Problem Analysis Session

EUC 2024 judges

March 24, 2024

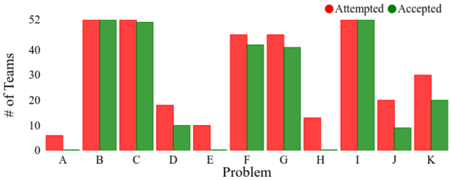
Our judges and problemsetters

- Federico Glaudo (Chief judge)
- Lucian Bicsi
- Martin Kacer
- Petr Mitrichev
- Giovanni Paolini
- Anton Trygub
- Michael Zündorf

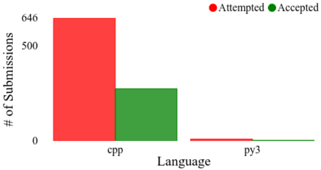
Statistics (at freeze)

Total number of **submissions**: 657

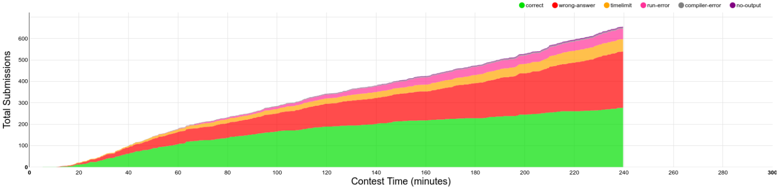
of which **accepted**: 277 ($\sim$ 42%)



Problems attempted/solved



Language statistics



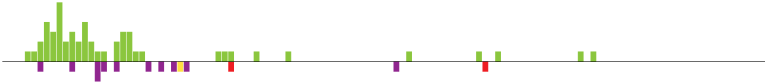
Submissions over time

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Number of **submissions**: 66
of which **accepted**: 52 ($\sim$ 79%)

First solved by **treenity (University of Cambridge)** after 10m



B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

The problem

- You are given two sets of dishes: appetizers and main dishes.

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

The problem

- You are given two sets of dishes: appetizers and main dishes.
- Each dish has spiciness.

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

The problem

- You are given two sets of dishes: appetizers and main dishes.
- Each dish has spiciness.
- Meal is appetizer + main dish

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

The problem

- You are given two sets of dishes: appetizers and main dishes.
- Each dish has spiciness.
- Meal is appetizer + main dish
- The charm of a meal is defined as an absolute value of the difference in these spicinesses.

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

The problem

- You are given two sets of dishes: appetizers and main dishes.
- Each dish has spiciness.
- Meal is appetizer + main dish
- The charm of a meal is defined as an absolute value of the difference in these spicinesses.
- **Goal:** Form meals in a way that maximizes minimum charm.

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

The problem

- You are given two sets of dishes: appetizers and main dishes.
- Each dish has spiciness.
- Meal is appetizer + main dish
- The charm of a meal is defined as an absolute value of the difference in these spicinesses.
- **Goal:** Form meals in a way that maximizes minimum charm.

Formal problem

You are given two arrays $a_1, a_2, \ldots, a_n$, $b_1, b_2, \ldots, b_n$. Over all permutations $(p_1, p_2, \ldots, p_n)$ of $(1, 2, \ldots, n)$, find the maximum possible value of

$$\min(|a_1 - b_{p_1}|, |a_2 - b_{p_2}|, \ldots, |a_n - b_{p_n}|)$$

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Lemma (Greedy ordering)

Let c, d be some nondecreasing arrays of length n. If there exists some permutation $\sigma(1), \sigma(2), \ldots, \sigma(n)$, such that $c_i \leq d_{\sigma(i)}$ for all i, then $c_i \leq d_i$ for all i.

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Lemma (Greedy ordering)

Let c, d be some nondecreasing arrays of length n. If there exists some permutation $\sigma(1), \sigma(2), \ldots, \sigma(n)$, such that $c_i \leq d_{\sigma(i)}$ for all i, then $c_i \leq d_i$ for all i.

Proof.

- Consider any $1 \leq i \leq n$

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Lemma (Greedy ordering)

Let c, d be some nondecreasing arrays of length n. If there exists some permutation $\sigma(1), \sigma(2), \ldots, \sigma(n)$, such that $c_i \leq d_{\sigma(i)}$ for all i, then $c_i \leq d_i$ for all i.

Proof.

- Consider any $1 \leq i \leq n$
- For any $j \geq i$ we have $d_{p_j} \geq c_j \geq c_i$

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Lemma (Greedy ordering)

Let c, d be some nondecreasing arrays of length n. If there exists some permutation $\sigma(1), \sigma(2), \dots, \sigma(n)$, such that $c_i \le d_{\sigma(i)}$ for all i, then $c_i \le d_i$ for all i.

Proof.

- Consider any $1 \le i \le n$
- For any $j \ge i$ we have $d_{p_j} \ge c_j \ge c_i$
- There can be at most $i - 1$ js with $d_j < c_i \implies d_i \ge c_i$

$\square$

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Assume $a_1 \leq a_2 \leq \ldots \leq a_n$, $b_1 \leq b_2 \leq \ldots \leq b_n$.

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Assume $a_1 \le a_2 \le \ldots \le a_n$, $b_1 \le b_2 \le \ldots \le b_n$.

Consider some optimal pairing, assume we paired a_i with $b_{\sigma(i)}$, and got a minimum charm of k. Let's call pairs with $a_i + k \le b_{\sigma(i)}$ **small**, and pairs with $a_i \ge b_{\sigma(i)} + k$ **large**.

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Assume $a_1 \le a_2 \le \ldots \le a_n$, $b_1 \le b_2 \le \ldots \le b_n$.

Consider some optimal pairing, assume we paired a_i with $b_{\sigma(i)}$, and got a minimum charm of k. Let's call pairs with $a_i + k \le b_{\sigma(i)}$ **small**, and pairs with $a_i \ge b_{\sigma(i)} + k$ **large**.

Observation 1

For elements of small pairs, we can assume they are paired in a sorted order. Same for large pairs.

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Assume $a_1 \leq a_2 \leq \ldots \leq a_n$, $b_1 \leq b_2 \leq \ldots \leq b_n$.

Consider some optimal pairing, assume we paired a_i with $b_{\sigma(i)}$, and got a minimum charm of k. Let's call pairs with $a_i + k \leq b_{\sigma(i)}$ **small**, and pairs with $a_i \geq b_{\sigma(i)} + k$ **large**.

Observation 1

For elements of small pairs, we can assume they are paired in a sorted order. Same for large pairs.

Proof.

For small pairs, we have $a_i \leq b_{p_i} + k$, so we can apply the greedy ordering lemma. $\square$

Observation 2

Let the number of small pairs be t. Then we can assume that we paired smallest t a_is with largest t b_is, and largest $n - t$ a_is with smallest $n - t$ b_is, and we will still have a charm of at least k.

Observation 2

Let the number of small pairs be t. Then we can assume that we paired smallest t a_is with largest t b_is, and largest $n-t$ a_is with smallest $n-t$ b_is, and we will still have a charm of at least k.

More formally:

- For $1 \leq i \leq t$, pair a_i with $b_{i+(n-t)}$;
- For $t+1 \leq i \leq n$, pair the a_i with b_{i-t}.

Observation 2

Let the number of small pairs be t. Then we can assume that we paired smallest t a_is with largest t b_is, and largest $n-t$ a_is with smallest $n-t$ b_is, and we will still have a charm of at least k.

More formally:

- For $1 \le i \le t$, pair a_i with $b_{i+(n-t)}$;
- For $t+1 \le i \le n$, pair the a_i with b_{i-t}.

Proof.

Trivial greedy! $\square$

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Solution

- Sort a_is, b_is.

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Solution

- Sort a_is, b_is.
- For each t from 0 to n:

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Solution

- Sort a_is, b_is.
- For each t from 0 to n:
  - Find the minimum charm when the smallest t a_is are paired with largest t b_is, and the largest $n-t$ a_is with the smallest $n-t$ b_is.

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Solution

- Sort a_is, b_is.
- For each t from 0 to n:
    - Find the minimum charm when the smallest t a_is are paired with largest t b_is, and the largest $n - t$ a_is with the smallest $n - t$ b_is.
- Return the largest of these values

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Solution

- Sort a_is, b_is.
- For each t from 0 to n:
  - Find the minimum charm when the smallest t a_is are paired with largest t b_is, and the largest $n-t$ a_is with the smallest $n-t$ b_is.
- Return the largest of these values

B Charming Meals

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Solution

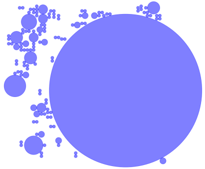
- Sort a_is, b_is.
- For each t from 0 to n:
  - Find the minimum charm when the smallest t a_is are paired with largest t b_is, and the largest $n-t$ a_is with the smallest $n-t$ b_is.
- Return the largest of these values

Running time: $O(n^2)$.

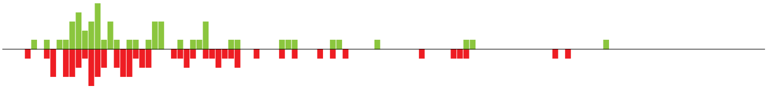
I Disks

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Number of **submissions**: 114
of which **accepted**: 52 ($\sim$ 46%)



First solved by **Heroes of the C (Universidade do Porto)** after 13m



I Disks

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

The problem

Given n non-overlapping disks on the plane, determine whether you can change their radii so that tangent disks remain tangent, there is no overlap, and the sum of all radii strictly decreases.

I Disks

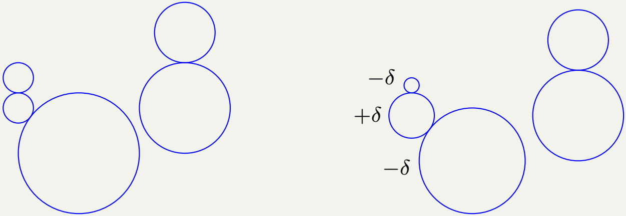
AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

The problem

Given n non-overlapping disks on the plane, determine whether you can change their radii so that tangent disks remain tangent, there is no overlap, and the sum of all radii strictly decreases.

Solution

- If we change the radius of a disk by δ, the radius of any tangent disk has to change by $-\delta$.



I Disks

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- In general, if we change the radii by real numbers $\delta_1, \ldots, \delta_n$, we must have $\delta_i = -\delta_j$ whenever the i-th and j-th disks are tangent.

I Disks

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

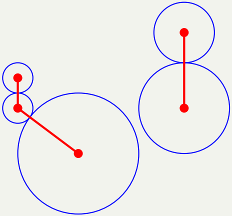
Solution

- In general, if we change the radii by real numbers $\delta_1, \dots, \delta_n$, we must have $\delta_i = -\delta_j$ whenever the i-th and j-th disks are tangent.
- This condition is also sufficient, provided that $\delta_1, \dots, \delta_n$ are small enough in absolute value (to avoid overlaps and to keep all the radii positive).

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- In general, if we change the radii by real numbers $\delta_1, \ldots, \delta_n$, we must have $\delta_i = -\delta_j$ whenever the i-th and j-th disks are tangent.
- This condition is also sufficient, provided that $\delta_1, \ldots, \delta_n$ are small enough in absolute value (to avoid overlaps and to keep all the radii positive).
- Build the graph that describes the tangency relation between the disks:

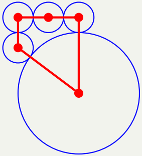


I Disks

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- If a connected component has an odd cycle, the radii of the disks in that component cannot be changed.

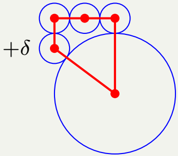


I Disks

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- If a connected component has an odd cycle, the radii of the disks in that component cannot be changed.

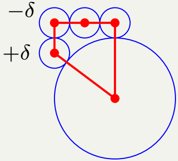


I Disks

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- If a connected component has an odd cycle, the radii of the disks in that component cannot be changed.

