BAPC 2022

Solutions presentation

October 22, 2022

DAPC 2022

Solutions presentation

September 30, 2022

BAPC 2021 Preliminaries

Solutions presentation

October 9, 2021

A: Adjusted Average

Problem Author: Ludo Pulles

- **Problem:** Given $n \leq 1500$ integers a_i, remove at most $k \leq 4$ of them to get an average as close as possible to the target $\overline{x}$.

A: Adjusted Average

Problem Author: Ludo Pulles

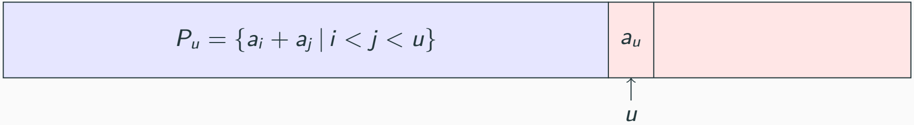
- **Problem:** Given $n \leq 1500$ integers a_i, remove at most $k \leq 4$ of them to get an average as close as possible to the target $\overline{x}$.
- When removing exactly $0 \leq \ell \leq k$ numbers, we need to find ℓ integers with sum as close as possible to $S_\ell = \sum_i a_i - \ell \cdot \overline{x}$.

A: Adjusted Average

Problem Author: Ludo Pulles

- **Problem:** Given $n \leq 1500$ integers a_i, remove at most $k \leq 4$ of them to get an average as close as possible to the target $\overline{x}$.
- When removing exactly $0 \leq \ell \leq k$ numbers, we need to find ℓ integers with sum as close as possible to $S_\ell = \sum_i a_i - \ell \cdot \overline{x}$.
- For $\ell \leq 2$: iterate over all combinations $\longrightarrow \mathcal{O}(n^\ell/\ell!)$ time.

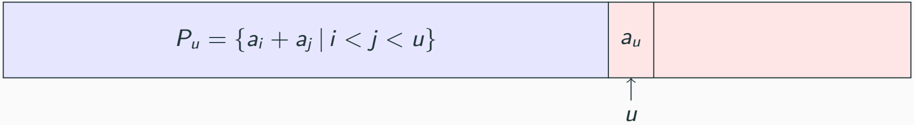
- **Problem:** Given $n \leq 1500$ integers a_i, remove at most $k \leq 4$ of them to get an average as close as possible to the target $\overline{x}$.
- When removing exactly $0 \leq \ell \leq k$ numbers, we need to find ℓ integers with sum as close as possible to $S_\ell = \sum_i a_i - \ell \cdot \overline{x}$.
- For $\ell \leq 2$: iterate over all combinations $\longrightarrow \mathcal{O}(n^\ell / \ell!)$ time.
- For $\ell = 3, 4$: this is too slow so use *meet-in-the-middle*:



A: Adjusted Average

Problem Author: Ludo Pulles

- **Problem:** Given $n \leq 1500$ integers a_i, remove at most $k \leq 4$ of them to get an average as close as possible to the target $\overline{x}$.
- When removing exactly $0 \leq \ell \leq k$ numbers, we need to find ℓ integers with sum as close as possible to $S_\ell = \sum_i a_i - \ell \cdot \overline{x}$.
- For $\ell \leq 2$: iterate over all combinations $\longrightarrow \mathcal{O}(n^\ell / \ell!)$ time.
- For $\ell = 3, 4$: this is too slow so use *meet-in-the-middle*:

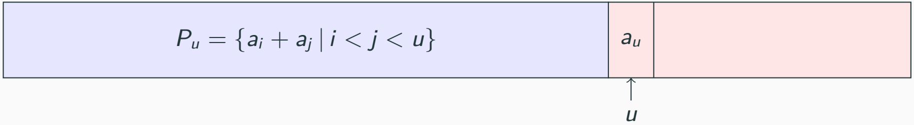


- For $\ell = 3$: loop over $u = 1, \ldots, n$, update P_u and then take $s \in P_u$ closest to $S_\ell - a_u$.

A: Adjusted Average

Problem Author: Ludo Pulles

- **Problem:** Given $n \leq 1500$ integers a_i, remove at most $k \leq 4$ of them to get an average as close as possible to the target $\overline{x}$.
- When removing exactly $0 \leq \ell \leq k$ numbers, we need to find ℓ integers with sum as close as possible to $S_\ell = \sum_i a_i - \ell \cdot \overline{x}$.
- For $\ell \leq 2$: iterate over all combinations $\longrightarrow \mathcal{O}(n^\ell/\ell!)$ time.
- For $\ell = 3, 4$: this is too slow so use *meet-in-the-middle*:

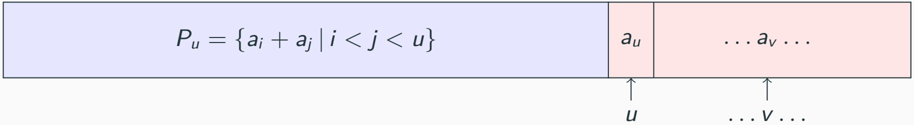


- For $\ell = 3$: loop over $u = 1, \ldots, n$, update P_u and then take $s \in P_u$ closest to $S_\ell - a_u$.
- Use an ordered set (BBST) for P_u, giving the time complexity $\mathcal{O}(n^2 \log n)$.

A: Adjusted Average

Problem Author: Ludo Pulles

- **Problem:** Given $n \leq 1500$ integers a_i, remove at most $k \leq 4$ of them to get an average as close as possible to the target $\overline{x}$.
- When removing exactly $0 \leq \ell \leq k$ numbers, we need to find ℓ integers with sum as close as possible to $S_\ell = \sum_i a_i - \ell \cdot \overline{x}$.
- For $\ell \leq 2$: iterate over all combinations $\longrightarrow \mathcal{O}(n^\ell / \ell!)$ time.
- For $\ell = 3, 4$: this is too slow so use *meet-in-the-middle*:

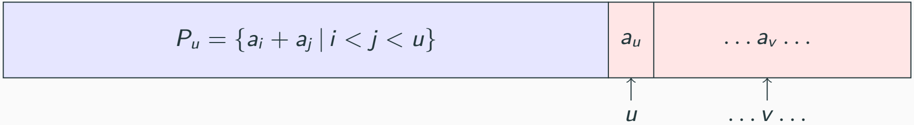


- For $\ell = 3$: loop over $u = 1, \ldots, n$, update P_u and then take $s \in P_u$ closest to $S_\ell - a_u$.
- Use an ordered set (BBST) for P_u, giving the time complexity $\mathcal{O}(n^2 \log n)$.
- For $\ell = 4$: For fixed u, loop over v with $v > u$ and pick $s \in P_u$ closest to $S_\ell - a_u - a_v$. This is still $\mathcal{O}(n^2 \log n)$.

A: Adjusted Average

Problem Author: Ludo Pulles

- **Problem:** Given $n \leq 1500$ integers a_i, remove at most $k \leq 4$ of them to get an average as close as possible to the target $\overline{x}$.
- When removing exactly $0 \leq \ell \leq k$ numbers, we need to find ℓ integers with sum as close as possible to $S_\ell = \sum_i a_i - \ell \cdot \overline{x}$.
- For $\ell \leq 2$: iterate over all combinations $\longrightarrow \mathcal{O}(n^\ell/\ell!)$ time.
- For $\ell = 3, 4$: this is too slow so use *meet-in-the-middle*:



- For $\ell = 3$: loop over $u = 1, \ldots, n$, update P_u and then take $s \in P_u$ closest to $S_\ell - a_u$.
- Use an ordered set (BBST) for P_u, giving the time complexity $\mathcal{O}(n^2 \log n)$.
- For $\ell = 4$: For fixed u, loop over v with $v > u$ and pick $s \in P_u$ closest to $S_\ell - a_u - a_v$. This is still $\mathcal{O}(n^2 \log n)$.

