GAPC 2023

Solutions presentation

May 7, 2023

D: Discrete Structures

Problem Author: Wietze Koops and Franciszek Szewczyk

- **Problem:** Compute the final grade from the sum of its components.

D: Discrete Structures

Problem Author: Wietze Koops and Franciszek Szewczyk

- **Problem:** Compute the final grade from the sum of its components.
- Two best essays are each multiplied by 0.15

D: Discrete Structures

Problem Author: Wietze Koops and Franciszek Szewczyk

- **Problem:** Compute the final grade from the sum of its components.
- Two best essays are each multiplied by 0.15
- The midterm is multiplied by 0.2

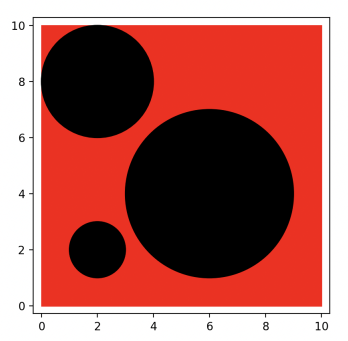
D: Discrete Structures

Problem Author: Wietze Koops and Franciszek Szewczyk

- **Problem:** Compute the final grade from the sum of its components.
- Two best essays are each multiplied by 0.15
- The midterm is multiplied by 0.2
- The final exam is multiplied by 0.5

A: A Rod in a Dot

Problem Author: Franciszek Szewczyk



- **Problem:** What's the expected number of hit coasters?

A: A Rod in a Dot

Problem Author: Franciszek Szewczyk

- We know that Majki's throws are uniformly distributed on the board.

A: A Rod in a Dot

Problem Author: Franciszek Szewczyk

- We know that Majki's throws are uniformly distributed on the board.
- Then probability of hitting a coaster $P(hit)$ is just the fraction of the board covered by coasters.

A: A Rod in a Dot

Problem Author: Franciszek Szewczyk

- We know that Majki's throws are uniformly distributed on the board.
- Then probability of hitting a coaster $P(hit)$ is just the fraction of the board covered by coasters.
- Expected amount of hits is $P(hit) \cdot n$

A: A Rod in a Dot

Problem Author: Franciszek Szewczyk

- We know that Majki's throws are uniformly distributed on the board.
- Then probability of hitting a coaster $P(hit)$ is just the fraction of the board covered by coasters.
- Expected amount of hits is $P(hit) \cdot n$
- If you enjoy pretty symbols: $\frac{\sum_{i=1}^{n} \pi \cdot r_i^2}{s^2} \cdot n$

A: A Rod in a Dot

Problem Author: Franciszek Szewczyk

- We know that Majki's throws are uniformly distributed on the board.
- Then probability of hitting a coaster $P(hit)$ is just the fraction of the board covered by coasters.
- Expected amount of hits is $P(hit) \cdot n$
- If you enjoy pretty symbols: $\frac{\sum_{i=1}^{n} \pi \cdot r_i^2}{s^2} \cdot n$
- Watch out for overflows

I: International Interpolation

Problem Author: Michal Te#nar

- **Problem:** Replace "**#**" characters in a string to be (alphabetically) between the characters around. If there are two "**#**" in a row, output impossible.

I: International Interpolation

Problem Author: Michal Te#nar

- **Problem:** Replace "**#**" characters in a string to be (alphabetically) between the characters around. If there are two "**#**" in a row, output impossible.
- First linearly search the strings to see if there are two "**#**", if yes, print "impossible" and break.

I: International Interpolation

Problem Author: Michal Te#nar

- **Problem:** Replace "#" characters in a string to be (alphabetically) between the characters around. If there are two "#" in a row, output impossible.
- First linearly search the strings to see if there are two "#", if yes, print "impossible" and break.
- Resolve "#" at the start and the end of a word by replacing it by "a" or "z", to avoid accessing outside of the string.

I: International Interpolation

Problem Author: Michal Te#nar

- **Problem:** Replace "#" characters in a string to be (alphabetically) between the characters around. If there are two "#" in a row, output impossible.
- First linearly search the strings to see if there are two "#", if yes, print "impossible" and break.
- Resolve "#" at the start and the end of a word by replacing it by "a" or "z", to avoid accessing outside of the string.
- For the rest of the "#", convert the neighbors into integers ((`ord()`) in Python).

