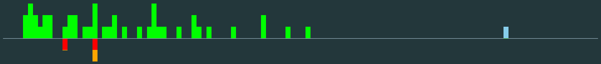
Freshmen Programming Contest 2023

Solutions presentation

May 27, 2023

E: Expected Eyes

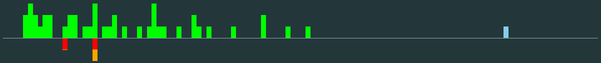
Problem Author: Maarten Sijm and Leon van der Waal



- **Problem:** Given list of dice and the number of faces of each one of them, calculate the expected value of throwing all of them at once.

E: Expected Eyes

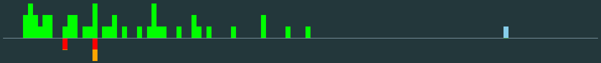
Problem Author: Maarten Sijm and Leon van der Waal



- **Problem:** Given list of dice and the number of faces of each one of them, calculate the expected value of throwing all of them at once.
- **Naive solution:** Calculate all possible throws, sum them and then divide them by the number of possible outcomes.

E: Expected Eyes

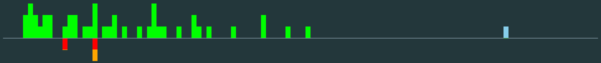
Problem Author: Maarten Sijm and Leon van der Waal



- **Problem:** Given list of dice and the number of faces of each one of them, calculate the expected value of throwing all of them at once.
- **Naive solution:** Calculate all possible throws, sum them and then divide them by the number of possible outcomes.
- **Complexity:** $\mathcal{O}({x_{\text{max}}}^n)$. Gets accepted!

E: Expected Eyes

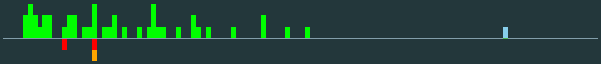
Problem Author: Maarten Sijm and Leon van der Waal



- **Problem:** Given list of dice and the number of faces of each one of them, calculate the expected value of throwing all of them at once.
- **Naive solution:** Calculate all possible throws, sum them and then divide them by the number of possible outcomes.
- **Complexity:** $\mathcal{O}({x_{\text{max}}}^n)$. Gets accepted!
- **Observation:**
  - $E[X + Y] = E[X] + E[Y]$, for any independent X and Y
  - $E[\text{dice with k faces}] = \frac{1}{k}\sum_{x=1}^{k} x = \frac{k+1}{2}$

E: Expected Eyes

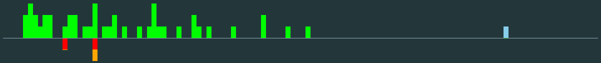
Problem Author: Maarten Sijm and Leon van der Waal



- **Problem:** Given list of dice and the number of faces of each one of them, calculate the expected value of throwing all of them at once.
- **Naive solution:** Calculate all possible throws, sum them and then divide them by the number of possible outcomes.
- **Complexity:** $\mathcal{O}({x_{\text{max}}}^n)$. Gets accepted!
- **Observation:**
  - $E[X+Y] = E[X] + E[Y]$, for any independent X and Y
  - $E[\text{dice with k faces}] = \frac{1}{k}\sum_{x=1}^{k} x = \frac{k+1}{2}$
- **Fast solution:** Calculate the expected value of every dice, then sum them up.

E: Expected Eyes

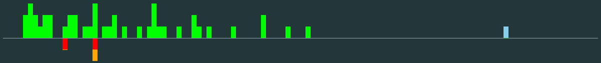
Problem Author: Maarten Sijm and Leon van der Waal



- **Problem:** Given list of dice and the number of faces of each one of them, calculate the expected value of throwing all of them at once.
- **Naive solution:** Calculate all possible throws, sum them and then divide them by the number of possible outcomes.
- **Complexity:** $\mathcal{O}({x_{\max}}^n)$. Gets accepted!
- **Observation:**
  - $E[X+Y] = E[X] + E[Y]$, for any independent X and Y
  - $E[\text{dice with k faces}] = \frac{1}{k}\sum_{x=1}^{k} x = \frac{k+1}{2}$
- **Fast solution:** Calculate the expected value of every dice, then sum them up.
- **Complexity:** $\mathcal{O}(n)$.

E: Expected Eyes

Problem Author: Maarten Sijm and Leon van der Waal

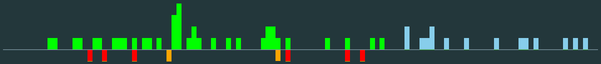


- **Problem:** Given list of dice and the number of faces of each one of them, calculate the expected value of throwing all of them at once.
- **Naive solution:** Calculate all possible throws, sum them and then divide them by the number of possible outcomes.
- **Complexity:** $\mathcal{O}({x_{\max}}^n)$. Gets accepted!
- **Observation:**
  - $E[X+Y] = E[X] + E[Y]$, for any independent X and Y
  - $E[\text{dice with k faces}] = \frac{1}{k}\sum_{x=1}^{k} x = \frac{k+1}{2}$
- **Fast solution:** Calculate the expected value of every dice, then sum them up.
- **Complexity:** $\mathcal{O}(n)$.

Statistics: 48 submissions, 44 accepted, 1 unknown

A: Admiring Droplets

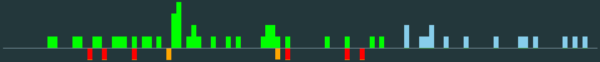
Problem Author: Davina van Meer and Maarten Sijm



- **Problem:** Calculate the time that it takes for the fully coalesced droplet to reach the bottom of the window.

A: Admiring Droplets

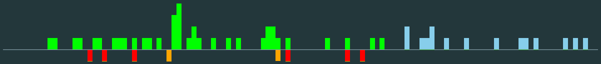
Problem Author: Davina van Meer and Maarten Sijm



- **Problem:** Calculate the time that it takes for the fully coalesced droplet to reach the bottom of the window.
- **Solution:** Perform a simulation of the droplets rolling down the window, one by one.

A: Admiring Droplets

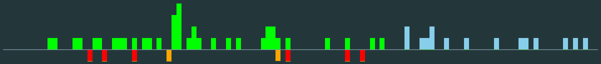
Problem Author: Davina van Meer and Maarten Sijm



- **Problem:** Calculate the time that it takes for the fully coalesced droplet to reach the bottom of the window.
- **Solution:** Perform a simulation of the droplets rolling down the window, one by one.
- **Pitfalls:**
  - Check your unit conversions: $1\ \text{m} = 1000\ \text{mm}$, $1\ \text{m}^3 = 10^9\ \text{mm}^3$.

A: Admiring Droplets

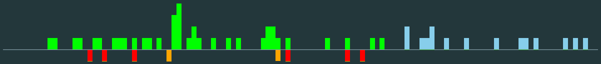
Problem Author: Davina van Meer and Maarten Sijm



- **Problem:** Calculate the time that it takes for the fully coalesced droplet to reach the bottom of the window.
- **Solution:** Perform a simulation of the droplets rolling down the window, one by one.
- **Pitfalls:**
  - Check your unit conversions: $1 \text{ m} = 1000 \text{ mm}$, $1 \text{ m}^3 = 10^9 \text{ mm}^3$.
  - Off-by-one errors are easy to make.

A: Admiring Droplets

Problem Author: Davina van Meer and Maarten Sijm

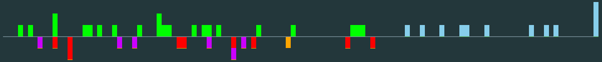


- **Problem:** Calculate the time that it takes for the fully coalesced droplet to reach the bottom of the window.
- **Solution:** Perform a simulation of the droplets rolling down the window, one by one.
- **Pitfalls:**
  - Check your unit conversions: $1\ \text{m} = 1000\ \text{mm}$, $1\ \text{m}^3 = 10^9\ \text{mm}^3$.
  - Off-by-one errors are easy to make.

