KTH Challenge 2020 Solutions

2020-04-25

G - Triple Texting

Proposed & prepared by Nils Gustafsson

Given string repeated three times with possible error, recover it.

G - Triple Texting

Proposed & prepared by Nils Gustafsson

Given string repeated three times with possible error, recover it.

`acceptedaxceptedaccepted`

G - Triple Texting

Proposed & prepared by Nils Gustafsson

Given string repeated three times with possible error, recover it.

`acceptedaxceptedaccepted`

1: split input into three strings

`accepted`

`axcepted`

`accepted`

G - Triple Texting

Proposed & prepared by Nils Gustafsson

Given string repeated three times with possible error, recover it.

`acceptedaxceptedaccepted`

2: sort the three strings

`accepted`

`accepted`

`axcepted`

G - Triple Texting

Proposed & prepared by Nils Gustafsson

Given string repeated three times with possible error, recover it.

`acceptedaxceptedaccepted`

3: output the middle one

`accepted`

`accepted`

`axcepted`

F - Proofs

Proposed & prepared by Joseph Swernofsky

Given assumptions used and conclusions in each step of a proof, check if correct.

F - Proofs

Proposed & prepared by Joseph Swernofsky

Given assumptions used and conclusions in each step of a proof, check if correct.

---

1. Keep dictionary of already proved conclusions.

F - Proofs

Proposed & prepared by Joseph Swernofsky

Given assumptions used and conclusions in each step of a proof, check if correct.

---

1. Keep dictionary of already proved conclusions.

2. For each line, look up assumptions in dictionary (if any missing, we found error), then add conclusion to dictionary

F - Proofs

Proposed & prepared by Joseph Swernofsky

Given assumptions used and conclusions in each step of a proof, check if correct.

---

1. Keep dictionary of already proved conclusions.

2. For each line, look up assumptions in dictionary (if any missing, we found error), then add conclusion to dictionary

Time complexity $O(n \cdot \text{dictionary\_lookup\_time})$

Make sure to use data structure with fast lookup time to avoid quadratic running time.

A - AI Jeopardy

Proposed & prepared by Per Austrin

Given $X \leq 10^{100}$, find smallest n and k such that $\binom{n}{k} = X$.

A - AI Jeopardy

Proposed & prepared by Per Austrin

Given $X \leq 10^{100}$, find smallest n and k such that $\binom{n}{k} = X$.

Idea: n could be huge, but k cannot.

A - AI Jeopardy

Proposed & prepared by Per Austrin

Given $X \leq 10^{100}$, find smallest n and k such that $\binom{n}{k} = X$.

Idea: n could be huge, but k cannot.

Smallest possible value of $\binom{n}{k}$ for a fixed k is $\binom{2k}{k} \approx 2^{2k}$.

A - AI Jeopardy

Proposed & prepared by Per Austrin

Given $X \leq 10^{100}$, find smallest n and k such that $\binom{n}{k} = X$.

Idea: n could be huge, but k cannot.

Smallest possible value of $\binom{n}{k}$ for a fixed k is $\binom{2k}{k} \approx 2^{2k}$.

So must have $k \leq O(\log X)$

A - AI Jeopardy

Proposed & prepared by Per Austrin

Given $X \leq 10^{100}$, find smallest n and k such that $\binom{n}{k} = X$.

Idea: n could be huge, but k cannot.

Smallest possible value of $\binom{n}{k}$ for a fixed k is $\binom{2k}{k} \approx 2^{2k}$.

So must have $k \leq O(\log X)$

Algorithm: try all possible values of k, and for each value binary search for n.

A - AI Jeopardy

Proposed & prepared by Per Austrin

Given $X \leq 10^{100}$, find smallest n and k such that $\binom{n}{k} = X$.

Idea: n could be huge, but k cannot.

Smallest possible value of $\binom{n}{k}$ for a fixed k is $\binom{2k}{k} \approx 2^{2k}$.

So must have $k \leq O(\log X)$

Algorithm: try all possible values of k, and for each value binary search for n.

Time complexity is $O(\log(X)^4)$.

A - AI Jeopardy

Proposed & prepared by Per Austrin

Given $X \leq 10^{100}$, find smallest n and k such that $\binom{n}{k} = X$.

Idea: n could be huge, but k cannot.

Smallest possible value of $\binom{n}{k}$ for a fixed k is $\binom{2k}{k} \approx 2^{2k}$.

So must have $k \leq O(\log X)$

Algorithm: try all possible values of k, and for each value binary search for n.

