UKIEPC 2017

Summary and solution outlines

rgl@google.com

UKIEPC Names

Organisers: **Max Wilson**, **James Davenport**, **Rachid Hourizi**

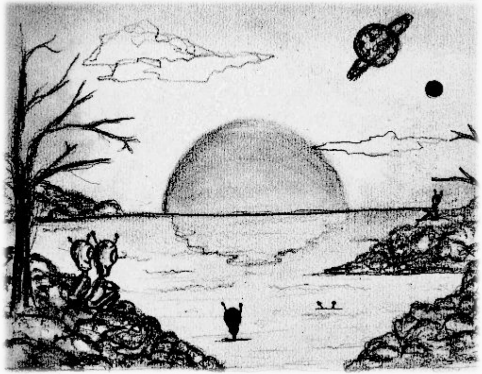
Writers: **Robin Lee, Jim Grimmett, Kiril Dichev**

Reviewers: **Ian Pratt-Hartmann, Sander Alewijnse**

Infrastructure: **Neil Francis**, **Matt Richards**

Illustrator: **Lisa Abose**

Problem Solutions



A - Alien Sunset

102 correct • solved at: **00:26** by
Tractor_Specialists (Oxford)

Author: **Jim**

Overview

- A number of planets, each with varying sunrise and sunset and different length days.
- Identify the earliest time all of the planets are in darkness.
- If it takes too long, output impossible.

Alien Sunset - Solution

Techniques

- LCM
- Intervals

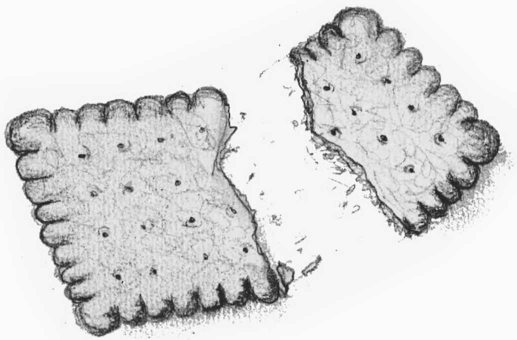
Algorithm

- Find the longest day length, multiply by 1825, and iterate through every hour up to there starting from 1.

- For each planet, is the hour chosen in their nighttime?
  - If yes, we have a solution.
  - Otherwise, try the next hour.

- If we reach the end of the timespan, “impossible”.

- O(Hours * Planets).



B - Break Biscuits

23 correct • solved at: **00:36** by
AibohphobiA (Edinburgh)

Author: **Robin**

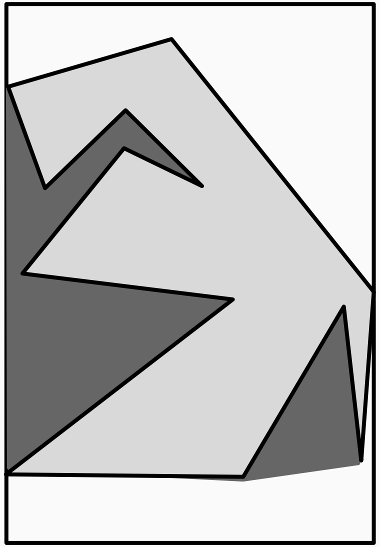
Overview

- A given irregularly-shaped biscuit will be dunked into an infinitely deep, straight-sided mug.
- How wide does this infinite mug need to be to accommodate the biscuit?

Break Biscuits - Solution

Techniques

- Convex hull
- Rotating calipers

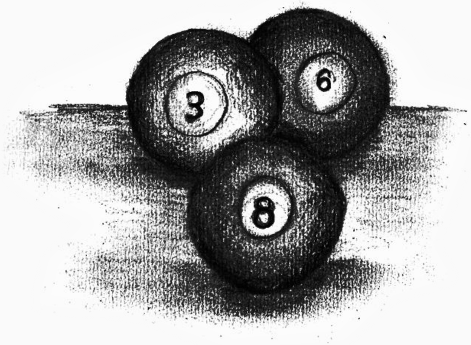


Algorithm

- If the mug is just large enough to contain the biscuit, the biscuit can't rotate once it's inside. At least 3 of its vertices must touch the side of the mug.

- Find the convex hull of the biscuit using, for example, Andrew's monotone chain algorithm.