Statistics: 61 submissions, 38 accepted, 15 unknown

B: Beaking Spackwards

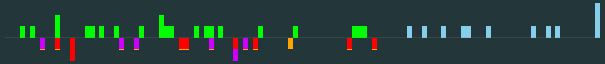
Problem Author: Jeroen Op de Beek



- **Problem:** Generate a palindrome with exactly $n \leq 10^9$ palindrome substrings.

B: Beaking Spackwards

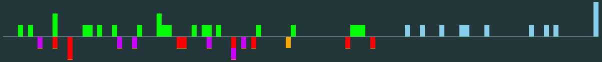
Problem Author: Jeroen Op de Beek



- **Problem:** Generate a palindrome with exactly $n \leq 10^9$ palindrome substrings.
- **Observation:** A string of length ℓ with identical characters has $\frac{\ell(\ell+1)}{2}$ palindromes.
  - Thus, the length of the palindrome-esque word is $\mathcal{O}(\sqrt{n})$.

B: Beaking Spackwards

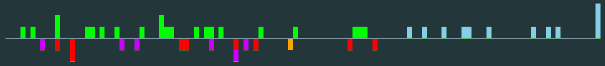
Problem Author: Jeroen Op de Beek



- **Problem:** Generate a palindrome with exactly $n \leq 10^9$ palindrome substrings.
- **Observation:** A string of length ℓ with identical characters has $\frac{\ell(\ell+1)}{2}$ palindromes.
  - Thus, the length of the palindrome-esque word is $\mathcal{O}(\sqrt{n})$.
- **Solution:** Find an ℓ such that $\frac{\ell(\ell+1)}{2} \leq n$, generate "a" $\times\, \ell$, and fill up the remainder with either:
  - non-palindromes (e.g., "`bcdbcd...`").
  - recursively applying the same strategy until the remaining length is 0.

B: Beaking Spackwards

Problem Author: Jeroen Op de Beek

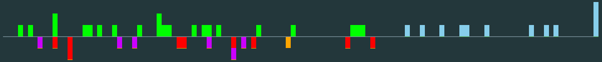


- **Problem:** Generate a palindrome with exactly $n \leq 10^9$ palindrome substrings.
- **Observation:** A string of length ℓ with identical characters has $\frac{\ell(\ell+1)}{2}$ palindromes.
  - Thus, the length of the palindrome-esque word is $\mathcal{O}(\sqrt{n})$.
- **Solution:** Find an ℓ such that $\frac{\ell(\ell+1)}{2} \leq n$, generate "a" $\times\, \ell$, and fill up the remainder with either:
  - non-palindromes (e.g., "`bcdbcd`...").
  - recursively applying the same strategy until the remaining length is 0.
- ℓ can be found using (linear or binary) search, or exactly:

$$\ell = \left\lfloor \frac{\sqrt{8n+1} - 1}{2} \right\rfloor$$

B: Beaking Spackwards

Problem Author: Jeroen Op de Beek



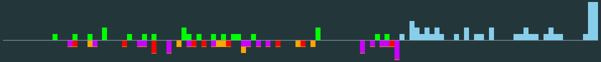
- **Problem:** Generate a palindrome with exactly $n \leq 10^9$ palindrome substrings.
- **Observation:** A string of length ℓ with identical characters has $\frac{\ell(\ell+1)}{2}$ palindromes.
  - Thus, the length of the palindrome-esque word is $\mathcal{O}(\sqrt{n})$.
- **Solution:** Find an ℓ such that $\frac{\ell(\ell+1)}{2} \leq n$, generate "a" $\times\, \ell$, and fill up the remainder with either:
  - non-palindromes (e.g., "`bcdbcd`. . . ").
  - recursively applying the same strategy until the remaining length is 0.
- ℓ can be found using (linear or binary) search, or exactly:

$$\ell = \left\lfloor \frac{\sqrt{8n+1}-1}{2} \right\rfloor$$

Statistics: 50 submissions, 22 accepted, 12 unknown

C: Catchy Tunes

Problem Author: Red Kaleb

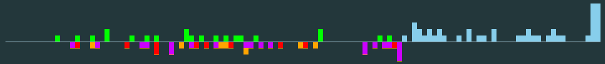


- **Problem:** Shuffle the playlist such that no two songs from the same artist are played in a row.

- **Problem:** Shuffle the playlist such that no two songs from the same artist are played in a row.
- **Guarantee:** At least half of the songs are from a unique artist.

C: Catchy Tunes

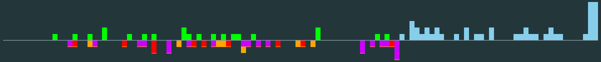
Problem Author: Red Kaleb



- **Problem:** Shuffle the playlist such that no two songs from the same artist are played in a row.
- **Guarantee:** At least half of the songs are from a unique artist.
- **Solution:** Interleave the songs that have a unique artist with other songs.

C: Catchy Tunes

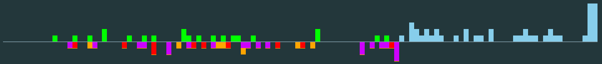
Problem Author: Red Kaleb



- **Problem:** Shuffle the playlist such that no two songs from the same artist are played in a row.
- **Guarantee:** At least half of the songs are from a unique artist.
- **Solution:** Interleave the songs that have a unique artist with other songs.
- Many other fancy strategies are possible, but not required.

C: Catchy Tunes

Problem Author: Red Kaleb

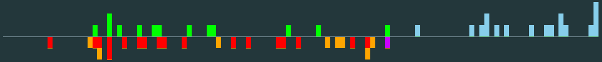


- **Problem:** Shuffle the playlist such that no two songs from the same artist are played in a row.
- **Guarantee:** At least half of the songs are from a unique artist.
- **Solution:** Interleave the songs that have a unique artist with other songs.
- Many other fancy strategies are possible, but not required.

Statistics: 103 submissions, 21 accepted, 44 unknown

I: Industry Improvements

Problem Author: Cristian-Alexandru Botocan

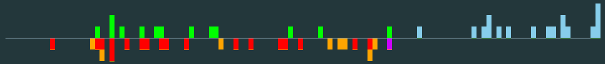


- **Problem:** Given a list of n boxes that need to be processed by a machine line in at most k runs, determine the minimum summed weight that the machine needs to handle in one run.

- **Problem:** Given a list of n boxes that need to be processed by a machine line in at most k runs, determine the minimum summed weight that the machine needs to handle in one run.
- **Naive solution:** Iterate through all possible capacities and simulate the machine line to see if it finishes in less than k runs.
  $\mathcal{O}\left(n \cdot \sum x\right)$ is too slow! Where $\sum x$ is the sum of all the weights.

I: Industry Improvements

Problem Author: Cristian-Alexandru Botocan

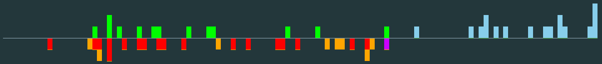


- **Problem:** Given a list of n boxes that need to be processed by a machine line in at most k runs, determine the minimum summed weight that the machine needs to handle in one run.
- **Naive solution:** Iterate through all possible capacities and simulate the machine line to see if it finishes in less than k runs.
  $\mathcal{O}\left(n \cdot \sum x\right)$ is too slow! Where $\sum x$ is the sum of all the weights.
- **Observation:** If the machine line processes everything in less than k runs with a capacity of a, then it will also work for a capacity of b, where $a < b$.

- **Problem:** Given a list of n boxes that need to be processed by a machine line in at most k runs, determine the minimum summed weight that the machine needs to handle in one run.
- **Naive solution:** Iterate through all possible capacities and simulate the machine line to see if it finishes in less than k runs.
  $\mathcal{O}\left(n \cdot \sum x\right)$ is too slow! Where $\sum x$ is the sum of all the weights.
- **Observation:** If the machine line processes everything in less than k runs with a capacity of a, then it will also work for a capacity of b, where $a < b$.
- **Solution:** Binary search the capacity of the machine line.