I: International Interpolation

Problem Author: Michal Te#nar

- **Problem:** Replace "#" characters in a string to be (alphabetically) between the characters around. If there are two "#" in a row, output impossible.
- First linearly search the strings to see if there are two "#", if yes, print "impossible" and break.
- Resolve "#" at the start and the end of a word by replacing it by "a" or "z", to avoid accessing outside of the string.
- For the rest of the "#", convert the neighbors into integers (`(ord())` in Python).
- Find the number between the two (and floor if not an integer).

I: International Interpolation

Problem Author: Michal Te#nar

- **Problem:** Replace "`#`" characters in a string to be (alphabetically) between the characters around. If there are two "`#`" in a row, output impossible.
- First linearly search the strings to see if there are two "`#`", if yes, print "impossible" and break.
- Resolve "`#`" at the start and the end of a word by replacing it by "a" or "z", to avoid accessing outside of the string.
- For the rest of the "`#`", convert the neighbors into integers (`(ord())` in Python).
- Find the number between the two (and floor if not an integer).
- Find the corresponding character in the alphabet (`chr()` in Python).

I: International Interpolation

Problem Author: Michal Te#nar

- **Problem:** Replace "#" characters in a string to be (alphabetically) between the characters around. If there are two "#" in a row, output impossible.
- First linearly search the strings to see if there are two "#", if yes, print "impossible" and break.
- Resolve "#" at the start and the end of a word by replacing it by "a" or "z", to avoid accessing outside of the string.
- For the rest of the "#", convert the neighbors into integers (`(ord())` in Python).
- Find the number between the two (and floor if not an integer).
- Find the corresponding character in the alphabet (`chr()` in Python).
- Replace "#" by the new character in the string (in Python immutable, convert to list first).

I: International Interpolation

Problem Author: Michal Te#nar

- **Problem:** Replace "`#`" characters in a string to be (alphabetically) between the characters around. If there are two "`#`" in a row, output impossible.
- First linearly search the strings to see if there are two "`#`", if yes, print "impossible" and break.
- Resolve "`#`" at the start and the end of a word by replacing it by "a" or "z", to avoid accessing outside of the string.
- For the rest of the "`#`", convert the neighbors into integers (`(ord())` in Python).
- Find the number between the two (and floor if not an integer).
- Find the corresponding character in the alphabet (`chr()` in Python).
- Replace "`#`" by the new character in the string (in Python immutable, convert to list first).
- *Clarification:* If length is 1 and the only character is "`#`", then the answer is "a".

H: Hasty Guesses

Problem Author: Maarten Sijm and Wojtek Trejter

- **Problem:** Find the second closest distinct number to a given target.

H: Hasty Guesses

Problem Author: Maarten Sijm and Wojtek Trejter

- **Problem:** Find the second closest distinct number to a given target.
- **Solution:**

H: Hasty Guesses

Problem Author: Maarten Sijm and Wojtek Trejter

- **Problem:** Find the second closest distinct number to a given target.
- **Solution:**
  - Keep track of the first and second closest numbers.

H: Hasty Guesses

Problem Author: Maarten Sijm and Wojtek Trejter

- **Problem:** Find the second closest distinct number to a given target.
- **Solution:**
  - Keep track of the first and second closest numbers.
  - Iterate throught the input, comparing the newly encountered number to the current first and second closest numbers

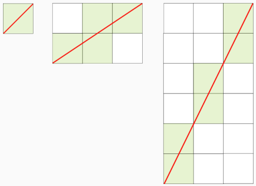
H: Hasty Guesses

Problem Author: Maarten Sijm and Wojtek Trejter

- **Problem:** Find the second closest distinct number to a given target.
- **Solution:**
  - Keep track of the first and second closest numbers.
  - Iterate throught the input, comparing the newly encountered number to the current first and second closest numbers
- **Slower solution:** Since $n \leq 10^5$, it is also possible to create a set of numbers from the input, sort it by the absolute value to the target number, and print the second number from the set.