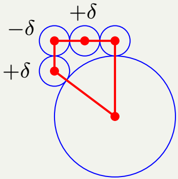


I Disks

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- If a connected component has an odd cycle, the radii of the disks in that component cannot be changed.

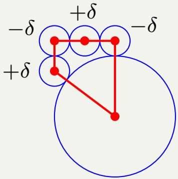


I Disks

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

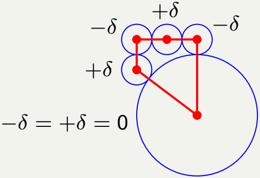
Solution

- If a connected component has an odd cycle, the radii of the disks in that component cannot be changed.



Solution

- If a connected component has an odd cycle, the radii of the disks in that component cannot be changed.

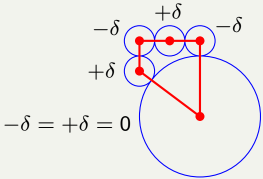


I Disks

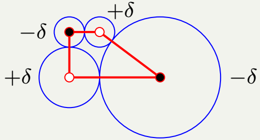
AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- If a connected component has an odd cycle, the radii of the disks in that component cannot be changed.



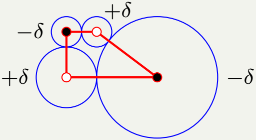
- If a connected component has no odd cycles, then it is bipartite, and we can change radii of white-colored disks by $+\delta$ and black-colored by $-\delta$.



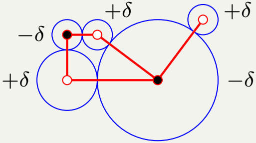
AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- To strictly decrease the sum of the radii, we need a bipartite connected component with a different number of white and black disks.



Not good

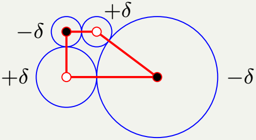


Good

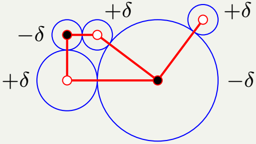
AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- To strictly decrease the sum of the radii, we need a bipartite connected component with a different number of white and black disks.



Not good

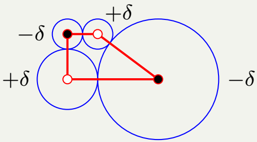


Good

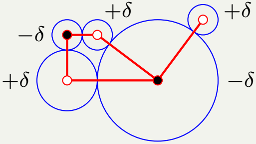
- To summarize:

Solution

- To strictly decrease the sum of the radii, we need a bipartite connected component with a different number of white and black disks.



Not good



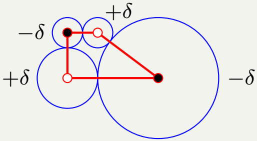
Good

- To summarize:
  - Construct the tangency graph — $O(n^2)$ is enough.

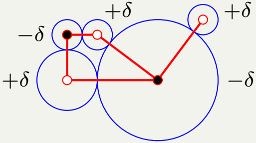
AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- To strictly decrease the sum of the radii, we need a bipartite connected component with a different number of white and black disks.



Not good



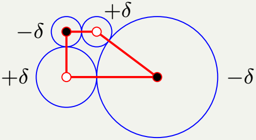
Good

- To summarize:
  - Construct the tangency graph — $O(n^2)$ is enough.
  - Visit each connected component, check if it is bipartite and the number of white and black disks is different — in $O(n)$.

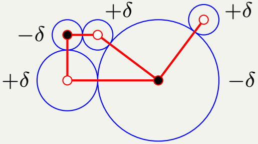
AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- To strictly decrease the sum of the radii, we need a bipartite connected component with a different number of white and black disks.



Not good



Good

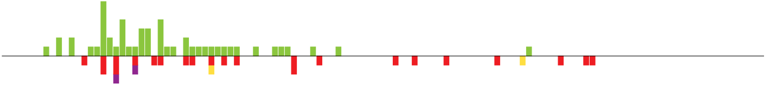
- To summarize:
  - Construct the tangency graph — $O(n^2)$ is enough.
  - Visit each connected component, check if it is bipartite and the number of white and black disks is different — in $O(n)$.
  - The answer is YES if and only if at least one component is good.

C Annual Ants Gathering

AUTHORED BY: Petr Mitrichev PREPARED BY: Michael Zündorf

Number of **submissions**: 79
of which **accepted**: 51 ($\sim$ 65%)

First solved by **<(OvO)> (Saarland University)** after 17m



C Annual Ants Gathering

AUTHORED BY: Petr Mitrichev PREPARED BY: Michael Zündorf

The problem

Given a tree T where one ant is at every vertex. Is it possible to move all ants to the same vertex when you can only move ants from vertex u to vertex v if v does not contain less ants.

C Annual Ants Gathering

AUTHORED BY: Petr Mitrichev PREPARED BY: Michael Zündorf

The problem

Given a tree T where one ant is at every vertex. Is it possible to move all ants to the same vertex when you can only move ants from vertex u to vertex v if v does not contain less ants.

First Solution

- Since ants can not move to an empty vertex the the region where of non empty homes always needs to stay connected.

C Annual Ants Gathering

AUTHORED BY: Petr Mitrichev PREPARED BY: Michael Zündorf

The problem

Given a tree T where one ant is at every vertex. Is it possible to move all ants to the same vertex when you can only move ants from vertex u to vertex v if v does not contain less ants.

First Solution

- Since ants can not move to an empty vertex the the region where of non empty homes always needs to stay connected.
- Therefore, only ants at a leaf of the tree (of non empty vertices) can ever move and they can only move in one direction.

C Annual Ants Gathering

AUTHORED BY: Petr Mitrichev PREPARED BY: Michael Zündorf

The problem

Given a tree T where one ant is at every vertex. Is it possible to move all ants to the same vertex when you can only move ants from vertex u to vertex v if v does not contain less ants.

First Solution

- Since ants can not move to an empty vertex the the region where of non empty homes always needs to stay connected.
- Therefore, only ants at a leaf of the tree (of non empty vertices) can ever move and they can only move in one direction.
- Notice that if the smallest group of ants at a leaf can not move the solution is impossible (all other leaf groups are larger and can therefore also never be merged into the parent group of the smallest group)

C Annual Ants Gathering

AUTHORED BY: Petr Mitrichev PREPARED BY: Michael Zündorf

The problem

Given a tree T where one ant is at every vertex. Is it possible to move all ants to the same vertex when you can only move ants from vertex u to vertex v if v does not contain less ants.

First Solution

- Since ants can not move to an empty vertex the the region where of non empty homes always needs to stay connected.
- Therefore, only ants at a leaf of the tree (of non empty vertices) can ever move and they can only move in one direction.
- Notice that if the smallest group of ants at a leaf can not move the solution is impossible (all other leaf groups are larger and can therefore also never be merged into the parent group of the smallest group)
- We can simply simulate this process.

C Annual Ants Gathering

AUTHORED BY: Petr Mitrichev PREPARED BY: Michael Zündorf

The problem

Given a tree T where one ant is at every vertex. Is it possible to move all ants to the same vertex when you can only move ants from vertex u to vertex v if v does not contain less ants.

C Annual Ants Gathering

AUTHORED BY: Petr Mitrichev PREPARED BY: Michael Zündorf

The problem

Given a tree T where one ant is at every vertex. Is it possible to move all ants to the same vertex when you can only move ants from vertex u to vertex v if v does not contain less ants.

Second Solution

- Notice that the centroids of the tree are the only homes where all ants can gather up.

C Annual Ants Gathering

AUTHORED BY: Petr Mitrichev PREPARED BY: Michael Zündorf

The problem

Given a tree T where one ant is at every vertex. Is it possible to move all ants to the same vertex when you can only move ants from vertex u to vertex v if v does not contain less ants.

Second Solution

- Notice that the centroids of the tree are the only homes where all ants can gather up.
- For a fixed root we can can greedily simulate the process with a DFS for each centroid.

C Annual Ants Gathering

AUTHORED BY: Petr Mitrichev PREPARED BY: Michael Zündorf

The problem

Given a tree T where one ant is at every vertex. Is it possible to move all ants to the same vertex when you can only move ants from vertex u to vertex v if v does not contain less ants.

Second Solution

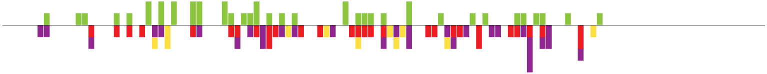
- Notice that the centroids of the tree are the only homes where all ants can gather up.
- For a fixed root we can can greedily simulate the process with a DFS for each centroid.
- Solve problem for all subtrees, sort them by ascending size and try to merge them into the parent.

F Dating

AUTHORED BY: Federico Glaudo PREPARED BY: Lucian Bicsi

Number of **submissions**: 114
of which **accepted**: 42 ($\sim$ 37%)

First solved by **KNU_0_GB_RAM (Taras Shevchenko National University of Kyiv)** after 4h 5m



F Dating

AUTHORED BY: Federico Glaudo PREPARED BY: Lucian Bicsi

The problem

Given n sets $S_1, S_2, \ldots, S_n$ of activities, find a pair (a, b) such that all three sets $S_a \setminus S_b, S_b \setminus S_a, S_a \cap S_b$ are non-empty.

F Dating

AUTHORED BY: Federico Glaudo PREPARED BY: Lucian Bicsi

The problem

Given n sets $S_1, S_2, \ldots, S_n$ of activities, find a pair (a, b) such that all three sets $S_a \setminus S_b, S_b \setminus S_a, S_a \cap S_b$ are non-empty.

Solution

- A pair (a, b) is good if and only if S_a and S_b are neither disjoint nor one included in the other.

F Dating

AUTHORED BY: Federico Glaudo PREPARED BY: Lucian Bicsi

The problem

Given n sets $S_1, S_2, \ldots, S_n$ of activities, find a pair (a, b) such that all three sets $S_a \setminus S_b, S_b \setminus S_a, S_a \cap S_b$ are non-empty.

Solution

- A pair (a, b) is good if and only if S_a and S_b are neither disjoint nor one included in the other.
- If there are no such good pairs, then the activities must induce a **tree** structure!

F Dating

AUTHORED BY: Federico Glaudo PREPARED BY: Lucian Bicsi

The problem

Given n sets $S_1, S_2, \ldots, S_n$ of activities, find a pair (a, b) such that all three sets $S_a \setminus S_b, S_b \setminus S_a, S_a \cap S_b$ are non-empty.

Solution

- A pair (a, b) is good if and only if S_a and S_b are neither disjoint nor one included in the other.
- If there are no such good pairs, then the activities must induce a **tree** structure!
- **Rough idea:** try to write a checker by building such a tree, and see if/when it fails.

Top-down approach

Top-down approach

- Sort the activity sets in decreasing order of lengths

Top-down approach

- Sort the activity sets in decreasing order of lengths
- For all activity sets S_i in this order, create a parent-child relationship $j = p(i)$ with the last processed index j that contains some activity (or -1, if there is no such index)

Top-down approach

- Sort the activity sets in decreasing order of lengths
- For all activity sets S_i in this order, create a parent-child relationship $j = p(i)$ with the last processed index j that contains some activity (or -1, if there is no such index)
  - If $j \neq -1$ and $S_i \not\subseteq S_j$, then (i, j) is a good pair

Top-down approach

- Sort the activity sets in decreasing order of lengths
- For all activity sets S_i in this order, create a parent-child relationship $j = p(i)$ with the last processed index j that contains some activity (or -1, if there is no such index)
  - If $j \neq -1$ and $S_i \not\subseteq S_j$, then (i, j) is a good pair
  - If some activity $x \in S_i$ is also in some other set S_k where $p(k) = p(i)$, then (i, k) is a good pair

Top-down approach

- Sort the activity sets in decreasing order of lengths
- For all activity sets S_i in this order, create a parent-child relationship $j = p(i)$ with the last processed index j that contains some activity (or -1, if there is no such index)
  - If $j \neq -1$ and $S_i \not\subseteq S_j$, then (i, j) is a good pair
  - If some activity $x \in S_i$ is also in some other set S_k where $p(k) = p(i)$, then (i, k) is a good pair
  - Otherwise, the activity sets still form a tree, therefore no good pairs exist (yet)

Top-down approach

- Sort the activity sets in decreasing order of lengths
- For all activity sets S_i in this order, create a parent-child relationship $j = p(i)$ with the last processed index j that contains some activity (or -1, if there is no such index)
  - If $j \neq -1$ and $S_i \not\subseteq S_j$, then (i, j) is a good pair
  - If some activity $x \in S_i$ is also in some other set S_k where $p(k) = p(i)$, then (i, k) is a good pair
  - Otherwise, the activity sets still form a tree, therefore no good pairs exist (yet)
- Complexity is $O(k \log k)$ or $O(k)$, where k is the total number of activities, depending on whether one uses ordered sets or hash maps.

