Freshmen Programming Contest 2022

Solutions presentation

May 11, 2022

A: Avant-garde

Problem Author: Jeroen Op de Beek

- **Problem:** Find the area of the union of at most 10 circles.

A: Avant-garde

Problem Author: Jeroen Op de Beek

- **Problem:** Find the area of the union of at most 10 circles.
- But the relative error allowed is 10%, so simple approximations are good enough

A: Avant-garde

Problem Author: Jeroen Op de Beek

- **Problem:** Find the area of the union of at most 10 circles.
- But the relative error allowed is 10%, so simple approximations are good enough
- **Solutions:** There are many different ways to approximate the area.

A: Avant-garde

Problem Author: Jeroen Op de Beek

- **Problem:** Find the area of the union of at most 10 circles.
- But the relative error allowed is 10%, so simple approximations are good enough
- **Solutions:** There are many different ways to approximate the area.
    - Handy formula for all points (x, y) that lie on a circle: $(x - c_x)^2 + y^2 = r^2$

A: Avant-garde

Problem Author: Jeroen Op de Beek

- **Problem:** Find the area of the union of at most 10 circles.
- But the relative error allowed is 10%, so simple approximations are good enough
- **Solutions:** There are many different ways to approximate the area.
  - Handy formula for all points (x, y) that lie on a circle: $(x - c_x)^2 + y^2 = r^2$
  - Draw a bounding box around the circles and check if randomly sampled points lie inside of at least one circle or not. Output $A_{\text{bounding box}} \cdot$ proportion of random points that hit a circle.

A: Avant-garde

Problem Author: Jeroen Op de Beek

- **Problem:** Find the area of the union of at most 10 circles.
- But the relative error allowed is 10%, so simple approximations are good enough
- **Solutions:** There are many different ways to approximate the area.
    - Handy formula for all points (x, y) that lie on a circle: $(x - c_x)^2 + y^2 = r^2$
    - Draw a bounding box around the circles and check if randomly sampled points lie inside of at least one circle or not. Output $A_{\text{bounding box}} \cdot$ proportion of random points that hit a circle.
    - Split the canvas in very small squares, and for each square, check if it overlaps with some circle.

A: Avant-garde

Problem Author: Jeroen Op de Beek

- **Problem:** Find the area of the union of at most 10 circles.
- But the relative error allowed is 10%, so simple approximations are good enough
- **Solutions:** There are many different ways to approximate the area.
  - Handy formula for all points (x, y) that lie on a circle: $(x - c_x)^2 + y^2 = r^2$
  - Draw a bounding box around the circles and check if randomly sampled points lie inside of at least one circle or not. Output $A_{\text{bounding box}} \cdot$ proportion of random points that hit a circle.
  - Split the canvas in very small squares, and for each square, check if it overlaps with some circle.
  - Define the function $f(x) =$ Highest y coordinate of any circle at this x. Calculate the integral of $f(x)$ numerically with small rectangles.

A: Avant-garde

Problem Author: Jeroen Op de Beek

- **Problem:** Find the area of the union of at most 10 circles.
- But the relative error allowed is 10%, so simple approximations are good enough
- **Solutions:** There are many different ways to approximate the area.
  - Handy formula for all points (x, y) that lie on a circle: $(x - c_x)^2 + y^2 = r^2$
  - Draw a bounding box around the circles and check if randomly sampled points lie inside of at least one circle or not. Output $A_{\text{bounding box}} \cdot$ proportion of random points that hit a circle.
  - Split the canvas in very small squares, and for each square, check if it overlaps with some circle.
  - Define the function $f(x) =$ Highest y coordinate of any circle at this x. Calculate the integral of $f(x)$ numerically with small rectangles.
- **Alternative solution:** Calculate all intersection points of all circles. Find all circular arcs that are on the outside of the resulting shape. Use formulas to calculate the total area.

A: Avant-garde

Problem Author: Jeroen Op de Beek

- **Problem:** Find the area of the union of at most 10 circles.
- But the relative error allowed is 10%, so simple approximations are good enough
- **Solutions:** There are many different ways to approximate the area.
  - Handy formula for all points (x, y) that lie on a circle: $(x - c_x)^2 + y^2 = r^2$
  - Draw a bounding box around the circles and check if randomly sampled points lie inside of at least one circle or not. Output $A_{\text{bounding box}} \cdot$ proportion of random points that hit a circle.
  - Split the canvas in very small squares, and for each square, check if it overlaps with some circle.
  - Define the function $f(x) =$ Highest y coordinate of any circle at this x. Calculate the integral of $f(x)$ numerically with small rectangles.
- **Alternative solution:** Calculate all intersection points of all circles. Find all circular arcs that are on the outside of the resulting shape. Use formulas to calculate the total area.
- **Pitfalls:**

A: Avant-garde

Problem Author: Jeroen Op de Beek

- **Problem:** Find the area of the union of at most 10 circles.
- But the relative error allowed is 10%, so simple approximations are good enough
- **Solutions:** There are many different ways to approximate the area.
  - Handy formula for all points (x, y) that lie on a circle: $(x - c_x)^2 + y^2 = r^2$
  - Draw a bounding box around the circles and check if randomly sampled points lie inside of at least one circle or not. Output $A_{\text{bounding box}} \cdot$ proportion of random points that hit a circle.
  - Split the canvas in very small squares, and for each square, check if it overlaps with some circle.
  - Define the function $f(x) =$ Highest y coordinate of any circle at this x. Calculate the integral of $f(x)$ numerically with small rectangles.
- **Alternative solution:** Calculate all intersection points of all circles. Find all circular arcs that are on the outside of the resulting shape. Use formulas to calculate the total area.
- **Pitfalls:**
  - Use too low resolution for your approximation technique, by setting the stepsize too big or not sampling enough random points.