F: Flatland Zoo

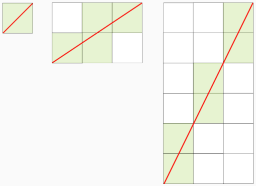
Problem Author: Anton Chernev and Vitor Greati



- **Problem:** Compute the number of square hits in a traversal of a $m \times n$ rectangular grid from the bottom-left corner to the top-right corner.

F: Flatland Zoo

Problem Author: Anton Chernev and Vitor Greati



- **Problem:** Compute the number of square hits in a traversal of a $m \times n$ rectangular grid from the bottom-left corner to the top-right corner.

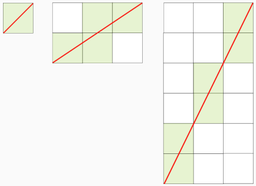
- *Solution:* The first square is always counted, and whenever Jimmy crosses a line of the grid, he enters a new square. There are $(m-1)+(n-1)$ lines to cross in this path, plus the first square, so $m+n-1$ in principle.

  However, when he crosses a vertex, he enters a new square, but then two lines will not be crossed.

  So, the answer is $m+n-1+N_{\text{vertices}}-2N_{\text{vertices}}=m+n-1-N_{\text{vertices}}$.

F: Flatland Zoo

Problem Author: Anton Chernev and Vitor Greati



- **Problem:** Compute the number of square hits in a traversal of a $m \times n$ rectangular grid from the bottom-left corner to the top-right corner.

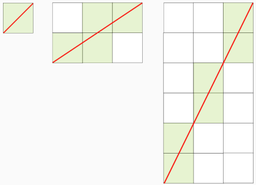
- *Solution:* The first square is always counted, and whenever Jimmy crosses a line of the grid, he enters a new square. There are $(m-1)+(n-1)$ lines to cross in this path, plus the first square, so $m+n-1$ in principle.

  However, when he crosses a vertex, he enters a new square, but then two lines will not be crossed.

  So, the answer is $m+n-1+N_{\text{vertices}}-2N_{\text{vertices}} = m+n-1-N_{\text{vertices}}$. But how to compute N_{vertices}?
  - A vertex is a pair of integers (x, y) such that $\frac{x}{y} = \frac{m}{n}$.
  - Note that $\frac{m}{n} = \frac{\gcd(m,n) \cdot m'}{\gcd(m,n) \cdot n'}$.
  - Thus, the vertices to cross are $(1 \cdot m', 1 \cdot n'), \ldots, ((\gcd(m,n)-1) \cdot m', (\gcd(m,n)-1) \cdot n')$.

F: Flatland Zoo

Problem Author: Anton Chernev and Vitor Greati



- **Problem:** Compute the number of square hits in a traversal of a $m \times n$ rectangular grid from the bottom-left corner to the top-right corner.

- *Solution:* The first square is always counted, and whenever Jimmy crosses a line of the grid, he enters a new square. There are $(m-1)+(n-1)$ lines to cross in this path, plus the first square, so $m+n-1$ in principle.

  However, when he crosses a vertex, he enters a new square, but then two lines will not be crossed.

  So, the answer is $m+n-1+N_{\text{vertices}}-2N_{\text{vertices}} = m+n-1-N_{\text{vertices}}$. But how to compute N_{vertices}?
  - A vertex is a pair of integers (x, y) such that $\frac{x}{y} = \frac{m}{n}$.
  - Note that $\frac{m}{n} = \frac{\gcd(m,n) \cdot m'}{\gcd(m,n) \cdot n'}$.
  - Thus, the vertices to cross are $(1 \cdot m', 1 \cdot n'), \ldots, ((\gcd(m,n)-1) \cdot m', (\gcd(m,n)-1) \cdot n')$.

  The number of square hits is, then, $m+n-1-N_{\text{vertices}} = m+n-\gcd(m,n)$.

  Then just output $m+n-\gcd(m,n)$. Complexity: $\mathcal{O}(\log(\min(m,n)))$.

J: Just in Time

Problem Author: Michal Tešnar

- **Problem**: Figure out how many overlaps are there between delayed orders.