Top-down approach

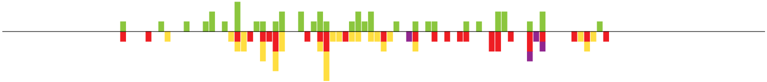
- Sort the activity sets in decreasing order of lengths
- For all activity sets S_i in this order, create a parent-child relationship $j = p(i)$ with the last processed index j that contains some activity (or -1, if there is no such index)
  - If $j \neq -1$ and $S_i \not\subseteq S_j$, then (i, j) is a good pair
  - If some activity $x \in S_i$ is also in some other set S_k where $p(k) = p(i)$, then (i, k) is a good pair
  - Otherwise, the activity sets still form a tree, therefore no good pairs exist (yet)
- Complexity is $O(k \log k)$ or $O(k)$, where k is the total number of activities, depending on whether one uses ordered sets or hash maps.
- It is possible to implement this solution in $O(k)$ using just arrays, but this is not required. Alternative complexities such as $O(k\sqrt{k})$ should also pass, with careful implementation.

G Scooter

AUTHORED BY: Giovanni Paolini PREPARED BY: Lucian Bicsi

Number of **submissions**: 103
of which **accepted**: 41 ($\sim$ 40%)

First solved by **Zagreb 1 (University of Zagreb)** after 49m



G Scooter

AUTHORED BY: Giovanni Paolini PREPARED BY: Lucian Bicsi

The problem

Devise an itinerary to drive professors to their corresponding classes, without reaching the same building twice.

G Scooter

AUTHORED BY: Giovanni Paolini PREPARED BY: Lucian Bicsi

The problem

Devise an itinerary to drive professors to their corresponding classes, without reaching the same building twice.

Solution...

- Ignore professors that are already in a proper building.

G Scooter

AUTHORED BY: Giovanni Paolini PREPARED BY: Lucian Bicsi

The problem

Devise an itinerary to drive professors to their corresponding classes, without reaching the same building twice.

Solution...

- Ignore professors that are already in a proper building.
- Pick up a professor from some building and drive them to a building where a corresponding class is held.

G Scooter

AUTHORED BY: Giovanni Paolini   PREPARED BY: Lucian Bicsi

The problem

Devise an itinerary to drive professors to their corresponding classes, without reaching the same building twice.

Solution...

- Ignore professors that are already in a proper building.
- Pick up a professor from some building and drive them to a building where a corresponding class is held.
  - How to make sure we don't visit the same place twice?

G Scooter

AUTHORED BY: Giovanni Paolini   PREPARED BY: Lucian Bicsi

The problem

Devise an itinerary to drive professors to their corresponding classes, without reaching the same building twice.

Solution...

- Ignore professors that are already in a proper building.
- Pick up a professor from some building and drive them to a building where a corresponding class is held.
  - How to make sure we don't visit the same place twice?
  - How to implement such a strategy with a positive attitude?

G Scooter

AUTHORED BY: Giovanni Paolini PREPARED BY: Lucian Bicsi

The problem

Devise an itinerary to drive professors to their corresponding classes, without reaching the same building twice.

Solution...

- Ignore professors that are already in a proper building.
- Pick up a professor from some building and drive them to a building where a corresponding class is held.
  - How to make sure we don't visit the same place twice?
  - How to implement such a strategy with a positive attitude?
- **Achtung!** Don't rush with the implementation! You might regret it...

Solution!

- Let's imagine that there are the same number of math professors as there are math classes (same with computer science).

Solution!

- Let's imagine that there are the same number of math professors as there are math classes (same with computer science).
- **The problem becomes easier!**

Solution!

- Let's imagine that there are the same number of math professors as there are math classes (same with computer science).
- **The problem becomes easier!**
    1. Drive to a building where there's a professor and no class, and drop them off at a building where there is a corresponding class.

Solution!

- Let's imagine that there are the same number of math professors as there are math classes (same with computer science).
- **The problem becomes easier!**
  1. Drive to a building where there's a professor and no class, and drop them off at a building where there is a corresponding class.
  2. Prioritize the buildings where there are professors to the ones which don't have any

Solution!

- Let's imagine that there are the same number of math professors as there are math classes (same with computer science).
- **The problem becomes easier!**
    1. Drive to a building where there's a professor and no class, and drop them off at a building where there is a corresponding class.
    2. Prioritize the buildings where there are professors to the ones which don't have any
    3. Repeat...

Solution!

- Let's imagine that there are the same number of math professors as there are math classes (same with computer science).
- **The problem becomes easier!**
  1. Drive to a building where there's a professor and no class, and drop them off at a building where there is a corresponding class.
  2. Prioritize the buildings where there are professors to the ones which don't have any
  3. Repeat...
- **Key condition:** There is a building with a professor and no class.

Solution!

- Let's imagine that there are the same number of math professors as there are math classes (same with computer science).
- **The problem becomes easier!**
  1. Drive to a building where there's a professor and no class, and drop them off at a building where there is a corresponding class.
  2. Prioritize the buildings where there are professors to the ones which don't have any
  3. Repeat...
- **Key condition:** There is a building with a professor and no class.
- Convert the initial problem to an instance of this simpler variant.

Solution!

- Let's imagine that there are the same number of math professors as there are math classes (same with computer science).
- **The problem becomes easier!**
  1. Drive to a building where there's a professor and no class, and drop them off at a building where there is a corresponding class.
  2. Prioritize the buildings where there are professors to the ones which don't have any
  3. Repeat...
- **Key condition:** There is a building with a professor and no class.
- Convert the initial problem to an instance of this simpler variant.
  - Remove all "excess" professors one by one.

Solution!

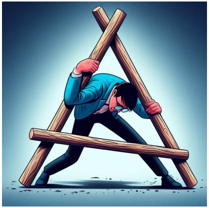
- Let's imagine that there are the same number of math professors as there are math classes (same with computer science).
- **The problem becomes easier!**
  1. Drive to a building where there's a professor and no class, and drop them off at a building where there is a corresponding class.
  2. Prioritize the buildings where there are professors to the ones which don't have any
  3. Repeat...
- **Key condition:** There is a building with a professor and no class.
- Convert the initial problem to an instance of this simpler variant.
  - Remove all "excess" professors one by one.
  - Prioritize removing professors from buildings where classes are held.

Solution!

- Let's imagine that there are the same number of math professors as there are math classes (same with computer science).
- **The problem becomes easier!**
  1. Drive to a building where there's a professor and no class, and drop them off at a building where there is a corresponding class.
  2. Prioritize the buildings where there are professors to the ones which don't have any
  3. Repeat...
- **Key condition:** There is a building with a professor and no class.
- Convert the initial problem to an instance of this simpler variant.
  - Remove all "excess" professors one by one.
  - Prioritize removing professors from buildings where classes are held.
- Complexity: $O(n)$

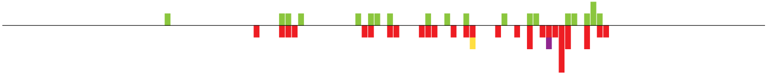
K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub



Number of **submissions**: 54
of which **accepted**: 20 ($\sim$ 37%)

First solved by **NewJeans (University of Oxford)** after 1h 6m



K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

The problem

- You are given n positive integers $x_1, x_2, \ldots, x_n$.
- You want to split them into three groups of sizes n_a, n_b, n_c so that:
  - Let s_a, s_b, s_c be the sums of numbers in these groups. Then s_a, s_b, s_c are the sides of a triangle with positive area.

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

The problem

- You are given n positive integers $x_1, x_2, \ldots, x_n$.
- You want to split them into three groups of sizes n_a, n_b, n_c so that:
  - Let s_a, s_b, s_c be the sums of numbers in these groups. Then s_a, s_b, s_c are the sides of a triangle with positive area.

Formal problem

Doesn't get more formal than that.

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Wlog $n_a \leq n_b \leq n_c$, and $x_1 \leq x_2 \leq \ldots \leq x_n$.

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Wlog $n_a \le n_b \le n_c$, and $x_1 \le x_2 \le \ldots \le x_n$.

Let the sum of all numbers be S. We just need the sum in each group to be smaller than $\frac{S}{2}$.

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Wlog $n_a \le n_b \le n_c$, and $x_1 \le x_2 \le \ldots \le x_n$.

Let the sum of all numbers be S. We just need the sum in each group to be smaller than $\frac{S}{2}$.

What are some obvious necessary constraints?

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Wlog $n_a \leq n_b \leq n_c$, and $x_1 \leq x_2 \leq \ldots \leq x_n$.

Let the sum of all numbers be S. We just need the sum in each group to be smaller than $\frac{S}{2}$.

What are some obvious necessary constraints?

- The largest **group** is not too large: $x_1 + x_2 + \ldots + x_{n_c} < \frac{S}{2}$;

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Wlog $n_a \leq n_b \leq n_c$, and $x_1 \leq x_2 \leq \ldots \leq x_n$.

Let the sum of all numbers be S. We just need the sum in each group to be smaller than $\frac{S}{2}$.

What are some obvious necessary constraints?

- The largest **group** is not too large: $x_1 + x_2 + \ldots + x_{n_c} < \frac{S}{2}$;
- The largest **item** is not too large: $x_n + (x_1 + x_2 + \ldots + x_{n_a - 1}) < \frac{S}{2}$.

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Wlog $n_a \leq n_b \leq n_c$, and $x_1 \leq x_2 \leq \ldots \leq x_n$.

Let the sum of all numbers be S. We just need the sum in each group to be smaller than $\frac{S}{2}$.

What are some obvious necessary constraints?

- The largest **group** is not too large: $x_1 + x_2 + \ldots + x_{n_c} < \frac{S}{2}$;
- The largest **item** is not too large: $x_n + (x_1 + x_2 + \ldots + x_{n_a - 1}) < \frac{S}{2}$.

Is it enough?

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Wlog $n_a \leq n_b \leq n_c$, and $x_1 \leq x_2 \leq \ldots \leq x_n$.

Let the sum of all numbers be S. We just need the sum in each group to be smaller than $\frac{S}{2}$.

What are some obvious necessary constraints?

- The largest **group** is not too large: $x_1 + x_2 + \ldots + x_{n_c} < \frac{S}{2}$;
- The largest **item** is not too large: $x_n + (x_1 + x_2 + \ldots + x_{n_a-1}) < \frac{S}{2}$.

Is it enough?

Of course, as in all my problems.

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Wlog $n_a \le n_b \le n_c$, and $x_1 \le x_2 \le \ldots \le x_n$.

Let the sum of all numbers be S. We just need the sum in each group to be smaller than $\frac{S}{2}$.

What are some obvious necessary constraints?

- The largest **group** is not too large: $x_1 + x_2 + \ldots + x_{n_c} < \frac{S}{2}$;
- The largest **item** is not too large: $x_n + (x_1 + x_2 + \ldots + x_{n_a - 1}) < \frac{S}{2}$.

Is it enough?

Of course, as in all my problems.

But we need construction too...

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Let's put elements into groups from largest to smallest. How to check if we can put an element into a given group?