A: Avant-garde

Problem Author: Jeroen Op de Beek

- **Problem:** Find the area of the union of at most 10 circles.
- But the relative error allowed is 10%, so simple approximations are good enough
- **Solutions:** There are many different ways to approximate the area.
  - Handy formula for all points (x, y) that lie on a circle: $(x - c_x)^2 + y^2 = r^2$
  - Draw a bounding box around the circles and check if randomly sampled points lie inside of at least one circle or not. Output $A_{\text{bounding box}} \cdot$ proportion of random points that hit a circle.
  - Split the canvas in very small squares, and for each square, check if it overlaps with some circle.
  - Define the function $f(x) =$ Highest y coordinate of any circle at this x. Calculate the integral of $f(x)$ numerically with small rectangles.
- **Alternative solution:** Calculate all intersection points of all circles. Find all circular arcs that are on the outside of the resulting shape. Use formulas to calculate the total area.
- **Pitfalls:**
  - Use too low resolution for your approximation technique, by setting the stepsize too big or not sampling enough random points.
  - Only sample points between -10 and 10 is not enough, circles also have a radius of at most 10.

A: Avant-garde

Problem Author: Jeroen Op de Beek

- **Problem:** Find the area of the union of at most 10 circles.
- But the relative error allowed is 10%, so simple approximations are good enough
- **Solutions:** There are many different ways to approximate the area.
    - Handy formula for all points (x, y) that lie on a circle: $(x - c_x)^2 + y^2 = r^2$
    - Draw a bounding box around the circles and check if randomly sampled points lie inside of at least one circle or not. Output $A_{\text{bounding box}} \cdot$ proportion of random points that hit a circle.
    - Split the canvas in very small squares, and for each square, check if it overlaps with some circle.
    - Define the function $f(x) =$ Highest y coordinate of any circle at this x. Calculate the integral of $f(x)$ numerically with small rectangles.
- **Alternative solution:** Calculate all intersection points of all circles. Find all circular arcs that are on the outside of the resulting shape. Use formulas to calculate the total area.
- **Pitfalls:**
    - Use too low resolution for your approximation technique, by setting the stepsize too big or not sampling enough random points.
    - Only sample points between -10 and 10 is not enough, circles also have a radius of at most 10.
    - Spending too much time on debugging a solution which tries to compute the area with exact formulas.

A: Avant-garde

Problem Author: Jeroen Op de Beek

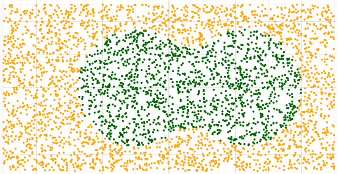


Figure 1: Monte Carlo random sampling

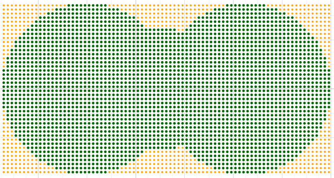


Figure 2: Pixellation based approximation

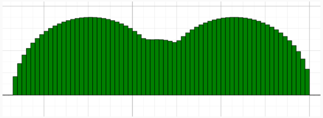


Figure 3: Approximation by numerically integrating a function
(Have to multiply the area by two at the end)

Statistics: 38 submissions, 10 accepted, 15 unknown

B: Balance by Elimination

Problem Author: Jeroen Op de Beek

- **Problem:** Make a binary tree height-balanced by removing at most one leaf.

B: Balance by Elimination

Problem Author: Jeroen Op de Beek

- **Problem:** Make a binary tree height-balanced by removing at most one leaf.
- **Naive solution:** Make a Depth First Search that calculates subtree heights and checks if a binary tree is balanced.

B: Balance by Elimination

Problem Author: Jeroen Op de Beek

- **Problem:** Make a binary tree height-balanced by removing at most one leaf.
- **Naive solution:** Make a Depth First Search that calculates subtree heights and checks if a binary tree is balanced.
    - Remove all leaves one by one and check if the tree becomes balanced with a DFS. $\mathcal{O}(n^2)$

B: Balance by Elimination

Problem Author: Jeroen Op de Beek

- **Problem:** Make a binary tree height-balanced by removing at most one leaf.
- **Naive solution:** Make a Depth First Search that calculates subtree heights and checks if a binary tree is balanced.
    - Remove all leaves one by one and check if the tree becomes balanced with a DFS. $\mathcal{O}(n^2)$
- **Solution:** Notice that there's only one candidate leaf that could possibly balance the tree.

B: Balance by Elimination

Problem Author: Jeroen Op de Beek

- **Problem:** Make a binary tree height-balanced by removing at most one leaf.
- **Naive solution:** Make a Depth First Search that calculates subtree heights and checks if a binary tree is balanced.
    - Remove all leaves one by one and check if the tree becomes balanced with a DFS. $\mathcal{O}(n^2)$
- **Solution:** Notice that there's only one candidate leaf that could possibly balance the tree.
- This is the deepest leaf in the subtree of the deepest unbalanced node.

B: Balance by Elimination

Problem Author: Jeroen Op de Beek

- **Problem:** Make a binary tree height-balanced by removing at most one leaf.
- **Naive solution:** Make a Depth First Search that calculates subtree heights and checks if a binary tree is balanced.
  - Remove all leaves one by one and check if the tree becomes balanced with a DFS. $\mathcal{O}(n^2)$
- **Solution:** Notice that there's only one candidate leaf that could possibly balance the tree.
- This is the deepest leaf in the subtree of the deepest unbalanced node.
- Now we only need to do two DFS's: A DFS for finding the candidate leaf, and a DFS for checking if the tree became balanced. $\mathcal{O}(n)$

B: Balance by Elimination

Problem Author: Jeroen Op de Beek

- **Problem:** Make a binary tree height-balanced by removing at most one leaf.
- **Naive solution:** Make a Depth First Search that calculates subtree heights and checks if a binary tree is balanced.
    - Remove all leaves one by one and check if the tree becomes balanced with a DFS. $\mathcal{O}(n^2)$
- **Solution:** Notice that there's only one candidate leaf that could possibly balance the tree.
- This is the deepest leaf in the subtree of the deepest unbalanced node.
- Now we only need to do two DFS's: A DFS for finding the candidate leaf, and a DFS for checking if the tree became balanced. $\mathcal{O}(n)$
- **Pitfall:** Checking the globally deepest leaf, instead of the deepest leaf in the correct subtree.