Statistics: 25 submissions, 3 accepted, 13 unknown

B: Bellevue

Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given the profile of an island, find the point with the largest viewing angle of the sea.

B: Bellevue

Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given the profile of an island, find the point with the largest viewing angle of the sea.
- **Observation:** If point P is blocking the view of the edge of the island from point Q, you can see more sea in P than Q.

B: Bellevue

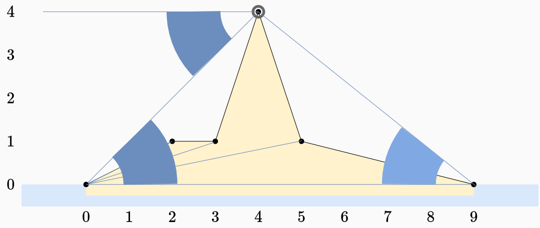
Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given the profile of an island, find the point with the largest viewing angle of the sea.
- **Observation:** If point P is blocking the view of the edge of the island from point Q, you can see more sea in P than Q.
- The answer is always an angle from the start/end of the island to another point.

B: Bellevue

Problem Author: Ragnar Groot Koerkamp

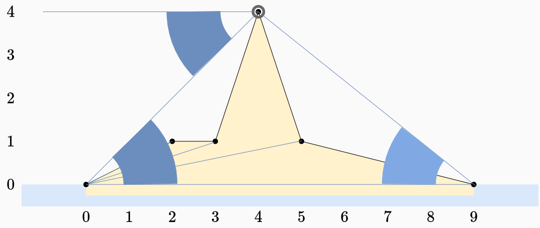
- **Problem:** Given the profile of an island, find the point with the largest viewing angle of the sea.
- **Observation:** If point P is blocking the view of the edge of the island from point Q, you can see more sea in P than Q.
- The answer is always an angle from the start/end of the island to another point.
- **Solution:** take the maximum angle around the start/end.



B: Bellevue

Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given the profile of an island, find the point with the largest viewing angle of the sea.
- **Observation:** If point P is blocking the view of the edge of the island from point Q, you can see more sea in P than Q.
- The answer is always an angle from the start/end of the island to another point.
- **Solution:** take the maximum angle around the start/end.

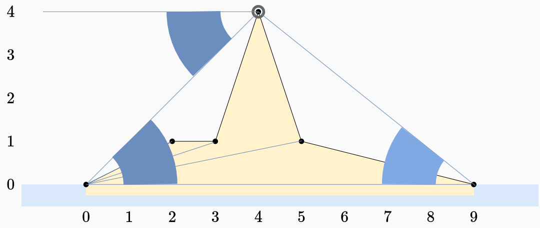


- **Alternative solution:** compute the convex hull and iterate over it.

B: Bellevue

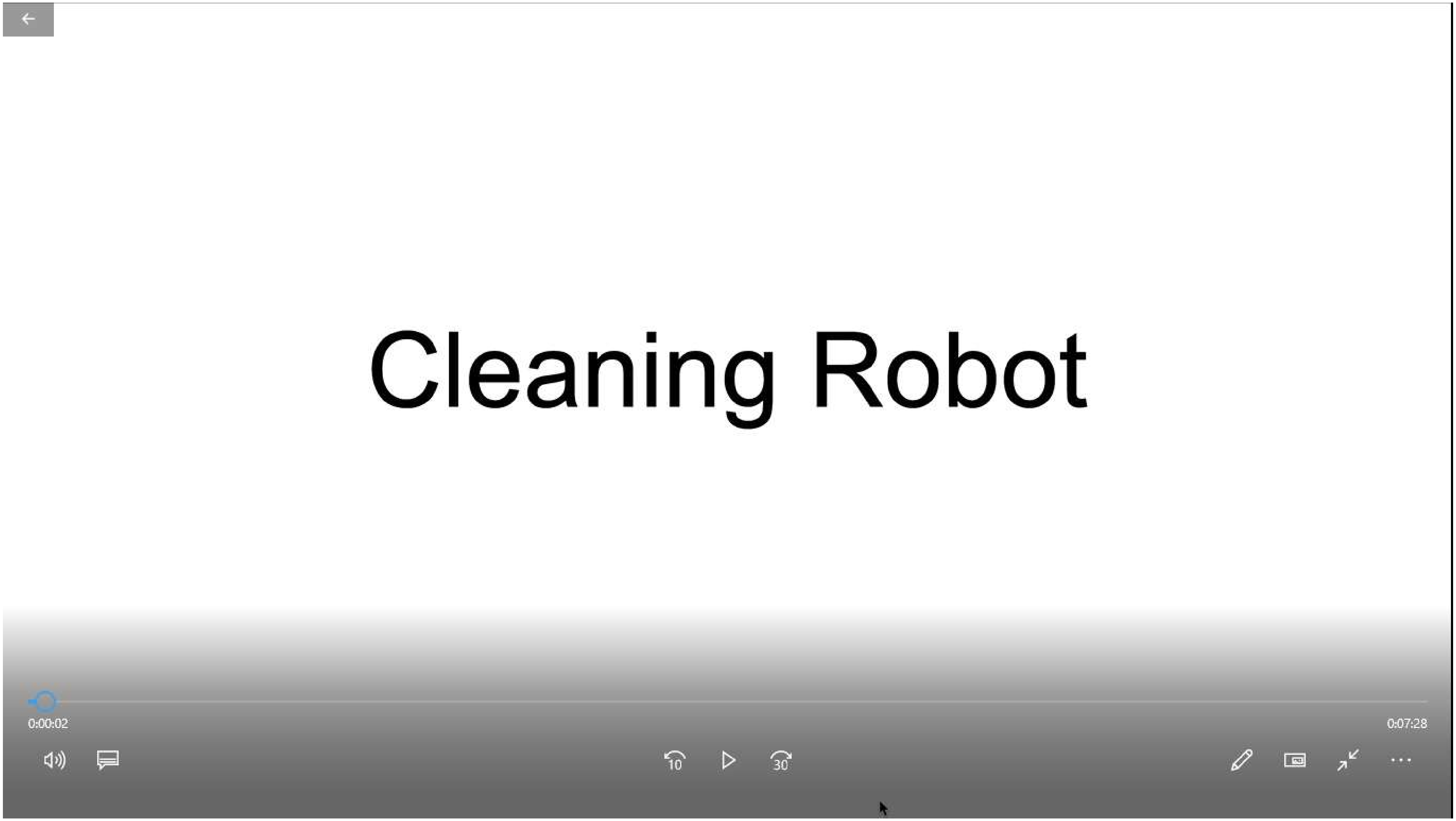
Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given the profile of an island, find the point with the largest viewing angle of the sea.
- **Observation:** If point P is blocking the view of the edge of the island from point Q, you can see more sea in P than Q.
- The answer is always an angle from the start/end of the island to another point.
- **Solution:** take the maximum angle around the start/end.



- **Alternative solution:** compute the convex hull and iterate over it.

Statistics: 102 submissions, 41 accepted, 18 unknown


Cleaning Robot
0:00:02
0:07:28
10
30

Cleaning Robot

Two steps:

- Preprocess the room so we can answer: what's the largest square that fits, starting here?
- Binary search on robot size, testing connectivity and coverage.

Cleaning Robot

Calculate suffix lengths on rows, right to left:

		4	3	2	1
	5	4	3	2	1
6	5	4	3	2	1
6	5	4	3	2	1
5	4	3	2	1	
4	3	2	1		

Cleaning Robot

Calculate suffix lengths on columns, bottom to top:

		6	6	5	4
	5	5	5	4	3
4	4	4	4	3	2
3	3	3	3	2	1
2	2	2	2	1	
1	1	1	1		

Cleaning Robot

Calculate biggest square, reverse raster order.

