four elements in S.

Leftover cases

This solves all cases except for two:

- The case where $a = b = c = 1$ and $v = 1$. This is the first sample.





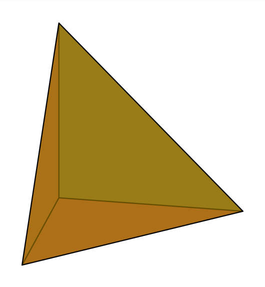
C: Cutting Edge

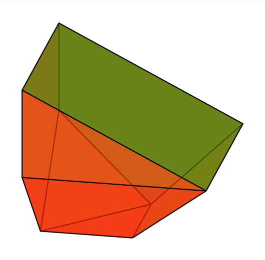
Problem Author: Paul Wild

Leftover cases

This solves all cases except for two:

- The case where $a = b = c = 1$ and $v = 1$. This is the first sample.
- The case where $a = b = c = 2$ and $v = 25$. Then $r = 23$. This is the third sample.

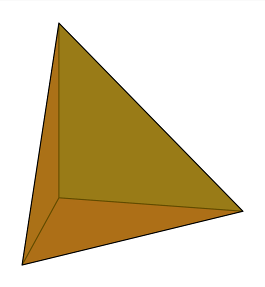


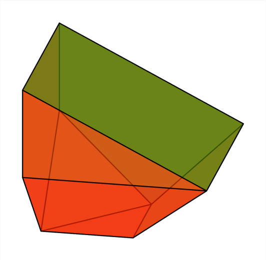


Leftover cases

This solves all cases except for two:

- The case where $a = b = c = 1$ and $v = 1$. This is the first sample.
- The case where $a = b = c = 2$ and $v = 25$. Then $r = 23$. This is the third sample.





Statistics: 4 submissions, 0 accepted, 4 unknown

Random facts

Jury work

- 632 commits

[1] *After* codegolfing

Random facts

Jury work

- 632 commits
- 681 secret test cases (last year: 486) ($\approx$ 57 per problem!)

[1] *After* codegolfing

Random facts

Jury work

- 632 commits
- 681 secret test cases (last year: 486) (≈ 57 per problem!)
- 248 jury solutions (last year: 232)

[1] *After* codegolfing

Random facts

Jury work

- 632 commits
- 681 secret test cases (last year: 486) (≈ 57 per problem!)
- 248 jury solutions (last year: 232)
- The minimum[1] number of lines the jury needed to solve all problems is

$$10 + 114 + 27 + 5 + 64 + 51 + 42 + 32 + 43 + 23 + 10 + 6 = 427$$

  On average 35.5 lines per problem, up from 9.6 in the BAPC

[1] *After* codegolfing