I: Industry Improvements

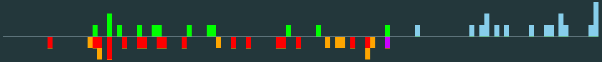
Problem Author: Cristian-Alexandru Botocan



- **Problem:** Given a list of n boxes that need to be processed by a machine line in at most k runs, determine the minimum summed weight that the machine needs to handle in one run.
- **Naive solution:** Iterate through all possible capacities and simulate the machine line to see if it finishes in less than k runs.
  $\mathcal{O}\left(n \cdot \sum x\right)$ is too slow! Where $\sum x$ is the sum of all the weights.
- **Observation:** If the machine line processes everything in less than k runs with a capacity of a, then it will also work for a capacity of b, where $a < b$.
- **Solution:** Binary search the capacity of the machine line.
- **Complexity:** $\mathcal{O}\left(n \log\left(\sum x\right)\right)$.

I: Industry Improvements

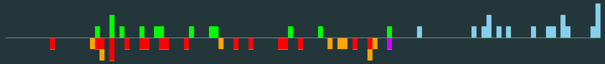
Problem Author: Cristian-Alexandru Botocan



- **Problem:** Given a list of n boxes that need to be processed by a machine line in at most k runs, determine the minimum summed weight that the machine needs to handle in one run.
- **Naive solution:** Iterate through all possible capacities and simulate the machine line to see if it finishes in less than k runs.
  $\mathcal{O}\left(n \cdot \sum x\right)$ is too slow! Where $\sum x$ is the sum of all the weights.
- **Observation:** If the machine line processes everything in less than k runs with a capacity of a, then it will also work for a capacity of b, where $a < b$.
- **Solution:** Binary search the capacity of the machine line.
- **Complexity:** $\mathcal{O}\left(n \log\left(\sum x\right)\right)$.
- **Pitfall:** Do not start binary search at 0, because machine capacity should be larger than the largest box.

I: Industry Improvements

Problem Author: Cristian-Alexandru Botocan



- **Problem:** Given a list of n boxes that need to be processed by a machine line in at most k runs, determine the minimum summed weight that the machine needs to handle in one run.
- **Naive solution:** Iterate through all possible capacities and simulate the machine line to see if it finishes in less than k runs.
  $\mathcal{O}\left(n \cdot \sum x\right)$ is too slow! Where $\sum x$ is the sum of all the weights.
- **Observation:** If the machine line processes everything in less than k runs with a capacity of a, then it will also work for a capacity of b, where $a < b$.
- **Solution:** Binary search the capacity of the machine line.
- **Complexity:** $\mathcal{O}\left(n \log\left(\sum x\right)\right)$.
- **Pitfall:** Do not start binary search at 0, because machine capacity should be larger than the largest box.

Statistics: 57 submissions, 13 accepted, 17 unknown

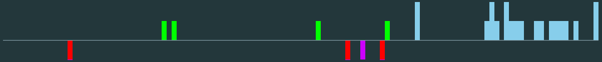
H: Hunting the Mavericks

Problem Author: Angel Karchev

- **Problem:** Determine in which level to start your playthrough, so that you miss the least armour upgrades.

H: Hunting the Mavericks

Problem Author: Angel Karchev



- **Problem:** Determine in which level to start your playthrough, so that you miss the least armour upgrades.
- **Solution:**
  - Calculate for each level i how many armour upgrades it contains (c_i) and how often it is required to obtain another armour upgrade (r_i).

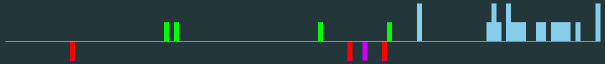
H: Hunting the Mavericks

Problem Author: Angel Karchev

- **Problem:** Determine in which level to start your playthrough, so that you miss the least armour upgrades.
- **Solution:**
  - Calculate for each level i how many armour upgrades it contains (c_i) and how often it is required to obtain another armour upgrade (r_i).
  - Determine how many upgrades you would miss by starting at level 1 and store this in a variable ans.

H: Hunting the Mavericks

Problem Author: Angel Karchev



- **Problem:** Determine in which level to start your playthrough, so that you miss the least armour upgrades.
- **Solution:**
  - Calculate for each level i how many armour upgrades it contains (c_i) and how often it is required to obtain another armour upgrade (r_i).
  - Determine how many upgrades you would miss by starting at level 1 and store this in a variable ans.
  - For each level i in ascending order, determine how many you would miss by doing level i last instead of first: $ans := ans + r_i - c_i$.

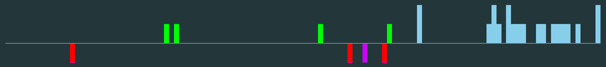
H: Hunting the Mavericks

Problem Author: Angel Karchev

- **Problem:** Determine in which level to start your playthrough, so that you miss the least armour upgrades.
- **Solution:**
  - Calculate for each level i how many armour upgrades it contains (c_i) and how often it is required to obtain another armour upgrade (r_i).
  - Determine how many upgrades you would miss by starting at level 1 and store this in a variable ans.
  - For each level i in ascending order, determine how many you would miss by doing level i last instead of first: $ans := ans + r_i - c_i$.
  - Output the minimal value for ans over all levels.

H: Hunting the Mavericks

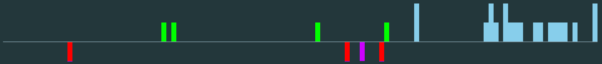
Problem Author: Angel Karchev



- **Problem:** Determine in which level to start your playthrough, so that you miss the least armour upgrades.
- **Solution:**
  - Calculate for each level i how many armour upgrades it contains (c_i) and how often it is required to obtain another armour upgrade (r_i).
  - Determine how many upgrades you would miss by starting at level 1 and store this in a variable ans.
  - For each level i in ascending order, determine how many you would miss by doing level i last instead of first: $ans := ans + r_i - c_i$.
  - Output the minimal value for ans over all levels.
- **Pitfalls:** Brute force quadratic solutions are too slow.

H: Hunting the Mavericks

Problem Author: Angel Karchev

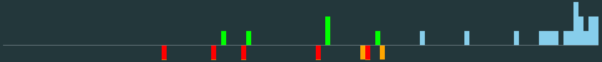


- **Problem:** Determine in which level to start your playthrough, so that you miss the least armour upgrades.
- **Solution:**
  - Calculate for each level i how many armour upgrades it contains (c_i) and how often it is required to obtain another armour upgrade (r_i).
  - Determine how many upgrades you would miss by starting at level 1 and store this in a variable *ans*.
  - For each level i in ascending order, determine how many you would miss by doing level i last instead of first: $ans := ans + r_i - c_i$.
  - Output the minimal value for *ans* over all levels.
- **Pitfalls:** Brute force quadratic solutions are too slow.

Statistics: 28 submissions, 4 accepted, 20 unknown

D: Dungeon of Darkness

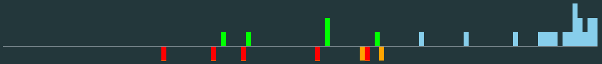
Problem Author: Maarten Sijm and Angel Karchev



- **Problem:** Get to the final room in a dungeon, where you only see the symbols of the doors leading from the current room.

D: Dungeon of Darkness

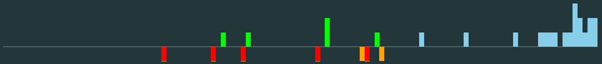
Problem Author: Maarten Sijm and Angel Karchev



- **Problem:** Get to the final room in a dungeon, where you only see the symbols of the doors leading from the current room.
- **Solution:** Use recursive DFS to delve deeper into the dungeon.