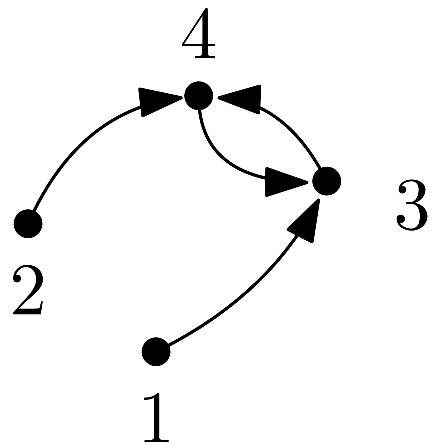
Time complexity is $O(\log(X)^4)$.

(because computing a single binomial coefficient $\binom{n}{k}$ takes $O(k \log(n)) = O(\log(X)^2)$ time)

D - Gaggle

Proposed & prepared by Per Austrin

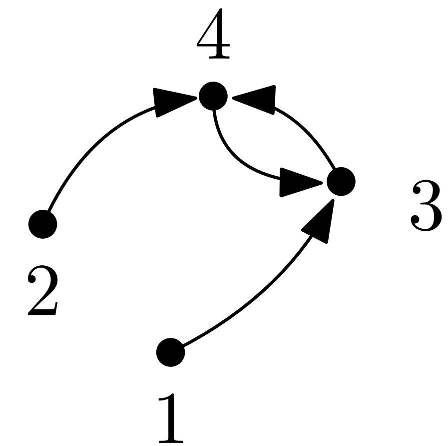
Transform function graph into a cycle, try to preserve out-edges from lower-numbered vertices.



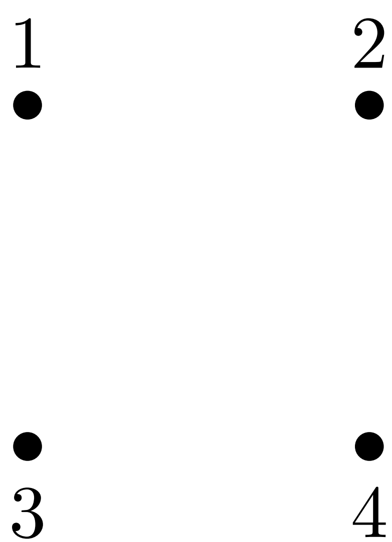
D - Gaggle

Proposed & prepared by Per Austrin

Transform function graph into a cycle, try to preserve out-edges from lower-numbered vertices.



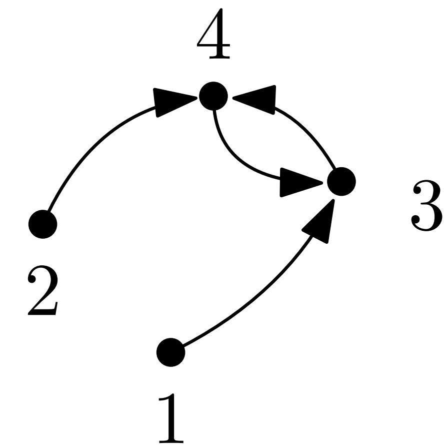
For each node, add best possible outgoing edge as long as indegrees ≤ 1 and no cycles formed (until last node)



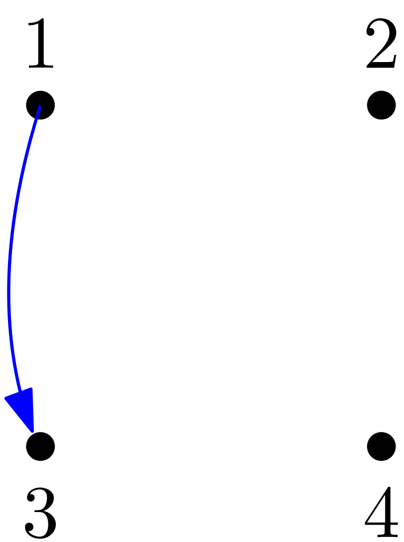
D - Gaggle

Proposed & prepared by Per Austrin

Transform function graph into a cycle, try to preserve out-edges from lower-numbered vertices.



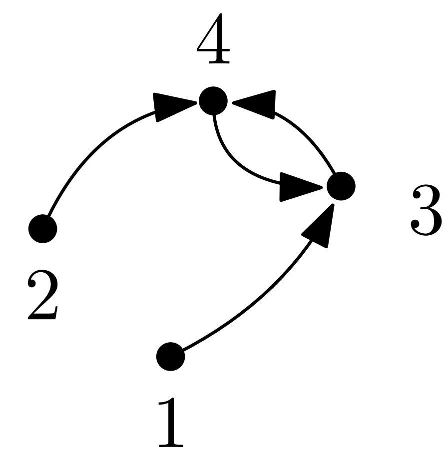
For each node, add best possible outgoing edge as long as indegrees ≤ 1 and no cycles formed (until last node)