- Iterate over the edges of the biscuit. Keep track of the "farthest vertex" from the current edge. The answer is the smallest such distance among all edges.



C - Cued In

162 correct • solved at: **00:07** by
AKSLOP-7991 (Cambridge)

Author: **Jim**

Overview

- Various coloured balls are on the snooker table. The value of each ball is given.
- Balls must be potted in alternating red:colour:red:colour order until no reds are left.
- Work out the maximum remaining score if all balls are potted.

Cued In - Solution

Techniques

- Greedy
- String matching

Algorithm

- We must pot reds with colours.
- Read in the colour of each ball, determine its value.
- To gain the maximum score, pot each red with the highest value colour (C_{max}). We then pot the remaining colours.
- As each red has a value of 1, if the number of reds is R and the sum of the colours is S the maximum is:

$$R(1+C_{max}) + S$$

- Special case: 100% of the balls are red

D - Deranging Hat

71 correct • solved at: **00:09** by
PrimeGoal (Cambridge)

Author: **Robin**

Overview

- Generate a sorting network to “un-sort” an array back into its original state.
- Use at most 10 * MAX_N swaps.

Deranging Hat - Solution

Techniques

- Data structures
- Permutations

Algorithm

- The mapping from the sorted string to the original string is a permutation.
  - Because, in a permutation, every vertex has one in-edge and one out-edge, a permutation graph is a set of disjoint cycles.

- If we need to move [0] to [1], [1] to [2], [2] to [3], and [3] to [0],this can be done in N-1 swaps
  - Swap [2] with [3] (2 3) if [2]>[3]; (3 2) otherwise.
  - Swap [1] with [2] (1 2) if [1]>[2]; (2 1) otherwise.
  - Swap [0] with [1] (0 1) if [0]>[1]; (1 0) otherwise.

- Pitfall: Ordering of letters changes over time. Simulate the swaps on a sorted string to keep track.

E - Education

58 correct • solved at: **00:36** by
FakeMaths (Cambridge)

Author: **Jim**

Overview

- Given a list of Departments and the number of students in each.
- Given a list of available rooms, their capacities and costs.
- Identify the minimum cost to house all of the Departments in separate rooms.

Education - Solution

Techniques

- Greedy matching
- Sorting
- Assignment problem

Algorithm

- Greedy: keep putting the largest unassigned class in the cheapest building available.
  - This works because for any two class sizes A < B, the choice of buildings for B is a subset of the choice for A with identical scores.

- Implementation
  - Keep two sorted sets of buildings, both initially full:
    - P (sorted by price first)
    - S (sorted by size first)
  - Iterate through classes C in decreasing order of size
  - While the max element of S is too large, delete from S and P.
  - Remove the smallest element of P and assign it to C.

F - Flipping Coins

49 correct • solved at: **00:16** by
Me[N]ta∭ca (Cambridge)

Author: **Robin**

Overview

- We have N coins face-down.
- You must pick up a single coin and flip it randomly, exactly K times.
- If you can choose the next coin to flip, what's the maximum expected number of coins heads-up at the end?

Flipping Coins - Solution

Techniques

- Combinatorics
- Dynamic programming
- Personal finance

Algorithm

- Dynamic programming state: {**head_count**, **flips_left**}

- If we have no flips left, the answer is the number of heads we have
  - $f(h, 0) = h$

- If at least one tail is left, flipping it gives a 50% chance of 1 extra head, or a 50% chance of nothing changing.
  - $f(h, k) = 0.5 * f(h+1,k-1) + 0.5 * f(h, k-1)$

- Otherwise, it's necessary to flip a head and have a 50% chance of *reducing* the score.
  - $f(N, k) = 0.5 * f(N,k-1) + 0.5 * f(N-1,k-1)$
  - $f(N, k) = N - 0.5$

Flipping Coins - Solution (alternative)

Techniques

- Combinatorics
- Dynamic programming
- Personal finance

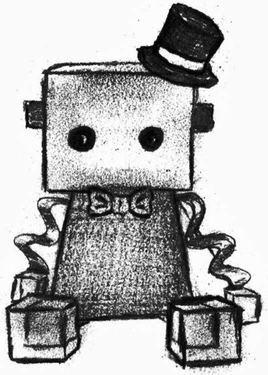
Algorithm

- Either we make **fewer** successful flips to heads (X) than N, or a **greater or equal** amount of flips.