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Let's put elements into groups from largest to smallest. How to check if we can put an element into a given group?

Assume we already placed a few largest numbers. Assume current sum in group g is S_g, the number of empty spots in group g is n'_g for $g \in \{a, b, c\}$, and there are $n'_a + n'_b + n'_c = n'$ numbers remaining, $x_1 \leq x_2 \leq \ldots \leq x_{n'}$.

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Let's put elements into groups from largest to smallest. How to check if we can put an element into a given group?

Assume we already placed a few largest numbers. Assume current sum in group g is S_g, the number of empty spots in group g is n'_g for $g \in \{a, b, c\}$, and there are $n'_a + n'_b + n'_c = n'$ numbers remaining, $x_1 \leq x_2 \leq \ldots \leq x_{n'}$.

What are some obvious constraints here?

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Let's put elements into groups from largest to smallest. How to check if we can put an element into a given group?

Assume we already placed a few largest numbers. Assume current sum in group g is S_g, the number of empty spots in group g is n'_g for $g \in \{a, b, c\}$, and there are $n'_a + n'_b + n'_c = n'$ numbers remaining, $x_1 \leq x_2 \leq \ldots \leq x_{n'}$.

What are some obvious constraints here?

- No group is too large: for any group g we have

$$S_g + x_1 + x_2 + \ldots + x_{n'_g} < \frac{S}{2}$$

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Let's put elements into groups from largest to smallest. How to check if we can put an element into a given group?

Assume we already placed a few largest numbers. Assume current sum in group g is S_g, the number of empty spots in group g is n'_g for $g \in \{a, b, c\}$, and there are $n'_a + n'_b + n'_c = n'$ numbers remaining, $x_1 \leq x_2 \leq \ldots \leq x_{n'}$.

What are some obvious constraints here?

- No group is too large: for any group g we have

$$S_g + x_1 + x_2 + \ldots + x_{n'_g} < \frac{S}{2}$$

- The largest item is not too large: there exists a group g with $n'_g > 0$, such that

$$S_g + x_{n'} + (x_1 + x_2 + \ldots + x_{n'_g - 1}) < \frac{S}{2}$$

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Proof.

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Proof.

Left as an exercise to the reader. □

Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Proof.

Left as an exercise to the reader. □

Solution

- Sort all x_is, and calculate prefix sums.

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Proof.

Left as an exercise to the reader. □

Solution

- Sort all x_is, and calculate prefix sums.

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Proof.

Left as an exercise to the reader. □

Solution

- Sort all x_is, and calculate prefix sums.
- Check if conditions hold, if no return NO.

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Proof.

Left as an exercise to the reader. □

Solution

- Sort all x_is, and calculate prefix sums.
- Check if conditions hold, if no return NO.
- For each element from largest to smallest:

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Proof.

Left as an exercise to the reader. □

Solution

- Sort all x_is, and calculate prefix sums.
- Check if conditions hold, if no return NO.
- For each element from largest to smallest:
  - Try to put it in each group, check if conditions hold.

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Proof.

Left as an exercise to the reader. □

Solution

- Sort all x_is, and calculate prefix sums.
- Check if conditions hold, if no return NO.
- For each element from largest to smallest:
  - Try to put it in each group, check if conditions hold.
  - If no, continue to the next group.

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Proof.

Left as an exercise to the reader. □

Solution

- Sort all x_is, and calculate prefix sums.
- Check if conditions hold, if no return NO.
- For each element from largest to smallest:
  - Try to put it in each group, check if conditions hold.
  - If no, continue to the next group.

K Make Triangle

AUTHORED BY: Anton Trygub PREPARED BY: Anton Trygub

Proof.

Left as an exercise to the reader. □

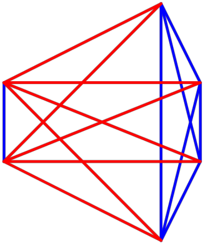
Solution

- Sort all x_is, and calculate prefix sums.
- Check if conditions hold, if no return NO.
- For each element from largest to smallest:
  - Try to put it in each group, check if conditions hold.
  - If no, continue to the next group.

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

D Funny or Scary

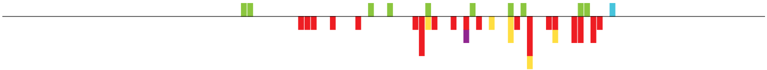
AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev



Number of **submissions**: 44

of which **accepted**: 10 ($\sim$ 23%)

First solved by **ELTE 1 (Eötvös Loránd University)** after 1h 37m



D Funny or Scary

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

The problem

You are given a complete undirected graph with n vertices and $\leq \lfloor \frac{n}{2} \rfloor$ edges colored red or blue. Color all remaining edges red or blue so that there is no simple monochromatic path with $> \lceil \frac{3n}{4} \rceil$ edges.

Funny or Scary

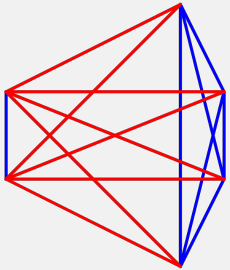
AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

The problem

You are given a complete undirected graph with n vertices and $\leq \lfloor \frac{n}{2} \rfloor$ edges colored red or blue. Color all remaining edges red or blue so that there is no simple monochromatic path with $> \lceil \frac{3n}{4} \rceil$ edges.

Solution

- If no edges are colored, split vertices into the small part and the big part.



D Funny or Scary

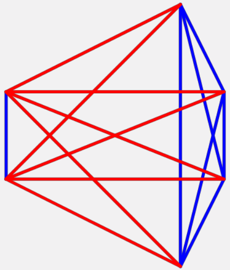
AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

The problem

You are given a complete undirected graph with n vertices and $\leq \lfloor \frac{n}{2} \rfloor$ edges colored red or blue. Color all remaining edges red or blue so that there is no simple monochromatic path with $> \lceil \frac{3n}{4} \rceil$ edges.

Solution

- If no edges are colored, split vertices into the small part and the big part.
- Max blue path: the size of the big part.



D Funny or Scary

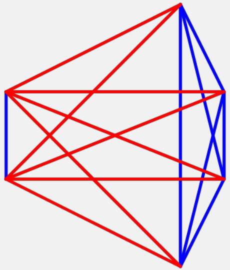
AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

The problem

You are given a complete undirected graph with n vertices and $\leq \lfloor \frac{n}{2} \rfloor$ edges colored red or blue. Color all remaining edges red or blue so that there is no simple monochromatic path with $> \lceil \frac{3n}{4} \rceil$ edges.

Solution

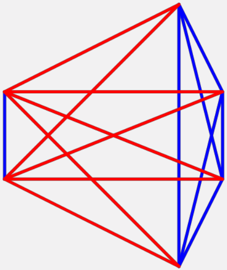
- If no edges are colored, split vertices into the small part and the big part.
- Max blue path: the size of the big part.
- Max red path: two times the size of the small part.



AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

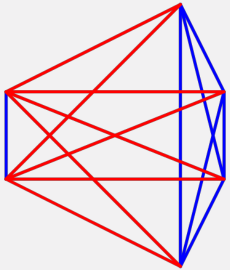
- Incorrect blue edges are bad: direct to $n-1$.



AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

- Incorrect blue edges are bad: direct to $n-1$.
- Incorrect red edges are not so bad: $+1$.

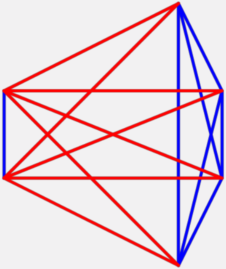


D Funny or Scary

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

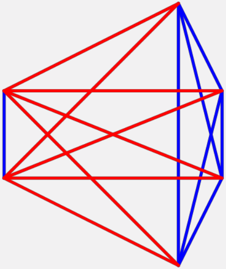
- Incorrect blue edges are bad: direct to $n-1$.
- Incorrect red edges are not so bad: $+1$.
- Take blue connected components together into small/big.



AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

- Incorrect blue edges are bad: direct to $n-1$.
- Incorrect red edges are not so bad: $+1$.
- Take blue connected components together into small/big.
- Choose red that is $\leq \lfloor \frac{n}{4} \rfloor$.

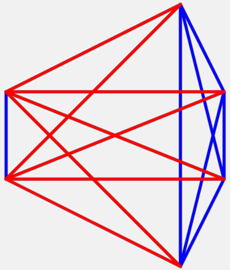


D Funny or Scary

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

- Incorrect blue edges are bad: direct to $n-1$.
- Incorrect red edges are not so bad: $+1$.
- Take blue connected components together into small/big.
- Choose red that is $\leq \lfloor \frac{n}{4} \rfloor$.
- Make small part of size $\lfloor \frac{n}{4} \rfloor$ or $\lfloor \frac{n}{4} \rfloor - 1$.



D Funny or Scary

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

- Why just $n \leq 24$, the solution is $O(n^2)$?

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

- Why just $n \leq 24$, the solution is $O(n^2)$?
- Checker is slow: $O(n^2 \cdot 2^n)$.

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

- Why just $n \leq 24$, the solution is $O(n^2)$?
- Checker is slow: $O(n^2 \cdot 2^n)$.
- Longest path heuristics are faster, but cannot find the longest paths on the cases we have here: an unbalanced bipartite graph plus a few edges.

D Funny or Scary

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

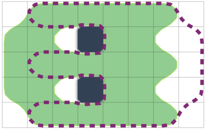
Solution

- Why just $n \leq 24$, the solution is $O(n^2)$?
- Checker is slow: $O(n^2 \cdot 2^n)$.
- Longest path heuristics are faster, but cannot find the longest paths on the cases we have here: an unbalanced bipartite graph plus a few edges.
- $n \leq 24$ allows to skip finding connected components, just try 2^n options for the small part.

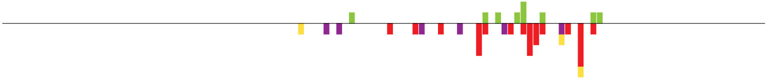
J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Number of **submissions**: 39
of which **accepted**: 9 ($\sim$ 23%)



First solved by **Jagiellonian teapots (Jagiellonian University in Krakow)** after 2h 18m



J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

The problem

Transform amoeba body from a given initial position to a given final position by moving pixels one-by-one while keeping the body connected at all times.

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

The problem

Transform amoeba body from a given initial position to a given final position by moving pixels one-by-one while keeping the body connected at all times.

Solution

- Build two trees, one spanning the initial position, one for the final one

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

The problem

Transform amoeba body from a given initial position to a given final position by moving pixels one-by-one while keeping the body connected at all times.

Solution

- Build two trees, one spanning the initial position, one for the final one
- Remove pixels from the first tree, bottom-up from leaves to its root

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

The problem

Transform amoeba body from a given initial position to a given final position by moving pixels one-by-one while keeping the body connected at all times.

Solution

- Build two trees, one spanning the initial position, one for the final one
- Remove pixels from the first tree, bottom-up from leaves to its root
- Add those pixels to the second tree, top-down from the root to leaves

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

The problem

Transform amoeba body from a given initial position to a given final position by moving pixels one-by-one while keeping the body connected at all times.