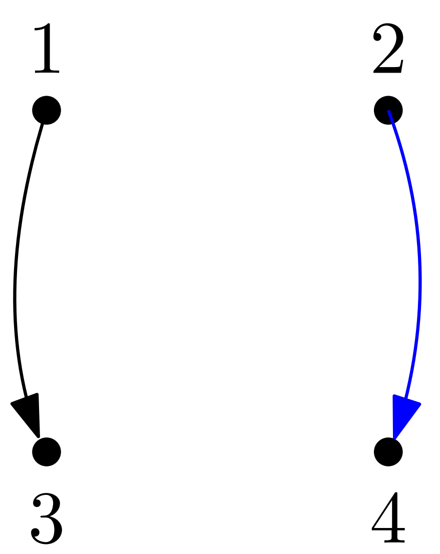
D - Gaggle

Proposed & prepared by Per Austrin

Transform function graph into a cycle, try to preserve out-edges from lower-numbered vertices.



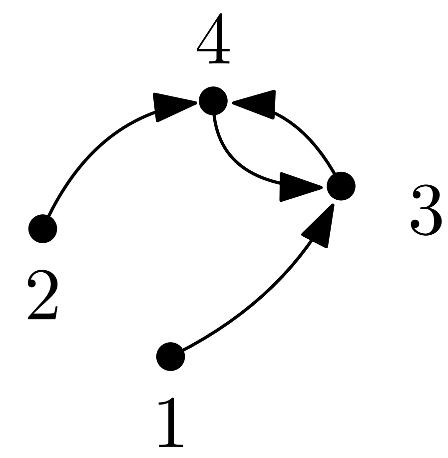
For each node, add best possible outgoing edge as long as indegrees ≤ 1 and no cycles formed (until last node)



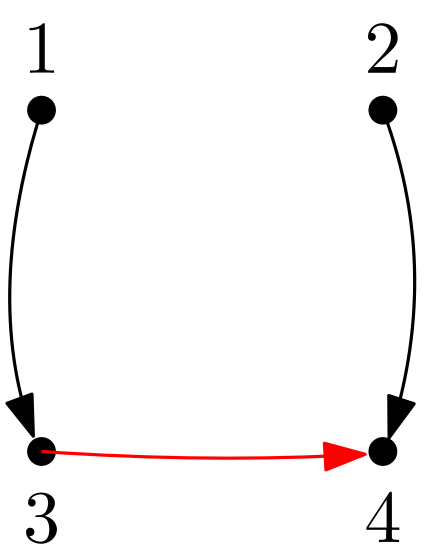
D - Gaggle

Proposed & prepared by Per Austrin

Transform function graph into a cycle, try to preserve out-edges from lower-numbered vertices.



For each node, add best possible outgoing edge as long as indegrees ≤ 1 and no cycles formed (until last node)

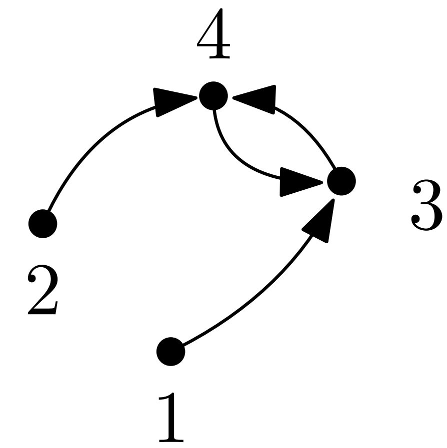


3 prefers pointing to 4, but 4 is taken so this is invalid

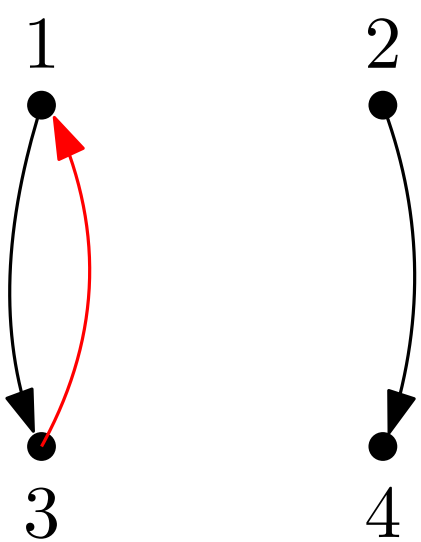
D - Gaggle

Proposed & prepared by Per Austrin

Transform function graph into a cycle, try to preserve out-edges from lower-numbered vertices.