B: Balance by Elimination

Problem Author: Jeroen Op de Beek

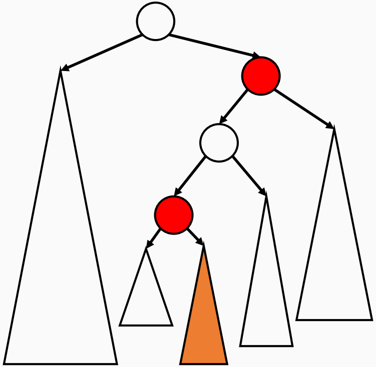


Figure 4: Proof by picture: The only candidate leaf is the leaf underneath the deepest unbalanced node.

Statistics: 21 submissions, 6 accepted, 8 unknown

C: Cake Promise

Problem Author: Maarten Sijm

- **Problem:** Calculate how many teams beat prof. Wright in the programming contest.

C: Cake Promise

Problem Author: Maarten Sijm

- **Problem:** Calculate how many teams beat prof. Wright in the programming contest.
- **Solution:** Count how many teams solved *more* problems than prof. Wright.

C: Cake Promise

Problem Author: Maarten Sijm

- **Problem:** Calculate how many teams beat prof. Wright in the programming contest.
- **Solution:** Count how many teams solved *more* problems than prof. Wright.
- If a team solved the same number of problems as prof. Wright, check whether the total amount of time needed is *less than or equal to* the time required by the professor.

C: Cake Promise

Problem Author: Maarten Sijm

- **Problem:** Calculate how many teams beat prof. Wright in the programming contest.
- **Solution:** Count how many teams solved *more* problems than prof. Wright.
- If a team solved the same number of problems as prof. Wright, check whether the total amount of time needed is *less than or equal to* the time required by the professor.

Statistics: 64 submissions, 43 accepted, 5 unknown

D: Dale ‘n’ Chip

Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea

- **Problem:** Given a list of moves (`Rock`, `Paper`, `Scissors`), calculate for a set of intervals, each move’s frequency within those intervals.

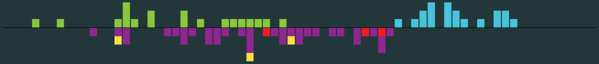
D: Dale ‘n’ Chip

Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea

- **Problem:** Given a list of moves (`Rock`, `Paper`, `Scissors`), calculate for a set of intervals, each move’s frequency within those intervals.
- **Key observation:** Players should be placed in a certain pattern: `R P S R P S ....`
  Answer for query range $[i, j]$ is calculated as: $3 \cdot \min(\mathit{freq}(\texttt{R})[i, j], \mathit{freq}(\texttt{P})[i, j], \mathit{freq}(\texttt{S})[i, j])$

D: Dale ‘n’ Chip

Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea



- **Problem:** Given a list of moves (`Rock`, `Paper`, `Scissors`), calculate for a set of intervals, each move’s frequency within those intervals.
- **Key observation:** Players should be placed in a certain pattern: `R P S R P S ...`.
  Answer for query range $[i, j]$ is calculated as: $3 \cdot \min(\mathit{freq}(\mathtt{R})[i, j], \mathit{freq}(\mathtt{P})[i, j], \mathit{freq}(\mathtt{S})[i, j])$
- **Naive solution:** For all queries, iterate over the interval and count each move’s frequency.

D: Dale ‘n’ Chip

Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea

- **Problem:** Given a list of moves (`Rock`, `Paper`, `Scissors`), calculate for a set of intervals, each move’s frequency within those intervals.
- **Key observation:** Players should be placed in a certain pattern: `R P S R P S ....`
  Answer for query range $[i, j]$ is calculated as: $3 \cdot \min(\mathit{freq}(\mathtt{R})[i, j], \mathit{freq}(\mathtt{P})[i, j], \mathit{freq}(\mathtt{S})[i, j])$
- **Naive solution:** For all queries, iterate over the interval and count each move’s frequency.
  - Too slow. $\mathcal{O}(n * q)$

D: Dale ‘n’ Chip

Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea

- **Problem:** Given a list of moves (`Rock`, `Paper`, `Scissors`), calculate for a set of intervals, each move’s frequency within those intervals.
- **Key observation:** Players should be placed in a certain pattern: `R P S R P S ....`
  Answer for query range $[i, j]$ is calculated as: $3 \cdot \min(\mathit{freq}(\mathtt{R})[i,j], \mathit{freq}(\mathtt{P})[i,j], \mathit{freq}(\mathtt{S})[i,j])$
- **Naive solution:** For all queries, iterate over the interval and count each move’s frequency.
  - Too slow. $\mathcal{O}(n * q)$
- **Solution:** Notice that we can pre-compute each move’s frequency in $\mathcal{O}(n)$ time for all the intervals $[1, i]$ $(1 \leq i \leq n)$.

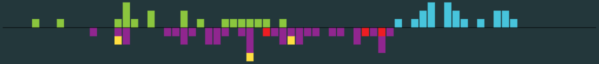
D: Dale ‘n’ Chip

Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea

- **Problem:** Given a list of moves (`Rock`, `Paper`, `Scissors`), calculate for a set of intervals, each move’s frequency within those intervals.
- **Key observation:** Players should be placed in a certain pattern: `R P S R P S ...`.
  Answer for query range $[i, j]$ is calculated as: $3 \cdot \min(\mathit{freq}(\mathtt{R})[i, j], \mathit{freq}(\mathtt{P})[i, j], \mathit{freq}(\mathtt{S})[i, j])$
- **Naive solution:** For all queries, iterate over the interval and count each move’s frequency.
  - Too slow. $\mathcal{O}(n * q)$
- **Solution:** Notice that we can pre-compute each move’s frequency in $\mathcal{O}(n)$ time for all the intervals $[1, i]$ $(1 \leq i \leq n)$.
  - Computation: $\mathit{freq}(\mathtt{R})[1, i] = \mathit{freq}(\mathtt{R})[1, i - 1] + (\ 1$ if ith move is “`Rock`”, 0 otherwise).
    Same for `Paper` and `Scissors`.