Minimum of:

- Row suffix sum,
- Column suffix sum,
- 1+square to southeast

We call this the unit square's *label.*

		4	3	2	1
	4	3	3	2	1
4	3	3	2	2	1
3	3	2	2	1	1
2	2	2	1	1	
1	1	1	1		

Cleaning Robot

Binary search on possible size of robot.

Monotonic function, so binary search will work:

- If a bot of size s works, so do all smaller bots;
- If a bot larger than s fails to work, so do all larger bots.

Cleaning Robot

To test if size s works, we need to check:

- Connectedness: are all unit squares labeled s or larger connected?
- Coverage: do all unobstructed squares have a unit square labeled s or more within s units up and to the left?

Cleaning Robot

Connectedness test at size s is just breadth-first search on unit squares labeled s or greater.

4? No.

3? Yes.

		4	3	2	1
	4	3	3	2	1
4	3	3	2	2	1
3	3	2	2	1	1
2	2	2	1	1	
1	1	1	1		

Cleaning Robot

Coverage test. Initialize a row vector v with zeros.

For each row, do the following:

- Extend from squares labeled s or bigger, else row-1.
- Check each unit square for coverage

		4	3	2	1
	4	3	3	2	1
4	3	3	2	2	1
3	3	2	2	1	1
2	2	2	1	1	
1	1	1	1		

Cleaning Robot

Extension of row k at size s:

```
j = 0
for i in [0,w):
  if label(k, i) >= s:
    while j<i+s:
      v[j++] = s
  else:
    if j==i:
      v[j++]--
```

		4	3	2	1
	4	3	3	2	1
4	3	3	2	2	1
3	3	2	2	1	1
2	2	2	1	1	
1	1	1	1		

Cleaning Robot

Start with [0 0 0 0 0 0]; s=3:

Row 1: [-1 -1 3 3 3 3]

Row 2: [-2 3 3 3 3 3]

Row 3: [3 3 3 3 3 2]

Row 4: [3 3 3 3 2 1]

Row 5: [2 2 2 2 1 0]

Row 6: [1 1 1 1 0 -1]

		4	3	2	1
	4	3	3	2	1
4	3	3	2	2	1
3	3	2	2	1	1
2	2	2	1	1	
1	1	1	1		

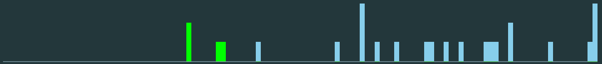
Cleaning Robot

Asymptotic complexity for board of size n x m is

$O(n\ m \log(\min(n,m)))$

D: Dimensional Debugging

Problem Author: Ragnar Groot Koerkamp



- **Problem:** Given n algorithms that only work when their input $\vec{x}$ is small enough ($\vec{x} \leq \vec{H}$), can you verify the correctness of all of them on sufficiently large inputs ($\vec{x} \geq \vec{L}$)?

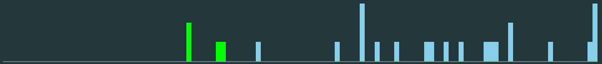
D: Dimensional Debugging

Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given n algorithms that only work when their input $\vec{x}$ is small enough ($\vec{x} \leq \vec{H}$), can you verify the correctness of all of them on sufficiently large inputs ($\vec{x} \geq \vec{L}$)?
- Since you know the answer in $\vec{0}$, you can verify the correctness of all algorithms with $\vec{L} = \vec{0}$.

D: Dimensional Debugging

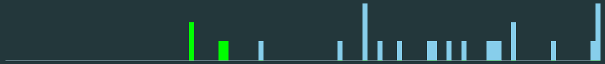
Problem Author: Ragnar Groot Koerkamp



- **Problem:** Given n algorithms that only work when their input $\vec{x}$ is small enough ($\vec{x} \leq \vec{H}$), can you verify the correctness of all of them on sufficiently large inputs ($\vec{x} \geq \vec{L}$)?
- Since you know the answer in $\vec{0}$, you can verify the correctness of all algorithms with $\vec{L} = \vec{0}$.
- Once algorithm i has been verified, you can verify other algorithms j for which $\vec{L}_j \leq \vec{H}_i$.

D: Dimensional Debugging

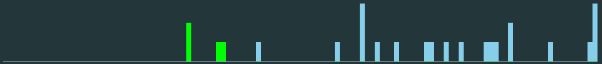
Problem Author: Ragnar Groot Koerkamp



- **Problem:** Given n algorithms that only work when their input $\vec{x}$ is small enough ($\vec{x} \leq \vec{H}$), can you verify the correctness of all of them on sufficiently large inputs ($\vec{x} \geq \vec{L}$)?
- Since you know the answer in $\vec{0}$, you can verify the correctness of all algorithms with $\vec{L} = \vec{0}$.
- Once algorithm i has been verified, you can verify other algorithms j for which $\vec{L}_j \leq \vec{H}_i$.
- More generally, the number of algorithms reachable in this way can be counted using BFS or DFS (floodfill).

D: Dimensional Debugging

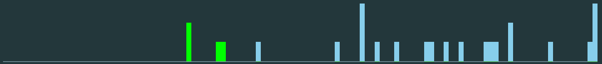
Problem Author: Ragnar Groot Koerkamp



- **Problem:** Given n algorithms that only work when their input $\vec{x}$ is small enough ($\vec{x} \leq \vec{H}$), can you verify the correctness of all of them on sufficiently large inputs ($\vec{x} \geq \vec{L}$)?
- Since you know the answer in $\vec{0}$, you can verify the correctness of all algorithms with $\vec{L} = \vec{0}$.
- Once algorithm i has been verified, you can verify other algorithms j for which $\vec{L}_j \leq \vec{H}_i$.
- More generally, the number of algorithms reachable in this way can be counted using BFS or DFS (floodfill).
- Complexity: $\mathcal{O}(n^2)$

D: Dimensional Debugging

Problem Author: Ragnar Groot Koerkamp



- **Problem:** Given n algorithms that only work when their input $\vec{x}$ is small enough ($\vec{x} \leq \vec{H}$), can you verify the correctness of all of them on sufficiently large inputs ($\vec{x} \geq \vec{L}$)?
- Since you know the answer in $\vec{0}$, you can verify the correctness of all algorithms with $\vec{L} = \vec{0}$.
- Once algorithm i has been verified, you can verify other algorithms j for which $\vec{L}_j \leq \vec{H}_i$.
- More generally, the number of algorithms reachable in this way can be counted using BFS or DFS (floodfill).
- Complexity: $\mathcal{O}(n^2)$

Statistics: 25 submissions, 4 accepted, 21 unknown

E: Equalising Audio

Problem Author: Abe Wits

- **Problem:** Normalise the amplitudes $a_1, \ldots, a_n$ to a perceived loudness of $\frac{1}{n}\sum_{i=1}^{n} a_i^2 = x$.

E: Equalising Audio

Problem Author: Abe Wits

- **Problem:** Normalise the amplitudes $a_1, \ldots, a_n$ to a perceived loudness of $\frac{1}{n}\sum_{i=1}^{n} a_i^2 = x$.
- **Solution:** First, compute the current perceived loudness $x' = \frac{1}{n}\sum_{i=1}^{n} a_i^2$.
  Then, output $\sqrt{x/x'} \cdot a_1, \ldots, \sqrt{x/x'} \cdot a_n$ with sufficient precision.