- If we made **fewer**, we'll have X heads at the end.
  - And there are **N choose X** ways of getting there.

- Otherwise, the answer depends on the final flip.
  - If successful (result = N), there are **N-1 choose X-1** ways.
  - If unsuccessful (result = N-1), there are **N-1 choose X** ways.

- Add the possible ways up for all X, and divide by $\mathbf{2^K}$.



G - GentleBots

33 correct • solved at: **01:19** by
PrimeGoal (Cambridge)

Author: **Kiril**

Overview

- Two polite robots need to navigate from points A to B without bumping into each other.
- Find any list of moves that accomplish this in no more than 7000 steps.

GentleBots - Solution

Techniques

- Ad-hoc
- Escape problem

Algorithm

- Many strategies that work. Most have edge cases -- it's very easy to accidentally get stuck in a loop.

- An easy-to-implement one:
  - Move both robots towards their goals repeatedly.
  - If the robots collide, undo the last move, pick a random robot and move it in a random direction instead.

- Other approaches that work:
  - Plan one robot's path, then plan the other path around it.
  - Resolve conflicts with a small depth-first-search.
  - Maximum flow!

H - Hiker Safety

6 correct • solved at: **02:32** by
FakeMaths (Cambridge)

Author: **Robin**

Overview

- We have a one-dimensional track we need to move people along.
- Everyone has a specific minimum and maximum distance they need to keep to their neighbour.
- Once someone reaches the end of the track, their constraints don't matter any more.

Hiker Safety - Solution

Techniques

- Greedy
- Two pointers

Algorithm

- If we have just two people, greedy works:
  - As long as either of the two hikers can move without violating constraints, move them forward.
  - We end up with a sequence of moves like AABBABBAB

- When there are three or more, solve for adjacent pairs of people and then merge all the solutions together.
  - Eg. AABAB + CBBC = AACBABC

- A hiker is eligible to move whenever they're first in all lists they appear in. Keep count and update "who's next" after every move.

I - I Work All Day

163 correct • solved at: **00:04** by
did you do D (Cambridge)

Author: **Robin**

Overview

- A tree-chopping machine keeps cutting down an N-height tree into logs of size L, or smaller if necessary.
    - For example, if N=15, L=4, we get "4 4 4 3".
    - We want to make the size of the last log as small as possible.
- Given several possible values of L, what's the best one to use?

I Work All Day - Solution

Techniques

- Modulo
- Sorting

Algorithm

- The chopping robot is an over-explained implementation of the modulo function (%).

- Map each of the inputs into a tuple (tree % X, X) and sort it using your language's built-in sort() function.
    - The answer is now the first element in the list.

J - Just a Minim

158 correct • solved at: **00:06** by
-= [B]ichael [B] [B]iggins =- (DCU)

Author: **Jim**

Overview

- Given are a number of notes in a tune and the length of each of those notes.
- Find the length of the tune, in seconds.

Just a Minim - Solution

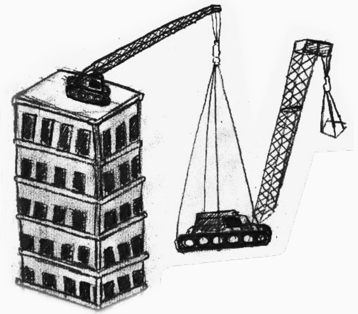
Techniques

- Floating point
- Powers

Algorithm

- Write a function to convert numbers to notes.
    - Hardcode it, or
    - Special-case 0 and use $f(x) = 1.0 \div \text{(double)}\ x$ for the rest

- Iterate through the array calling the function. Accumulate all the answers together at the end.



K - Knightsbridge

15 correct • solved at: **02:13** by

Hello World (Edinburgh)

Author: **Robin**

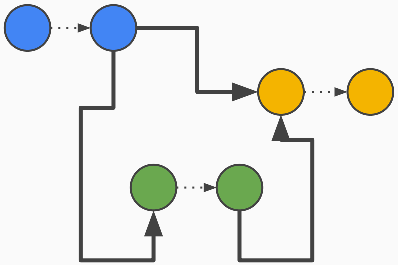
Overview

- Several buildings each need a crane on top that can lift some given weight.
- A crane can only be lifted up onto the top of a building if another crane strong enough to lift its weight is already there, or if its weight is 0.
- Find a sequence of distinct cranes to put on top of each building.