D: Dungeon of Darkness

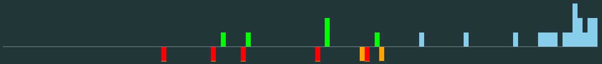
Problem Author: Maarten Sijm and Angel Karchev



- **Problem:** Get to the final room in a dungeon, where you only see the symbols of the doors leading from the current room.
- **Solution:** Use recursive DFS to delve deeper into the dungeon.
- If you don't find the final room in a recursive call, print the symbol of the door you went through to go back.

D: Dungeon of Darkness

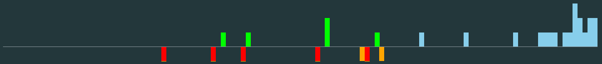
Problem Author: Maarten Sijm and Angel Karchev



- **Problem:** Get to the final room in a dungeon, where you only see the symbols of the doors leading from the current room.
- **Solution:** Use recursive DFS to delve deeper into the dungeon.
- If you don't find the final room in a recursive call, print the symbol of the door you went through to go back.
- **Pitfall:** Not being careful about when you print the symbol of a door.

D: Dungeon of Darkness

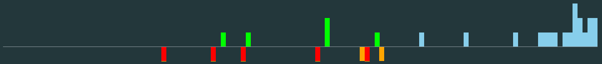
Problem Author: Maarten Sijm and Angel Karchev



- **Problem:** Get to the final room in a dungeon, where you only see the symbols of the doors leading from the current room.
- **Solution:** Use recursive DFS to delve deeper into the dungeon.
- If you don't find the final room in a recursive call, print the symbol of the door you went through to go back.
- **Pitfall:** Not being careful about when you print the symbol of a door.
- On that note, if you solved this problem, you're probably qualified to be in the jury for next year (the point above took the author > 3 hours to debug).

D: Dungeon of Darkness

Problem Author: Maarten Sijm and Angel Karchev

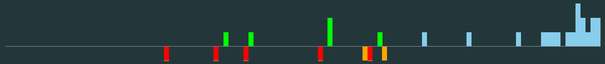


- **Problem:** Get to the final room in a dungeon, where you only see the symbols of the doors leading from the current room.
- **Solution:** Use recursive DFS to delve deeper into the dungeon.
- If you don't find the final room in a recursive call, print the symbol of the door you went through to go back.
- **Pitfall:** Not being careful about when you print the symbol of a door.
- On that note, if you solved this problem, you're probably qualified to be in the jury for next year (the point above took the author >3 hours to debug).

"If there is a way forward, you are never lost"
—*The guy two invented DFS, probably*

D: Dungeon of Darkness

Problem Author: Maarten Sijm and Angel Karchev



- **Problem:** Get to the final room in a dungeon, where you only see the symbols of the doors leading from the current room.
- **Solution:** Use recursive DFS to delve deeper into the dungeon.
- If you don't find the final room in a recursive call, print the symbol of the door you went through to go back.
- **Pitfall:** Not being careful about when you print the symbol of a door.
- On that note, if you solved this problem, you're probably qualified to be in the jury for next year (the point above took the author >3 hours to debug).

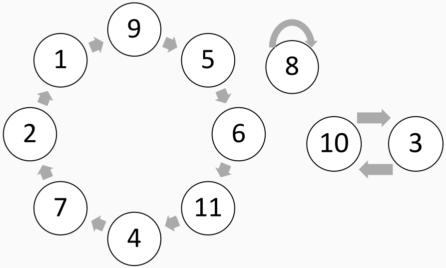
"If there is a way forward, you are never lost"
—*The guy two invented DFS, probably*

Statistics: 31 submissions, 5 accepted, 19 unknown

F: Feline Friendship

Problem Author: Jeroen Op de Beek

- **Problem:** Given a special functional graph, change the least number of edges such that there is a maximal path of length k.
- **Observation 1:** The array in the input is a permutation $\rightarrow$ the favourite cat relations form disjoint cycles.

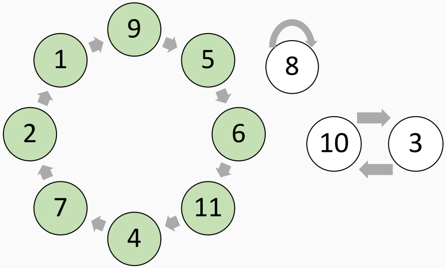


Cats are visualised as circles, with arrows being the favourite playing cat relations

F: Feline Friendship

Problem Author: Jeroen Op de Beek

- **Problem:** Given a special functional graph, change the least number of edges such that there is a maximal path of length k.
- **Observation 2:** If there is a cycle of length k, 0 operations suffice.

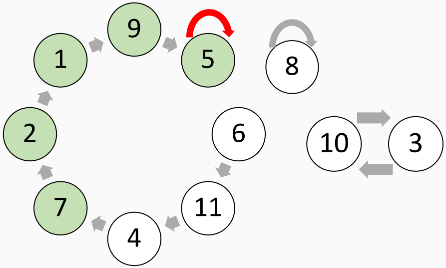


Take whole cycle as team.

F: Feline Friendship

Problem Author: Jeroen Op de Beek

- **Problem:** Given a special functional graph, change the least number of edges such that there is a maximal path of length k.
- **Observation 3:** Else if there is a cycle of length $> k$, 1 operation is enough.

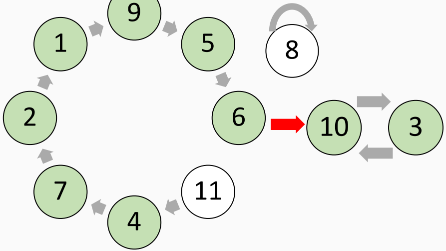


Make cycle into a path, and then choose where to start.

F: Feline Friendship

Problem Author: Jeroen Op de Beek

- **Problem:** Given a special functional graph, change the least number of edges such that there is a maximal path of length k.
- **Observation 4:** If you can make teams of length a and b, can make team of length $a+1, a+2, \ldots, a+b$ in 1 operation.

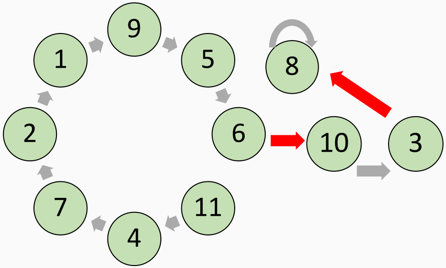


Merge two cycles.

F: Feline Friendship

Problem Author: Jeroen Op de Beek

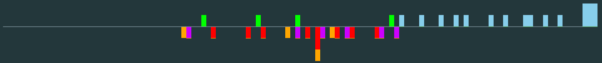
- **Problem:** Given a special functional graph, change the least number of edges such that there is a maximal path of length k.
- **Solution:** Greedily merge largest cycles in this way, until sum becomes $\geq k$.
- **Complexity:** $\mathcal{O}(n)$ for finding the disjoint cycles and $\mathcal{O}(n \log n)$ or $\mathcal{O}(n)$ for sorting the cycle sizes.



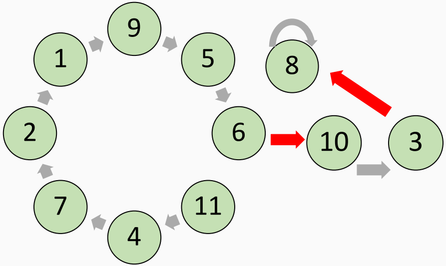
Repeatedly connect cycles together.

F: Feline Friendship

Problem Author: Jeroen Op de Beek



- **Problem:** Given a special functional graph, change the least number of edges such that there is a maximal path of length k.
- **Solution:** Greedily merge largest cycles in this way, until sum becomes $\geq k$.
- **Complexity:** $\mathcal{O}(n)$ for finding the disjoint cycles and $\mathcal{O}(n \log n)$ or $\mathcal{O}(n)$ for sorting the cycle sizes.



Repeatedly connect cycles together.