For each node, add best possible outgoing edge as long as indegrees ≤ 1 and no cycles formed (until last node)

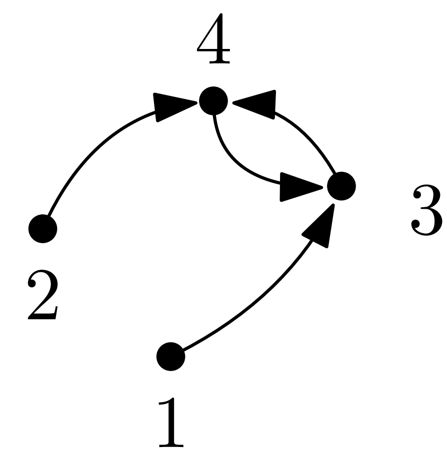


next best choice would be 1, but creates cycle so also invalid

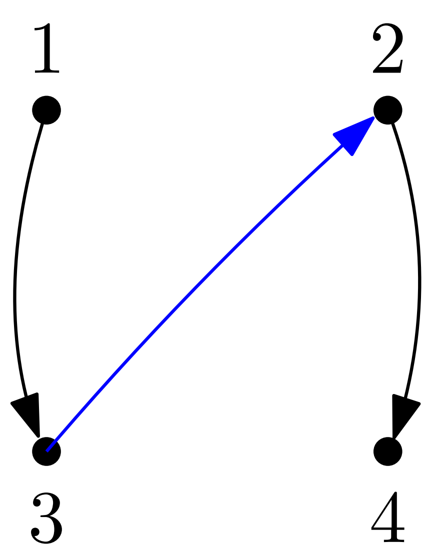
D - Gaggle

Proposed & prepared by Per Austrin

Transform function graph into a cycle, try to preserve out-edges from lower-numbered vertices.



For each node, add best possible outgoing edge as long as indegrees ≤ 1 and no cycles formed (until last node)

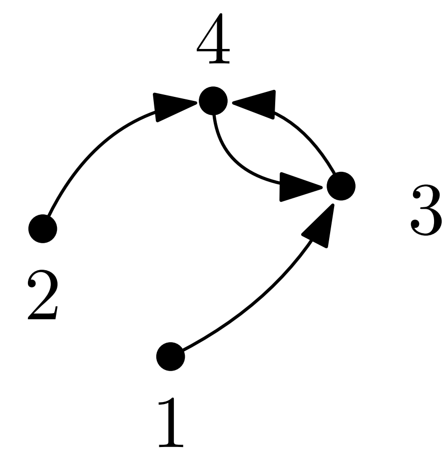


next best choice is 2, this is valid

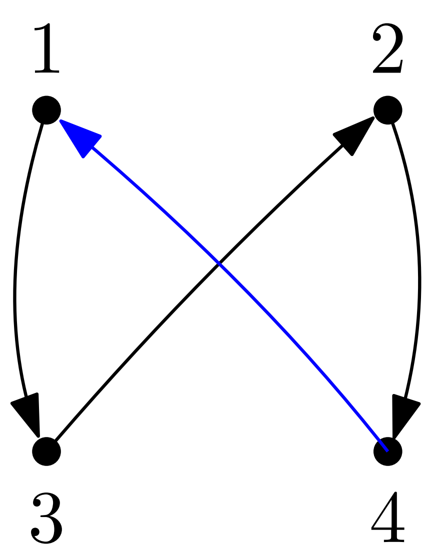
D - Gaggle

Proposed & prepared by Per Austrin

Transform function graph into a cycle, try to preserve out-edges from lower-numbered vertices.



For each node, add best possible outgoing edge as long as indegrees ≤ 1 and no cycles formed (until last node)

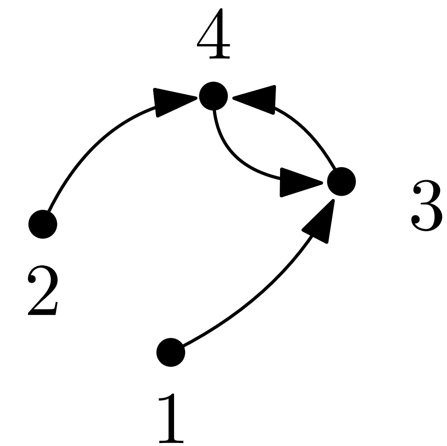


when at last step, we have built a long path, tie it up into a cycle

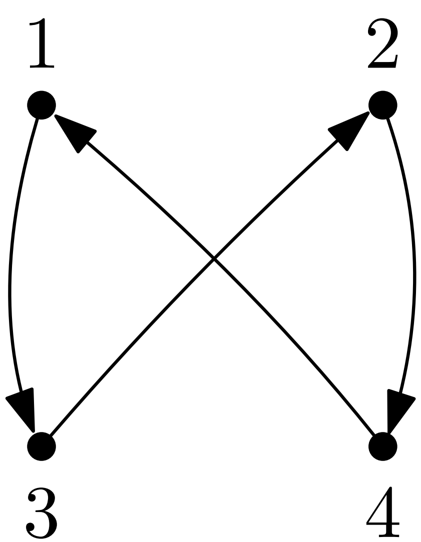
D - Gaggle

Proposed & prepared by Per Austrin

Transform function graph into a cycle, try to preserve out-edges from lower-numbered vertices.



