UKIEPC 2016

Post-Contest Presentation
rgl@google.com

UKIEPC Numbers

2013: **52** teams; **5** sites
2014: **61** teams; **9** sites
2015: **142** teams; **12** sites
2016: **171** teams; **13** sites

First correct submission: **00:03:36 – Grass Seed Inc, IRL** (Cambridge)
Last correct submission: **04:59:23 – Fridge, @ tvoj otec** (Southampton)
Number of submissions: **1433**

506 lines of code to solve the whole set.

UKIEPC Names

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

Writers: **Robin Lee, Jim Grimmett, James Stanley**

Reviewers: **Ximo Lerma, Per Austrin**

Sysadmins: **Neil Francis, Matt Richards, Rob Perkins**

Illustrator: **Lisa Abose**

Fun Facts

We restarted the domjudge server **3 times** during this contest

We had **2 judgehosts** fail after the 3rd restart (but we brought them back!)

We threw away **1 question** a week before the contest

Problem Solutions

A - Taxing

30 correct • solved at: **00:35** by

Just To Fail One More Time (Taras Shevchenko)

Author: **Jim**

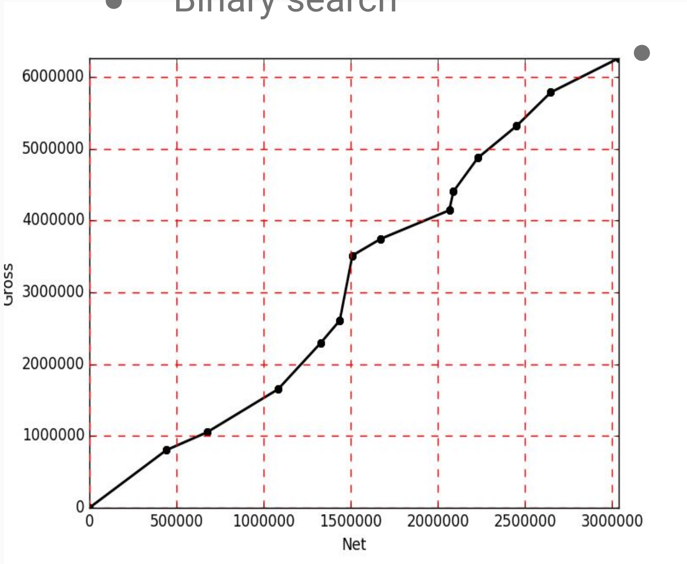
Overview

- A number of tax bands, each with a certain tax percentage.
- A number of friends with earnings and net present size.
- Determine the gross present size for each friend.

Taxing Problem - Solution

Techniques

- Geometric series
- Binary search



Algorithm

- For each friend, ‘fill-up’ tax bands one-by-one.
- Start filling up the first tax band with any space left.
  - ○ If the gift will not fit in this first band, work out the tax on this part of the gift and move onto the next tax band.
  - ○ If the gift does fit, calculate tax, and tax on tax, etc. If that total would leave us in the same band, we are done.
  - ○ If not, work out what portion of the tax will overlap, move to the next band and repeat.

B - Build a Boat

1 correct • solved at: **04:34** by

Catz CS Society (Oxford)

Author: **Robin**

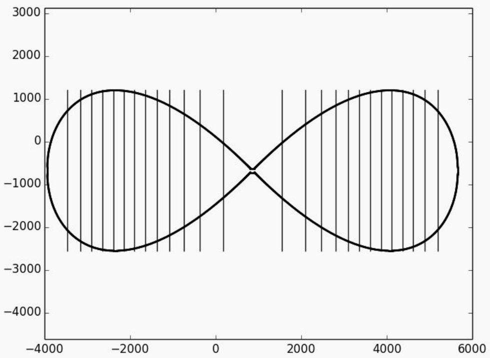
Overview

- Given a polygon with edges going strictly left-right (a monotone polygon)
- Partition the polygon into as many equal slices as possible, above a minimum size

