NWERC 2016
Presentation of solutions

The Jury

2016-11-20

NWERC 2016 Jury

- François Aubry (Université catholique de Louvain)
- Per Austrin (KTH Royal Institute of Technology)
- Gregor Behnke (Ulm University)
- Jeroen Bransen (Chordify)
- Egor Dranischnikow (CST AG, Darmstadt)
- Tommy Färnqvist (Swedish National Forensic Centre)
- Jim Grimmett (LifeJak)
- Eduard Kalinicenko (Palantir)
- Robin Lee (Google)
- Lukáš Poláček (Google)
- Tobias Werth (Google)

Big thanks to our test solvers

- Michal Forišek (Comenius University)
- Barış Kaya (Google)
- Jan Kuipers (AppTornado)
- Alexey Zayakin (University of Latvia)

E – Exam Redistribution

Problem

Given $s_1, \ldots, s_n$, find safe ordering of them.

Solution

Statistics: 143 submissions, 112 accepted

E – Exam Redistribution

Problem

Given $s_1, \ldots, s_n$, find safe ordering of them.

Solution

1. There are two ways in which an ordering can fail to be safe:

Statistics: 143 submissions, 112 accepted

E – Exam Redistribution

Problem

Given $s_1, \ldots, s_n$, find safe ordering of them.

Solution

1. There are two ways in which an ordering can fail to be safe:
   (a) There is a chance that someone gets their own exam
   Observation 1: this happens when size of first room larger than sum of other room sizes

Statistics: 143 submissions, 112 accepted

E – Exam Redistribution

Problem

Given $s_1, \ldots, s_n$, find safe ordering of them.

Solution

1. There are two ways in which an ordering can fail to be safe:
   (a) There is a chance that someone gets their own exam
   Observation 1: this happens when size of first room larger than sum of other room sizes
   (b) We don't have enough in our pile when entering a room
   Observation 2: this happens when size of first room smaller than some other room

Statistics: 143 submissions, 112 accepted

E – Exam Redistribution

Problem

Given $s_1, \ldots, s_n$, find safe ordering of them.

Solution

1. There are two ways in which an ordering can fail to be safe:
   (a) There is a chance that someone gets their own exam

   Observation 1: this happens when size of first room larger than sum of other room sizes

   (b) We don't have enough in our pile when entering a room

   Observation 2: this happens when size of first room smaller than some other room
2. $\Rightarrow$ Possible if $\max s_i \geq \frac{1}{2} \sum s_i$, and any ordering which puts a largest room first works.

Statistics: 143 submissions, 112 accepted

H – Hamiltonian Hypercube

Problem

Given two code-words a and b of an n-bit Gray Code, compute the number of code-words between them.

Solution

Statistics: 220 submissions, 98 accepted

H – Hamiltonian Hypercube

Problem

Given two code-words a and b of an n-bit Gray Code, compute the number of code-words between them.

Solution

1. Output does not depend on n

Statistics: 220 submissions, 98 accepted

H – Hamiltonian Hypercube

Problem

Given two code-words a and b of an n-bit Gray Code, compute the number of code-words between them.

Solution

1. Output does not depend on n
2. Reduce the problem to determining the index of a in the Gray-Code.
   $dist(a, b) = ind(b) - ind(a) - 1$

Statistics: 220 submissions, 98 accepted

H – Hamiltonian Hypercube

Problem

Given two code-words a and b of an n-bit Gray Code, compute the number of code-words between them.

Solution

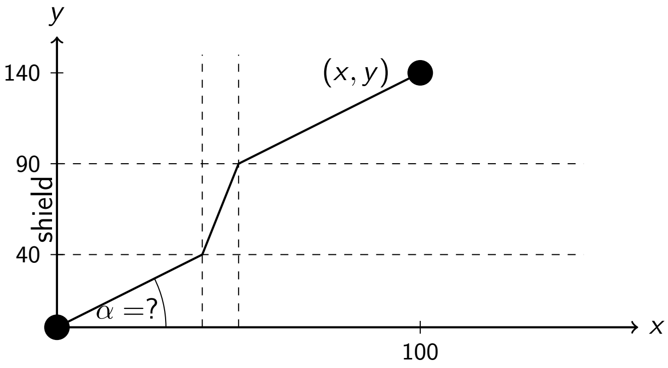
1. Output does not depend on n
2. Reduce the problem to determining the index of a in the Gray-Code.
   $dist(a, b) = ind(b) - ind(a) - 1$
3. Can be solved by a simple recursion:
   $$ind(x) = \begin{cases} ind(y) & \text{if } x = 0y \\ 2^{len(x)} - ind(y) - 1 & \text{if } x = 1y \end{cases}$$