For each node, add best possible outgoing edge as long as indegrees ≤ 1 and no cycles formed (until last node)



In general throughout the algorithm the partial solution is a set of paths.

Keep track of which endpoints are connected to each other to avoid creating cycles.

Can be done in $O(n)$.

E - Pitch Performance

Proposed by Johan Sannemo & prepared by Per Austrin

Given piecewise-constant function f and piecewise-quadratic function g, compute $\int_{x=0}^{T} |f(x) - g(x)| \mathrm{d}x$

E - Pitch Performance

Proposed by Johan Sannemo & prepared by Per Austrin

Given piecewise-constant function f and piecewise-quadratic function g, compute $\int_{x=0}^{T} |f(x) - g(x)| \mathrm{d}x$

Separate the calculation into intervals so that $f(x)$ is constant and $g(x)$ quadratic on each interval.

E - Pitch Performance

Proposed by Johan Sannemo & prepared by Per Austrin

Given piecewise-constant function f and piecewise-quadratic function g, compute $\int_{x=0}^{T} |f(x) - g(x)| \mathrm{d}x$

Separate the calculation into intervals so that $f(x)$ is constant and $g(x)$ quadratic on each interval.

For each interval:
If $f(x)$ crosses $g(x)$ in the interval (happens at most twice), subdivide further based on these crossing points

E - Pitch Performance

Proposed by Johan Sannemo & prepared by Per Austrin

Given piecewise-constant function f and piecewise-quadratic function g, compute $\int_{x=0}^{T} |f(x) - g(x)| \mathrm{d}x$

Separate the calculation into intervals so that $f(x)$ is constant and $g(x)$ quadratic on each interval.

For each interval:
If $f(x)$ crosses $g(x)$ in the interval (happens at most twice), subdivide further based on these crossing points

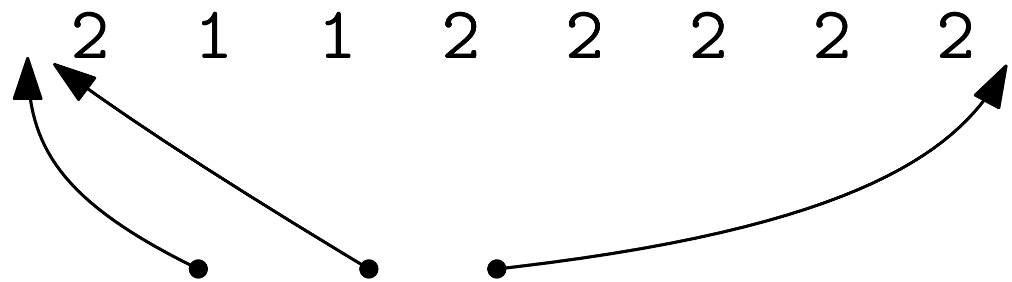
Now just need to integrate a quadratic function on each interval, use standard formula

H - Winning the Vote

Proposed & prepared by Nils Gustafsson

We get sequence of 1s and 2s, and some positions where we count who is in the lead

A count gives $+1$, 0, or -1 points. What is minimum distance counters need to be moved so that total score is positive?



H - Winning the Vote

Proposed & prepared by Nils Gustafsson

We get sequence of 1s and 2s, and some positions where we count who is in the lead

A count gives $+1$, 0, or -1 points. What is minimum distance counters need to be moved so that total score is positive?

$$\boxed{0}\ 2\ \boxed{-1}\ 1\ \boxed{0}\ 1\ \boxed{+1}\ 2\ \boxed{0}\ 2\ \boxed{-1}\ 2\ \boxed{-1}\ 2\ \boxed{-1}\ 2\ \boxed{-1}$$

Initial score $= 0 + 0 + (-1)$

Need to get 2 more points.

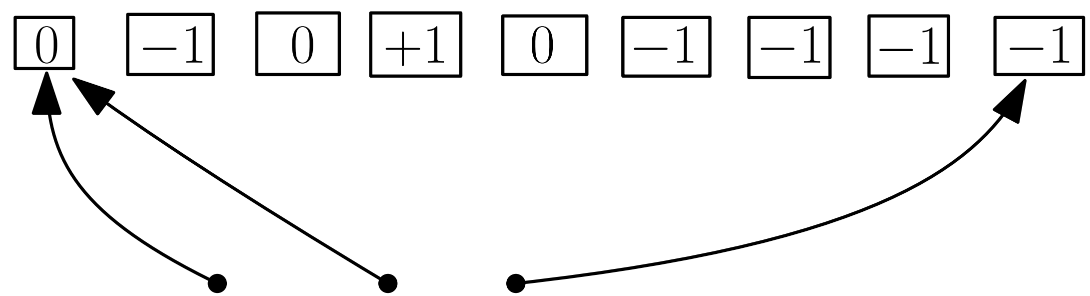
1. Compute the point value for each position.