E: Equalising Audio

Problem Author: Abe Wits

- **Problem:** Normalise the amplitudes $a_1, \ldots, a_n$ to a perceived loudness of $\frac{1}{n}\sum_{i=1}^{n} a_i^2 = x$.
- **Solution:** First, compute the current perceived loudness $x' = \frac{1}{n}\sum_{i=1}^{n} a_i^2$.
  Then, output $\sqrt{x/x'} \cdot a_1, \ldots, \sqrt{x/x'} \cdot a_n$ with sufficient precision.
- Verification:

$$\frac{1}{n}\sum_{i=1}^{n}(\sqrt{x/x'}a_i)^2 = \frac{x}{x'} \cdot \frac{1}{n}\sum_{i=1}^{n} a_i^2 = \frac{x}{x'} \cdot x' = x.$$

E: Equalising Audio

Problem Author: Abe Wits

- **Problem:** Normalise the amplitudes $a_1, \dots, a_n$ to a perceived loudness of $\frac{1}{n}\sum_{i=1}^{n} a_i^2 = x$.
- **Solution:** First, compute the current perceived loudness $x' = \frac{1}{n}\sum_{i=1}^{n} a_i^2$.
  Then, output $\sqrt{x/x'} \cdot a_1, \dots, \sqrt{x/x'} \cdot a_n$ with sufficient precision.
- Verification:
$$\frac{1}{n}\sum_{i=1}^{n}(\sqrt{x/x'}a_i)^2 = \frac{x}{x'} \cdot \frac{1}{n}\sum_{i=1}^{n} a_i^2 = \frac{x}{x'} \cdot x' = x.$$
- Exception: if $x' = 0$, return $0, \dots, 0$.

E: Equalising Audio

Problem Author: Abe Wits

- **Problem:** Normalise the amplitudes $a_1, \ldots, a_n$ to a perceived loudness of $\frac{1}{n}\sum_{i=1}^{n} a_i^2 = x$.
- **Solution:** First, compute the current perceived loudness $x' = \frac{1}{n}\sum_{i=1}^{n} a_i^2$.
  Then, output $\sqrt{x/x'} \cdot a_1, \ldots, \sqrt{x/x'} \cdot a_n$ with sufficient precision.
- Verification:

$$\frac{1}{n}\sum_{i=1}^{n}(\sqrt{x/x'}a_i)^2 = \frac{x}{x'} \cdot \frac{1}{n}\sum_{i=1}^{n} a_i^2 = \frac{x}{x'} \cdot x' = x.$$

- Exception: if $x' = 0$, return $0, \ldots, 0$.
  - And apparently, we forgot to test the case `-1 1`, where the sum is 0

E: Equalising Audio

Problem Author: Abe Wits

- **Problem:** Normalise the amplitudes $a_1, \ldots, a_n$ to a perceived loudness of $\frac{1}{n}\sum_{i=1}^n a_i^2 = x$.
- **Solution:** First, compute the current perceived loudness $x' = \frac{1}{n}\sum_{i=1}^n a_i^2$.
  Then, output $\sqrt{x/x'} \cdot a_1, \ldots, \sqrt{x/x'} \cdot a_n$ with sufficient precision.
- Verification:

$$\frac{1}{n}\sum_{i=1}^{n}(\sqrt{x/x'}a_i)^2 = \frac{x}{x'} \cdot \frac{1}{n}\sum_{i=1}^{n} a_i^2 = \frac{x}{x'} \cdot x' = x.$$

- Exception: if $x' = 0$, return $0, \ldots, 0$.
  - And apparently, we forgot to test the case `-1 1`, where the sum is 0
- Pitfall: multiplying two `ints` causes integer overflow, use `long` instead!

E: Equalising Audio

Problem Author: Abe Wits

- **Problem:** Normalise the amplitudes $a_1, \ldots, a_n$ to a perceived loudness of $\frac{1}{n}\sum_{i=1}^{n} a_i^2 = x$.
- **Solution:** First, compute the current perceived loudness $x' = \frac{1}{n}\sum_{i=1}^{n} a_i^2$.
  Then, output $\sqrt{x/x'} \cdot a_1, \ldots, \sqrt{x/x'} \cdot a_n$ with sufficient precision.
- Verification:

$$\frac{1}{n}\sum_{i=1}^{n}(\sqrt{x/x'}a_i)^2 = \frac{x}{x'} \cdot \frac{1}{n}\sum_{i=1}^{n} a_i^2 = \frac{x}{x'} \cdot x' = x.$$

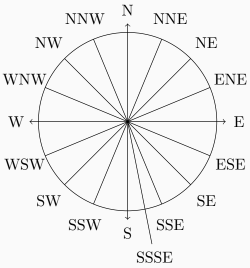
- Exception: if $x' = 0$, return $0, \ldots, 0$.
  - And apparently, we forgot to test the case `-1 1`, where the sum is 0
- Pitfall: multiplying two `int`s causes integer overflow, use `long` instead!

Statistics: 127 submissions, 50 accepted, 6 unknown

F: Failing Flagship

Problem Author: Ruben Brokkelkamp

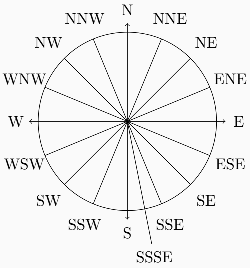
- **Problem:** Compute the minimum angle in degrees between two wind directions.



F: Failing Flagship

Problem Author: Ruben Brokkelkamp

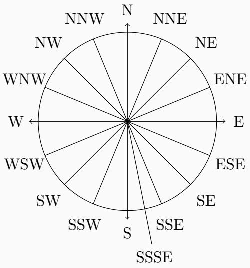
- **Problem:** Compute the minimum angle in degrees between two wind directions.
- **Solution:**



F: Failing Flagship

Problem Author: Ruben Brokkelkamp

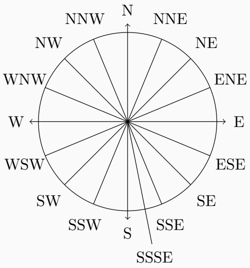
- **Problem:** Compute the minimum angle in degrees between two wind directions.
- **Solution:**
  - Convert both wind directions into degrees $d_1 \geq d_2$.



F: Failing Flagship

Problem Author: Ruben Brokkelkamp

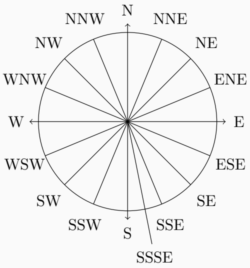
- **Problem:** Compute the minimum angle in degrees between two wind directions.
- **Solution:**
  - Convert both wind directions into degrees $d_1 \geq d_2$.
  - **Observation:** For every extra letter the degrees it represents halves: $45, 22.5, 11.25, 6.125, \ldots$.



F: Failing Flagship

Problem Author: Ruben Brokkelkamp

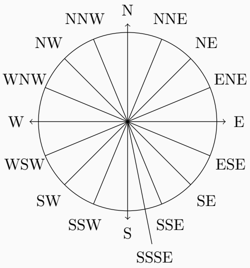
- **Problem:** Compute the minimum angle in degrees between two wind directions.
- **Solution:**
  - Convert both wind directions into degrees $d_1 \geq d_2$.
  - **Observation:** For every extra letter the degrees it represents halves: $45, 22.5, 11.25, 6.125, \ldots$.
  - Return $\min(d_2 - d_1, 360 + d_1 - d_2)$.