Statistics: 220 submissions, 98 accepted

C – Careful Ascent

Problem

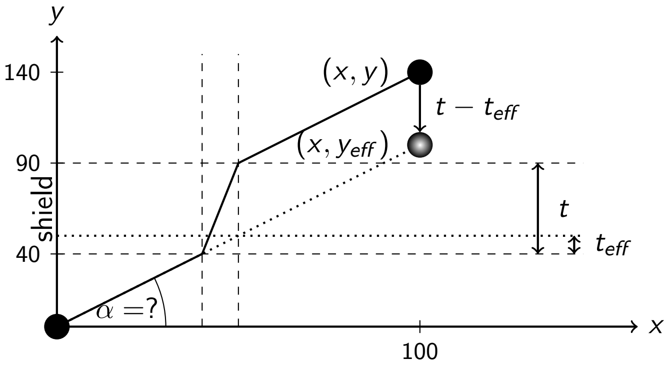
Given target's coordinates (x, y) and descriptions of shields and taking into account the interference of the shields, what is the right horizontal velocity for hitting the target?



C – Careful Ascent

Solution

1. Easy, if there are no shields: $v_{hor} = \cot(\alpha) = x/y$.
2. Replace every shield (l_i, h_i, f_i) by a layer with effective thickness $t_{eff_i} = f_i \cdot (u_i - l_i)$ and calculate y_{eff}.
3. $v_{hor} = \cot(\alpha) = x/y_{eff}$.



C – Careful Ascent

Alternative solution

1. For given v_{hor}, simulate the flight.
2. Adjust v_{hor} depending on whether Mal is too far to the right or too far to the left from the Firefly.
3. Use binary search to reach the needed precision fast enough.

Statistics: 180 submissions, 110 accepted

F – Free Weights

Problem

Given two rows of weights, find the largest weight that must be moved so that all weights can be put into pairs.

Solution

F – Free Weights

Problem

Given two rows of weights, find the largest weight that must be moved so that all weights can be put into pairs.

Solution

1. Decision problem: can we solve it by moving all weights smaller than or equal to M?

F – Free Weights

Problem

Given two rows of weights, find the largest weight that must be moved so that all weights can be put into pairs.

Solution

1. Decision problem: can we solve it by moving all weights smaller than or equal to M?
   - Remove all weights that are smaller than M.

F – Free Weights

Problem

Given two rows of weights, find the largest weight that must be moved so that all weights can be put into pairs.

Solution

1. Decision problem: can we solve it by moving all weights smaller than or equal to M?
   - Remove all weights that are smaller than M.
   - For every i, W_{2i} should be equal to W_{2i+1}.

F – Free Weights

Problem

Given two rows of weights, find the largest weight that must be moved so that all weights can be put into pairs.

Solution

1. Decision problem: can we solve it by moving all weights smaller than or equal to M?
    - Remove all weights that are smaller than M.
    - For every i, W_{2i} should be equal to W_{2i+1}.
    - Otherwise, it isn't possible.

F – Free Weights

Problem

Given two rows of weights, find the largest weight that must be moved so that all weights can be put into pairs.

Solution

1. Decision problem: can we solve it by moving all weights smaller than or equal to M?
   - Remove all weights that are smaller than M.
   - For every i, W_{2i} should be equal to W_{2i+1}.
   - Otherwise, it isn't possible.
2. So binary search on the maximum weight moved.

F – Free Weights

Problem

Given two rows of weights, find the largest weight that must be moved so that all weights can be put into pairs.