D: Dale ‘n’ Chip

Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea



- **Problem:** Given a list of moves (`Rock`, `Paper`, `Scissors`), calculate for a set of intervals, each move’s frequency within those intervals.
- **Key observation:** Players should be placed in a certain pattern: `R P S R P S ....` Answer for query range $[i, j]$ is calculated as: $3 \cdot \min(\mathit{freq}(\mathtt{R})[i, j], \mathit{freq}(\mathtt{P})[i, j], \mathit{freq}(\mathtt{S})[i, j])$
- **Naive solution:** For all queries, iterate over the interval and count each move’s frequency.
  - Too slow. $\mathcal{O}(n * q)$
- **Solution:** Notice that we can pre-compute each move’s frequency in $\mathcal{O}(n)$ time for all the intervals $[1, i]$ $(1 \leq i \leq n)$.
  - Computation: $\mathit{freq}(\mathtt{R})[1, i] = \mathit{freq}(\mathtt{R})[1, i-1] +$ (1 if ith move is “`Rock`”, 0 otherwise). Same for `Paper` and `Scissors`.
- For a given interval $\mathit{freq}(\mathtt{R})[i, j] = \mathit{freq}(\mathtt{R})[1, j] - \mathit{freq}(\mathtt{R})[1, i-1]$. Lookups take $\mathcal{O}(1)$ time.

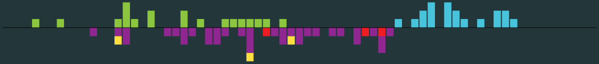
D: Dale ‘n’ Chip

Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea

- **Problem:** Given a list of moves (`Rock`, `Paper`, `Scissors`), calculate for a set of intervals, each move’s frequency within those intervals.
- **Key observation:** Players should be placed in a certain pattern: `R P S R P S` . . . .
  Answer for query range $[i, j]$ is calculated as: $3 \cdot \min(\mathit{freq}(\mathtt{R})[i, j], \mathit{freq}(\mathtt{P})[i, j], \mathit{freq}(\mathtt{S})[i, j])$
- **Naive solution:** For all queries, iterate over the interval and count each move’s frequency.
  - Too slow. $\mathcal{O}(n * q)$
- **Solution:** Notice that we can pre-compute each move’s frequency in $\mathcal{O}(n)$ time for all the intervals $[1, i]$ $(1 \leq i \leq n)$.
  - Computation: $\mathit{freq}(\mathtt{R})[1, i] = \mathit{freq}(\mathtt{R})[1, i-1] + ($ 1 if ith move is “`Rock`”, 0 otherwise $)$.
    Same for `Paper` and `Scissors`.
- For a given interval $\mathit{freq}(\mathtt{R})[i, j] = \mathit{freq}(\mathtt{R})[1, j] - \mathit{freq}(\mathtt{R})[1, i-1]$. Lookups take $\mathcal{O}(1)$ time.
- Time complexity: $\mathcal{O}(n + q)$

D: Dale ‘n’ Chip

Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea



- **Problem:** Given a list of moves (`Rock`, `Paper`, `Scissors`), calculate for a set of intervals, each move’s frequency within those intervals.
- **Key observation:** Players should be placed in a certain pattern: `R P S R P S ....`
  Answer for query range $[i, j]$ is calculated as: $3 \cdot \min(\mathit{freq}(\texttt{R})[i,j], \mathit{freq}(\texttt{P})[i,j], \mathit{freq}(\texttt{S})[i,j])$
- **Naive solution:** For all queries, iterate over the interval and count each move’s frequency.
  - Too slow. $\mathcal{O}(n * q)$
- **Solution:** Notice that we can pre-compute each move’s frequency in $\mathcal{O}(n)$ time for all the intervals $[1, i]$ $(1 \leq i \leq n)$.
  - Computation: $\mathit{freq}(\texttt{R})[1,i] = \mathit{freq}(\texttt{R})[1,i-1] +$ (1 if ith move is “`Rock`”, 0 otherwise). Same for `Paper` and `Scissors`.
- For a given interval $\mathit{freq}(\texttt{R})[i,j] = \mathit{freq}(\texttt{R})[1,j] - \mathit{freq}(\texttt{R})[1,i-1]$. Lookups take $\mathcal{O}(1)$ time.
- Time complexity: $\mathcal{O}(n + q)$

Statistics: 78 submissions, 19 accepted, 19 unknown

E: Eurovision

Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea

- **Problem:** Given a series of numbers and k cuts allowed, choose where to cut the list such that the sum of the largest interval (S) is the smallest.

E: Eurovision

Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea

- **Problem:** Given a series of numbers and k cuts allowed, choose where to cut the list such that the sum of the largest interval (S) is the smallest.
- **First step:** Transform the initial input into a list of numbers which represent groups of song fragments that can *not* be divided. Each song fragment is a part between two local minima.

E: Eurovision

Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea

- **Problem:** Given a series of numbers and k cuts allowed, choose where to cut the list such that the sum of the largest interval (S) is the smallest.
- **First step:** Transform the initial input into a list of numbers which represent groups of song fragments that can *not* be divided. Each song fragment is a part between two local minima.
- **Second step:** Find where to cut the list of song fragments.

E: Eurovision

Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea

- **Problem:** Given a series of numbers and k cuts allowed, choose where to cut the list such that the sum of the largest interval (S) is the smallest.
- **First step:** Transform the initial input into a list of numbers which represent groups of song fragments that can *not* be divided. Each song fragment is a part between two local minima.
- **Second step:** Find where to cut the list of song fragments.
- **Note:** For a given S, you can calculate whether it is possible
to perform the song using at most k breaths in $\mathcal{O}(n)$ time.

E: Eurovision

Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea

- **Problem:** Given a series of numbers and k cuts allowed, choose where to cut the list such that the sum of the largest interval (S) is the smallest.
- **First step:** Transform the initial input into a list of numbers which represent groups of song fragments that can *not* be divided. Each song fragment is a part between two local minima.
- **Second step:** Find where to cut the list of song fragments.
- **Note:** For a given S, you can calculate whether it is possible to perform the song using at most k breaths in $\mathcal{O}(n)$ time.
- Therefore, it is possible to find S using binary search:
  - If it is possible to perform a song for a given S, search lower; else, search higher.