F: Failing Flagship

Problem Author: Ruben Brokkelkamp

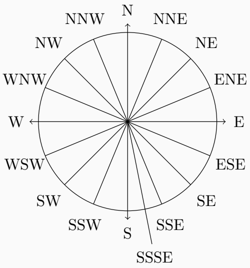
- **Problem:** Compute the minimum angle in degrees between two wind directions.
- **Solution:**
  - Convert both wind directions into degrees $d_1 \geq d_2$.
  - **Observation:** For every extra letter the degrees it represents halves: $45, 22.5, 11.25, 6.125, \ldots$.
  - Return $\min(d_2 - d_1, 360 + d_1 - d_2)$.
- **Remark:** Only 29 characters are needed for the required precision.



F: Failing Flagship

Problem Author: Ruben Brokkelkamp

- **Problem:** Compute the minimum angle in degrees between two wind directions.
- **Solution:**
  - Convert both wind directions into degrees $d_1 \geq d_2$.
  - **Observation:** For every extra letter the degrees it represents halves: $45, 22.5, 11.25, 6.125, \ldots$.
  - Return $\min(d_2 - d_1, 360 + d_1 - d_2)$.
- **Remark:** Only 29 characters are needed for the required precision.



Statistics: 144 submissions, 39 accepted, 21 unknown

G: Grinding Gravel

Problem Author: Daan van Gent, Onno Berrevoets

- **Problem:** Given $n \leq 100$ integers, split them into groups of size $k \leq 8$ making as few cuts as possible.

G: Grinding Gravel

Problem Author: Daan van Gent, Onno Berrevoets

- **Problem:** Given $n \leq 100$ integers, split them into groups of size $k \leq 8$ making as few cuts as possible.
- **Equivalent problem:** Given n integers, partition them into as many groups as possible with sum a multiple of k.

G: Grinding Gravel

Problem Author: Daan van Gent, Onno Berrevoets

- **Problem:** Given $n \leq 100$ integers, split them into groups of size $k \leq 8$ making as few cuts as possible.
- **Equivalent problem:** Given n integers, partition them into as many groups as possible with sum a multiple of k.
- Greedy 1: Each number $x \geq k$ is replaced by $x \mod k$. Count the numbers with each remainder.

G: Grinding Gravel

Problem Author: Daan van Gent, Onno Berrevoets

- **Problem:** Given $n \leq 100$ integers, split them into groups of size $k \leq 8$ making as few cuts as possible.
- **Equivalent problem:** Given n integers, partition them into as many groups as possible with sum a multiple of k.
- Greedy 1: Each number $x \geq k$ is replaced by $x \mod k$. Count the numbers with each remainder.
- Greedy 2: For $x < k/2$, we can pair up x and $k - x$. Each $x = 0$ is its own group.

G: Grinding Gravel

Problem Author: Daan van Gent, Onno Berrevoets

- **Problem:** Given $n \leq 100$ integers, split them into groups of size $k \leq 8$ making as few cuts as possible.
- **Equivalent problem:** Given n integers, partition them into as many groups as possible with sum a multiple of k.
- Greedy 1: Each number $x \geq k$ is replaced by $x \mod k$. Count the numbers with each remainder.
- Greedy 2: For $x < k/2$, we can pair up x and $k - x$. Each $x = 0$ is its own group.
- We are left with at most 4 different values: 1 or 7, 2 or 6, 3 or 5, and at most one 4.

G: Grinding Gravel

Problem Author: Daan van Gent, Onno Berrevoets

- **Problem:** Given $n \leq 100$ integers, split them into groups of size $k \leq 8$ making as few cuts as possible.
- **Equivalent problem:** Given n integers, partition them into as many groups as possible with sum a multiple of k.
- Greedy 1: Each number $x \geq k$ is replaced by $x \mod k$. Count the numbers with each remainder.
- Greedy 2: For $x < k/2$, we can pair up x and $k - x$. Each $x = 0$ is its own group.
- We are left with at most 4 different values: 1 or 7, 2 or 6, 3 or 5, and at most one 4.
- Now, do a DP on state $[c_1, \dots, c_{k-1}]$, the counts for each remainder.

G: Grinding Gravel

Problem Author: Daan van Gent, Onno Berrevoets

- **Problem:** Given $n \leq 100$ integers, split them into groups of size $k \leq 8$ making as few cuts as possible.
- **Equivalent problem:** Given n integers, partition them into as many groups as possible with sum a multiple of k.
- Greedy 1: Each number $x \geq k$ is replaced by $x \mod k$. Count the numbers with each remainder.
- Greedy 2: For $x < k/2$, we can pair up x and $k - x$. Each $x = 0$ is its own group.
- We are left with at most 4 different values: 1 or 7, 2 or 6, 3 or 5, and at most one 4.
- Now, do a DP on state $[c_1, \ldots, c_{k-1}]$, the counts for each remainder.
  - For each precomputed (minimal) subset with sum $0 \mod k$ remove it and recurse.

G: Grinding Gravel

Problem Author: Daan van Gent, Onno Berrevoets

- **Problem:** Given $n \leq 100$ integers, split them into groups of size $k \leq 8$ making as few cuts as possible.
- **Equivalent problem:** Given n integers, partition them into as many groups as possible with sum a multiple of k.
- Greedy 1: Each number $x \geq k$ is replaced by $x \mod k$. Count the numbers with each remainder.
- Greedy 2: For $x < k/2$, we can pair up x and $k - x$. Each $x = 0$ is its own group.
- We are left with at most 4 different values: 1 or 7, 2 or 6, 3 or 5, and at most one 4.
- Now, do a DP on state $[c_1, \ldots, c_{k-1}]$, the counts for each remainder.
  - For each precomputed (minimal) subset with sum $0 \mod k$ remove it and recurse.
  - **Simpler alternative:** Merge the largest remainder with another one, and update the state. $\rightarrow$ Too slow when counts are $1 \times 4, 30 \times 5, 30 \times 6, 30 \times 7$.

G: Grinding Gravel

Problem Author: Daan van Gent, Onno Berrevoets

- **Problem:** Given $n \leq 100$ integers, split them into groups of size $k \leq 8$ making as few cuts as possible.
- **Equivalent problem:** Given n integers, partition them into as many groups as possible with sum a multiple of k.
- Greedy 1: Each number $x \geq k$ is replaced by $x \mod k$. Count the numbers with each remainder.
- Greedy 2: For $x < k/2$, we can pair up x and $k - x$. Each $x = 0$ is its own group.
- We are left with at most 4 different values: 1 or 7, 2 or 6, 3 or 5, and at most one 4.
- Now, do a DP on state $[c_1, \ldots, c_{k-1}]$, the counts for each remainder.
  - For each precomputed (minimal) subset with sum $0 \mod k$ remove it and recurse.
  - **Simpler alternative:** Merge the largest remainder with another one, and update the state. $\rightarrow$ Too slow when counts are $1 \times 4, 30 \times 5, 30 \times 6, 30 \times 7$.
  - **Instead:** merge the least-occurring element with one of the others.

G: Grinding Gravel

Problem Author: Daan van Gent, Onno Berrevoets

- **Problem:** Given $n \leq 100$ integers, split them into groups of size $k \leq 8$ making as few cuts as possible.
- **Equivalent problem:** Given n integers, partition them into as many groups as possible with sum a multiple of k.
- Greedy 1: Each number $x \geq k$ is replaced by $x \mod k$. Count the numbers with each remainder.
- Greedy 2: For $x < k/2$, we can pair up x and $k - x$. Each $x = 0$ is its own group.
- We are left with at most 4 different values: 1 or 7, 2 or 6, 3 or 5, and at most one 4.
- Now, do a DP on state $[c_1, \ldots, c_{k-1}]$, the counts for each remainder.
  - For each precomputed (minimal) subset with sum $0 \mod k$ remove it and recurse.
  - **Simpler alternative:** Merge the largest remainder with another one, and update the state. $\rightarrow$ Too slow when counts are $1 \times 4, 30 \times 5, 30 \times 6, 30 \times 7$.
  - **Instead:** merge the least-occurring element with one of the others.
  - **Even simpler:** remove any one of the remaining elements. If this makes the total sum be $0 \mod k$, add one.