Build a Boat - Solution

Techniques

- Polygon area
- Integration
- Binary search



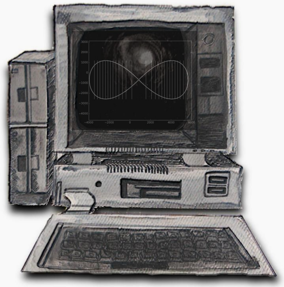
Algorithm

- Create a function that takes a width, crops the polygon vertices to that width, and calculates its area, eg. with cross-products:
    - total_area = sum(vertex[i] x vertex[i+1]) / 2

- Precompute the function for every “interesting” width (X coordinates of vertices) and interpolate in between

- Work out segment sizes from total area:
    - segment_area = total_area / floor(total_area / min_area)

- Run binary search repeatedly to find the segment positions, given the areas they need to occupy



C - Compiler

4 correct • solved at: **02:36** by
Catz CS Society (Oxford)

Author: **Robin**

Overview

- A simple processor supplied with limited instructions, three registers, and a small stack.
- No program can be longer than 40 instructions.
- Write a program that will write the assembly language to output a number between 0 and 255

Compiler - Solution

Techniques

- Dynamic programming
- Shortest paths

Algorithm

- 3 registers and 256 bytes of stack is overkill. All we need is:
    - 2 registers
    - 1 item on the stack

- Let `state={X,Y,Stack1}` --- that's 257^3 = 16,974,593 choices
    - Breadth-first search over all possible CPU states
        - Worst case: 38 instructions

- Another approach from Per
    - Factorise one register recursively via (PH S)*T, AD*(x-1), PL
        - Worst case: 40 instructions

D - Darkness

Not solved

Author: **Jim**

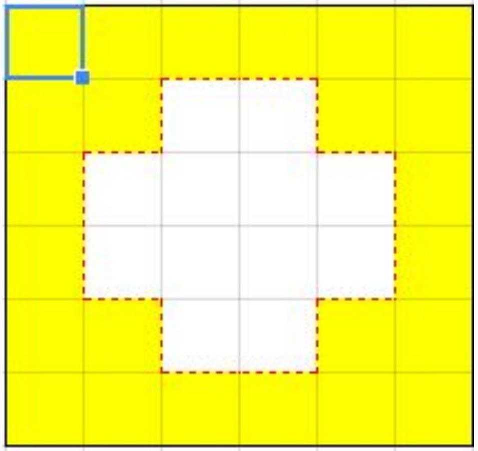
Overview

- Wall off badly lit areas of a nightclub
- You may also wall off well-lit areas, but this comes at a hefty price

Darkness - Solution

Techniques

- Minimum cut
- Maximum flow
- Fractions



Algorithm

- Find the cheapest way of cutting off the “inside” from the “outside”
  - £11 to remove an edge between adjacent cells
  - £43 if the cells were both lit

- First, find which cells are above the threshold
  - One big loop is fine

- Next, add edges between cells for fence costs
  - And infinite edges from **Source** for boundary cells
  - And infinite edges to **Sink** for unlit cells

- Solve with your favourite maxflow algorithm

E - Showroom

48 correct • solved at: **00:19** by

Me[∰]tallica (Cambridge)

Author: **Jim**

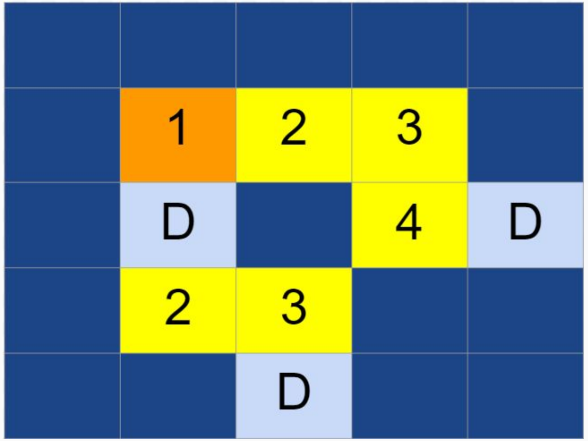
Overview

- A map of a car showroom with doors, cars and walls.
- There can be many doors in the outer wall leading to the target.
- Given the coordinates of a car in the showroom, how many cars must be moved in total.