Statistics: 40 submissions, 4 accepted, 17 unknown

G: Gridlock

Problem Author: Matei Tinca

- **Problem:** Given a grid full of arrows, remove them one by one. When removing an arrow, it must not point to another arrow in the grid.

G: Gridlock

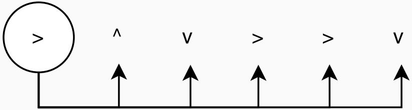
Problem Author: Matei Tinca

- **Problem:** Given a grid full of arrows, remove them one by one. When removing an arrow, it must not point to another arrow in the grid.
- **Observation:** If we want to remove an arrow, all of the other arrows that it points to must be removed first.

G: Gridlock

Problem Author: Matei Tinca

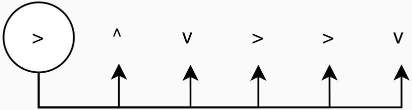
- **Problem:** Given a grid full of arrows, remove them one by one. When removing an arrow, it must not point to another arrow in the grid.
- **Observation:** If we want to remove an arrow, all of the other arrows that it points to must be removed first.
- This means that the arrow that we want to remove has a bunch of dependencies.



G: Gridlock

Problem Author: Matei Tinca

- **Problem:** Given a grid full of arrows, remove them one by one. When removing an arrow, it must not point to another arrow in the grid.
- **Observation:** If we want to remove an arrow, all of the other arrows that it points to must be removed first.
- This means that the arrow that we want to remove has a bunch of dependencies.

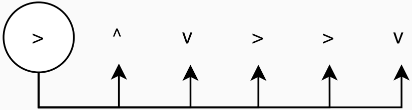


- **Naive Solution:** We can build a graph with all the dependencies, then compute the topological sorting.

G: Gridlock

Problem Author: Matei Tinca

- **Problem:** Given a grid full of arrows, remove them one by one. When removing an arrow, it must not point to another arrow in the grid.
- **Observation:** If we want to remove an arrow, all of the other arrows that it points to must be removed first.
- This means that the arrow that we want to remove has a bunch of dependencies.



- **Naive Solution:** We can build a graph with all the dependencies, then compute the topological sorting.
  - If one such sorting exists, we have a solution.
  - If one does not exist, then it is impossible to solve the puzzle.

G: Gridlock

Problem Author: Matei Tinca

- **Naive Solution:** We can build a graph with all the dependencies, then compute the topological sorting.
- **Complexity:**
  - Since an arrow points to an entire row or column of arrows, a cell has $\mathcal{O}(h)$ or $\mathcal{O}(w)$ dependencies, therefore, we have $\mathcal{O}(hw)$ nodes and $\mathcal{O}(hw(h+w))$ edges.

G: Gridlock

Problem Author: Matei Tinca

- **Naive Solution:** We can build a graph with all the dependencies, then compute the topological sorting.
- **Complexity:**
  - Since an arrow points to an entire row or column of arrows, a cell has $\mathcal{O}(h)$ or $\mathcal{O}(w)$ dependencies, therefore, we have $\mathcal{O}(hw)$ nodes and $\mathcal{O}(hw(h+w))$ edges.
  - Since the topological sorting has $\mathcal{O}(V+E)$ complexity, then this solution yields $\mathcal{O}(hw(h+w))$ complexity, which is not enough to pass the time limit.

- **Observation:** Maybe we can try to solve this in the opposite way.

G: Gridlock

Problem Author: Matei Tinca

- **Observation:** Maybe we can try to solve this in the opposite way.
- Instead of starting from a cell and recursively solving all of its dependencies, we can instead start from the trivial nodes, with no dependencies and then solve the nodes with more dependencies.

- **Observation:** Maybe we can try to solve this in the opposite way.
- **Solution:** Start from the margins of the puzzle and remove the cells one by one.

G: Gridlock

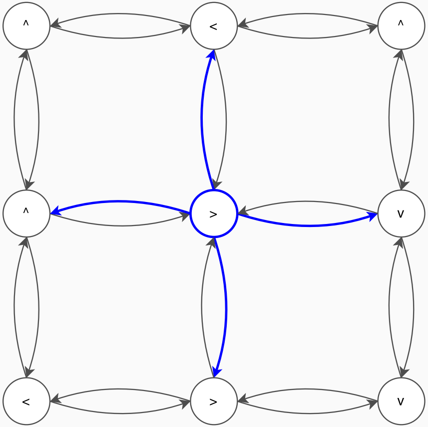
Problem Author: Matei Tinca

- **Observation:** Maybe we can try to solve this in the opposite way.
- **Solution:** Start from the margins of the puzzle and remove the cells one by one.
- When removing a cell, we can then attempt to remove all of its remaining neighbours.

G: Gridlock

Problem Author: Matei Tinca

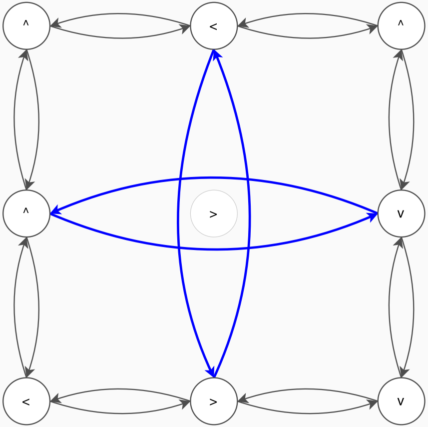
- **Observation:** Maybe we can try to solve this in the opposite way.
- **Solution:** Start from the margins of the puzzle and remove the cells one by one.
- When removing a cell, we can then attempt to remove all of its remaining neighbours.
- To find the neighbours of a cell quickly, we can maintain for each cell a link to its neighbours.



G: Gridlock

Problem Author: Matei Tinca

- **Observation:** Maybe we can try to solve this in the opposite way.
- **Solution:** Start from the margins of the puzzle and remove the cells one by one.
- When removing a cell, we can then attempt to remove all of its remaining neighbours.
- To find the neighbours of a cell quickly, we can maintain for each cell a link to its neighbours.
- When removing a cell, we must link its left and right neighbours between each other. The same goes for the neighbours from above and below.



G: Gridlock

Problem Author: Matei Tinca

- **Solution:** Start from the margins of the puzzle and remove the cells one by one.
- When removing a cell, we can then attempt to remove all of its remaining neighbours.
- **Complexity:**
  - We remove $\mathcal{O}(h \cdot w)$ cells.

G: Gridlock

Problem Author: Matei Tinca

- **Solution:** Start from the margins of the puzzle and remove the cells one by one.
- When removing a cell, we can then attempt to remove all of its remaining neighbours.
- **Complexity:**
  - We remove $\mathcal{O}(h \cdot w)$ cells.
  - Whenever we remove a cell, we check for at most 4 neighbours (up, down, left, right).

G: Gridlock

Problem Author: Matei Tinca

- **Solution:** Start from the margins of the puzzle and remove the cells one by one.
- When removing a cell, we can then attempt to remove all of its remaining neighbours.
- **Complexity:**
  - We remove $\mathcal{O}(h \cdot w)$ cells.
  - Whenever we remove a cell, we check for at most 4 neighbours (up, down, left, right).
  - Changing the links of neighbouring cells takes $\mathcal{O}(1)$ time.

G: Gridlock

Problem Author: Matei Tinca

- **Solution:** Start from the margins of the puzzle and remove the cells one by one.
- When removing a cell, we can then attempt to remove all of its remaining neighbours.
- **Complexity:**
  - We remove $\mathcal{O}(h \cdot w)$ cells.
  - Whenever we remove a cell, we check for at most 4 neighbours (up, down, left, right).
  - Changing the links of neighbouring cells takes $\mathcal{O}(1)$ time.
  - In total, the solution takes $\mathcal{O}(h \cdot w)$ time, which is enough to pass the time limit.