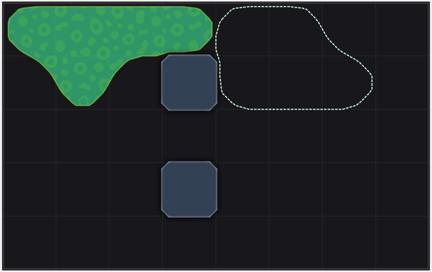
Solution

- Build two trees, one spanning the initial position, one for the final one
- Remove pixels from the first tree, bottom-up from leaves to its root
- Add those pixels to the second tree, top-down from the root to leaves
- Resolve the cases where those two trees overlap

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution



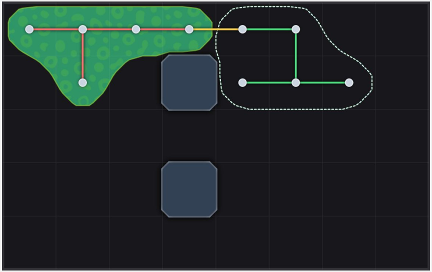
The easiest case:

- Remove pixels from the first tree, bottom-up from leaves to its root
- Add those pixels to the second tree, top-down from the root to leaves

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution



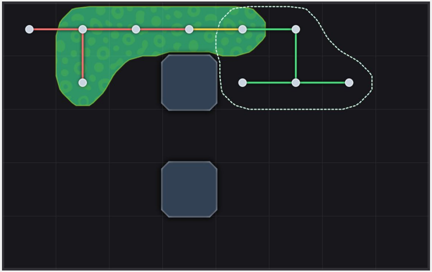
The easiest case:

- Remove pixels from the first tree, bottom-up from leaves to its root
- Add those pixels to the second tree, top-down from the root to leaves

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

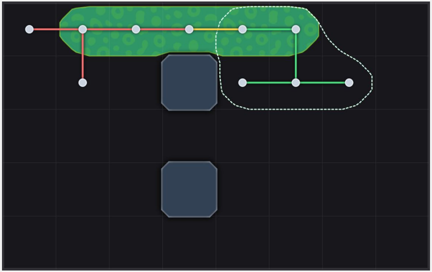
Solution



The easiest case:

- Remove pixels from the first tree, bottom-up from leaves to its root
- Add those pixels to the second tree, top-down from the root to leaves

Solution

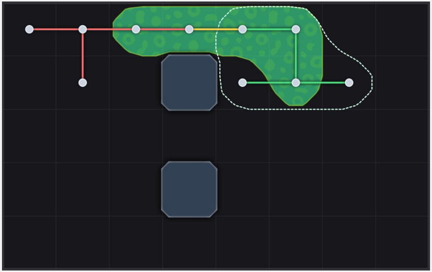


The easiest case:

- Remove pixels from the first tree, bottom-up from leaves to its root
- Add those pixels to the second tree, top-down from the root to leaves

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution

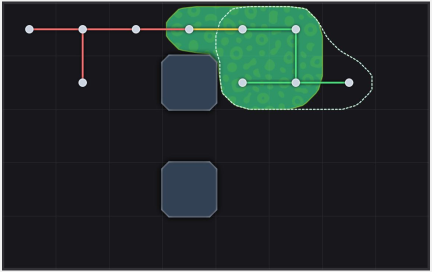


The easiest case:

- Remove pixels from the first tree, bottom-up from leaves to its root
- Add those pixels to the second tree, top-down from the root to leaves

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution

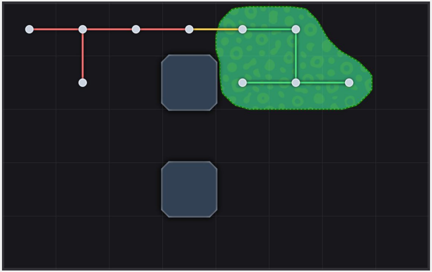


The easiest case:

- Remove pixels from the first tree, bottom-up from leaves to its root
- Add those pixels to the second tree, top-down from the root to leaves

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution

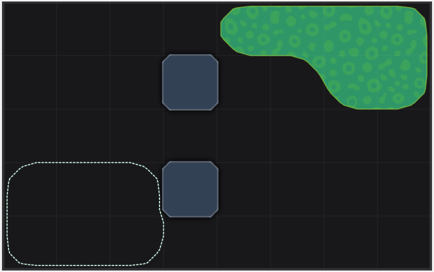


The easiest case:

- Remove pixels from the first tree, bottom-up from leaves to its root
- Add those pixels to the second tree, top-down from the root to leaves

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution



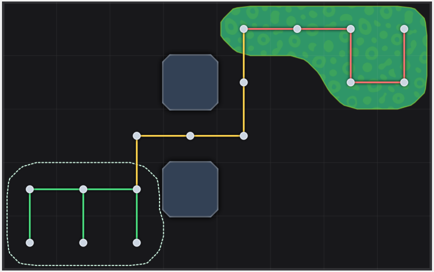
What if the positions are further apart?

- Find a path between the two positions
- Fill the path first and the proceed to the final position

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution

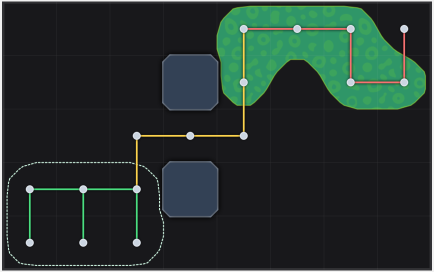


What if the positions are further apart?

- Find a path between the two positions
- Fill the path first and the proceed to the final position

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution



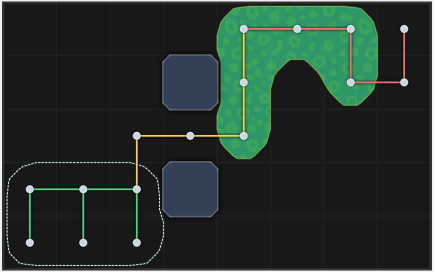
What if the positions are further apart?

- Find a path between the two positions
- Fill the path first and the proceed to the final position

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

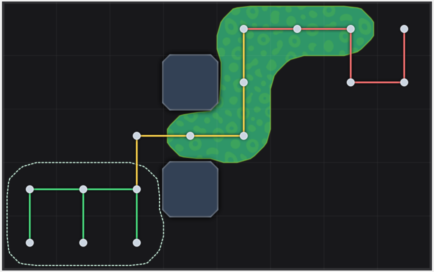
Solution



What if the positions are further apart?

- Find a path between the two positions
- Fill the path first and the proceed to the final position

Solution



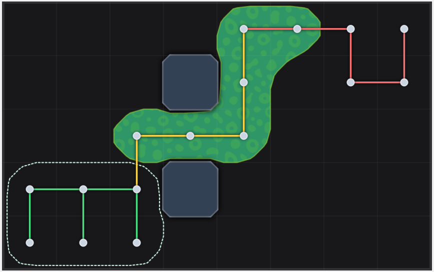
What if the positions are further apart?

- Find a path between the two positions
- Fill the path first and the proceed to the final position

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution



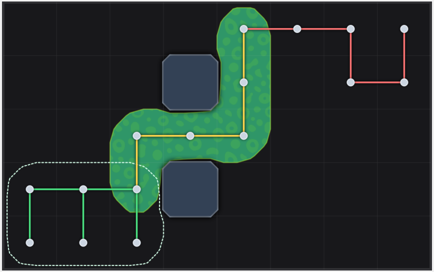
What if the positions are further apart?

- Find a path between the two positions
- Fill the path first and the proceed to the final position

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution

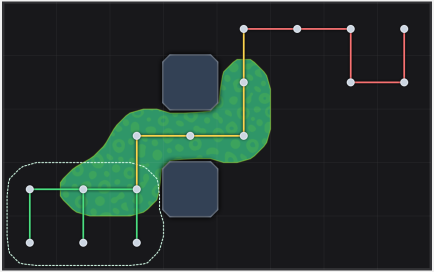


What if the positions are further apart?

- Find a path between the two positions
- Fill the path first and the proceed to the final position

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution



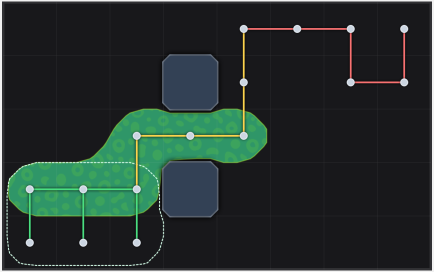
What if the positions are further apart?

- Find a path between the two positions
- Fill the path first and the proceed to the final position

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution

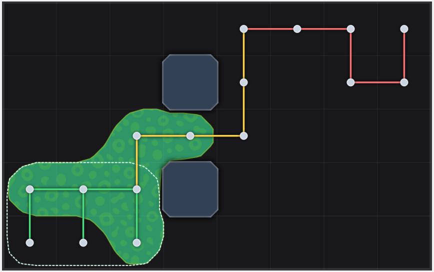


What if the positions are further apart?

- Find a path between the two positions
- Fill the path first and the proceed to the final position

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution

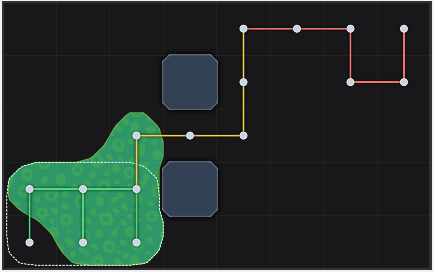


What if the positions are further apart?

- Find a path between the two positions
- Fill the path first and the proceed to the final position

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution

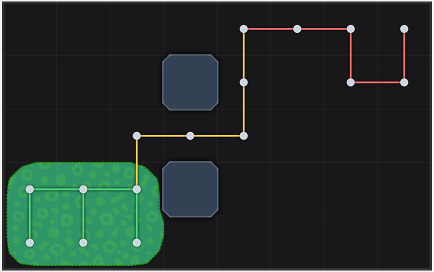


What if the positions are further apart?

- Find a path between the two positions
- Fill the path first and the proceed to the final position

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

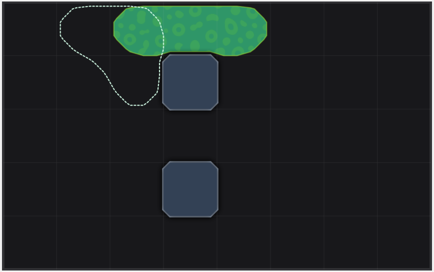
Solution



What if the positions are further apart?

- Find a path between the two positions
- Fill the path first and the proceed to the final position

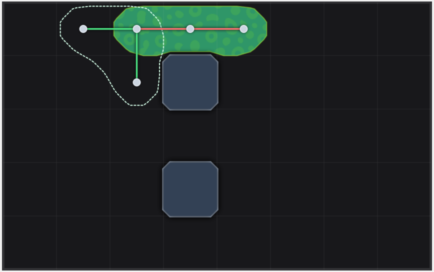
Solution



What if the positions overlap? (for start, in one pixel only)

- The shared pixel stays
- Other than that, the same procedure is used

Solution

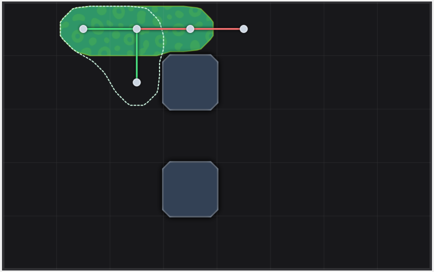


What if the positions overlap? (for start, in one pixel only)

- The shared pixel stays
- Other than that, the same procedure is used

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution

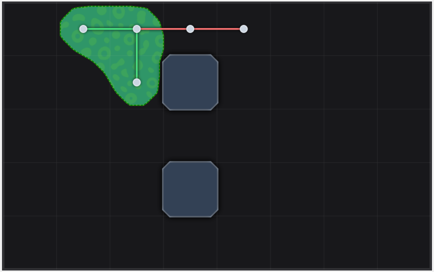


What if the positions overlap? (for start, in one pixel only)

- The shared pixel stays
- Other than that, the same procedure is used

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution

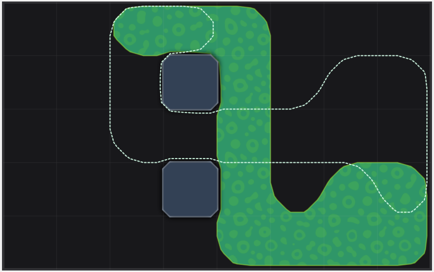


What if the positions overlap? (for start, in one pixel only)