H - Winning the Vote

Proposed & prepared by Nils Gustafsson

We get sequence of 1s and 2s, and some positions where we count who is in the lead

A count gives $+1$, 0, or -1 points. What is minimum distance counters need to be moved so that total score is positive?



Cost to increase by 1:	3	3	4
Cost to increase by 2:	∞	∞	5

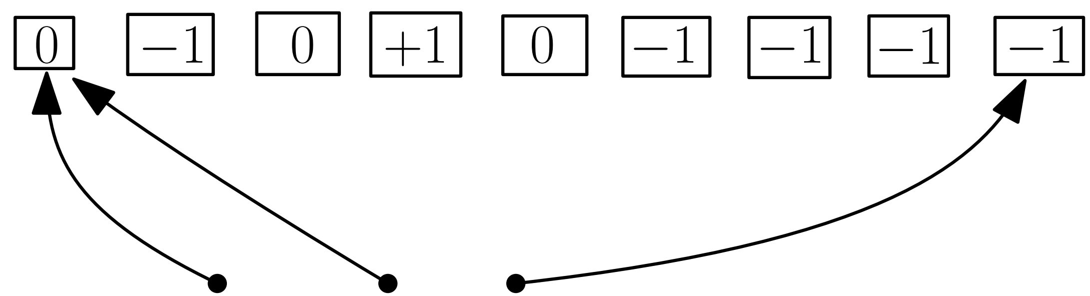
2. Compute how far each counter would have to move to get $+1$ or $+2$ additional score

H - Winning the Vote

Proposed & prepared by Nils Gustafsson

We get sequence of 1s and 2s, and some positions where we count who is in the lead

A count gives $+1$, 0, or -1 points. What is minimum distance counters need to be moved so that total score is positive?



Cost to increase by 1:	3	3	4
Cost to increase by 2:	∞	∞	5

3. Find cheapest way of getting the needed total increase using dynamic programming: "what is minimum cost to get total score increase x using only the first i counters?"

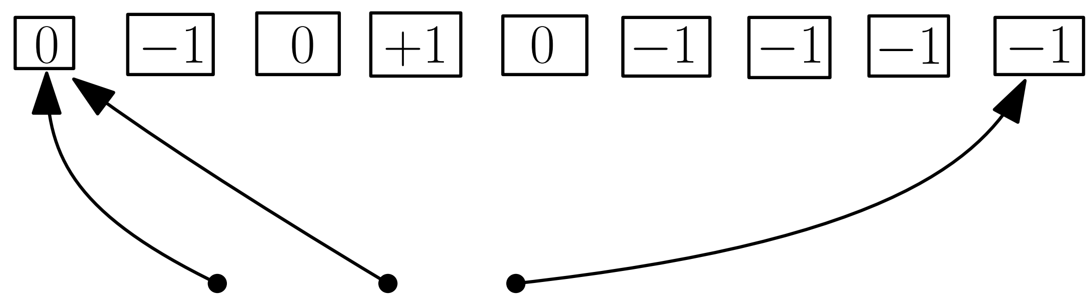
Time complexity $O(\#\text{counters} \cdot \#\text{voters}) = O(n^2)$

H - Winning the Vote

Proposed & prepared by Nils Gustafsson

We get sequence of 1s and 2s, and some positions where we count who is in the lead

A count gives $+1$, 0, or -1 points. What is minimum distance counters need to be moved so that total score is positive?



Cost to increase by 1:	3	3	4
Cost to increase by 2:	∞	∞	5

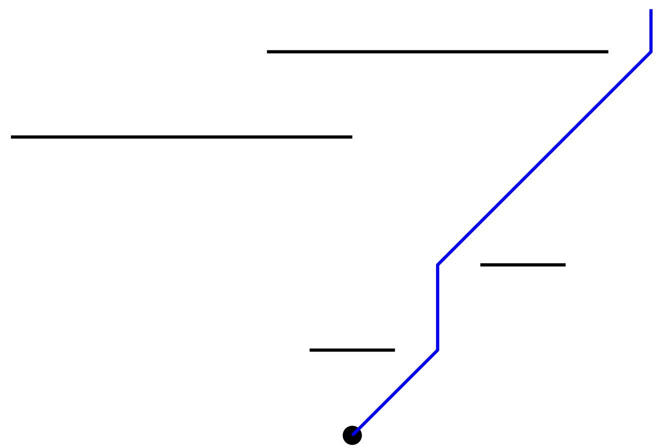
4. Bonus challenge: there is also a greedy strategy for the last step, gives an $O(n \log n)$ solution

C - Friendly Fire

Proposed & prepared by Nils Gustafsson

Find path from origin avoiding some lines.