G: Grinding Gravel

Problem Author: Daan van Gent, Onno Berrevoets

- **Problem:** Given $n \leq 100$ integers, split them into groups of size $k \leq 8$ making as few cuts as possible.
- **Equivalent problem:** Given n integers, partition them into as many groups as possible with sum a multiple of k.
- Greedy 1: Each number $x \geq k$ is replaced by $x \mod k$. Count the numbers with each remainder.
- Greedy 2: For $x < k/2$, we can pair up x and $k - x$. Each $x = 0$ is its own group.
- We are left with at most 4 different values: 1 or 7, 2 or 6, 3 or 5, and at most one 4.
- Now, do a DP on state $[c_1, \ldots, c_{k-1}]$, the counts for each remainder.
  - For each precomputed (minimal) subset with sum $0 \mod k$ remove it and recurse.
  - **Simpler alternative:** Merge the largest remainder with another one, and update the state. $\rightarrow$ Too slow when counts are $1 \times 4, 30 \times 5, 30 \times 6, 30 \times 7$.
  - **Instead:** merge the least-occurring element with one of the others.
  - **Even simpler:** remove any one of the remaining elements. If this makes the total sum be $0 \mod k$, add one.
- Instead of 4-deep nested loops, we can use a dictionary of tuples.

G: Grinding Gravel

Problem Author: Daan van Gent, Onno Berrevoets

- **Problem:** Given $n \leq 100$ integers, split them into groups of size $k \leq 8$ making as few cuts as possible.
- **Equivalent problem:** Given n integers, partition them into as many groups as possible with sum a multiple of k.
- Greedy 1: Each number $x \geq k$ is replaced by $x \mod k$. Count the numbers with each remainder.
- Greedy 2: For $x < k/2$, we can pair up x and $k - x$. Each $x = 0$ is its own group.
- We are left with at most 4 different values: 1 or 7, 2 or 6, 3 or 5, and at most one 4.
- Now, do a DP on state $[c_1, \ldots, c_{k-1}]$, the counts for each remainder.
  - For each precomputed (minimal) subset with sum $0 \mod k$ remove it and recurse.
  - **Simpler alternative:** Merge the largest remainder with another one, and update the state. $\rightarrow$ Too slow when counts are $1 \times 4, 30 \times 5, 30 \times 6, 30 \times 7$.
  - **Instead:** merge the least-occurring element with one of the others.
  - **Even simpler:** remove any one of the remaining elements. If this makes the total sum be $0 \mod k$, add one.
- Instead of 4-deep nested loops, we can use a dictionary of tuples.

Statistics: 5 submissions, 1 accepted, 1 unknown

H: Heavy Hauling

Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given n boxes at given positions. Moving a box d positions costs d^2. What is the minimal cost to make all box positions distinct?

H: Heavy Hauling

Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given n boxes at given positions. Moving a box d positions costs d^2.
  What is the minimal cost to make all box positions distinct?
- **Observation:** The boxes will remain in their original order (they will never overtake each other).

H: Heavy Hauling

Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given n boxes at given positions. Moving a box d positions costs d^2.
  What is the minimal cost to make all box positions distinct?
- **Observation:** The boxes will remain in their original order (they will never overtake each other).
- **Observation:** Groups of consecutive boxes map to an interval.

H: Heavy Hauling

Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given n boxes at given positions. Moving a box d positions costs d^2. What is the minimal cost to make all box positions distinct?
- **Observation:** The boxes will remain in their original order (they will never overtake each other).
- **Observation:** Groups of consecutive boxes map to an interval.
- The cost of moving a box from position p to a position x, can be modelled with a quadratic function $C_p(x) = (x - p)^2$.
  - Example: For one box with original position 3 moved to position x, $C_3(x) = (x - 3)^2 = x^2 - 6x + 9$.

H: Heavy Hauling

Problem Author: Ragnar Groot Koerkamp

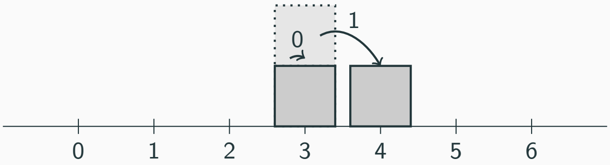
- **Problem:** Given n boxes at given positions. Moving a box d positions costs d^2.
  What is the minimal cost to make all box positions distinct?
- **Observation:** The boxes will remain in their original order (they will never overtake each other).
- **Observation:** Groups of consecutive boxes map to an interval.
- The cost of moving a box from position p to a position x, can be modelled with a quadratic function $C_p(x) = (x - p)^2$.
  - Example: For one box with original position 3 moved to position x, $C_3(x) = (x-3)^2 = x^2 - 6x + 9$.
- When adding the costs of two groups of boxes that overlap together, *translate* the cost function of the right group of boxes by the size of the left group.
  - Example: For two boxes with original position 3, moved such that the left-most box is at position x, the summed cost is $C_{3,3}(x) = C_3(x) + C_3(x+1) = (x-3)^2 + (x-2)^2 = 2x^2 - 10x + 13$.

H: Heavy Hauling

Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given n boxes at given positions. Moving a box d positions costs d^2.
  What is the minimal cost to make all box positions distinct?
- The cost of a box at a position x, starting at position p, can be modelled with a quadratic function $C_p(x) = (x - p)^2$.
  - For two boxes that start at position 3, the summed cost is
    $C_{3,3}(x) = C_3(x) + C_3(x+1) = (x-3)^2 + (x-2)^2 = 2x^2 - 10x + 13$.

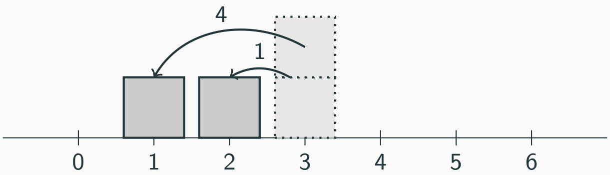
Proof by example:



$$C_{3,3}(3) = 2 \cdot 3^2 - 10 \cdot 3 + 13 = 1$$

- **Problem:** Given n boxes at given positions. Moving a box d positions costs d^2.
  What is the minimal cost to make all box positions distinct?
- The cost of a box at a position x, starting at position p, can be modelled with a quadratic function $C_p(x) = (x - p)^2$.
  - For two boxes that start at position 3, the summed cost is
    $C_{3,3}(x) = C_3(x) + C_3(x+1) = (x-3)^2 + (x-2)^2 = 2x^2 - 10x + 13.$

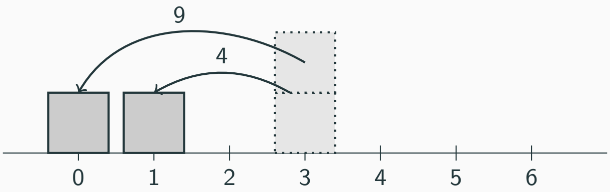
Proof by example:



$$C_{3,3}(1) = 2 \cdot 1^2 - 10 \cdot 1 + 13 = 5$$

- **Problem:** Given n boxes at given positions. Moving a box d positions costs d^2.
  What is the minimal cost to make all box positions distinct?
- The cost of a box at a position x, starting at position p, can be modelled with a quadratic function $C_p(x) = (x - p)^2$.
  - For two boxes that start at position 3, the summed cost is
    $C_{3,3}(x) = C_3(x) + C_3(x + 1) = (x - 3)^2 + (x - 2)^2 = 2x^2 - 10x + 13.$

Proof by example:



$$C_{3,3}(0) = 2 \cdot 0^2 - 10 \cdot 0 + 13 = 13$$

H: Heavy Hauling

Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given n boxes at given positions. Moving a box d positions costs d^2.
  What is the minimal cost to make all box positions distinct?
- **Solution:** Add every box from left to right, maintaining the optimal placement by maintaining the cost function for every group of boxes.