- The shared pixel stays
- Other than that, the same procedure is used

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution

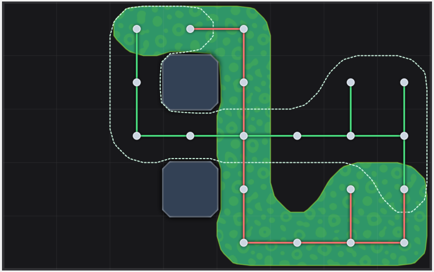


If the positions overlap in more places:

- Both trees are rooted in the same pixel
- The principle of the algorithm remains the same

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

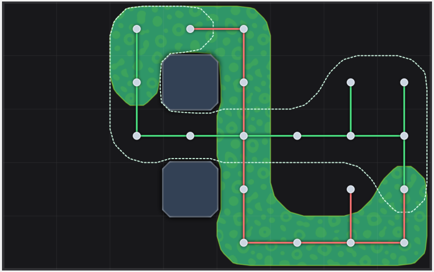
Solution



If the positions overlap in more places:

- Both trees are rooted in the same pixel
- The principle of the algorithm remains the same

Solution

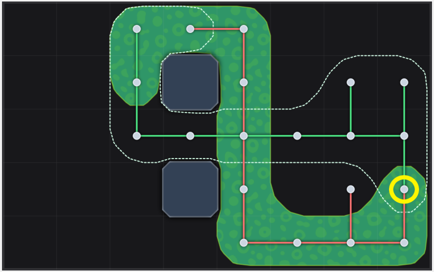


If the positions overlap in more places:

- Both trees are rooted in the same pixel
- The principle of the algorithm remains the same

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

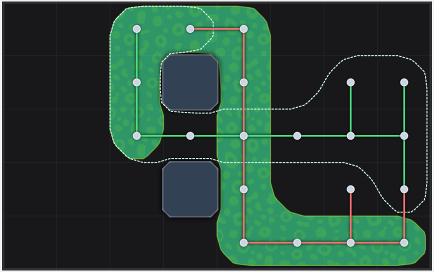
Solution



If the positions overlap in more places:

- If some pixel is to be removed although it belongs to the final position, it must be removed anyway, to keep the body connected (the same pixel will be added back later)

Solution



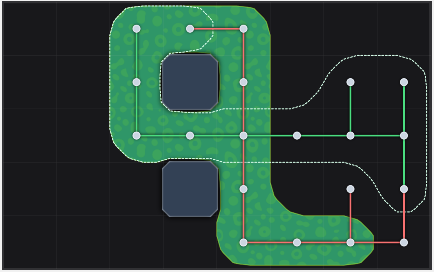
If the positions overlap in more places:

- If some pixel is to be removed although it belongs to the final position, it must be removed anyway, to keep the body connected
(the same pixel will be added back later)

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution

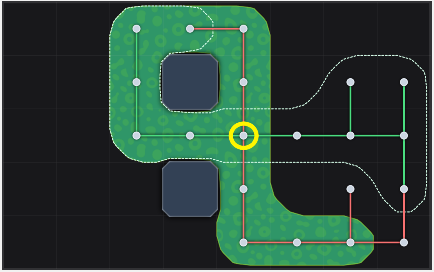


If the positions overlap in more places:

- If some pixel is to be removed although it belongs to the final position, it must be removed anyway, to keep the body connected
  (the same pixel will be added back later)

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

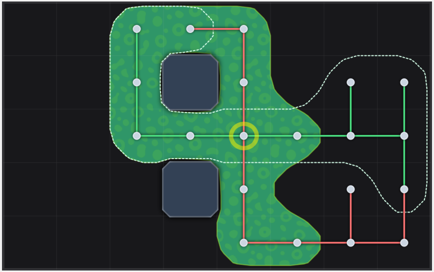
Solution



If the positions overlap in more places:

- If a pixel should be added, but it is already occupied (not removed yet), it is skipped and the algorithm continues with the next pixel
- The respective pixel is marked to not be removed anymore

Solution

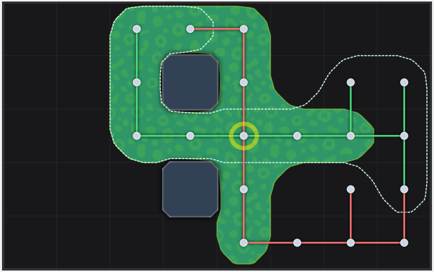


If the positions overlap in more places:

- If a pixel should be added, but it is already occupied (not removed yet), it is skipped and the algorithm continues with the next pixel
- The respective pixel is marked to not be removed anymore

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution



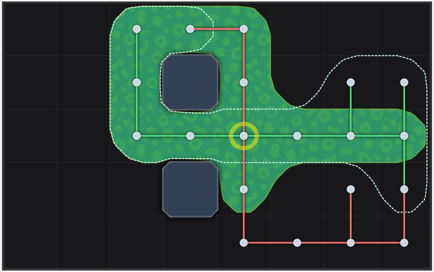
If the positions overlap in more places:

- If a pixel should be added, but it is already occupied (not removed yet), it is skipped and the algorithm continues with the next pixel
- The respective pixel is marked to not be removed anymore

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution

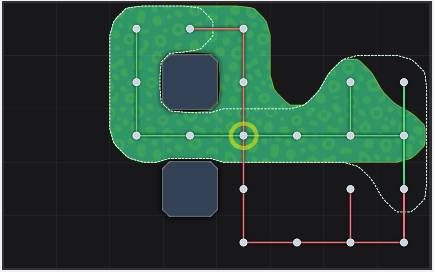


If the positions overlap in more places:

- If a pixel should be added, but it is already occupied (not removed yet), it is skipped and the algorithm continues with the next pixel
- The respective pixel is marked to not be removed anymore

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution



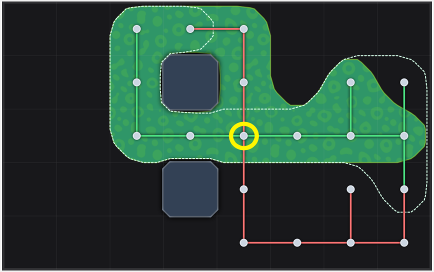
If the positions overlap in more places:

- If a pixel should be added, but it is already occupied (not removed yet), it is skipped and the algorithm continues with the next pixel
- The respective pixel is marked to not be removed anymore

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

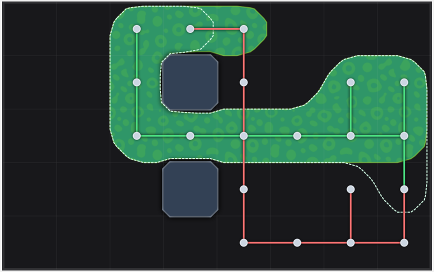
Solution



If the positions overlap in more places:

- The respective pixel is marked to not be removed anymore
- When the removal comes to such a pixel, it is skipped

Solution



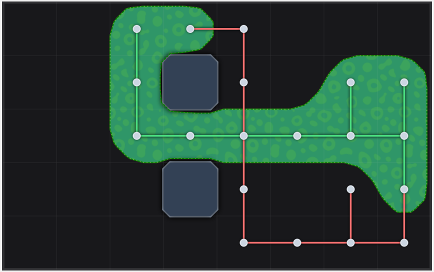
If the positions overlap in more places:

- The respective pixel is marked to not be removed anymore
- When the removal comes to such a pixel, it is skipped

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

Solution



If the positions overlap in more places:

- The respective pixel is marked to not be removed anymore
- When the removal comes to such a pixel, it is skipped

J Amanda the Amoeba

AUTHORED BY: Lucian Bicsi PREPARED BY: Martin Kacer

The problem

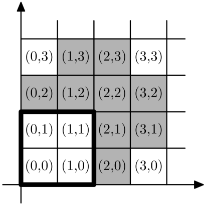
Transform amoeba body from a given initial position to a given final position by moving pixels one-by-one while keeping the body connected at all times.

Solution Summary

1. Build two trees of the two positions, with a path between their roots (the path may possibly be empty)
2. Removal order: the initial tree bottom-up, then the path between
3. Adding order: the path between, then the final tree top-down
4. Consecutively remove pixels one-by-one and add them, in the given order
5. If a pixel should be added but it is already part of the body, it is skipped and must be marked to *not* be removed later

H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev



Number of **submissions**: 21
of which **accepted**: 0 ($\sim$ 0%)

First solved by **N/A** after N/A

H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

The problem

We maintain a set of alive cells on the grid. In one operation, we replace $(a, b) \to (a+1, b) + (a, b+1)$. Initially just $(0, 0)$ is alive. Given a set of n forbidden cells, can we get to a state where none of them are alive? We are not allowed to have two copies of the same cell.

H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

The problem

We maintain a set of alive cells on the grid. In one operation, we replace $(a, b) \to (a+1, b) + (a, b+1)$. Initially just $(0, 0)$ is alive. Given a set of n forbidden cells, can we get to a state where none of them are alive? We are not allowed to have two copies of the same cell.

Solution

- We can allow multiple copies of the same cell temporarily, but not in final state.

H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

The problem

We maintain a set of alive cells on the grid. In one operation, we replace $(a, b) \to (a+1, b) + (a, b+1)$. Initially just $(0, 0)$ is alive. Given a set of n forbidden cells, can we get to a state where none of them are alive? We are not allowed to have two copies of the same cell.

Solution

- We can allow multiple copies of the same cell temporarily, but not in final state.
- Given a sequence of operations that has multiple copies of the same cell temporarily, we can reorder to get rid of the multiple copies.

H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

The problem

We maintain a set of alive cells on the grid. In one operation, we replace $(a, b) \to (a+1, b) + (a, b+1)$. Initially just $(0, 0)$ is alive. Given a set of n forbidden cells, can we get to a state where none of them are alive? We are not allowed to have two copies of the same cell.

Solution

- We can allow multiple copies of the same cell temporarily, but not in final state.
- Given a sequence of operations that has multiple copies of the same cell temporarily, we can reorder to get rid of the multiple copies.
- Now the order of operations does not matter!

H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

The problem

We maintain a set of alive cells on the grid. In one operation, we replace $(a, b) \to (a+1, b) + (a, b+1)$. Initially just $(0, 0)$ is alive. Given a set of n forbidden cells, can we get to a state where none of them are alive? We are not allowed to have two copies of the same cell.

Solution

- We can allow multiple copies of the same cell temporarily, but not in final state.
- Given a sequence of operations that has multiple copies of the same cell temporarily, we can reorder to get rid of the multiple copies.
- Now the order of operations does not matter!
- The operations we need to do are uniquely determined, we just need to execute them fast.

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

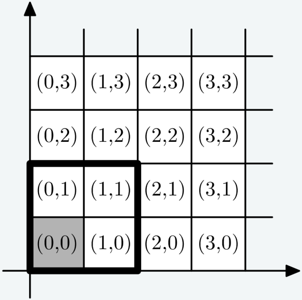
- Go by diagonals $a + b = z$.
  $d_{a+b,a}$ — the number of occurrences of (a, b).