Solution

1. Decision problem: can we solve it by moving all weights smaller than or equal to M?
   - Remove all weights that are smaller than M.
   - For every i, W_{2i} should be equal to W_{2i+1}.
   - Otherwise, it isn't possible.
2. So binary search on the maximum weight moved.

Possible pitfalls

1. Not checking for weights split across rows.
2. Trying to put the weights into ascending order.

Statistics: 285 submissions, 80 accepted

I – Iron and Coal

Problem

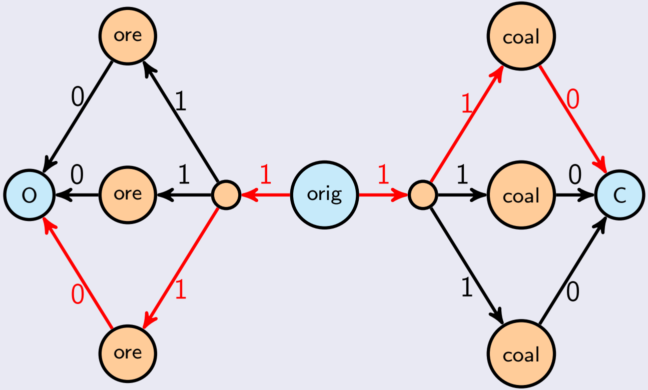
Given a game board with interconnected cells and resources, find an optimal way to seize coal and ore, starting with a single origin cell.

Solution: Reduction to Steiner tree

1. Every cell on the board is a node.
2. Add directed edges to accessible neighbors with weight 0.
3. All nodes with resource “coal” have a directed edge with weight 0 to a super-node C.
4. All nodes with resource “ore” have a directed edge with weight 0 to a super-node 0.
5. Find a minimal Steiner tree for nodes O, C, and the node corresponding to the origin cell.

I – Iron and Coal

A possible resulting graph with a minimal Steiner tree marked with red:



I – Iron and Coal

Solution: Polynomial Algorithm

1. Normally, finding a minimal Steiner tree is a NP-complete problem.
2. In this special case with three nodes, a polynomial algorithm is possible:
   - find distances from the origin node to every other node via bfs.
   - find distances from the super-node O to every other node via bfs on the reversed graph.
   - the same for the super-node C.
   - find a node with the minimal sum of distances to O and to C and from the origin node.
3. Running time: $O(n)$, with n - number of cells on the board.

Statistics: 158 submissions, 56 accepted

A – Arranging Hat

Problem

Given a list of decimal numbers, how many digits need to be substituted to make the list lexicographically sorted?

A – Arranging Hat

Solution

1. Choose some quantity X and for all of the first X numbers, set the first digit to 0.

A – Arranging Hat

Solution

1. Choose some quantity X and for all of the first X numbers, set the first digit to 0.
2. Now we have two subproblems to recurse into:

A – Arranging Hat

Solution

1. Choose some quantity X and for all of the first X numbers, set the first digit to 0.
2. Now we have two subproblems to recurse into:
   - Sort the remainder of the numbers starting with 0.

A – Arranging Hat

Solution

1. Choose some quantity X and for all of the first X numbers, set the first digit to 0.
2. Now we have two subproblems to recurse into:
   - Sort the remainder of the numbers starting with 0.
   - Sort all of the other numbers, using digits greater than or equal to 0 as prefixes.

A – Arranging Hat

Solution

1. Choose some quantity X and for all of the first X numbers, set the first digit to 0.
2. Now we have two subproblems to recurse into:
   - Sort the remainder of the numbers starting with 0.
   - Sort all of the other numbers, using digits greater than or equal to 0 as prefixes.
3. We look at every subrange of numbers, every starting index, and every starting digit.

A – Arranging Hat

Solution

1. Choose some quantity X and for all of the first X numbers, set the first digit to 0.
2. Now we have two subproblems to recurse into:
   - Sort the remainder of the numbers starting with 0.
   - Sort all of the other numbers, using digits greater than or equal to 0 as prefixes.
3. We look at every subrange of numbers, every starting index, and every starting digit.
4. With memoisation, overall complexity is $O(N^3 \cdot M)$.