E: Eurovision

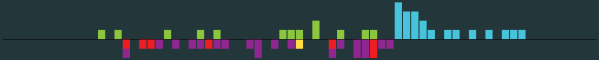
Problem Author: Jeroen Op de Beek and Dragos-Paul Vecerdea

- **Problem:** Given a series of numbers and k cuts allowed, choose where to cut the list such that the sum of the largest interval (S) is the smallest.
- **First step:** Transform the initial input into a list of numbers which represent groups of song fragments that can *not* be divided. Each song fragment is a part between two local minima.
- **Second step:** Find where to cut the list of song fragments.
- **Note:** For a given S, you can calculate whether it is possible to perform the song using at most k breaths in $\mathcal{O}(n)$ time.
- Therefore, it is possible to find S using binary search:
  - If it is possible to perform a song for a given S, search lower; else, search higher.

Statistics: 10 submissions, 2 accepted, 4 unknown

F: Fastest Thing Alive

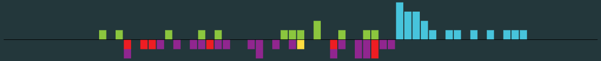
Problem Author: Angel Karchev



- **Problem:** Find a path with no spikes, while moving up to one lane to the side for each row.

F: Fastest Thing Alive

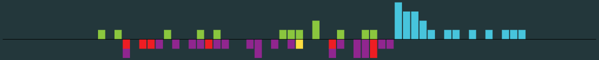
Problem Author: Angel Karchev



- **Problem:** Find a path with no spikes, while moving up to one lane to the side for each row.
- **Solution:** Traverse every possible path until a spike is reached by using DFS.

F: Fastest Thing Alive

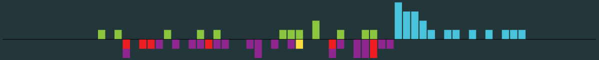
Problem Author: Angel Karchev



- **Problem:** Find a path with no spikes, while moving up to one lane to the side for each row.
- **Solution:** Traverse every possible path until a spike is reached by using DFS.
- For each visited field, remember the direction from which it was accessed in order to recover a correct path.

F: Fastest Thing Alive

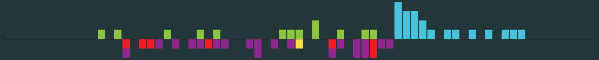
Problem Author: Angel Karchev



- **Problem:** Find a path with no spikes, while moving up to one lane to the side for each row.
- **Solution:** Traverse every possible path until a spike is reached by using DFS.
- For each visited field, remember the direction from which it was accessed in order to recover a correct path.
- **Pitfalls:** If you don’t keep track of already visited fields, the solution will take a long time.

F: Fastest Thing Alive

Problem Author: Angel Karchev

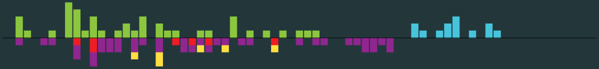


- **Problem:** Find a path with no spikes, while moving up to one lane to the side for each row.
- **Solution:** Traverse every possible path until a spike is reached by using DFS.
- For each visited field, remember the direction from which it was accessed in order to recover a correct path.
- **Pitfalls:** If you don't keep track of already visited fields, the solution will take a long time.

Statistics: 61 submissions, 13 accepted, 20 unknown

G: Glass Reflection

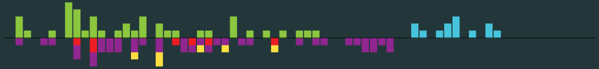
Problem Author: Maarten Sijm and Robert van Dijk



- **Problem:** Remove duplicated letters from a reflected word.

G: Glass Reflection

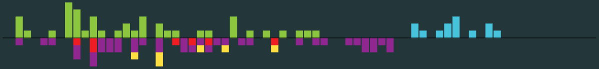
Problem Author: Maarten Sijm and Robert van Dijk



- **Problem:** Remove duplicated letters from a reflected word.
- **Solution:** For every letter in the word (starting from the second letter):
    - If the letter is equal to the previous letter, add it to the result.
    - Else, discard the letter.

G: Glass Reflection

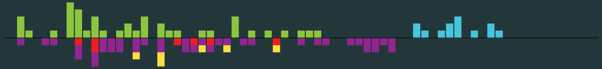
Problem Author: Maarten Sijm and Robert van Dijk



- **Problem:** Remove duplicated letters from a reflected word.
- **Solution:** For every letter in the word (starting from the second letter):
  - If the letter is equal to the previous letter, add it to the result.
  - Else, discard the letter.
- **Pitfalls:**
  - Do not use += to concatenate strings
  - When using Java, do not print letter-by-letter, because I/O is slow

G: Glass Reflection

Problem Author: Maarten Sijm and Robert van Dijk



- **Problem:** Remove duplicated letters from a reflected word.
- **Solution:** For every letter in the word (starting from the second letter):
    - If the letter is equal to the previous letter, add it to the result.
    - Else, discard the letter.
- **Pitfalls:**
    - Do not use += to concatenate strings
    - When using Java, do not print letter-by-letter, because I/O is slow

Statistics: 106 submissions, 41 accepted, 13 unknown

H: Highways of the Future

Problem Author: Cristian-Alexandru Botocan

Problem:

- Given a directed graph, how many directed edges should you add to get one big Strongly Connected Component?

H: Highways of the Future

Problem Author: Cristian-Alexandru Botocan

Problem:

- Given a directed graph, how many directed edges should you add to get one big Strongly Connected Component?

Solution:

- Firstly we have to compute the number the number of Strongly Connected Components (SCC) of the directed graph.
    - **Def:** A Strongly Connected Component is the portion of a directed graph in which there is a path from each vertex to another vertex.
- To determine the number of the SCCs, we can use the Kosaraju's algorithm or Tarjan's algorithm.
- If the graph consists of one single SCC, we will just output 0 and finish the program.