H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

- Go by diagonals $a + b = z$. $d_{a+b,a}$ — the number of occurrences of (a, b).
- In the first sample:
- $d_0 = (1)$

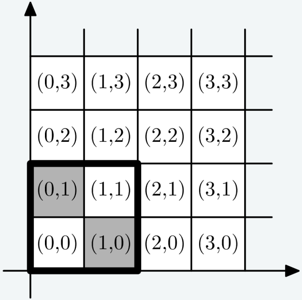


H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

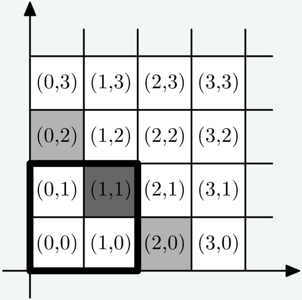
Solution

- Go by diagonals $a + b = z$. $d_{a+b,a}$ — the number of occurrences of (a, b).
- In the first sample:
- $d_0 = (1)$
- $d_1 = (1, 1)$



Solution

- Go by diagonals $a + b = z$. $d_{a+b,a}$ — the number of occurrences of (a, b).
- In the first sample:
- $d_0 = (1)$
- $d_1 = (1, 1)$
- $d_2 = (1, 2, 1)$

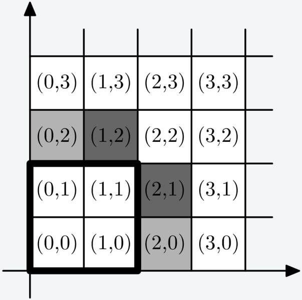


H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

- Go by diagonals $a + b = z$. $d_{a+b,a}$ — the number of occurrences of (a, b).
- In the first sample:
- $d_0 = (1)$
- $d_1 = (1, 1)$
- $d_2 = (1, 2, 1)$
- $d_3 = (0, 2, 2, 0)$

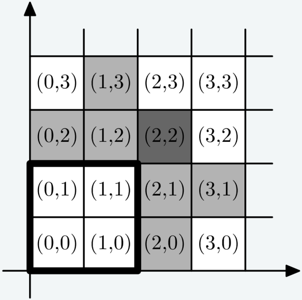


H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

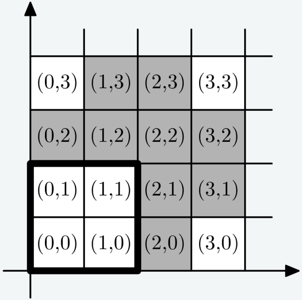
- Go by diagonals $a + b = z$. $d_{a+b,a}$ — the number of occurrences of (a, b).
- In the first sample:
- $d_0 = (1)$
- $d_1 = (1, 1)$
- $d_2 = (1, 2, 1)$
- $d_3 = (0, 2, 2, 0)$
- $d_4 = (0, 1, 2, 1, 0)$



AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

- Go by diagonals $a + b = z$. $d_{a+b,a}$ — the number of occurrences of (a, b).
- In the first sample:
- $d_0 = (1)$
- $d_1 = (1, 1)$
- $d_2 = (1, 2, 1)$
- $d_3 = (0, 2, 2, 0)$
- $d_4 = (0, 1, 2, 1, 0)$
- $d_5 = (0, 0, 1, 1, 0, 0)$

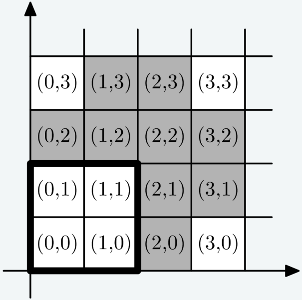


H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

- Go by diagonals $a + b = z$. $d_{a+b,a}$ — the number of occurrences of (a, b).
- In the first sample:
- $d_0 = (1)$
- $d_1 = (1, 1)$
- $d_2 = (1, 2, 1)$
- $d_3 = (0, 2, 2, 0)$
- $d_4 = (0, 1, 2, 1, 0)$
- $d_5 = (0, 0, 1, 1, 0, 0)$
- $d_6 = (0, 0, 0, 0, 0, 0, 0)$



H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

- If we get a 3, one of the adjacent numbers is a 2, and we never terminate: $3, 2 \to 2, 3 \to \ldots$.

Solution

- If we get a 3, one of the adjacent numbers is a 2, and we never terminate: $3, 2 \to 2, 3 \to \dots$.
- If we only have 0, 1, 2, and there are no forbidden cells, we will eventually terminate, for example:
  $1, 2, 2, 2, 1 \to 1, 2, 2, 1 \to 1, 2, 1 \to 1, 1 \to$.

H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

- If we get a 3, one of the adjacent numbers is a 2, and we never terminate: $3, 2 \to 2, 3 \to \dots$.
- If we only have 0, 1, 2, and there are no forbidden cells, we will eventually terminate, for example:
$1, 2, 2, 2, 1 \to 1, 2, 2, 1 \to 1, 2, 1 \to 1, 1 \to$.
- We have only $O(n)$ steps because without forbidden cells the number of 2s decreases, so this solution is $O(n^2)$.

H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

- If we get a 3, one of the adjacent numbers is a 2, and we never terminate: $3, 2 \to 2, 3 \to \dots$.
- If we only have 0, 1, 2, and there are no forbidden cells, we will eventually terminate, for example:
  $1, 2, 2, 2, 1 \to 1, 2, 2, 1 \to 1, 2, 1 \to 1, 1 \to$.
- We have only $O(n)$ steps because without forbidden cells the number of 2s decreases, so this solution is $O(n^2)$.
- To make it $O(n)$, we can notice that we always have either $(0, 0, \dots, 0, 1, 2, 2, \dots, 2, 1, 0, 0, \dots, 0)$ or $(0, 0, \dots, 0, 2, 2, 0, 0, \dots, 0)$.

H Division Avoidance

AUTHORED BY: Petr Mitrichev PREPARED BY: Petr Mitrichev

Solution

- If we get a 3, one of the adjacent numbers is a 2, and we never terminate: $3, 2 \to 2, 3 \to \dots$.
- If we only have 0, 1, 2, and there are no forbidden cells, we will eventually terminate, for example: $1, 2, 2, 2, 1 \to 1, 2, 2, 1 \to 1, 2, 1 \to 1, 1 \to$.
- We have only $O(n)$ steps because without forbidden cells the number of 2s decreases, so this solution is $O(n^2)$.
- To make it $O(n)$, we can notice that we always have either $(0, 0, \dots, 0, 1, 2, 2, \dots, 2, 1, 0, 0, \dots, 0)$ or $(0, 0, \dots, 0, 2, 2, 0, 0, \dots, 0)$.
- So we just need to maintain two integers and one boolean per diagonal.

E Damage per Second

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Number of **submissions**: 15
of which **accepted**: 0 ($\sim$ 0%)

First solved by **N/A** after N/A

E Damage per Second

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

The problem

You are facing n enemies with $h_1, \ldots, h_n$ health. You have k skill points each can be used to to either increase your *damage per hit* or your *hits per second* by one. How should you distribute the skill points to minimize the time needed to kill all enemies.

E Damage per Second

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

The problem

You are facing n enemies with $h_1, \ldots, h_n$ health. You have k skill points each can be used to to either increase your *damage per hit* or your *hits per second* by one. How should you distribute the skill points to minimize the time needed to kill all enemies.

Formal problem

- Given WLOG $h_1 \geq h_2 \geq \cdots \geq h_n$.
- Let $H = h_1 + \cdots + h_n$.
- Let $f(x) = \sum_{i=1}^{n} \left\lceil \frac{h_i}{x} \right\rceil$.
- Let $g(x) = \frac{f(x)}{k-x}$.
- Find the minimum of $g(x)$ in $[0, k]$.

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Observation 1: the "most interesting" values of x?

- Look at $g'(x) = \frac{H}{x(k-x)}$.

Observation 1: the "most interesting" values of x?

- Look at $g'(x) = \frac{H}{x(k-x)}$.
- Notice that $g(x) \geq g'(x)$.

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Observation 1: the "most interesting" values of x?

- Look at $g'(x) = \frac{H}{x(k-x)}$.
- Notice that $g(x) \geq g'(x)$.
- The minimum of $g'(x)$ is $\frac{k}{2}$.

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Observation 1: the "most interesting" values of x?

- Look at $g'(x) = \frac{H}{x(k-x)}$.
- Notice that $g(x) \geq g'(x)$.
- The minimum of $g'(x)$ is $\frac{k}{2}$.
- We expect the minimum of g to be close to $\frac{k}{2}$ as well.

E Damage per Second

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Observation 1: the "most interesting" values of x?

- Look at $g'(x) = \frac{H}{x(k-x)}$.
- Notice that $g(x) \geq g'(x)$.
- The minimum of $g'(x)$ is $\frac{k}{2}$.
- We expect the minimum of g to be close to $\frac{k}{2}$ as well.
- True if there were no good testcases...

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Observation 2: Speeding up the computation of g

- Iterate from $i = 1$ to n until $i < \frac{h_i}{x} \cdot \log(n)$.

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Observation 2: Speeding up the computation of g

- Iterate from $i = 1$ to n until $i < \frac{h_i}{x} \cdot \log(n)$.
- Let q_x be the first value of i where the inequality fails.

Observation 2: Speeding up the computation of g

- Iterate from $i = 1$ to n until $i < \frac{h_i}{x} \cdot \log(n)$.
- Let q_x be the first value of i where the inequality fails.
- For $q_x \leq i \leq n$ the quantity $\lceil \frac{h_i}{x} \rceil$ takes at most $\frac{h_{q_x}}{x}$ different values.

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Observation 2: Speeding up the computation of g

- Iterate from $i = 1$ to n until $i < \frac{h_i}{x} \cdot \log(n)$.
- Let q_x be the first value of i where the inequality fails.
- For $q_x \leq i \leq n$ the quantity $\lceil \frac{h_i}{x} \rceil$ takes at most $\frac{h_{q_x}}{x}$ different values.
- Then $\sum_{i=q_x+1}^{n} \lceil \frac{h_i}{x} \rceil$, can be computed in $O\left(\frac{h_{q_x+1}}{x} \log n\right) = O(q_x)$ with binary search.

E Damage per Second

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Observation 2: Speeding up the computation of g

- Iterate from $i = 1$ to n until $i < \frac{h_i}{x} \cdot \log(n)$.
- Let q_x be the first value of i where the inequality fails.
- For $q_x \leq i \leq n$ the quantity $\lceil \frac{h_i}{x} \rceil$ takes at most $\frac{h_{q_x}}{x}$ different values.
- Then $\sum_{i=q_x+1}^{n} \lceil \frac{h_i}{x} \rceil$, can be computed in $O\left(\frac{h_{q_x+1}}{x} \log n\right) = O(q_x)$ with binary search.
- Obviously, $\sum_{i=1}^{q_x-1} \lceil \frac{h_i}{x} \rceil$ can also be calculated in $O(q_x)$.

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Observation 2: Speeding up the computation of g

- Iterate from $i = 1$ to n until $i < \frac{h_i}{x} \cdot \log(n)$.
- Let q_x be the first value of i where the inequality fails.
- For $q_x \leq i \leq n$ the quantity $\lceil \frac{h_i}{x} \rceil$ takes at most $\frac{h_{q_x}}{x}$ different values.
- Then $\sum_{i=q_x+1}^{n} \lceil \frac{h_i}{x} \rceil$, can be computed in $O\left(\frac{h_{q_x+1}}{x} \log n\right) = O(q_x)$ with binary search.
- Obviously, $\sum_{i=1}^{q_x-1} \lceil \frac{h_i}{x} \rceil$ can also be calculated in $O(q_x)$.
- We can compute $g(x)$ in $O(q_x)$.

E Damage per Second

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Solution

- Maintain the running minimum m of $g(x)$.

Solution

- Maintain the running minimum m of $g(x)$.
- Iterate over $\frac{k}{2} \pm 0, \frac{k}{2} - 1, \frac{k}{2} + 1, \ldots$

E Damage per Second

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Solution

- Maintain the running minimum m of $g(x)$.
- Iterate over $\frac{k}{2} \pm 0, \frac{k}{2} - 1, \frac{k}{2} + 1, \ldots$
- For each x, if $g'(x) \geq m$ skip.

E Damage per Second

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Solution

- Maintain the running minimum m of $g(x)$.
- Iterate over $\frac{k}{2} \pm 0, \frac{k}{2} - 1, \frac{k}{2} + 1, \ldots$
- For each x, if $g'(x) \geq m$ skip.
- Compute $g(x)$ in $O(q_x)$ and update m accordingly.

E Damage per Second

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Solution

- Maintain the running minimum m of $g(x)$.
- Iterate over $\frac{k}{2} \pm 0, \frac{k}{2} - 1, \frac{k}{2} + 1, \ldots$
- For each x, if $g'(x) \geq m$ skip.
- Compute $g(x)$ in $O(q_x)$ and update m accordingly.

This algorithm produces the correct answer, but is it provably fast enough?