Each step movesupward but can choose straight or diagonal left/right

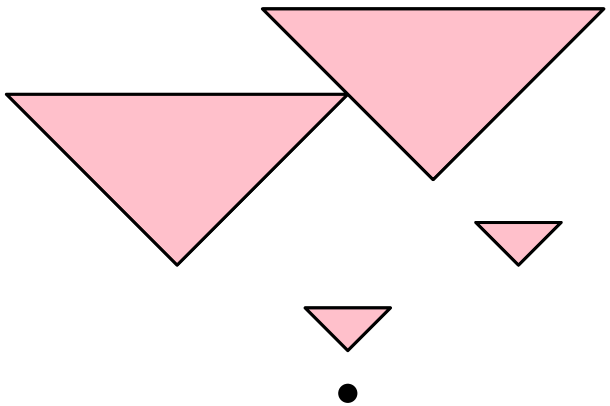


C - Friendly Fire

Proposed & prepared by Nils Gustafsson

Find path from origin avoiding some lines.

Each step movesupward but can choose straight or diagonal left/right



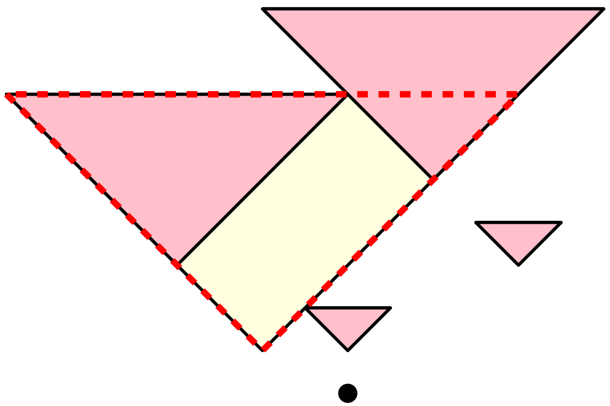
1. Each line gives rise to a triangle-shaped bad area
(If we go into the bad area we cannot avoid the line.)

C - Friendly Fire

Proposed & prepared by Nils Gustafsson

Find path from origin avoiding some lines.

Each step movesupward but can choose straight or diagonal left/right



2. When two triangles touch, the pocket formed is also bad.

We get a larger bad triangle.

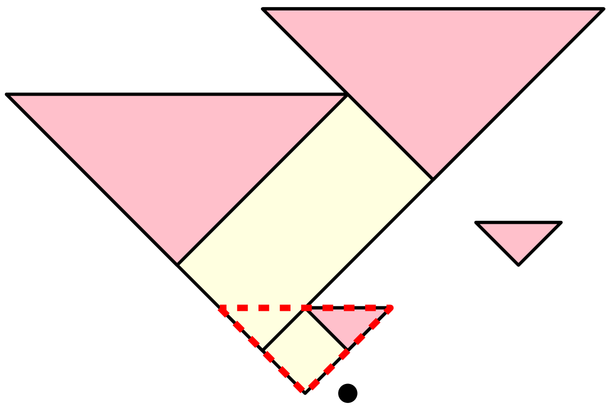
The new triangle could touch others and this repeats.

C - Friendly Fire

Proposed & prepared by Nils Gustafsson

Find path from origin avoiding some lines.

Each step movesupward but can choose straight or diagonal left/right



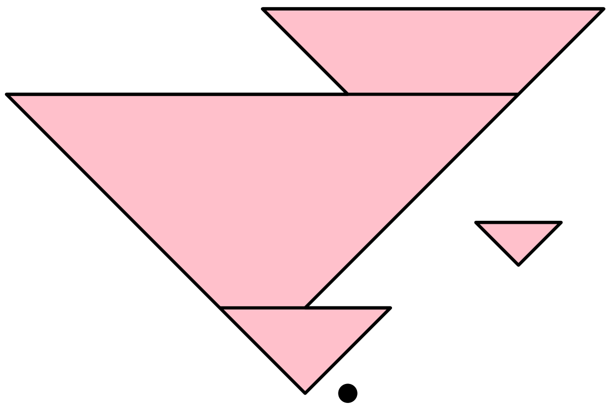
2. When two triangles touch, the pocket formed is also bad.
We get a larger bad triangle.
The new triangle could touch others and this repeats.

C - Friendly Fire

Proposed & prepared by Nils Gustafsson

Find path from origin avoiding some lines.

Each step movesupward but can choose straight or diagonal left/right

Final bad region: 

Given these extended triangles we can sweep upwards

Keeping a set of the currently active triangles we can efficiently check if an x-coordinate is bad or safe.