A – Arranging Hat

Solution

1. Choose some quantity X and for all of the first X numbers, set the first digit to 0.
2. Now we have two subproblems to recurse into:
   - Sort the remainder of the numbers starting with 0.
   - Sort all of the other numbers, using digits greater than or equal to 0 as prefixes.
3. We look at every subrange of numbers, every starting index, and every starting digit.
4. With memoisation, overall complexity is $O(N^3 \cdot M)$.

Possible pitfalls

1. Slow python recursion

A – Arranging Hat

Alternative dynamic programming solution

1. Let's consider whole numbers without splitting them by digits.
2. $a_{i,j} =$ the minimum number obtainable for i-th number, if we made j changes on the first i numbers.
3. From any state we can try changing some amount of digits in the $i+1$-th number.
4. Then we can greedily in $O(M)$ obtain the smallest number we can get from $i+1$-th number using fixed number of changes.
5. If it's greater or equal than $a_{i,j}$ – that's a valid transition.
6. $O(N^2 \cdot M)$ for the state and $O(M^2)$ for the transition.
7. But the answer is never going to be more than $N \cdot \log_{10} N$.
8. $O(N^2 \cdot \log_{10} N)$ for the state and $O(N \cdot \log_{10} N \cdot M)$ for the transition.

Statistics: 45 submissions, 12 accepted

J – Jupiter Orbiter

Problem

There are Q FIFO-queue with capacities c_i and N timeslots to remove data from the queues. The maximum amount removable d_i is given for each timeslot.
Each queue gets a given amount of data prior each such timeslot.
Is it possible to remove data from the queues in such a way that after the last timeslot no data is left in the queues?

Solution

J – Jupiter Orbiter

Problem

There are Q FIFO-queue with capacities c_i and N timeslots to remove data from the queues. The maximum amount removable d_i is given for each timeslot.
Each queue gets a given amount of data prior each such timeslot.
Is it possible to remove data from the queues in such a way that after the last timeslot no data is left in the queues?

Solution

1. Model the problem as a graph and run Max-Flow.

J – Jupiter Orbiter

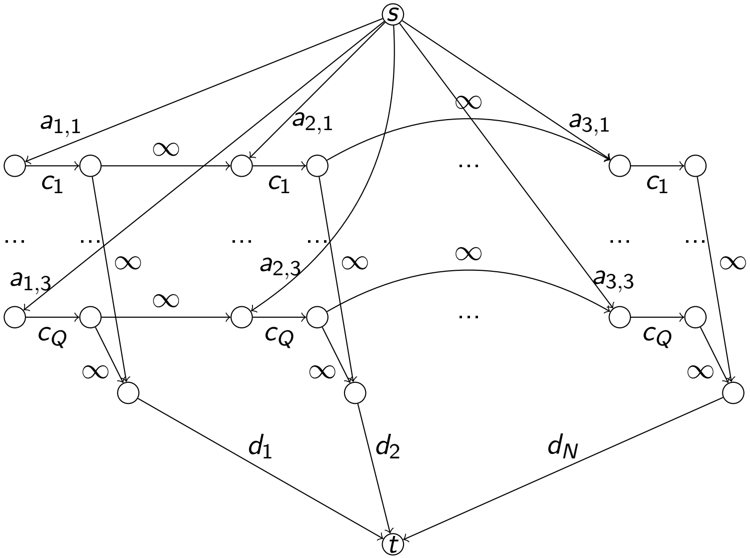
Problem

There are Q FIFO-queue with capacities c_i and N timeslots to remove data from the queues. The maximum amount removable d_i is given for each timeslot.
Each queue gets a given amount of data prior each such timeslot.
Is it possible to remove data from the queues in such a way that after the last timeslot no data is left in the queues?

Solution

1. Model the problem as a graph and run Max-Flow.
2. Check whether the flow is equal to the total amount of data generated.

J – Jupiter Orbiter



J – Jupiter Orbiter

Alternative solution

1. Simulate receiving of all the data.
2. Whenever the queue overfills, cut off the excess and record that you have to downlink that amount of data from that queue until this time moment – these form *restrictions*.
3. Simulate the whole process the second time.
4. You fail if and only if you violate one of the restrictions.
5. As such, whenever we have the opportunity to downlink data, we should downlink data to satisfy the restriction with the earliest possible time deadline.
6. So we can just downlink the data greedily based on sorted restriction list.