H: Highways of the Future

Problem Author: Cristian-Alexandru Botocan

Problem:

- Given a directed graph, how many directed edges should you add to get one big Strongly Connected Component?

Solution:

- If the graph does not consist of one single SCC, then we still have to do some operations.
- **Def:** A SCC-root has no incoming edges from a different SCC.
- **Def:** A SCC-leaf has no outgoing edges to a different SCC.
- **Note:** We can have the case where a single SCC is both SCC-root and SCC-leaf.

H: Highways of the Future

Problem Author: Cristian-Alexandru Botocan

Problem:

- Given a directed graph, how many directed edges should you add to get one big Strongly Connected Component?

Solution:

- The total number of edges which have to be added is represented by:

$$\max(\text{number of SCC-roots}, \text{number of SCC-leaves})$$

- Thus, after we computed the SCCs, we can just count the number of SCC-roots and SCC-leaves and print the maximum between those.

H: Highways of the Future

Problem Author: Cristian-Alexandru Botocan

Problem:

- Given a directed graph, how many directed edges should you add to get one big Strongly Connected Component?

Pitfalls:

- Compute the number of connected components using simple BFS/DFS instead considering Strong Connected components using Kosaraju's/Tarjan's algorithm.
- Computing the final answer as number of SCCs $-$ 1, instead of computing the maximum between the total number of SCC-roots and SCC-leaves.

Statistics: 5 submissions, 0 accepted, 5 unknown

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- **Problem:** Find the lexicographically minimal, valid assignment of m lecture halls with capacities c_j to n lectures. x_i students will come to lecture i.

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- **Problem:** Find the lexicographically minimal, valid assignment of m lecture halls with capacities c_j to n lectures. x_i students will come to lecture i.
- **First attempt**: Build the assignment from left to right, trying to optimize the niceness of the lecture hall of professor 1, then professor 2, $\cdots$.

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- **Problem:** Find the lexicographically minimal, valid assignment of m lecture halls with capacities c_j to n lectures. x_i students will come to lecture i.
- **First attempt**: Build the assignment from left to right, trying to optimize the niceness of the lecture hall of professor 1, then professor 2, $\cdots$.
    - Try all lecture halls from nicest to least nice, and check if $x_1 \leq c_j$.

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- **Problem:** Find the lexicographically minimal, valid assignment of m lecture halls with capacities c_j to n lectures. x_i students will come to lecture i.
- **First attempt**: Build the assignment from left to right, trying to optimize the niceness of the lecture hall of professor 1, then professor 2, $\cdots$.
  - Try all lecture halls from nicest to least nice, and check if $x_1 \leq c_j$.
  - After fixing the hall for professor 1, $m-1$ lecture halls and $n-1$ courses are left.

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- **Problem:** Find the lexicographically minimal, valid assignment of m lecture halls with capacities c_j to n lectures. x_i students will come to lecture i.
- **First attempt**: Build the assignment from left to right, trying to optimize the niceness of the lecture hall of professor 1, then professor 2, $\cdots$.
  - Try all lecture halls from nicest to least nice, and check if $x_1 \leq c_j$.
  - After fixing the hall for professor 1, $m - 1$ lecture halls and $n - 1$ courses are left.
  - To find out if there exists any valid assignment of these, sort the remaining lecture halls and courses decreasingly.

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- **Problem:** Find the lexicographically minimal, valid assignment of m lecture halls with capacities c_j to n lectures. x_i students will come to lecture i.
- **First attempt**: Build the assignment from left to right, trying to optimize the niceness of the lecture hall of professor 1, then professor 2, $\cdots$.
  - Try all lecture halls from nicest to least nice, and check if $x_1 \leq c_j$.
  - After fixing the hall for professor 1, $m-1$ lecture halls and $n-1$ courses are left.
  - To find out if there exists any valid assignment of these, sort the remaining lecture halls and courses decreasingly.
  - This gives two new sequences $x_1' \geq x_2' \geq \cdots \geq x_{n-1}'$ and $c_1' \geq c_2' \geq \cdots \geq c_{m-1}'$

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- **Problem:** Find the lexicographically minimal, valid assignment of m lecture halls with capacities c_j to n lectures. x_i students will come to lecture i.
- **First attempt**: Build the assignment from left to right, trying to optimize the niceness of the lecture hall of professor 1, then professor 2, $\cdots$.
    - Try all lecture halls from nicest to least nice, and check if $x_1 \leq c_j$.
    - After fixing the hall for professor 1, $m-1$ lecture halls and $n-1$ courses are left.
    - To find out if there exists any valid assignment of these, sort the remaining lecture halls and courses decreasingly.
    - This gives two new sequences $x'_1 \geq x'_2 \geq \cdots \geq x'_{n-1}$ and $c'_1 \geq c'_2 \geq \cdots \geq c'_{m-1}$
    - Check if the ith course in the order can be matched with the ith hall. $x'_i \leq c'_i$

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- **Problem:** Find the lexicographically minimal, valid assignment of m lecture halls with capacities c_j to n lectures. x_i students will come to lecture i.
- **First attempt**: Build the assignment from left to right, trying to optimize the niceness of the lecture hall of professor 1, then professor 2, $\cdots$.
  - Try all lecture halls from nicest to least nice, and check if $x_1 \leq c_j$.
  - After fixing the hall for professor 1, $m-1$ lecture halls and $n-1$ courses are left.
  - To find out if there exists any valid assignment of these, sort the remaining lecture halls and courses decreasingly.
  - This gives two new sequences $x'_1 \geq x'_2 \geq \cdots \geq x'_{n-1}$ and $c'_1 \geq c'_2 \geq \cdots \geq c'_{m-1}$
  - Check if the ith course in the order can be matched with the ith hall. $x'_i \leq c'_i$
  - It can be proven that if this greedy approach fails, a valid assignment does not exist.