G: Gridlock

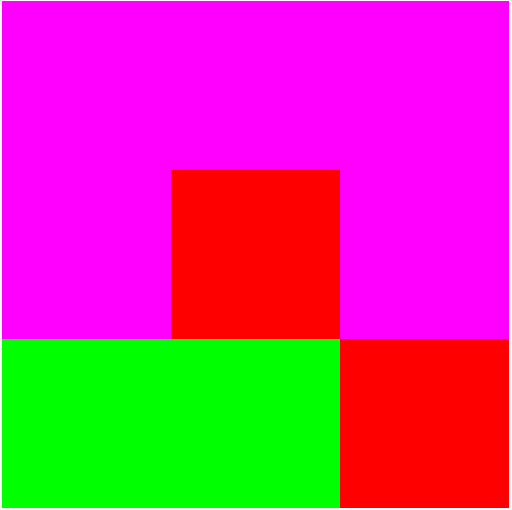
Problem Author: Matei Tinca

- **Solution:** Start from the margins of the puzzle and remove the cells one by one.
- When removing a cell, we can then attempt to remove all of its remaining neighbours.
- **Complexity:**
  - We remove $\mathcal{O}(h \cdot w)$ cells.
  - Whenever we remove a cell, we check for at most 4 neighbours (up, down, left, right).
  - Changing the links of neighbouring cells takes $\mathcal{O}(1)$ time.
  - In total, the solution takes $\mathcal{O}(h \cdot w)$ time, which is enough to pass the time limit.

Statistics: 47 submissions, 1 accepted, 27 unknown

G: Gridlock

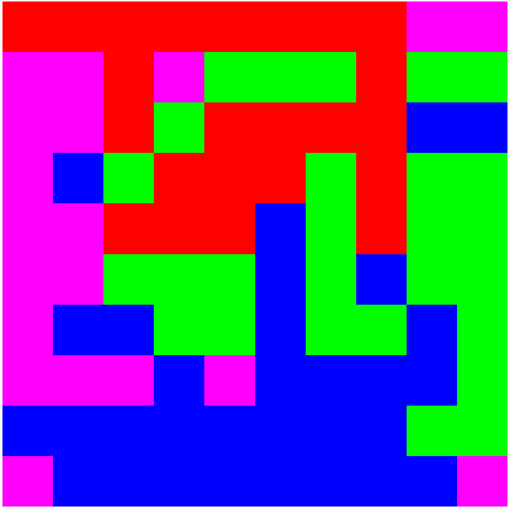
Problem Author: Matei Tinca



Magenta: Left Red: Up Green: Right Blue: Down (possible)

G: Gridlock

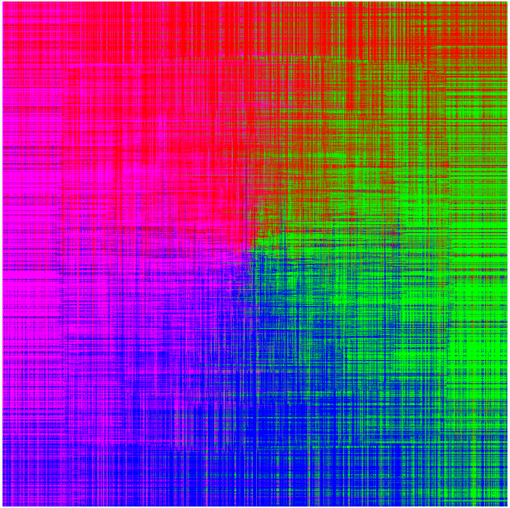
Problem Author: Matei Tinca



Magenta: Left Red: Up Green: Right Blue: Down (possible)

G: Gridlock

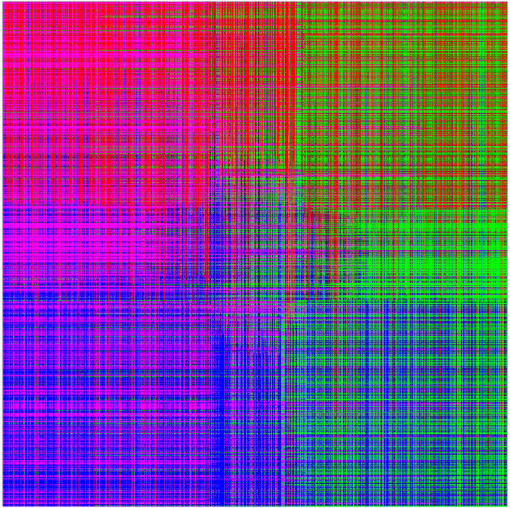
Problem Author: Matei Tinca



Magenta: Left Red: Up Green: Right Blue: Down (possible)

G: Gridlock

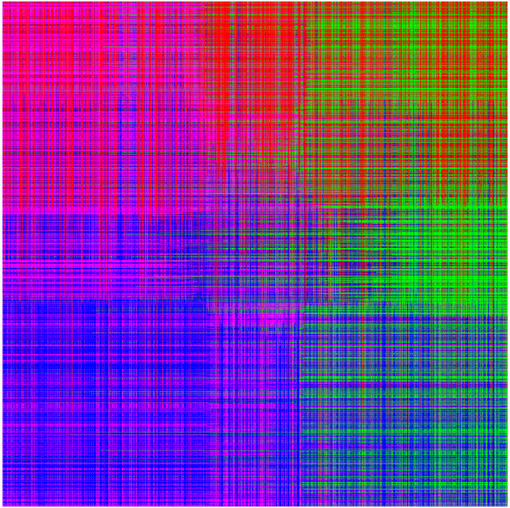
Problem Author: Matei Tinca



Magenta: Left Red: Up Green: Right Blue: Down (possible)

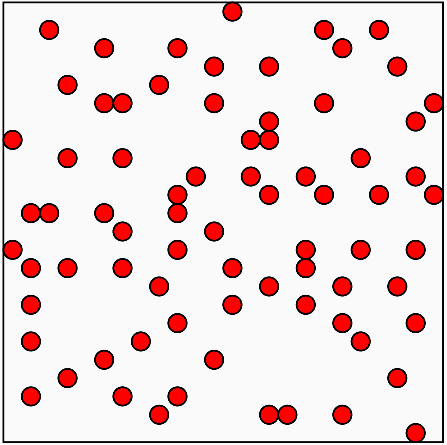
G: Gridlock

Problem Author: Matei Tinca

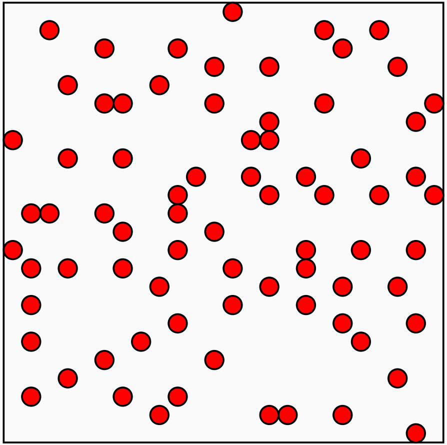


Magenta: Left Red: Up Green: Right Blue: Down (**im**possible)

- **Problem:** Given a set of **uniform random** points in a square, find the smallest perimeter among all triangles.

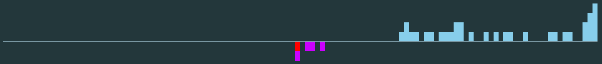


- **Problem:** Given a set of **uniform random** points in a square, find the smallest perimeter among all triangles.
- **Naive solution:** Calculate the perimeter of all possible triangles, and take the minimum.