This enables us to construct the path in $O(n \log n)$

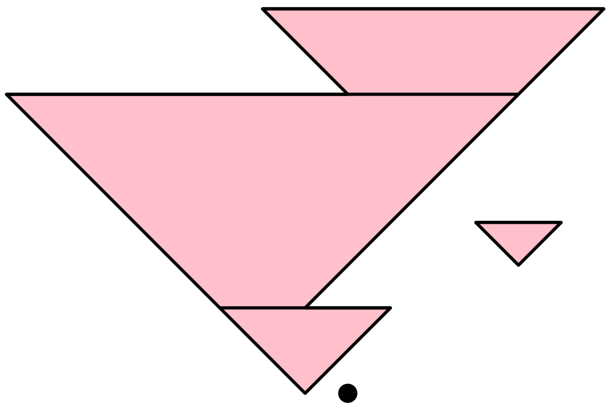
C - Friendly Fire

Proposed & prepared by Nils Gustafsson

Find path from origin avoiding some lines.

Each step movesupward but can choose straight or diagonal left/right

Final bad region:



Finding the extended triangles can be done by downwards sweep in $O(n \log n)$.

Again we keep set of active triangles.

When we add new one, check if it touches an existing one, and if so extend it.

B - Bling

Proposed by Johan Sannemo & prepared by Per Austrin

Find an optimal way of making money from fruits in game that is not Animal Crossing

B - Bling

Proposed by Johan Sannemo & prepared by Per Austrin

Find an optimal way of making money from fruits in game that is not Animal Crossing

---

Main difficulty in problem:

Most of the time we want to plant fruits into trees because this yields many more fruits and things grow exponentially.

But sometimes we want to instead sell a few fruits in order to be able to afford an exotic fruit in one of the next few days.

B - Bling

Proposed by Johan Sannemo & prepared by Per Austrin

Find an optimal way of making money from fruits in game that is not Animal Crossing

---

Main difficulty in problem:

Most of the time we want to plant fruits into trees because this yields many more fruits and things grow exponentially.

But sometimes we want to instead sell a few fruits in order to be able to afford an exotic fruit in one of the next few days.

This makes a greedy approach likely to fail (we do not know any greedy strategy that works)

B - Bling

Proposed by Johan Sannemo & prepared by Per Austrin

Find an optimal way of making money from fruits in game that is not Animal Crossing

---

When greedy fails, we look to dynamic programming for hope.

B - Bling

Proposed by Johan Sannemo & prepared by Per Austrin

Find an optimal way of making money from fruits in game that is not Animal Crossing

---

When greedy fails, we look to dynamic programming for hope.

Issue: we cannot use the entire state of the farm as our DP state because the number of trees etc will grow exponentially and there will be too many states.

B - Bling

Proposed by Johan Sannemo & prepared by Per Austrin

Find an optimal way of making money from fruits in game that is not Animal Crossing

---

When greedy fails, we look to dynamic programming for hope.

Issue: we cannot use the entire state of the farm as our DP state because the number of trees etc will grow exponentially and there will be too many states.

Solution: when a parameter has become large enough, we do not care about its exact value anymore.

B - Bling

Proposed by Johan Sannemo & prepared by Per Austrin

Find an optimal way of making money from fruits in game that is not Animal Crossing

---

When greedy fails, we look to dynamic programming for hope.

Issue: we cannot use the entire state of the farm as our DP state because the number of trees etc will grow exponentially and there will be too many states.

Solution: when a parameter has become large enough, we do not care about its exact value anymore.

For instance, 12 fruits are enough to buy exotic fruits in all the next three days.

So our DP state only needs to keep 13 possible values for how many fruits we have (we have 0, 1, $\ldots$, or 11 fruits, or we have ≥ 12 fruits)

B - Bling

Proposed by Johan Sannemo & prepared by Per Austrin

Find an optimal way of making money from fruits in game that is not Animal Crossing

---

Carefully thinking the DP state through for each parameter one ends up with fewer than 10^6 states

B - Bling

Proposed by Johan Sannemo & prepared by Per Austrin

Find an optimal way of making money from fruits in game that is not Animal Crossing

---

Carefully thinking the DP state through for each parameter one ends up with fewer than 10^6 states

For each state there are only a small constant number of options (at most 20ish)

B - Bling

Proposed by Johan Sannemo & prepared by Per Austrin

Find an optimal way of making money from fruits in game that is not Animal Crossing

---

Carefully thinking the DP state through for each parameter one ends up with fewer than 10^6 states

For each state there are only a small constant number of options (at most 20ish)

Leads to a fast algorithm.

(Large span of possible amounts of pruning and optimization that can be done on state.)