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- **Problem:** Find the lexicographically minimal, valid assignment of m lecture halls with capacities c_j to n lectures. x_i students will come to lecture i.
- **First attempt**: Build the assignment from left to right, trying to optimize the niceness of the lecture hall of professor 1, then professor 2, $\cdots$.
    - Try all lecture halls from nicest to least nice, and check if $x_1 \leq c_j$.
    - After fixing the hall for professor 1, $m-1$ lecture halls and $n-1$ courses are left.
    - To find out if there exists any valid assignment of these, sort the remaining lecture halls and courses decreasingly.
    - This gives two new sequences $x'_1 \geq x'_2 \geq \cdots \geq x'_{n-1}$ and $c'_1 \geq c'_2 \geq \cdots \geq c'_{m-1}$
    - Check if the ith course in the order can be matched with the ith hall. $x'_i \leq c'_i$
    - It can be proven that if this greedy approach fails, a valid assignment does not exist.
    - After a match for professor 1 is found, repeat this procedure for professor $2, 3, \cdots, n$

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- **Problem:** Find the lexicographically minimal, valid assignment of m lecture halls with capacities c_j to n lectures. x_i students will come to lecture i.
- **First attempt**: Build the assignment from left to right, trying to optimize the niceness of the lecture hall of professor 1, then professor 2, $\cdots$.
  - Try all lecture halls from nicest to least nice, and check if $x_1 \leq c_j$.
  - After fixing the hall for professor 1, $m-1$ lecture halls and $n-1$ courses are left.
  - To find out if there exists any valid assignment of these, sort the remaining lecture halls and courses decreasingly.
  - This gives two new sequences $x'_1 \geq x'_2 \geq \cdots \geq x'_{n-1}$ and $c'_1 \geq c'_2 \geq \cdots \geq c'_{m-1}$
  - Check if the ith course in the order can be matched with the ith hall. $x'_i \leq c'_i$
  - It can be proven that if this greedy approach fails, a valid assignment does not exist.
  - After a match for professor 1 is found, repeat this procedure for professor $2, 3, \cdots, n$
  - Runtime: $\mathcal{O}(n \cdot m \cdot (n \log n + m \log m))$, too slow!

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- First speedup: The log factors are easy to remove: Only sort the halls and courses once at the beginning, and remove items from the lists when necessary.

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- First speedup: The log factors are easy to remove: Only sort the halls and courses once at the beginning, and remove items from the lists when necessary.
- Second speedup:

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- First speedup: The log factors are easy to remove: Only sort the halls and courses once at the beginning, and remove items from the lists when necessary.
- Second speedup:
  - For each phase of the algorithm, when searching for the best lecture hall for the next professor, The greedy assignment checker checks very similar arrays x' and c' each time.

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- First speedup: The log factors are easy to remove: Only sort the halls and courses once at the beginning, and remove items from the lists when necessary.
- Second speedup:
  - For each phase of the algorithm, when searching for the best lecture hall for the next professor, The greedy assignment checker checks very similar arrays x' and c' each time.
  - In fact, per phase, x' stays the same, and c' changes only by one element.

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- First speedup: The log factors are easy to remove: Only sort the halls and courses once at the beginning, and remove items from the lists when necessary.
- Second speedup:
  - For each phase of the algorithm, when searching for the best lecture hall for the next professor, The greedy assignment checker checks very similar arrays x' and c' each time.
  - In fact, per phase, x' stays the same, and c' changes only by one element.
  - With clever precomputation per phase of $\mathcal{O}(n+m)$, it's possible to check whether the greedy will fail in $\mathcal{O}(1)$!

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- First speedup: The log factors are easy to remove: Only sort the halls and courses once at the beginning, and remove items from the lists when necessary.
- Second speedup:
  - For each phase of the algorithm, when searching for the best lecture hall for the next professor, The greedy assignment checker checks very similar arrays x' and c' each time.
  - In fact, per phase, x' stays the same, and c' changes only by one element.
  - With clever precomputation per phase of $\mathcal{O}(n+m)$, it's possible to check whether the greedy will fail in $\mathcal{O}(1)$!
- This gives a solution with $\mathcal{O}(n \cdot (n+m))$ complexity.

I: Inspiring Professors

Problem Author: Jeroen Op de Beek

- First speedup: The log factors are easy to remove: Only sort the halls and courses once at the beginning, and remove items from the lists when necessary.
- Second speedup:
  - For each phase of the algorithm, when searching for the best lecture hall for the next professor, The greedy assignment checker checks very similar arrays x' and c' each time.
  - In fact, per phase, x' stays the same, and c' changes only by one element.
  - With clever precomputation per phase of $\mathcal{O}(n+m)$, it's possible to check whether the greedy will fail in $\mathcal{O}(1)$!
- This gives a solution with $\mathcal{O}(n \cdot (n+m))$ complexity.

Statistics: 1 submissions, 0 accepted, 1 unknown

J: Journey to Mastery

Problem Author: Angel Karchev

- **Problem:** Use a combination of moves to hit the dummy before they hit you.

J: Journey to Mastery

Problem Author: Angel Karchev

- **Problem:** Use a combination of moves to hit the dummy before they hit you.
- **Solution:**

J: Journey to Mastery

Problem Author: Angel Karchev

- **Problem:** Use a combination of moves to hit the dummy before they hit you.
- **Solution:**
  - Always counter the dummy's Hadouken with your own, so you don't need to keep track of fireball positions.

J: Journey to Mastery

Problem Author: Angel Karchev

- **Problem:** Use a combination of moves to hit the dummy before they hit you.
- **Solution:**
  - Always counter the dummy's Hadouken with your own, so you don't need to keep track of fireball positions.
  - At distance 1, Shoryuken beats everything, so use that.

J: Journey to Mastery

Problem Author: Angel Karchev

- **Problem:** Use a combination of moves to hit the dummy before they hit you.
- **Solution:**
  - Always counter the dummy's Hadouken with your own, so you don't need to keep track of fireball positions.
  - At distance 1, Shoryuken beats everything, so use that.
  - At distance 2, the dummy using grounded kick beats everything, so never let the dummy reach distance 2.