Knightsbridge Rises - Solution

Techniques

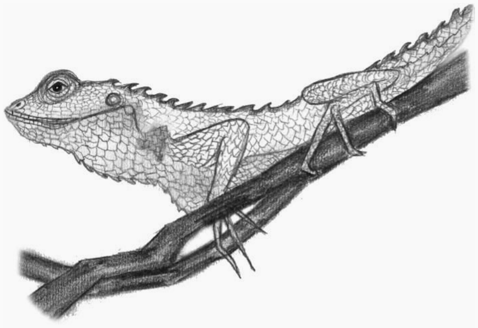
- Disjoint paths
- Maximum flow
- Directed acyclic graphs



Algorithm

- Draw a directed graph, adding an edge from crane A to crane B if and only if A.strength ≥ B.weight.
    - The answer will be a set of vertex-disjoint paths through this graph, one for each building, each ending in a crane strong enough for that building.
    - Classic maximum flow transformation: split every vertex into two virtual halves, "in" and "out".
        - Draw an edge between "in" and "out" with unit capacity. This means the vertex can only be used for one path.
        - Add a sink vertex for each building, an edge from the source to any crane with 0 weight.

- Apply any reasonable max-flow algorithm.



L - Lounge Lizards

18 correct • solved at: **01:41** by
Me[N]ta⟦ca (Cambridge)

Author: **Robin**

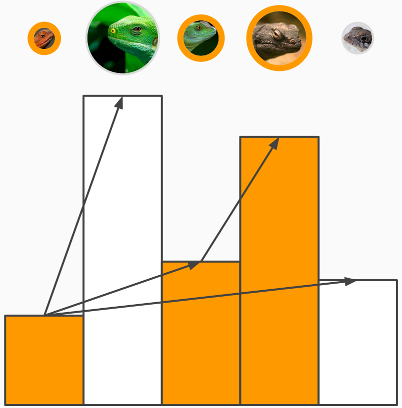
Overview

- Monitor lizards are sitting around a television screen.
- A lizard can see the TV if it is taller than all the lizards on the straight line to the television.
- How many lizards can see the screen at once if we remove an optimal subset?

Lounge Lizards - Solution

Techniques

- Patience algorithm
- Common divisors
- Geometry

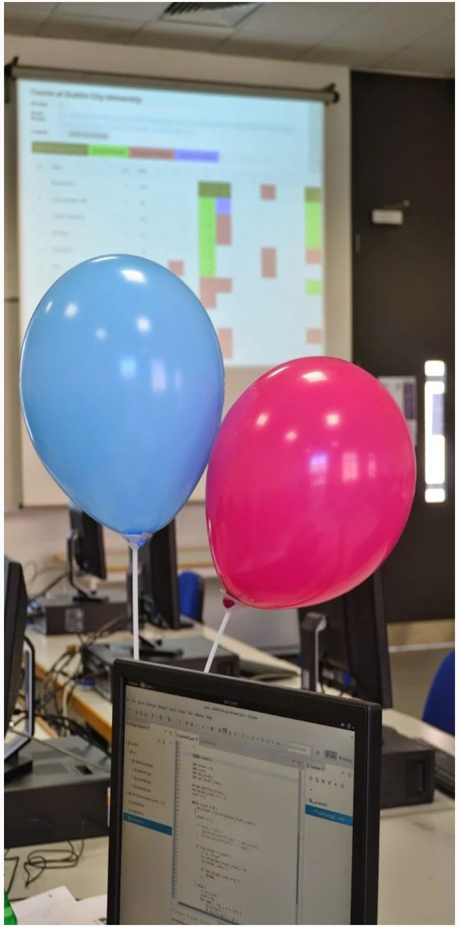


Algorithm

- Save lizard positions as vectors relative to the television.
- Two lizards **i** and **j** intersect iff **d[i] * t = d[j]** for some **t**
  - Canonicalise every vector by taking the GCD of its **x** and **y**, dividing through, and saving that part as its “length”.
    - (6, 12) becomes (1,2)*2
    - (-9, -6) becomes (-3,-2)*3
  - Group lizards by direction. Sort each group by GCD as a proxy for distance.

- The answer for any one group of lizards is its Longest Increasing Subsequence. Patience sorting is fast and easy to implement.
  - O(NlogN) dynamic programming is also fast enough.






Questions?

Or comments?

Final Standings

http://domjudge.bath.ac.uk/