J: Just in Time

Problem Author: Michal Tešnar

- **Problem**: Figure out how many overlaps are there between delayed orders.
- *Clarification:* The workers do not need to deliver the order.

J: Just in Time

Problem Author: Michal Tešnar

- **Problem**: Figure out how many overlaps are there between delayed orders.
- *Clarification:* The workers do not need to deliver the order.
- Treat each entry as two events: when order is started to be prepared, 1 more worker is needed, when order stops being prepared, 1 less worker is needed.

J: Just in Time

Problem Author: Michal Tešnar

- **Problem**: Figure out how many overlaps are there between delayed orders.
- *Clarification:* The workers do not need to deliver the order.
- Treat each entry as two events: when order is started to be prepared, 1 more worker is needed, when order stops being prepared, 1 less worker is needed.
- End of order has timestamp $a - c$, start of order has timestamp $a - b - c$.

J: Just in Time

Problem Author: Michal Tešnar

- **Problem**: Figure out how many overlaps are there between delayed orders.
- *Clarification:* The workers do not need to deliver the order.
- Treat each entry as two events: when order is started to be prepared, 1 more worker is needed, when order stops being prepared, 1 less worker is needed.
- End of order has timestamp $a - c$, start of order has timestamp $a - b - c$.
- Sort the events by time, linearly go through them and keep count:
  - $+1$ when order arrives,
  - -1 when order is finished.

J: Just in Time

Problem Author: Michal Tešnar

- **Problem**: Figure out how many overlaps are there between delayed orders.
- *Clarification:* The workers do not need to deliver the order.
- Treat each entry as two events: when order is started to be prepared, 1 more worker is needed, when order stops being prepared, 1 less worker is needed.
- End of order has timestamp $a - c$, start of order has timestamp $a - b - c$.
- Sort the events by time, linearly go through them and keep count:
  - $+1$ when order arrives,
  - -1 when order is finished.
- In the end, output maximum value of counter as the answer.

J: Just in Time

Problem Author: Michal Tešnar

- **Problem**: Figure out how many overlaps are there between delayed orders.
- *Clarification:* The workers do not need to deliver the order.
- Treat each entry as two events: when order is started to be prepared, 1 more worker is needed, when order stops being prepared, 1 less worker is needed.
- End of order has timestamp $a - c$, start of order has timestamp $a - b - c$.
- Sort the events by time, linearly go through them and keep count:
  - $+1$ when order arrives,
  - -1 when order is finished.
- In the end, output maximum value of counter as the answer.
- Complexity $\mathcal{O}(2n\log(2n) + 2n) = \mathcal{O}(n\log(n))$.

G: Gruesome CAPTCHAs

Problem Author: Anton Chernev and Wojtek Trejter

- **Problem:** Determine for q queries whether there is a graph centre with n nodes.

G: Gruesome CAPTCHAs

Problem Author: Anton Chernev and Wojtek Trejter

- **Problem:** Determine for q queries whether there is a graph centre with n nodes.
- Naive solution: For every node check whether it is connected to all other nodes. $\mathcal{O}(n^2 q)$ is too slow!

G: Gruesome CAPTCHAs

Problem Author: Anton Chernev and Wojtek Trejter

- **Problem:** Determine for q queries whether there is a graph centre with n nodes.
- Naive solution: For every node check whether it is connected to all other nodes. $\mathcal{O}(n^2 q)$ is too slow!
- Instead, we start with node 1, which is our centre candidate.

G: Gruesome CAPTCHAs

Problem Author: Anton Chernev and Wojtek Trejter

- **Problem:** Determine for q queries whether there is a graph centre with n nodes.
- Naive solution: For every node check whether it is connected to all other nodes. $\mathcal{O}(n^2q)$ is too slow!
- Instead, we start with node 1, which is our centre candidate.
- If 1 sees 2, then 1 is still a potential candidate and 2 is definitely not a center.