Elegant Showroom - Solution

Techniques

- Dijkstra's algorithm
- Breadth-first search



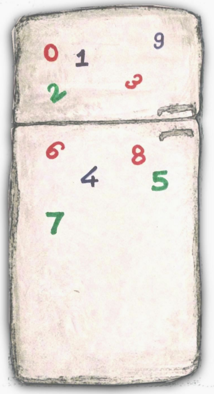
	1	2	3	
	D		4	D
	2	3		
		D		

Algorithm

- Read in the 'map' of the showroom and build a graph. Make a note of the doors on the edges.

- Use Dijkstra's algorithm to find the distance to the target car.
  - Weight each node. 1 for a car, 0 for a door.
  - Push all of the edge doors onto a priority queue at once, distance 0
  - Starting a new search from each door is slow.
    - About 1,500 times slower, in fact.

- See also: Sokoban for a harder challenge with the same idea



F - Fridge

114 correct • solved at: **00:12** by
Charles University in Prague

Author: **Robin**

Overview

- A single string of up to 1000 digits [0-9].
- Print the smallest positive integer that cannot be made without reusing any of those digits.
- Example:
  - 01123456789 → 22

Fridge - Solution

Techniques

- Counting
- Strings

```
    100...000
   111...111
  222...222
 333...333
444...444
 555...666
  777...777
   888...888
    999...999
```

Algorithm

- Find the digit with the fewest occurrences
  - The answer will be the digit repeated * (occurrences + 1)

- But in the case of zero, the answer has to be positive
  - So prepend a “1” as well

- Done!
  - Note: “up to 1000 digits” is a little too much to read into an unsigned long
    - And also slightly too large for iterating over all possibilities to work

G - Gondola

Not solved

Author: **Robin**

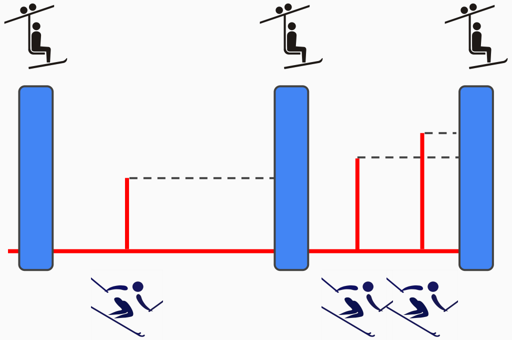
Overview

- People arrive at a mountain foot at certain times
  - They would like to get on their gondolas quickly
- You have a limited number of gondolas and must place them on the rotating track
- Minimise the sum of all waiting times

Gondola - Solution

Techniques

- Modular arithmetic
- Dynamic programming
- Convex hull trick



Algorithm

- First observations:
  - Arriving at time X is equivalent to arriving at time X+2×T
  - Gondolas should always coincide with someone arriving

- Assume we put the first gondola at X=2xT so cost=sum(arrivals)
  - We can add another gondola at time Y<X
    - This saves (X-Y) × count(arrival[i] <= Y)
    - And now we have a smaller instance
    - Dynamic programming takes O(N^3)
      - Or **$O(N^2)$** by using convexity properties

- One wrinkle: 2xT may not be the best place to put a gondola
  - So wrap the array around and try other end times

H - Rhyming Slang

93 correct • solved at: **00:17** by
ill_overflow_ur_NaN_m8 (Trinity College Dublin)

Author: **Jim**

Overview

- Read a number of lists of word endings. If two endings are in the same list words with those endings rhyme.