H: Heavy Hauling

Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given n boxes at given positions. Moving a box d positions costs d^2.
  What is the minimal cost to make all box positions distinct?
- **Solution:** Add every box from left to right, maintaining the optimal placement by maintaining the cost function for every group of boxes.
- If two groups of boxes touch or overlap, merge them into one group by summing their (possibly translated) costs.
  - This new group may overlap with its preceding group after the merge, so merge recursively.

H: Heavy Hauling

Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given n boxes at given positions. Moving a box d positions costs d^2.
  What is the minimal cost to make all box positions distinct?
- **Solution:** Add every box from left to right, maintaining the optimal placement by maintaining the cost function for every group of boxes.
- If two groups of boxes touch or overlap, merge them into one group by summing their (possibly translated) costs.
  - This new group may overlap with its preceding group after the merge, so merge recursively.
- For every group with cost $C(x) = ax^2 + bx + c$, the minimal cost is:

$$C\left(\left\lfloor \frac{-b}{2a} + \frac{1}{2} \right\rfloor\right)$$

H: Heavy Hauling

Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given n boxes at given positions. Moving a box d positions costs d^2.
  What is the minimal cost to make all box positions distinct?
- **Solution:** Add every box from left to right, maintaining the optimal placement by maintaining the cost function for every group of boxes.
- If two groups of boxes touch or overlap, merge them into one group by summing their (possibly translated) costs.
  - This new group may overlap with its preceding group after the merge, so merge recursively.
- For every group with cost $C(x) = ax^2 + bx + c$, the minimal cost is:

$$C\left(\left\lfloor \frac{-b}{2a} + \frac{1}{2} \right\rfloor\right)$$

- The total runtime is $\mathcal{O}(n)$ (after sorting): we do at most $n-1$ merges.

H: Heavy Hauling

Problem Author: Ragnar Groot Koerkamp

- **Problem:** Given n boxes at given positions. Moving a box d positions costs d^2.
  What is the minimal cost to make all box positions distinct?
- **Solution:** Add every box from left to right, maintaining the optimal placement by maintaining the cost function for every group of boxes.
- If two groups of boxes touch or overlap, merge them into one group by summing their (possibly translated) costs.
  - This new group may overlap with its preceding group after the merge, so merge recursively.
- For every group with cost $C(x) = ax^2 + bx + c$, the minimal cost is:

$$C\left(\left\lfloor \frac{-b}{2a} + \frac{1}{2} \right\rfloor\right)$$

- The total runtime is $\mathcal{O}(n)$ (after sorting): we do at most $n - 1$ merges.

Statistics: 10 submissions, 0 accepted, 9 unknown

I: Ice Growth

Problem Author: Jorke de Vlas

- Given a weather report for n days and k people that have a required minimal ice thickness, how many days can each person skate?

I: Ice Growth

Problem Author: Jorke de Vlas

- Given a weather report for n days and k people that have a required minimal ice thickness, how many days can each person skate?
- Compute and store the ice thickness for each day.
  - Ice-thickness can't be negative.
  - Use integers to count 'degrees of frost'.

I: Ice Growth

Problem Author: Jorke de Vlas

- Given a weather report for n days and k people that have a required minimal ice thickness, how many days can each person skate?
- Compute and store the ice thickness for each day.
  - Ice-thickness can't be negative.
  - Use integers to count 'degrees of frost'.
- Sort the days by ice thickness [$\mathcal{O}(n\log(n))$].

I: Ice Growth

Problem Author: Jorke de Vlas

- Given a weather report for n days and k people that have a required minimal ice thickness, how many days can each person skate?
- Compute and store the ice thickness for each day.
  - Ice-thickness can't be negative.
  - Use integers to count 'degrees of frost'.
- Sort the days by ice thickness [$\mathcal{O}(n \log(n))$].
- For each person binary search how many days have the required thickness [$\mathcal{O}(k \log(n))$].

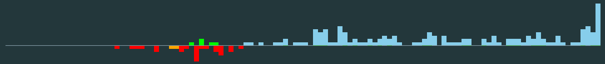
I: Ice Growth

Problem Author: Jorke de Vlas

- Given a weather report for n days and k people that have a required minimal ice thickness, how many days can each person skate?
- Compute and store the ice thickness for each day.
  - Ice-thickness can't be negative.
  - Use integers to count 'degrees of frost'.
- Sort the days by ice thickness [$\mathcal{O}(n \log(n))$].
- For each person binary search how many days have the required thickness [$\mathcal{O}(k \log(n))$].
- Alternative: store the number of days for each ice-thickness $\leq 10^6$, and accumulate once [$\mathcal{O}(k + n)$].

J: Jabbing Jets

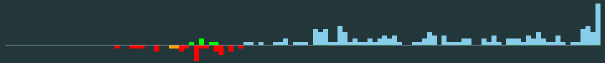
Problem Author: Abe Wits



- **Problem:** Given n concentric circles, find the maximal number of points on these circles such that the distance between any two points is at least e.

J: Jabbing Jets

Problem Author: Abe Wits



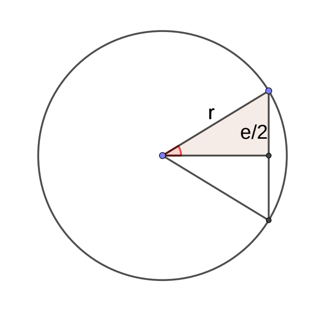
- **Problem:** Given n concentric circles, find the maximal number of points on these circles such that the distance between any two points is at least e.
- **Observation:** Because $r_{i+1} - r_i \geq e$, each circle can be considered separately.

J: Jabbing Jets

Problem Author: Abe Wits

- **Problem:** Given n concentric circles, find the maximal number of points on these circles such that the distance between any two points is at least e.
- **Observation:** Because $r_{i+1} - r_i \geq e$, each circle can be considered separately.
- The number of points on circle i is

$$\left\lfloor \frac{2\pi}{2\arcsin\left(\frac{e}{2r_i}\right)} \right\rfloor.$$



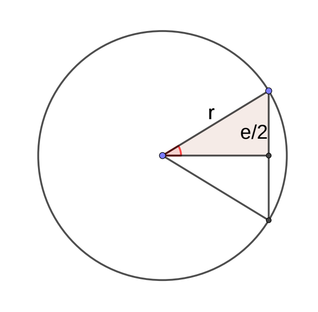
J: Jabbing Jets

Problem Author: Abe Wits

- **Problem:** Given n concentric circles, find the maximal number of points on these circles such that the distance between any two points is at least e.
- **Observation:** Because $r_{i+1} - r_i \geq e$, each circle can be considered separately.
- The number of points on circle i is

$$\left\lfloor \frac{2\pi}{2\arcsin\left(\frac{e}{2r_i}\right)} \right\rfloor.$$

- Edge case: if $2r_i < e$, the number of points is 1.