G: Gruesome CAPTCHAs

Problem Author: Anton Chernev and Wojtek Trejter

- **Problem:** Determine for q queries whether there is a graph centre with n nodes.
- Naive solution: For every node check whether it is connected to all other nodes. $\mathcal{O}(n^2 q)$ is too slow!
- Instead, we start with node 1, which is our centre candidate.
- If 1 sees 2, then 1 is still a potential candidate and 2 is definitely not a center.
- Otherwise, 1 is definitely not a center and 2 becomes our potential candidate.

G: Gruesome CAPTCHAs

Problem Author: Anton Chernev and Wojtek Trejter

- **Problem:** Determine for q queries whether there is a graph centre with n nodes.
- Naive solution: For every node check whether it is connected to all other nodes. $\mathcal{O}(n^2 q)$ is too slow!
- Instead, we start with node 1, which is our centre candidate.
- If 1 sees 2, then 1 is still a potential candidate and 2 is definitely not a center.
- Otherwise, 1 is definitely not a center and 2 becomes our potential candidate.
- Next, we check whether our current center candidate sees 3. If it does, it keeps being our center candidate.

G: Gruesome CAPTCHAs

Problem Author: Anton Chernev and Wojtek Trejter

- **Problem:** Determine for q queries whether there is a graph centre with n nodes.
- Naive solution: For every node check whether it is connected to all other nodes. $\mathcal{O}(n^2 q)$ is too slow!
- Instead, we start with node 1, which is our centre candidate.
- If 1 sees 2, then 1 is still a potential candidate and 2 is definitely not a center.
- Otherwise, 1 is definitely not a center and 2 becomes our potential candidate.
- Next, we check whether our current center candidate sees 3. If it does, it keeps being our center candidate.
- Otherwise, 3 becomes our center candidate. We continue in the same way with 4, 5, ...

G: Gruesome CAPTCHAs

Problem Author: Anton Chernev and Wojtek Trejter

- **Problem:** Determine for q queries whether there is a graph centre with n nodes.
- Naive solution: For every node check whether it is connected to all other nodes. $\mathcal{O}(n^2q)$ is too slow!
- Instead, we start with node 1, which is our centre candidate.
- If 1 sees 2, then 1 is still a potential candidate and 2 is definitely not a center.
- Otherwise, 1 is definitely not a center and 2 becomes our potential candidate.
- Next, we check whether our current center candidate sees 3. If it does, it keeps being our center candidate.
- Otherwise, 3 becomes our center candidate. We continue in the same way with 4, 5, ...
- At the end, we have a center candidate v and we know that either v is a center or no center exists.

G: Gruesome CAPTCHAs

Problem Author: Anton Chernev and Wojtek Trejter

- **Problem:** Determine for q queries whether there is a graph centre with n nodes.
- Naive solution: For every node check whether it is connected to all other nodes. $\mathcal{O}(n^2 q)$ is too slow!
- Instead, we start with node 1, which is our centre candidate.
- If 1 sees 2, then 1 is still a potential candidate and 2 is definitely not a center.
- Otherwise, 1 is definitely not a center and 2 becomes our potential candidate.
- Next, we check whether our current center candidate sees 3. If it does, it keeps being our center candidate.
- Otherwise, 3 becomes our center candidate. We continue in the same way with 4, 5, ...
- At the end, we have a center candidate v and we know that either v is a center or no center exists.
- Then we can run again through all vertices and check whether v sees them. Overall, we needed $\mathcal{O}(n)$ operations per query, so the overall complexity is $\mathcal{O}(e + nq)$.

C: Cutting Cake

Problem Author: Wietze Koops

- **Problem:** Compute the minimum number of cuts that needs to be made to cut a rectangular cake in at least k equal pieces, such that each piece is at most s% smaller than when cutting the cake in exactly k equal pieces.

C: Cutting Cake

Problem Author: Wietze Koops

- **Problem:** Compute the minimum number of cuts that needs to be made to cut a rectangular cake in at least k equal pieces, such that each piece is at most $s\%$ smaller than when cutting the cake in exactly k equal pieces.
- Naive solution (too slow): Try all possible number of pieces in $\left[k, \lfloor \frac{k}{1-s\%} \rfloor\right]$. For a fixed number of pieces, try all divisors up to $\sqrt{k}$.