- Read a single common word and a number of possible phrases that could be rhyming slang for the common word.

- Output YES if the word and phrase rhyme, NO otherwise.

Rhyming Slang - Solution

Techniques

- Substrings
- Hashmaps

Algorithm

- Read in all of the endings and the common word.
  - We only care about rhyming sets where the common word matches at least one ending in the list.
- Put the set of possible rhymes into a hash set.
- For each possible rhyming phrase iterate over all possible suffix lengths for the end word.
  - Look them up in the hash set.
  - If any exist in there (possibly more than 1), write YES.

I - Grass Seed

161 correct • solved at: **00:03** by

IRL (Cambridge)

Author: **Jim**

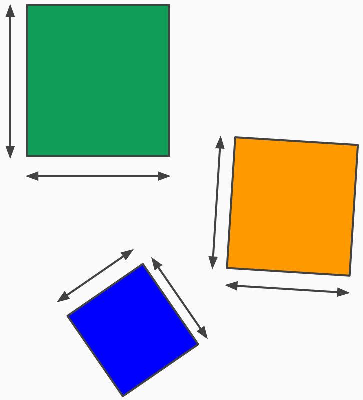
Overview

- Given:
  - The cost of seed for one square metre of lawn
  - Several lawn widths and lengths
- Calculate the total cost of seed.

Grass Seed - Solution

Techniques

- Floating point
- Multiplication



Algorithm

- For each lawn:
  - Read in width and height
  - Multiply to find the area
- Sum the lawn areas.
- Multiply the sum by the cost of the seed.
  - Print back out with %.6f, %.7f, etc.

J - Jack’s Beanbag

16 correct • solved at: **02:10** by

KTU United (Kaunas University of Technology)

Author: **Robin**

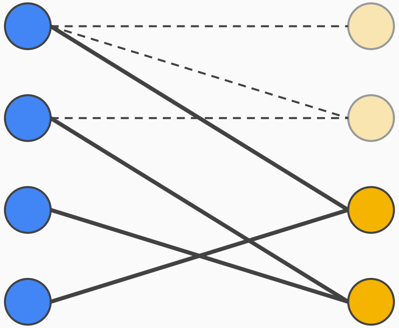
Overview

- N farmers each have a set, X
  - When asked, they will yield one item
  - But you can’t pick which one
- You want a certain number of each kind of bean
- After utilising the farmers’ supplies, how many more beans will you need to barter for?

Jack and the Beanbag - Solution

Techniques

- Brute force
- Combinations
- Set cover

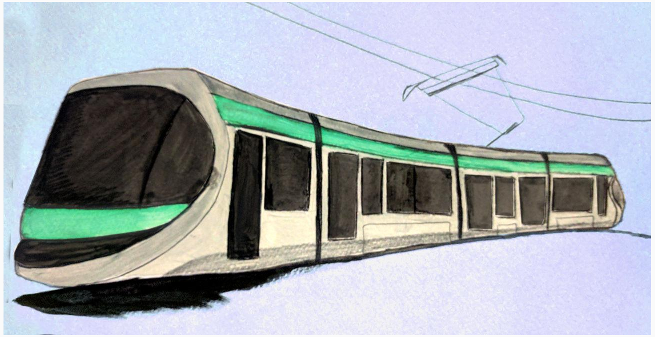


Algorithm

- Each farmer will give the full amount of at least one kind of bean.
  - Proof by induction: either you already had enough, or getting another bean brings Jack one step closer.