J: Journey to Mastery

Problem Author: Angel Karchev

- **Problem:** Use a combination of moves to hit the dummy before they hit you.
- **Solution:**
  - Always counter the dummy's Hadouken with your own, so you don't need to keep track of fireball positions.
  - At distance 1, Shoryuken beats everything, so use that.
  - At distance 2, the dummy using grounded kick beats everything, so never let the dummy reach distance 2.
  - At distance 3, kick if the dummy walks forward, and Shoryuken if the dummy jumps.

J: Journey to Mastery

Problem Author: Angel Karchev

- **Problem:** Use a combination of moves to hit the dummy before they hit you.
- **Solution:**
  - Always counter the dummy's Hadouken with your own, so you don't need to keep track of fireball positions.
  - At distance 1, Shoryuken beats everything, so use that.
  - At distance 2, the dummy using grounded kick beats everything, so never let the dummy reach distance 2.
  - At distance 3, kick if the dummy walks forward, and Shoryuken if the dummy jumps.
  - At distance 4, wait if the dummy walks forward, kick if they jump.

J: Journey to Mastery

Problem Author: Angel Karchev

- **Problem:** Use a combination of moves to hit the dummy before they hit you.
- **Solution:**
  - Always counter the dummy's Hadouken with your own, so you don't need to keep track of fireball positions.
  - At distance 1, Shoryuken beats everything, so use that.
  - At distance 2, the dummy using grounded kick beats everything, so never let the dummy reach distance 2.
  - At distance 3, kick if the dummy walks forward, and Shoryuken if the dummy jumps.
  - At distance 4, wait if the dummy walks forward, kick if they jump.
  - At distance 5, wait if the dummy uses a jump or walks.

J: Journey to Mastery

Problem Author: Angel Karchev

- **Problem:** Use a combination of moves to hit the dummy before they hit you.
- **Solution:**
  - Always counter the dummy's Hadouken with your own, so you don't need to keep track of fireball positions.
  - At distance 1, Shoryuken beats everything, so use that.
  - At distance 2, the dummy using grounded kick beats everything, so never let the dummy reach distance 2.
  - At distance 3, kick if the dummy walks forward, and Shoryuken if the dummy jumps.
  - At distance 4, wait if the dummy walks forward, kick if they jump.
  - At distance 5, wait if the dummy uses a jump or walks.
  - At any longer distance, you can pick whichever action you like, unless the dummy uses Hadouken.

J: Journey to Mastery

Problem Author: Angel Karchev

- **Problem:** Use a combination of moves to hit the dummy before they hit you.
- **Solution:**
  - Always counter the dummy's Hadouken with your own, so you don't need to keep track of fireball positions.
  - At distance 1, Shoryuken beats everything, so use that.
  - At distance 2, the dummy using grounded kick beats everything, so never let the dummy reach distance 2.
  - At distance 3, kick if the dummy walks forward, and Shoryuken if the dummy jumps.
  - At distance 4, wait if the dummy walks forward, kick if they jump.
  - At distance 5, wait if the dummy uses a jump or walks.
  - At any longer distance, you can pick whichever action you like, unless the dummy uses Hadouken.
- Angel's solution only uses Hadouken at long range and is 140 lines because of it, while Maarten's tries to match player cooldowns to the dummy and is 10 lines.

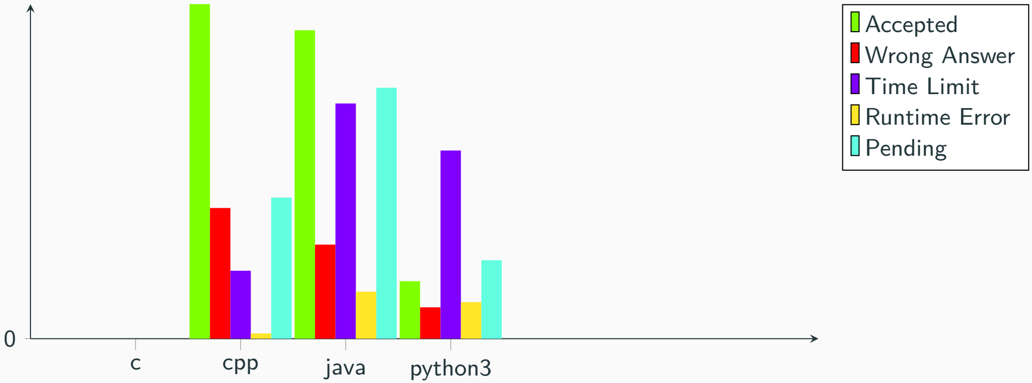
J: Journey to Mastery

Problem Author: Angel Karchev

- **Problem:** Use a combination of moves to hit the dummy before they hit you.
- **Solution:**
  - Always counter the dummy's Hadouken with your own, so you don't need to keep track of fireball positions.
  - At distance 1, Shoryuken beats everything, so use that.
  - At distance 2, the dummy using grounded kick beats everything, so never let the dummy reach distance 2.
  - At distance 3, kick if the dummy walks forward, and Shoryuken if the dummy jumps.
  - At distance 4, wait if the dummy walks forward, kick if they jump.
  - At distance 5, wait if the dummy uses a jump or walks.
  - At any longer distance, you can pick whichever action you like, unless the dummy uses Hadouken.
- Angel's solution only uses Hadouken at long range and is 140 lines because of it, while Maarten's tries to match player cooldowns to the dummy and is 10 lines.

Statistics: 0 submissions, 0 accepted, 0 unknown

Language stats



Other stats

Jury work

- 371 commits (last year: 323)
- 252 secret test cases (last year: 219)
- 59 accepted jury solutions (last year: 44)
- The minimum[1] number of lines the jury needed to solve all problems is

  $$4 + 12 + 3 + 3 + 8 + 16 + 1 + 37 + 16 + 4 = 104$$

  On average 10.4 lines per problem, down from 13.9 from last year

---

[1] *After* codegolfing

Thanks to:

The Proofreaders

- Aleksandar Lazarov
- Arnoud van der Leer
- Davina van Meer
- Robert van Dijk
- Thomas Verwoerd

The Jury

- Angel Karchev
- Cristian-Alexandru Botocan
- Dragos-Paul Vecerdea
- Jeroen Op de Beek
- Maarten Sijm