C: Cutting Cake

Problem Author: Wietze Koops

- **Problem:** Compute the minimum number of cuts that needs to be made to cut a rectangular cake in at least k equal pieces, such that each piece is at most $s\%$ smaller than when cutting the cake in exactly k equal pieces.
- Naive solution (too slow): Try all possible number of pieces in $\left[k, \lfloor \frac{k}{1-s\%} \rfloor\right]$. For a fixed number of pieces, try all divisors up to $\sqrt{k}$.
- Faster solution: Assume we make at least as many horizontal as vertical cuts. Try all $\lceil \sqrt{k} \rceil$ possible values for the number v of vertical cuts. Then we need $h = \lceil \frac{k}{v} \rceil$ horizontal cuts. Take the minimum value of $h + v$ that does not give too many pieces.

C: Cutting Cake

Problem Author: Wietze Koops

- **Problem:** Compute the minimum number of cuts that needs to be made to cut a rectangular cake in at least k equal pieces, such that each piece is at most $s\%$ smaller than when cutting the cake in exactly k equal pieces.
- Naive solution (too slow): Try all possible number of pieces in $\left[k, \lfloor \frac{k}{1-s\%} \rfloor\right]$. For a fixed number of pieces, try all divisors up to $\sqrt{k}$.
- Faster solution: Assume we make at least as many horizontal as vertical cuts. Try all $\lceil \sqrt{k} \rceil$ possible values for the number v of vertical cuts. Then we need $h = \lceil \frac{k}{v} \rceil$ horizontal cuts. Take the minimum value of $h + v$ that does not give too many pieces.
- Even faster: Without the constraint that each piece is at most $s\%$ smaller, the answer is always cutting the cake in $\lfloor \sqrt{k} \rfloor \times \lceil \sqrt{k} \rceil$ or $\lceil \sqrt{k} \rceil \times \lceil \sqrt{k} \rceil$ pieces. This needs at most $\sqrt{k}$ additional pieces. If $s > 0$, then this is possible for $k > 10000$.

E: Epic Party on a Boat

Problem Author: Wietze Koops

- **Problem:** Find the optimal worst-case cost of guessing a number between 1 and n, given that the cost of guessing k is k.

E: Epic Party on a Boat

Problem Author: Wietze Koops

- **Problem:** Find the optimal worst-case cost of guessing a number between 1 and n, given that the cost of guessing k is k.
- Dynamic Programming: for all $0 \leq a \leq b \leq n$, find the optimal worst-case cost $c[a][b]$ of guessing a number in the half-open interval $(a, b]$ given that the cost of guessing k is k.

E: Epic Party on a Boat

Problem Author: Wietze Koops

- **Problem:** Find the optimal worst-case cost of guessing a number between 1 and n, given that the cost of guessing k is k.
- Dynamic Programming: for all $0 \leq a \leq b \leq n$, find the optimal worst-case cost $c[a][b]$ of guessing a number in the half-open interval $(a, b]$ given that the cost of guessing k is k.
- Compute the $c[a][b]$ in increasing order of the length $b - a$ of the interval.

E: Epic Party on a Boat

Problem Author: Wietze Koops

- **Problem:** Find the optimal worst-case cost of guessing a number between 1 and n, given that the cost of guessing k is k.
- Dynamic Programming: for all $0 \leq a \leq b \leq n$, find the optimal worst-case cost $c[a][b]$ of guessing a number in the half-open interval $(a, b]$ given that the cost of guessing k is k.
- Compute the $c[a][b]$ in increasing order of the length $b - a$ of the interval.
- Then $c[a][a] = 0$ (since the interval is empty) and

$$c[a][b] = \min_{a<k\leq b} \left[k + \max\{c[a][k-1], c[k][b]\}\right].$$

E: Epic Party on a Boat

Problem Author: Wietze Koops

- **Problem:** Find the optimal worst-case cost of guessing a number between 1 and n, given that the cost of guessing k is k.
- Dynamic Programming: for all $0 \leq a \leq b \leq n$, find the optimal worst-case cost $c[a][b]$ of guessing a number in the half-open interval $(a, b]$ given that the cost of guessing k is k.
- Compute the $c[a][b]$ in increasing order of the length $b - a$ of the interval.
- Then $c[a][a] = 0$ (since the interval is empty) and

$$c[a][b] = \min_{a<k\leq b} \left[k + \max\{c[a][k-1], c[k][b]\}\right].$$

- The answer is $c[0][n]$.