- The worst case is when farmers collude:
  - Each picks a kind of bean to always give and puts it in set S
  - Cost = sum(beans \ S)

- There are at most 2^B such sets---generate all of them, check if each makes a valid farmer selection, and take the smallest.
  - This is known as the set cover problem
  - Complexity: **$O(2^B \times N)$**



K - Compensation

2 correct • solved at: **02:10** by

Charles University in Prague

Author: **Robin**

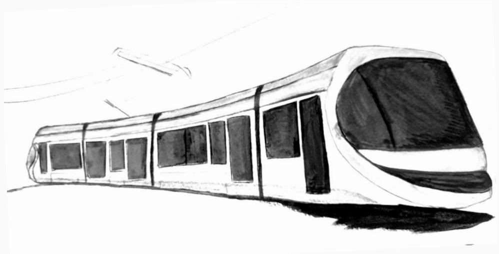
Overview

- Trains are scheduled at times X, Y
- But they are delayed, so actual departure/arrival times are X+C, Y+C
- What is the earliest train journey we can book so we are “delayed” by more than 1800 seconds?

Compensation - Solution

Techniques

- Dynamic programming
- Shortest paths
- Graphs



Algorithm

- Make two separate graphs, one “regular” version and one “delayed” version

- For every start train in the “regular” graph, find the shortest path provided we board exactly that train
    - (note: we booked it, so even if there’s a faster way, we must take the train we were scheduled to)
        - This caused a big sea of **WRONG-ANSWER**.
    - Cache and reuse repeated answers for {station,time}

- Another **fun** fact: we found a wrong judge solution halfway through. Luckily it was **not** the one we use for validating test data.

L - Secret Santa

61 correct • solved at: **00:11** by **IRL**(Cambridge)

Author: **James**

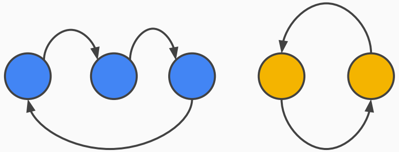
Overview

- We have N people in a town.
  - Each person picks up a unique name from the set, on a piece of paper
- What are the chances that someone (maybe several people) picked up their own name?

Secret Santa - Solution

Techniques

- Dynamic programming
- Permutations
- Infinite series

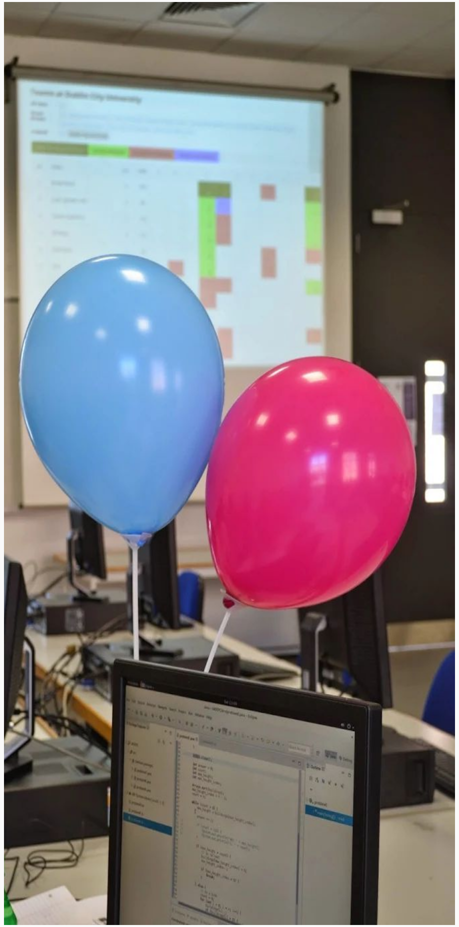


Algorithm

- Count the number of permutations with no fixed points
  - (also known as derangements)

- With N people, whoever person 1 gives a gift to may:
  - Give a gift in return
    - In which case answer[N] += answer[N-2] * (N-1)
  - Give a gift to someone else
    - In which case answer[N] += answer[N-1] * (N-1)

- Dynamic programming gives a fast solution for small N
- But N <= 10^12
  - Handily, the answer quickly converges to 1-(1/e)
  - After 8 in fact---so brute force works too


Teams at Dublin City University

Questions?

Or comments?

Final Standings

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