Statistics: 188 submissions, 36 accepted

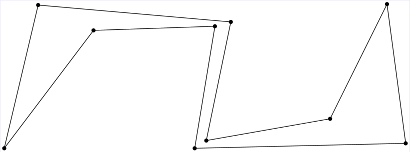
K – Kiwi Trees

Problem

Given a polygon with special properties, can you place 2 disjoint circles inside the polygon?

Solution

1. Every polygon with $n \geq 4$ has two ears.



Statistics: 59 submissions, 9 accepted

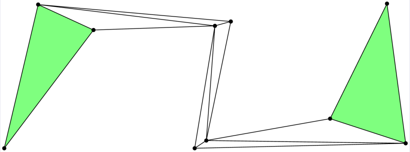
K – Kiwi Trees

Problem

Given a polygon with special properties, can you place 2 disjoint circles inside the polygon?

Solution

1. Every polygon with $n \geq 4$ has two ears.



2. Angle in ear between 18 and 144 degrees. Two sides of the angle are at least 30 meters.

Statistics: 59 submissions, 9 accepted

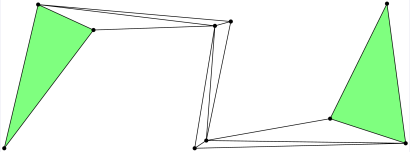
K – Kiwi Trees

Problem

Given a polygon with special properties, can you place 2 disjoint circles inside the polygon?

Solution

1. Every polygon with $n \geq 4$ has two ears.



2. Angle in ear between 18 and 144 degrees. Two sides of the angle are at least 30 meters.
3. Each ear can accommodate a tree of radius $\sim$ 4008 mm.

Statistics: 59 submissions, 9 accepted

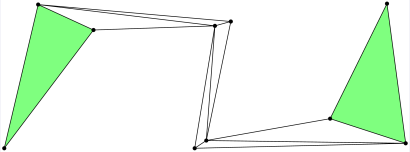
K – Kiwi Trees

Problem

Given a polygon with special properties, can you place 2 disjoint circles inside the polygon?

Solution

1. Every polygon with $n \geq 4$ has two ears.



2. Angle in ear between 18 and 144 degrees. Two sides of the angle are at least 30 meters.
3. Each ear can accommodate a tree of radius ~ 4008 mm.
4. Special case $n = 3$ (triangle). Can output "impossible".

Statistics: 59 submissions, 9 accepted

B – British Menu

Problem

Given a directed graph $G = (V, E)$, were every cycle contains at most 5 different nodes, compute the length of the longest path.

Solution

Statistics: 52 submissions, 8 accepted

B – British Menu

Problem

Given a directed graph $G = (V, E)$, were every cycle contains at most 5 different nodes, compute the length of the longest path.

Solution

1. This problem is in general $\mathbb{NP}$-complete, but can be solved for DAGs in $\mathcal{O}(n + m)$ using DP

Statistics: 52 submissions, 8 accepted

B – British Menu

Problem

Given a directed graph $G = (V, E)$, were every cycle contains at most 5 different nodes, compute the length of the longest path.

Solution

1. This problem is in general $\mathbb{NP}$-complete, but can be solved for DAGs in $\mathcal{O}(n + m)$ using DP
2. Form the cycle property follows that every SCC of G contains at most 5 vertices.

Statistics: 52 submissions, 8 accepted

B – British Menu

Problem

Given a directed graph $G = (V, E)$, were every cycle contains at most 5 different nodes, compute the length of the longest path.

Solution

1. This problem is in general $\mathbb{NP}$-complete, but can be solved for DAGs in $\mathcal{O}(n + m)$ using DP
2. Form the cycle property follows that every SCC of G contains at most 5 vertices.
3. Solve longest path for each SCC – from every vertex to every other – by complete exploration ($\mathcal{O}(n!)$ is sufficiently fast)

Statistics: 52 submissions, 8 accepted

B – British Menu

Problem

Given a directed graph $G = (V, E)$, were every cycle contains at most 5 different nodes, compute the length of the longest path.