B: Binary Speakers

Problem Author: Vitor Greati

Mister Bin understands someone from region R if he can express in his language (using only projections, s and superposition) the basic functions taught in R.

- **Problem:** Given $f : \{0,1\}^k \to \{0,1\}$, is f generated from projections and s by superposition?

B: Binary Speakers

Problem Author: Vitor Greati

Mister Bin understands someone from region R if he can express in his language (using only projections, s and superposition) the basic functions taught in R.

- **Problem:** Given $f : \{0,1\}^k \to \{0,1\}$, is f generated from projections and s by superposition?
- *Solution*:
  A binary function g is *x-preserving* if $g(x,\ldots,x) = x$, for each $x \in \{0,1\}$. Observe that s and all the other functions in the examples are both 0-preserving and 1-preserving.

B: Binary Speakers

Problem Author: Vitor Greati

Mister Bin understands someone from region R if he can express in his language (using only projections, s and superposition) the basic functions taught in R.

- **Problem:** Given $f : \{0, 1\}^k \to \{0, 1\}$, is f generated from projections and s by superposition?
- *Solution*:
  A binary function g is *x-preserving* if $g(x, \ldots, x) = x$, for each $x \in \{0, 1\}$. Observe that s and all the other functions in the examples are both 0-preserving and 1-preserving.

B: Binary Speakers

Problem Author: Vitor Greati

Mister Bin understands someone from region R if he can express in his language (using only projections, s and superposition) the basic functions taught in R.

- **Problem:** Given $f : \{0,1\}^k \to \{0,1\}$, is f generated from projections and s by superposition?
- *Solution*:

  A binary function g is *x-preserving* if $g(x, \ldots, x) = x$, for each $x \in \{0,1\}$. Observe that s and all the other functions in the examples are both 0-preserving and 1-preserving.

  This implies that projections and s cannot break this property when combined via superposition.

  Indeed, Mister Bin understands a function if and only if this function is 0- and 1-preserving.

B: Binary Speakers

Problem Author: Vitor Greati

Mister Bin understands someone from region R if he can express in his language (using only projections, s and superposition) the basic functions taught in R.

- **Problem:** Given $f : \{0,1\}^k \to \{0,1\}$, is f generated from projections and s by superposition?
- *Solution*:

  A binary function g is *x-preserving* if $g(x, \ldots, x) = x$, for each $x \in \{0,1\}$. Observe that s and all the other functions in the examples are both 0-preserving and 1-preserving.

  This implies that projections and s cannot break this property when combined via superposition.

  Indeed, Mister Bin understands a function if and only if this function is 0- and 1-preserving.

  That is, *checking whether the function is expressible or not is O(1)*, good for the `no` cases.

B: Binary Speakers

Problem Author: Vitor Greati

Mister Bin understands someone from region R if he can express in his language (using only projections, s and superposition) the basic functions taught in R.

- **Problem:** Given $f : \{0,1\}^k \to \{0,1\}$, is f generated from projections and s by superposition?
- *Solution*:

  A binary function g is *x-preserving* if $g(x, \ldots, x) = x$, for each $x \in \{0,1\}$. Observe that s and all the other functions in the examples are both 0-preserving and 1-preserving.

  This implies that projections and s cannot break this property when combined via superposition.

  Indeed, Mister Bin understands a function if and only if this function is 0- and 1-preserving.

  That is, *checking whether the function is expressible or not is O(1)*, good for the `no` cases.