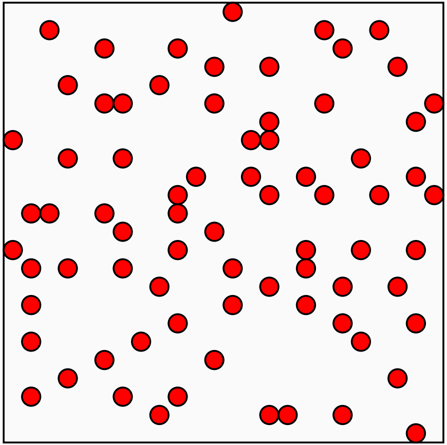


J: Jurassic Park

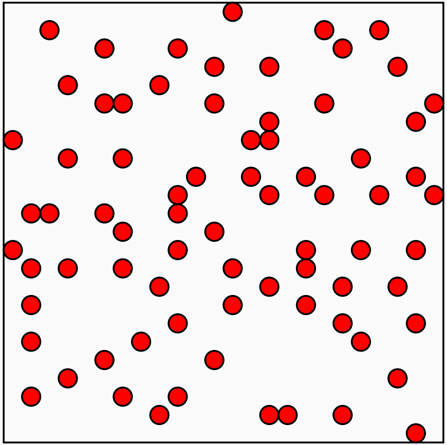
Problem Author: Leon van der Waal



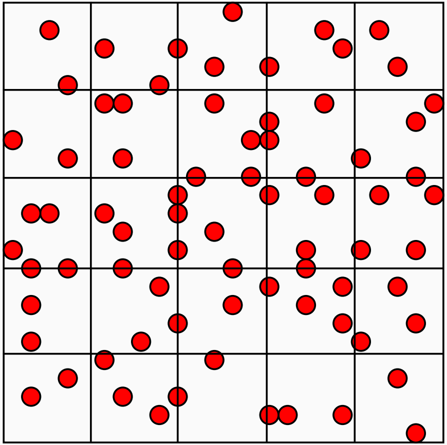
- **Problem:** Given a set of **uniform random** points in a square, find the smallest perimeter among all triangles.
- **Naive solution:** Calculate the perimeter of all possible triangles, and take the minimum.
- **Complexity:** This solution runs in $\mathcal{O}\left(n^3\right)$ time, too slow!



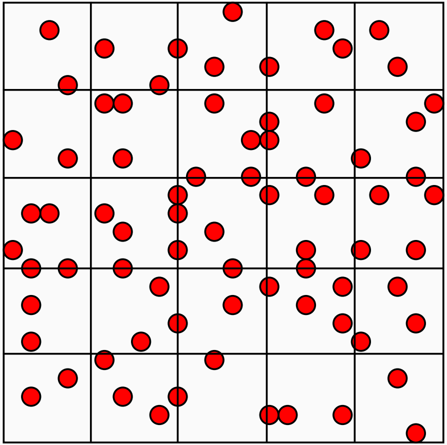
- **Problem:** Given a set of **uniform random** points in a square, find the smallest perimeter among all triangles.
- **Observation:** The points are randomly distributed, so there are no nasty testcases.
- How can we make use of this fact?



- **Problem:** Given a set of **uniform random** points in a square, find the smallest perimeter among all triangles.
- **Solution:** Divide the bounding box into a $\left\lfloor\sqrt{\frac{n}{3}}\right\rfloor \times \left\lfloor\sqrt{\frac{n}{3}}\right\rfloor$ grid, with sidelengths ℓ.



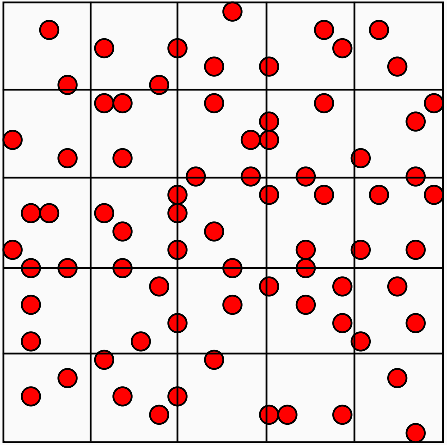
- **Problem:** Given a set of **uniform random** points in a square, find the smallest perimeter among all triangles.
- **Solution:** Divide the bounding box into a $\left\lfloor\sqrt{\frac{n}{3}}\right\rfloor \times \left\lfloor\sqrt{\frac{n}{3}}\right\rfloor$ grid, with sidelengths ℓ.
- **Observation:** By the pigeonhole principle, at least one of the tiles contains 3 points.



J: Jurassic Park

Problem Author: Leon van der Waal

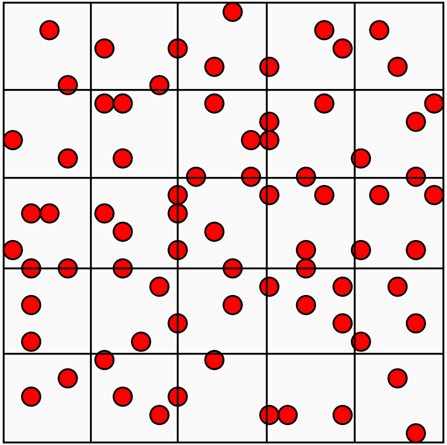
- **Problem:** Given a set of **uniform random** points in a square, find the smallest perimeter among all triangles.
- **Solution:** Divide the bounding box into a $\left\lfloor\sqrt{\frac{n}{3}}\right\rfloor \times \left\lfloor\sqrt{\frac{n}{3}}\right\rfloor$ grid, with sidelengths ℓ.
- **Observation:** By the pigeonhole principle, at least one of the tiles contains 3 points.
- **Observation:** The smallest perimeter is hence at most $\left(2+\sqrt{2}\right)\ell$.



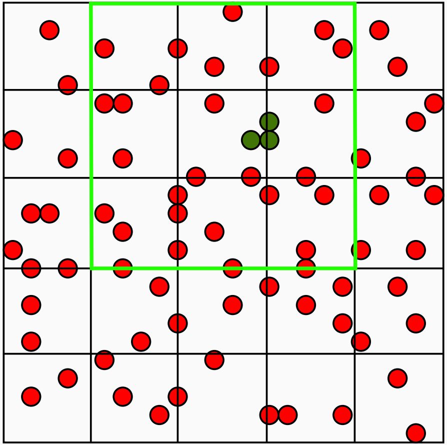
J: Jurassic Park

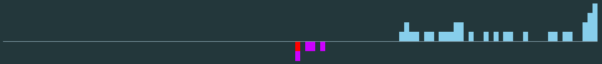
Problem Author: Leon van der Waal

- **Problem:** Given a set of **uniform random** points in a square, find the smallest perimeter among all triangles.
- **Solution:** Divide the bounding box into a $\left\lfloor\sqrt{\frac{n}{3}}\right\rfloor \times \left\lfloor\sqrt{\frac{n}{3}}\right\rfloor$ grid, with sidelengths ℓ.
- **Observation:** By the pigeonhole principle, at least one of the tiles contains 3 points.
- **Observation:** The smallest perimeter is hence at most $\left(2+\sqrt{2}\right)\ell$.
- **Observation:** This means that the distance between two points of the smallest triangle can at most be $\left(1+\frac{1}{2}\sqrt{2}\right)\ell < 2\ell$.

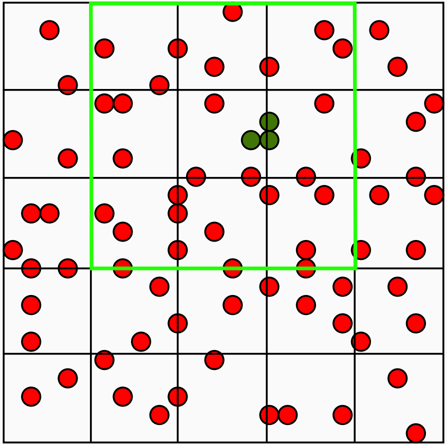


- **Problem:** Given a set of **uniform random** points in a square, find the smallest perimeter among all triangles.
- **Observation:** This means that the distance between two points of the smallest triangle can at most be $\left(1 + \frac{1}{2}\sqrt{2}\right)\ell < 2\ell$.
- **Solution:** Calculate the perimeter of the triangles contained in all blocks of 3×3 tiles.