Solution

1. This problem is in general $\mathbb{NP}$-complete, but can be solved for DAGs in $\mathcal{O}(n + m)$ using DP
2. Form the cycle property follows that every SCC of G contains at most 5 vertices.
3. Solve longest path for each SCC – from every vertex to every other – by complete exploration ($\mathcal{O}(n!)$ is sufficiently fast)
4. Reduce all SCCs to a single vertex

Statistics: 52 submissions, 8 accepted

B – British Menu

Problem

Given a directed graph $G = (V, E)$, were every cycle contains at most 5 different nodes, compute the length of the longest path.

Solution

1. This problem is in general $\mathbb{NP}$-complete, but can be solved for DAGs in $\mathcal{O}(n + m)$ using DP
2. Form the cycle property follows that every SCC of G contains at most 5 vertices.
3. Solve longest path for each SCC – from every vertex to every other – by complete exploration ($\mathcal{O}(n!)$ is sufficiently fast)
4. Reduce all SCCs to a single vertex
5. Now the graph is a DAG, so run DP (and keep in mind how the paths in each SCC look like)

Statistics: 52 submissions, 8 accepted

D – Driving in Optimistan

Problem

Given distances between all leaves of a tree, find the average distance of road signs placed every 1 kilometer of a road.

Solution, part 1

- Whole tree with all traffic signs is too big; only reconstruct leaves and intersections:
  1. Start with each port town being a separate node.
  2. Sort port town pairs in ascending order by distance.
  3. Go through the pairs and merge the two trees containing both port towns by adding a new root.

Statistics: 6 submissions, 2 accepted

D – Driving in Optimistan

Problem

Given distances between all leaves of a tree, find the average distance of road signs placed every 1 kilometer of a road.

Solution, part 2

- For each subtree T with root r, calculate: average length of shortest paths going through r with both ends (A_r^2) in T and with one end (A_r^1) in T.
- Both values A_r^1 and A_r^2 can be calculated using values A^1 of all children of r and their distances from r.

Statistics: 6 submissions, 2 accepted

G – Gotta Nudge 'Em All

Problem

Given a timed list of caught Nudgémon, figure out when to activate an item (which doubles the XP for catching Nudgémon and allows to evolve Nudgémon for additional XP) so that XP is maximized.

G – Gotta Nudge 'Em All

Solution

1. Use sliding window for Egg on caught Nudgémon.

Solution

1. Use sliding window for Egg on caught Nudgémon.
2. XP gained between Nudgémon families are independent.

G – Gotta Nudge 'Em All

Solution

1. Use sliding window for Egg on caught Nudgémon.
2. XP gained between Nudgémon families are independent.
3. After every catch recalculate XP for corresponding family.

G – Gotta Nudge 'Em All

Solution

1. Use sliding window for Egg on caught Nudgémon.
2. XP gained between Nudgémon families are independent.
3. After every catch recalculate XP for corresponding family.
4. Should only evolve the weakest Nudgémon in the family.

G – Gotta Nudge 'Em All

Solution

1. Use sliding window for Egg on caught Nudgémon.
2. XP gained between Nudgémon families are independent.
3. After every catch recalculate XP for corresponding family.
4. Should only evolve the weakest Nudgémon in the family.
5. Should only transfer the strongest Nudgémon in the family.

G – Gotta Nudge 'Em All

Solution

1. Use sliding window for Egg on caught Nudgémon.
2. XP gained between Nudgémon families are independent.
3. After every catch recalculate XP for corresponding family.
4. Should only evolve the weakest Nudgémon in the family.
5. Should only transfer the strongest Nudgémon in the family.
6. Hence, can greedily calculate the maximum XP for the family in $O(N)$ by grouping together Nudgémon of the same type.

G – Gotta Nudge 'Em All

Solution

1. Use sliding window for Egg on caught Nudgémon.
2. XP gained between Nudgémon families are independent.
3. After every catch recalculate XP for corresponding family.
4. Should only evolve the weakest Nudgémon in the family.
5. Should only transfer the strongest Nudgémon in the family.
6. Hence, can greedily calculate the maximum XP for the family in $O(N)$ by grouping together Nudgémon of the same type.
7. This can be sped up to $O(\log^2 N)$ (or even $O(\log N)$) by using segment trees for a total complexity of $O(N \cdot \log^2 N)$.