  Of course, generating an expression for an expressible function demands more work. Observe however that *s is a ternary conditional operator*, and this will help a lot:

  $$s(a, b, c) \text{ is } b \text{ if } a = 1 \text{ and is } c \text{ otherwise.}$$

First, the intuition. What happens if we pick a 3-ary f and fix its third argument? Look:

B: Binary Speakers

Problem Author: Vitor Greati

First, the intuition. What happens if we pick a 3-ary f and fix its third argument? Look:

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$\rightarrow$

x	y	z	$F_0(x,y,0)$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0

x	y	z	$F_1(x,y,1)$
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

B: Binary Speakers

Problem Author: Vitor Greati

First, the intuition. What happens if we pick a 3-ary f and fix its third argument? Look:

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$\rightarrow$

x	y	z	$F_0(x,y,0)$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0

x	y	z	$F_1(x,y,1)$
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

Then:

$$f(x,y,z) = \begin{cases} F_0(x,y,0) & \text{if } z = 0 \\ F_1(x,y,1) & \text{otherwise} \end{cases}$$

B: Binary Speakers

Problem Author: Vitor Greati

First, the intuition. What happens if we pick a 3-ary f and fix its third argument? Look:

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

$\rightarrow$

x	y	z	$F_0(x,y,0)$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0

x	y	z	$F_1(x,y,1)$
0	0	1	0
0	1	1	1
1	0	1	0
1	1	1	1

Then:

$$f(x,y,z) = \begin{cases} F_0(x,y,0) & \text{if } z = 0 \\ F_1(x,y,1) & \text{otherwise} \end{cases}$$

So, find F_0 and F_1 and combine them using s, which is an if-else!

B: Binary Speakers

Problem Author: Vitor Greati

The algorithm works by fixing suffixes in the arguments of the function, combining simpler functions to form more complex ones, until reaching f.

B: Binary Speakers

Problem Author: Vitor Greati

The algorithm works by fixing suffixes in the arguments of the function, combining simpler functions to form more complex ones, until reaching f.

For a suffix $b_1, \ldots, b_k \in \{0, 1\}$ of length k, two cases:

- $f(b_1, \ldots, b_k) = 0$: then at least one b_i is 0, so the expression is just the i^{th} projection.
- $f(b_1, \ldots, b_k) = 1$: then at least one b_i is 1, so the expression is just the i^{th} projection.

B: Binary Speakers

Problem Author: Vitor Greati

The algorithm works by fixing suffixes in the arguments of the function, combining simpler functions to form more complex ones, until reaching f.

For a suffix $b_1, \ldots, b_k \in \{0, 1\}$ of length k, two cases:

- $f(b_1, \ldots, b_k) = 0$: then at least one b_i is 0, so the expression is just the i^{th} projection.
- $f(b_1, \ldots, b_k) = 1$: then at least one b_i is 1, so the expression is just the i^{th} projection.

Recursively, assume we have expressions F_0 and F_1 for the fixed suffixes $0, b_{j+2}, \ldots, b_k$ and $1, b_{j+2}, \ldots, b_k$, that is, F_z is the expression for the function

$$f_z(x_1, \ldots, x_j) = f(x_1, \ldots, x_j, z, b_{j+2}, \ldots, b_k).$$

We then obtain an expression for the smaller fixed suffix $b_{j+2}, \ldots, b_k$:

$$s(\mathtt{pi}(j + 1)(x), F_0, F_1).$$

B: Binary Speakers

Problem Author: Vitor Greati

The algorithm works by fixing suffixes in the arguments of the function, combining simpler functions to form more complex ones, until reaching f.

For a suffix $b_1, \ldots, b_k \in \{0, 1\}$ of length k, two cases:

- $f(b_1, \ldots, b_k) = 0$: then at least one b_i is 0, so the expression is just the i^{th} projection.
- $f(b_1, \ldots, b_k) = 1$: then at least one b_i is 1, so the expression is just the i^{th} projection.

Recursively, assume we have expressions F_0 and F_1 for the fixed suffixes $0, b_{j+2}, \ldots, b_k$ and $1, b_{j+2}, \ldots, b_k$, that is, F_z is the expression for the function

$$f_z(x_1, \ldots, x_j) = f(x_1, \ldots, x_j, z, b_{j+2}, \ldots, b_k).$$

We then obtain an expression for the smaller fixed suffix $b_{j+2}, \ldots, b_k$:

$$s(\mathtt{pi}(j+1)(x), F_0, F_1).$$

Finally, the function f is just the case in which the length of the fixed suffix is 0. Complexity: $\mathcal{O}(2^k)$.