E Damage per Second

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

Solution

- Maintain the running minimum m of $g(x)$.
- Iterate over $\frac{k}{2} \pm 0, \frac{k}{2} - 1, \frac{k}{2} + 1, \ldots$
- For each x, if $g'(x) \geq m$ skip.
- Compute $g(x)$ in $O(q_x)$ and update m accordingly.

This algorithm produces the correct answer, but is it provably fast enough? Unexpectedly yes!

E Damage per Second

AUTHORED BY: Michael Zündorf, Federico Glaudo PREPARED BY: Michael Zündorf

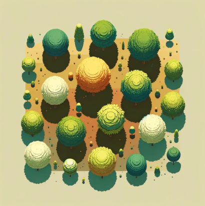
Solution

- Maintain the running minimum m of $g(x)$.
- Iterate over $\frac{k}{2} \pm 0, \frac{k}{2} - 1, \frac{k}{2} + 1, \ldots$
- For each x, if $g'(x) \geq m$ skip.
- Compute $g(x)$ in $O(q_x)$ and update m accordingly.

This algorithm produces the correct answer, but is it provably fast enough?
Unexpectedly yes!
This algorithm has the remarkable running time $O\left(k\sqrt{n\log(n)}\right)$.

A Grove

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini



Number of **submissions**: 8
of which **accepted**: 0 ($\sim$ 0%)

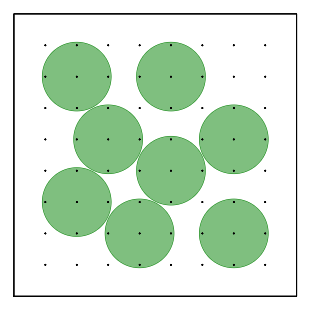
First solved by **N/A** after N/A

A Grove

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

The problem

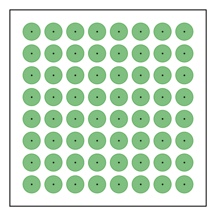
Plant as many trees as possible in an $n \times n$ lawn. Trees should be located at integer coordinates, and disks of radius r centered at these locations should not overlap.



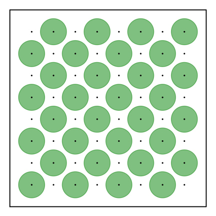
An optimal configuration for $n = 9, \; r = 1.1$

Solution

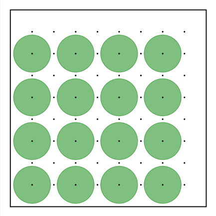
- Optimal configurations for small r can be constructed explicitly:



$r \in \left(0, \frac{1}{2}\right]$



$r \in \left(\frac{1}{2}, \frac{\sqrt{2}}{2}\right]$



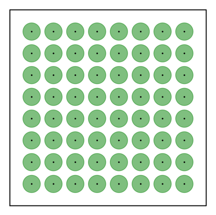
$r \in \left(\frac{\sqrt{2}}{2}, 1\right]$

A Grove

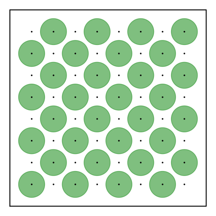
AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

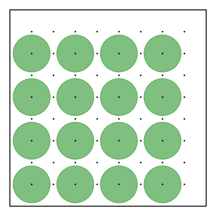
- Optimal configurations for small r can be constructed explicitly:



$r \in \left(0, \frac{1}{2}\right]$



$r \in \left(\frac{1}{2}, \frac{\sqrt{2}}{2}\right]$



$r \in \left(\frac{\sqrt{2}}{2}, 1\right]$

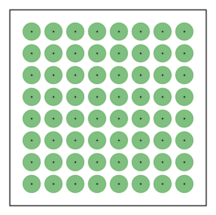
- In general, there is a finite number of intervals $(p, q]$ such that the valid configurations are the same for all radii $r \in (p, q]$.

A Grove

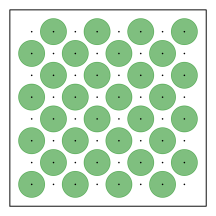
AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

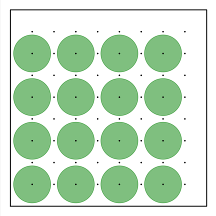
- Optimal configurations for small r can be constructed explicitly:



$r \in \left(0, \frac{1}{2}\right]$



$r \in \left(\frac{1}{2}, \frac{\sqrt{2}}{2}\right]$



$r \in \left(\frac{\sqrt{2}}{2}, 1\right]$

- In general, there is a finite number of intervals $(p, q]$ such that the valid configurations are the same for all radii $r \in (p, q]$.
- This makes it possible to pre-compute all optimal configurations offline (not necessary, but useful).

A Grove

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- Let $\delta = \lceil r \rceil$. All trees need to be planted at least δ away from the boundary of the lawn, so their coordinates must satisfy $\delta \le x, y \le n - \delta$.

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- Let $\delta = \lceil r \rceil$. All trees need to be planted at least δ away from the boundary of the lawn, so their coordinates must satisfy $\delta \leq x, y \leq n - \delta$.
- For integers $0 \leq a_\delta, \ldots, a_{n-\delta} \leq n - \delta$, denote by $f(a_\delta, \ldots, a_{n-\delta})$ any optimal configuration where we can only plant trees at locations (x, y) satisfying $x \leq a_y$.

A Grove

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- Let $\delta = \lceil r \rceil$. All trees need to be planted at least δ away from the boundary of the lawn, so their coordinates must satisfy $\delta \leq x, y \leq n - \delta$.
- For integers $0 \leq a_\delta, \ldots, a_{n-\delta} \leq n - \delta$, denote by $f(a_\delta, \ldots, a_{n-\delta})$ any optimal configuration where we can only plant trees at locations (x, y) satisfying $x \leq a_y$.
- Our task is to compute $f(n - \delta, \ldots, n - \delta)$.

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- Let $\delta = \lceil r \rceil$. All trees need to be planted at least δ away from the boundary of the lawn, so their coordinates must satisfy $\delta \leq x, y \leq n - \delta$.
- For integers $0 \leq a_\delta, \ldots, a_{n-\delta} \leq n - \delta$, denote by $f(a_\delta, \ldots, a_{n-\delta})$ any optimal configuration where we can only plant trees at locations (x, y) satisfying $x \leq a_y$.
- Our task is to compute $f(n - \delta, \ldots, n - \delta)$.
- We compute f recursively and store results of all recursive calls (memoization).

A Grove

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- Let $\delta = \lceil r \rceil$. All trees need to be planted at least δ away from the boundary of the lawn, so their coordinates must satisfy $\delta \leq x, y \leq n - \delta$.
- For integers $0 \leq a_\delta, \ldots, a_{n-\delta} \leq n - \delta$, denote by $f(a_\delta, \ldots, a_{n-\delta})$ any optimal configuration where we can only plant trees at locations (x, y) satisfying $x \leq a_y$.
- Our task is to compute $f(n - \delta, \ldots, n - \delta)$.
- We compute f recursively and store results of all recursive calls (memoization).
- If $a_1, \ldots, a_{n-1} < \delta$, then $f(a_1, \ldots, a_{n-1}) = \emptyset$, because any tree would be too close to the left boundary.

A Grove

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- To compute $f(a_1, \ldots, a_{n-1})$ in general, let $\bar{x} = \max\{a_1, \ldots, a_{n-1}\}$, and let $\bar{y}$ be such that $a_{\bar{y}} = \bar{x}$. This is a rightmost location where we can plant a tree.

A Grove

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- To compute $f(a_1, \ldots, a_{n-1})$ in general, let $\bar{x} = \max\{a_1, \ldots, a_{n-1}\}$, and let $\bar{y}$ be such that $a_{\bar{y}} = \bar{x}$. This is a rightmost location where we can plant a tree.
- If we do not plant a tree at $(\bar{x}, \bar{y})$, an optimal configuration is given by $f(a'_\delta, \ldots, a'_{n-\delta})$, where $a'_y = a_y$ if $y \neq \bar{y}$ and $a'_{\bar{y}} = a_{\bar{y}} - 1$.

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- To compute $f(a_1, \ldots, a_{n-1})$ in general, let $\bar{x} = \max\{a_1, \ldots, a_{n-1}\}$, and let $\bar{y}$ be such that $a_{\bar{y}} = \bar{x}$. This is a rightmost location where we can plant a tree.
- If we do not plant a tree at $(\bar{x}, \bar{y})$, an optimal configuration is given by $f(a'_\delta, \ldots, a'_{n-\delta})$, where $a'_y = a_y$ if $y \neq \bar{y}$ and $a'_{\bar{y}} = a_{\bar{y}} - 1$.
- If we plant a tree at $(\bar{x}, \bar{y})$, an optimal configuration is given by $f(a'_\delta, \ldots, a'_{n-\delta}) \cup \{(\bar{x}, \bar{y})\}$, where a'_y is the largest integer $\leq a_y$ such that locations (a'_y, y) and $(\bar{x}, \bar{y})$ are at distance $\geq 2r$.

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

- To compute $f(a_1, \ldots, a_{n-1})$ in general, let $\bar{x} = \max\{a_1, \ldots, a_{n-1}\}$, and let $\bar{y}$ be such that $a_{\bar{y}} = \bar{x}$. This is a rightmost location where we can plant a tree.
- If we do not plant a tree at $(\bar{x}, \bar{y})$, an optimal configuration is given by $f(a'_\delta, \ldots, a'_{n-\delta})$, where $a'_y = a_y$ if $y \neq \bar{y}$ and $a'_{\bar{y}} = a_{\bar{y}} - 1$.
- If we plant a tree at $(\bar{x}, \bar{y})$, an optimal configuration is given by $f(a'_\delta, \ldots, a'_{n-\delta}) \cup \{(\bar{x}, \bar{y})\}$, where a'_y is the largest integer $\leq a_y$ such that locations (a'_y, y) and $(\bar{x}, \bar{y})$ are at distance $\geq 2r$.
- Let $f(a_1, \ldots, a_{n-1})$ be the best of the two configurations found above.

AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

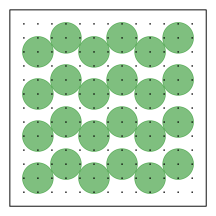
Solution

- The number of recursive calls is larger for small r, so it can be useful to solve by hand the small r cases (as shown at the beginning).

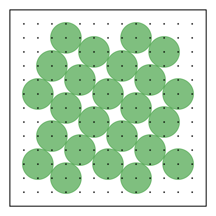
AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

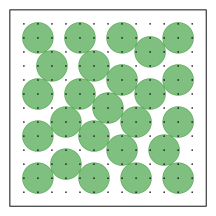
- The number of recursive calls is larger for small r, so it can be useful to solve by hand the small r cases (as shown at the beginning).
- **Achtung!** Already for $r \in \left(1, \frac{\sqrt{5}}{2}\right]$, the solution is not "obvious":



24 trees



24 trees



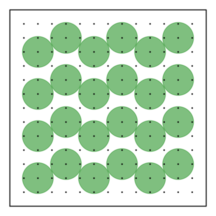
25 trees!

A Grove

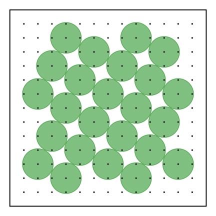
AUTHORED BY: Giovanni Paolini PREPARED BY: Giovanni Paolini

Solution

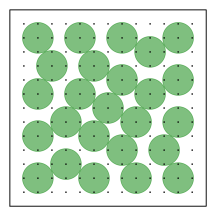
- The number of recursive calls is larger for small r, so it can be useful to solve by hand the small r cases (as shown at the beginning).
- **Achtung!** Already for $r \in \left(1, \frac{\sqrt{5}}{2}\right]$, the solution is not "obvious":



24 trees



24 trees



25 trees!

- Alternatively to the presented solution, one can use a generic max-clique algorithm and pre-compute all optimal configurations offline.