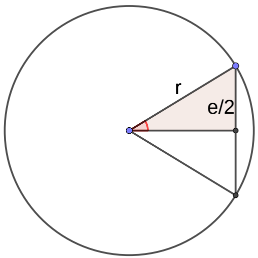
J: Jabbing Jets

Problem Author: Abe Wits

- **Problem:** Given n concentric circles, find the maximal number of points on these circles such that the distance between any two points is at least e.
- **Observation:** Because $r_{i+1} - r_i \geq e$, each circle can be considered separately.
- The number of points on circle i is

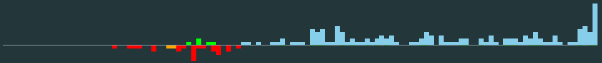
$$\left\lfloor \frac{2\pi}{2\arcsin\left(\frac{e}{2r_i}\right)} \right\rfloor.$$

- Edge case: if $2r_i < e$, the number of points is 1.
- Beware floating point issues!



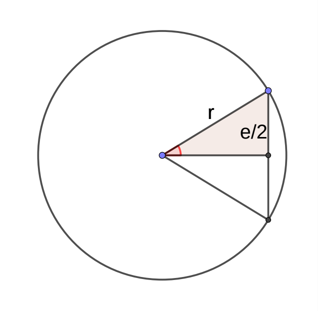
J: Jabbing Jets

Problem Author: Abe Wits



- **Problem:** Given n concentric circles, find the maximal number of points on these circles such that the distance between any two points is at least e.
- **Observation:** Because $r_{i+1} - r_i \geq e$, each circle can be considered separately.
- The number of points on circle i is

$$\left\lfloor \frac{2\pi}{2\arcsin\left(\frac{e}{2r_i}\right)} \right\rfloor.$$



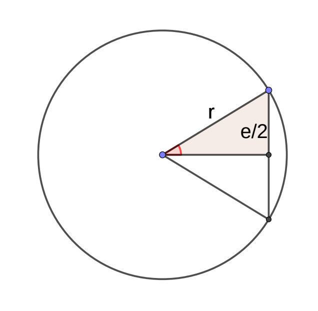
- Edge case: if $2r_i < e$, the number of points is 1.
- Beware floating point issues!
  - Add $0.5 \cdot 10^{-6}$ to every radius.

J: Jabbing Jets

Problem Author: Abe Wits

- **Problem:** Given n concentric circles, find the maximal number of points on these circles such that the distance between any two points is at least e.
- **Observation:** Because $r_{i+1} - r_i \geq e$, each circle can be considered separately.
- The number of points on circle i is

$$\left\lfloor \frac{2\pi}{2\arcsin\left(\frac{e}{2r_i}\right)} \right\rfloor.$$

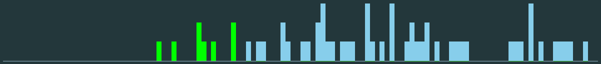


- Edge case: if $2r_i < e$, the number of points is 1.
- Beware floating point issues!
  - Add $0.5 \cdot 10^{-6}$ to every radius.

Statistics: 168 submissions, 5 accepted, 137 unknown

K: Knitting Patterns

Problem Author: Maarten Sijm



- **Problem:** Given a knitting pattern and amount of wool it costs for letting the wool strand unused, using the wool in a stitch, and for starting or ending the use of wool. Compute the minimal amount of wool required for every colour of wool.

K: Knitting Patterns

Problem Author: Maarten Sijm

- **Problem:** Given a knitting pattern and amount of wool it costs for letting the wool strand unused, using the wool in a stitch, and for starting or ending the use of wool. Compute the minimal amount of wool required for every colour of wool.
- **Observation:** Between two times a colour of wool is used, you either leave the strand through the back unused for the entire gap, or you immediately end the use at the beginning of the gap and start using it at the end.

K: Knitting Patterns

Problem Author: Maarten Sijm

- **Problem:** Given a knitting pattern and amount of wool it costs for letting the wool strand unused, using the wool in a stitch, and for starting or ending the use of wool. Compute the minimal amount of wool required for every colour of wool.
- **Observation:** Between two times a colour of wool is used, you either leave the strand through the back unused for the entire gap, or you immediately end the use at the beginning of the gap and start using it at the end.
- **Solution:** For every colour, iterate through the knitting pattern and remember the index of the last time the colour occurred. If you encounter the colour again, the marginal cost is the minimum between leaving the strand unused the whole time since the last time, and the sum of the costs for ending and starting. Runs in $\mathcal{O}(|w| \cdot n)$.

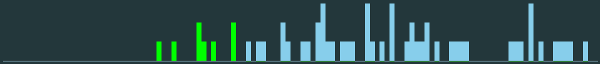
K: Knitting Patterns

Problem Author: Maarten Sijm

- **Problem:** Given a knitting pattern and amount of wool it costs for letting the wool strand unused, using the wool in a stitch, and for starting or ending the use of wool. Compute the minimal amount of wool required for every colour of wool.
- **Observation:** Between two times a colour of wool is used, you either leave the strand through the back unused for the entire gap, or you immediately end the use at the beginning of the gap and start using it at the end.
- **Solution:** For every colour, iterate through the knitting pattern and remember the index of the last time the colour occurred. If you encounter the colour again, the marginal cost is the minimum between leaving the strand unused the whole time since the last time, and the sum of the costs for ending and starting. Runs in $\mathcal{O}(|w| \cdot n)$.
- **Remark:** Can be done in $\mathcal{O}(n)$ by doing some bookkeeping and storing for every colour the last time it occurred.

K: Knitting Patterns

Problem Author: Maarten Sijm

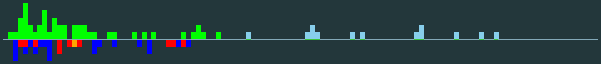


- **Problem:** Given a knitting pattern and amount of wool it costs for letting the wool strand unused, using the wool in a stitch, and for starting or ending the use of wool. Compute the minimal amount of wool required for every colour of wool.
- **Observation:** Between two times a colour of wool is used, you either leave the strand through the back unused for the entire gap, or you immediately end the use at the beginning of the gap and start using it at the end.
- **Solution:** For every colour, iterate through the knitting pattern and remember the index of the last time the colour occurred. If you encounter the colour again, the marginal cost is the minimum between leaving the strand unused the whole time since the last time, and the sum of the costs for ending and starting. Runs in $\mathcal{O}(|w| \cdot n)$.
- **Remark:** Can be done in $\mathcal{O}(n)$ by doing some bookkeeping and storing for every colour the last time it occurred.

Statistics: 58 submissions, 8 accepted, 50 unknown

L: Lots of Liquid

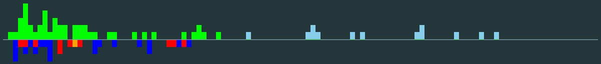
Problem Author: Maarten Sijm



- **Problem:** Find the length of the side of a cube that contains all liquid.

L: Lots of Liquid

Problem Author: Maarten Sijm

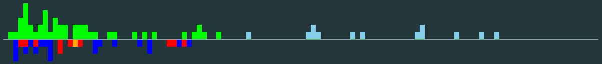


- **Problem:** Find the length of the side of a cube that contains all liquid.
- **Solution:** Calculate the value of following expression:

$$\sqrt[3]{\sum_{c} c^3}$$

L: Lots of Liquid

Problem Author: Maarten Sijm



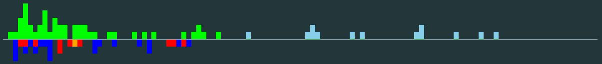
- **Problem:** Find the length of the side of a cube that contains all liquid.
- **Solution:** Calculate the value of following expression:

$$\sqrt[3]{\sum_{c} c^3}$$

- **Pitfall:** Make sure to use `double`, not `float`

L: Lots of Liquid

Problem Author: Maarten Sijm



- **Problem:** Find the length of the side of a cube that contains all liquid.
- **Solution:** Calculate the value of following expression:

$$\sqrt[3]{\sum_c c^3}$$

- **Pitfall:** Make sure to use `double`, not `float`

Statistics: 97 submissions, 46 accepted, 13 unknown