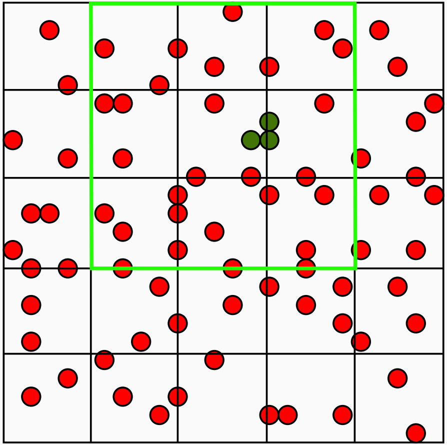




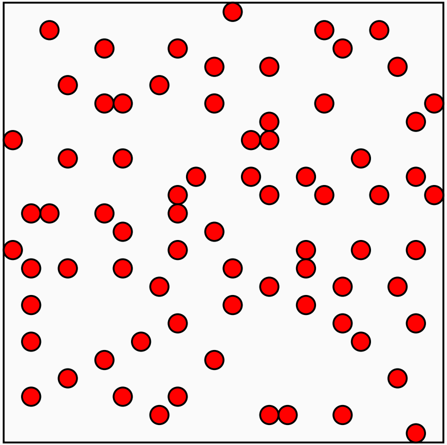
- **Problem:** Given a set of **uniform random** points in a square, find the smallest perimeter among all triangles.
- **Observation:** This means that the distance between two points of the smallest triangle can at most be $\left(1 + \frac{1}{2}\sqrt{2}\right)\ell < 2\ell$.
- **Solution:** Calculate the perimeter of the triangles contained in all blocks of 3×3 tiles.
- **Solution:** Because the points are uniformly distributed, the number of points inside the blocks is small with high probability.



- **Problem:** Given a set of **uniform random** points in a square, find the smallest perimeter among all triangles.
- **Observation:** This means that the distance between two points of the smallest triangle can at most be $\left(1 + \frac{1}{2}\sqrt{2}\right)\ell < 2\ell$.
- **Solution:** Calculate the perimeter of the triangles contained in all blocks of 3×3 tiles.
- **Solution:** Because the points are uniformly distributed, the number of points inside the blocks is small with high probability.
- **Complexity:** $\mathcal{O}(n)$, with high probability.



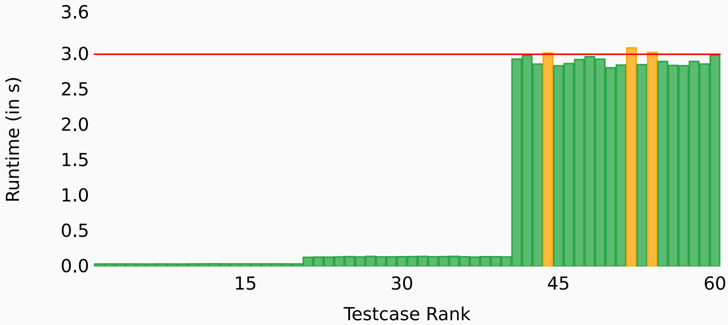
- **Problem:** Given a set of **uniform random** points in a square, find the smallest perimeter among all triangles.
- **Solution:** Many other solutions work using the randomness, as long as you somehow do not check all possible triangles.
- **Challenge:** Try to make an algorithm that does not use randomness, and runs in $\mathcal{O}(n \log(n))$ time.



J: Jurassic Park

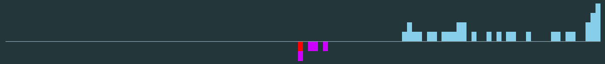
Problem Author: Leon van der Waal

- Time limits can be tricky...

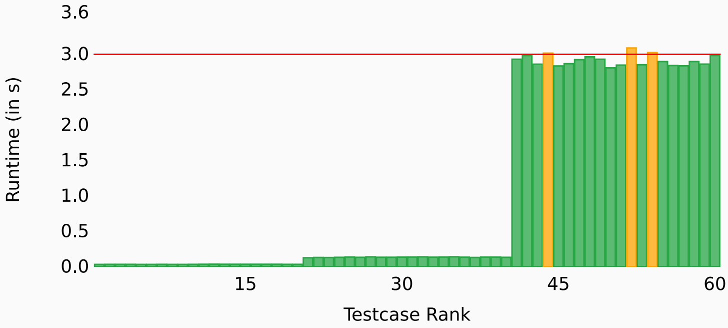


J: Jurassic Park

Problem Author: Leon van der Waal

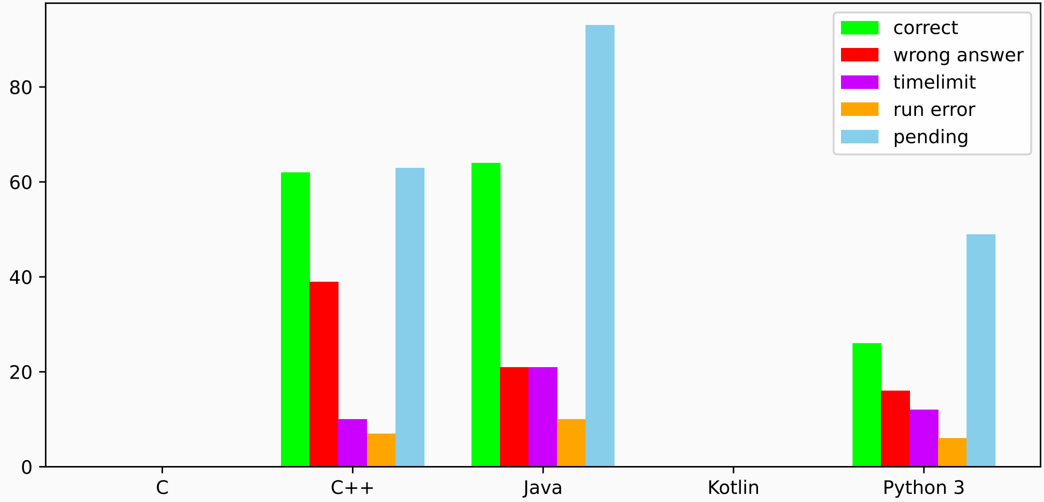


- Time limits can be tricky. . .



Statistics: 38 submissions, 0 accepted, 33 unknown

Language stats



Random facts

Jury work

- 361 commits (last year: 371)

[1] *After* codegolfing

Random facts

Jury work

- 361 commits (last year: 371)
- 339 secret test cases (last year: 252)

[1] *After* codegolfing

Random facts

Jury work

- 361 commits (last year: 371)
- 339 secret test cases (last year: 252)
- 96 accepted jury/proofreader solutions (last year: 59)

[1] *After* codegolfing

Random facts

Jury work

- 361 commits (last year: 371)
- 339 secret test cases (last year: 252)
- 96 accepted jury/proofreader solutions (last year: 59)
- The minimum[1] number of lines the jury needed to solve all problems is

$$4 + 6 + 5 + 9 + 1 + 12 + 14 + 5 + 3 + 5 = 64$$

  On average 6.4 lines per problem, down from 10.4 last year

[1] *After* codegolfing

Thanks to:

The Proofreaders

- Andrei Botocan (Bucharest, Romania)
- Davina van Meer (Delft)
- Michael Vasseur (VU Amsterdam)
- Michal Tešnar (RU Groningen)
- Nadyne Aretz (TU Delft)
- Thomas Verwoerd 🎈 (TU Delft)
- Wietze Koops (RU Groningen)

The Jury for FPC (TU Delft) and AAPJE (VU Amsterdam)

- Alexandru Bolfa (TU Delft)
- Angel Karchev (TU Delft)
- Jeroen Op de Beek (TU Delft)
- Leon van der Waal (TU Delft)
- Maarten Sijm (TU Delft)
- Matei Tinca (VU Amsterdam)
- Red Kalab (VU Amsterdam)