G – Gotta Nudge 'Em All

Solution

1. Use sliding window for Egg on caught Nudgémon.
2. XP gained between Nudgémon families are independent.
3. After every catch recalculate XP for corresponding family.
4. Should only evolve the weakest Nudgémon in the family.
5. Should only transfer the strongest Nudgémon in the family.
6. Hence, can greedily calculate the maximum XP for the family in $O(N)$ by grouping together Nudgémon of the same type.
7. This can be sped up to $O(\log^2 N)$ (or even $O(\log N)$) by using segment trees for a total complexity of $O(N \cdot \log^2 N)$.
8. Careful with transfers on the same level we're currently evolving – for the constraints given easier to start with -1 candies, have 4 for every catch and forget about transfers.

Alternative solution

1. When sliding egg window, keep track of all Nudgémons and their current level

Statistics: 6 submissions, 2 accepted

G – Gotta Nudge 'Em All

Alternative solution

1. When sliding egg window, keep track of all Nudgémons and their current level
2. Forget about transfers – amount of candy after X catches is effectively $4X - 1$

Statistics: 6 submissions, 2 accepted

Alternative solution

1. When sliding egg window, keep track of all Nudgémons and their current level
2. Forget about transfers – amount of candy after X catches is effectively $4X - 1$
3. When catching a new Nudgémon, try to increase number of upgrades in its family by repeating the following:

Statistics: 6 submissions, 2 accepted

G – Gotta Nudge 'Em All

Alternative solution

1. When sliding egg window, keep track of all Nudgémons and their current level
2. Forget about transfers – amount of candy after X catches is effectively $4X - 1$
3. When catching a new Nudgémon, try to increase number of upgrades in its family by repeating the following:
   1. If can afford cheapest currently available upgrade, buy it.

Statistics: 6 submissions, 2 accepted

G – Gotta Nudge 'Em All

Alternative solution

1. When sliding egg window, keep track of all Nudgémons and their current level
2. Forget about transfers – amount of candy after X catches is effectively $4X - 1$
3. When catching a new Nudgémon, try to increase number of upgrades in its family by repeating the following:
   1. If can afford cheapest currently available upgrade, buy it.
   2. Otherwise, if most expensive upgrade bought cost more than cheapest available one, then "undo" the expensive one.

Statistics: 6 submissions, 2 accepted

G – Gotta Nudge 'Em All

Alternative solution

1. When sliding egg window, keep track of all Nudgémons and their current level
2. Forget about transfers – amount of candy after X catches is effectively $4X - 1$
3. When catching a new Nudgémon, try to increase number of upgrades in its family by repeating the following:
    1. If can afford cheapest currently available upgrade, buy it.
    2. Otherwise, if most expensive upgrade bought cost more than cheapest available one, then "undo" the expensive one.
4. Using two maps, can easily do each iteration in $O(\log n)$ time.

Statistics: 6 submissions, 2 accepted

G – Gotta Nudge 'Em All

Alternative solution

1. When sliding egg window, keep track of all Nudgémons and their current level
2. Forget about transfers – amount of candy after X catches is effectively $4X - 1$
3. When catching a new Nudgémon, try to increase number of upgrades in its family by repeating the following:
    1. If can afford cheapest currently available upgrade, buy it.
    2. Otherwise, if most expensive upgrade bought cost more than cheapest available one, then "undo" the expensive one.
4. Using two maps, can easily do each iteration in $O(\log n)$ time.
5. Because of the constant amount of candies per catch, the total number of iterations is $O(n)$.

Statistics: 6 submissions, 2 accepted

Random numbers produced by the jury

1081 number of posts made in the jury's forum.
(NWERC 2015: 1217)

964 commits made to the problem set repository.
(NWERC 2015: 915)

370 number of lines of code used in total by the shortest judge solutions to solve the entire problem set.
(NWERC 2015: 416)

20.6 average number of jury solutions per problem, including incorrect ones.
(NWERC 2015: 16.6)