UKIEPC 2022

Summary and solution outlines

Problem Solutions

Alpine pass

19 correct • solved at: **01:31** by
lamelame
University of Oxford

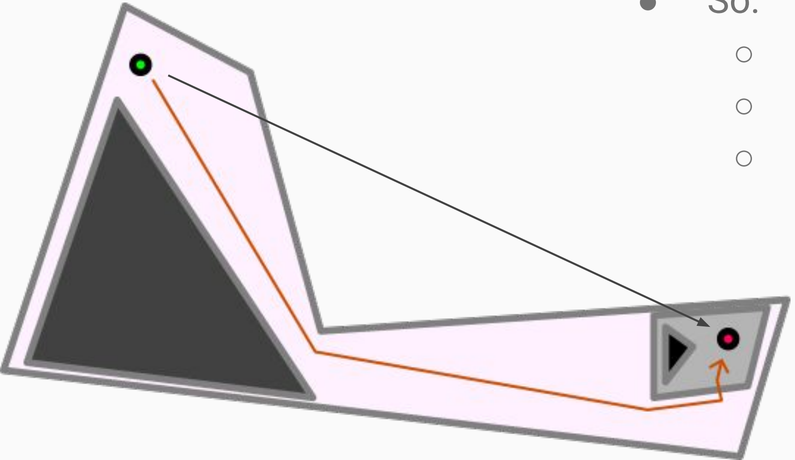
Overview

- Mountains are represented with contour lines.
- How high into the mountains do we need to go if we want to travel from point A to point B?

Alpine Pass

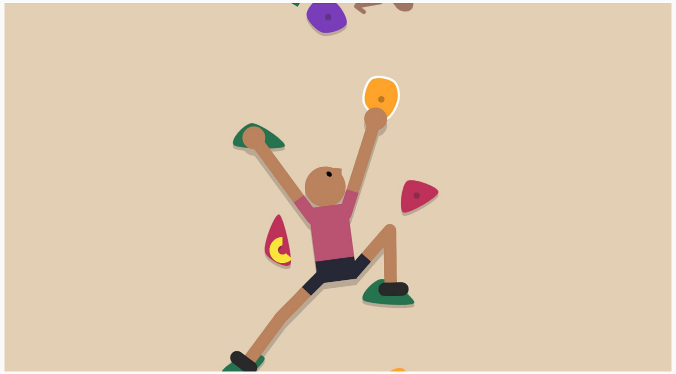
Techniques

- Topology
- Geometry



Algorithm

- Key observation about contours:
  - If we are **inside** and our goal is **outside**, we need to cross.
  - If we are **outside** and our goal is **inside**, we need to cross.
  - Otherwise, we will never need to cross this contour.
- So:
  - Find the lists of polygon contours containing the start
  - Find the lists of polygon contours containing the end
  - **XOR** the lists. These are the contours we need to cross.
    - We also need to take our start and end points into account, so find the smallest polygon enclosing each.



Boulder Wall

1 correct • solved at: **04:29** by
we <3 ronnie
University of Oxford

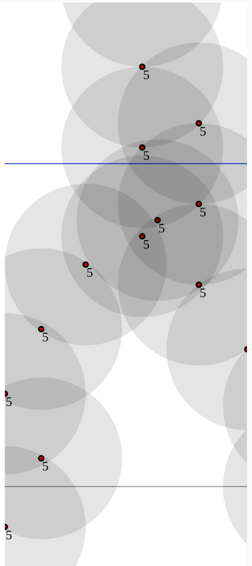
Overview

- Find a set of routes up and down a wall
- Multiple routes:
  - Must not touch horizontally
  - Must each consist of one colour
- Individual routes:
  - Must start and end below h=100
  - Must reach above h=H
  - Must go strictly up + strictly down
- Individual holds:
  - Must be at most 100cm apart

Boulder Wall

Techniques

- Dynamic programming
- Finger strength



Algorithm

- Separate colours are irrelevant; split them up right away and solve individually
- Assume we can get the optimal solution for a range of holds in x1…x2.
    - Use dynamic programming over this:
        - dp[T] = max(dp[u] + solve(u, T)] for all u < T
- How do we calculate solve(u, T)?
    - Imagine the ascent and descent are actually going up at the same time, and (assuming up1 > up2):
        - dp2[up1][up2] = interest[up1] + max(
          max(dp2[up2][i] forall 0 <= i < up2),
          max(dp2[i][up2] forall up2 < i < up1))
- Pitfalls:
    - Only link two nodes if distance^2 <= 10000
    - Need to introduce a special 0 node for “not used yet”
        - This node doesn’t care about distance, only height

Colour Mixing

63 correct • solved at: **00:00** by

TrinOI

University of Cambridge

Overview

- The expression R:G:B gives us a ratio of three numbers.
- We increase the values of R, G, and B one-by-one.
- How many unique ratios do we have along the way?

Colour Mixing

Techniques

- Fractions
- Greatest Divisors

Algorithm

- Keep a variable for the frequency of each of the three colours, and a hashset of ratios of colours each time.
- Before we put a new element in the ratio hashset, we must reduce it to a canonical form. For example 6:12:9 could equally become 2:3:4.
  - We'll create these canonical forms by taking the Greatest Common Divisor (GCD) of all three numbers
  - Classic recursive algorithm (shouting optional):
    - **WHILE** b > 0: (a, b) = (b, a **MOD** b)
    - **RETURN** b
  - In this case, brute force is also acceptable.
- Print the size of the hashset at the end.

Duck Crossing

47 correct • solved at: **00:33** by

Placeholder Team Name
University of Oxford

Overview

- People are lined up on one side
- Ducks are lined up on the other
- People match with ducks
  - Each person needs to hook a duck
  - Hooks are not allowed to cross
- How many phases of duck-hooking will we need to hook all the ducks without crossing hooks?

Duck Crossing

Techniques

- Sorting
- Longest Increasing Subsequence

Algorithm

- Consider ducks as (x, y) pairs. Sort them by x. The **longest increasing subsequence** is the most ducks we can have in one round at once (no inversions).
  - This can be found using Patience Sort, an algorithm which optimises the standard dynamic programming solution.
- But what about the number of subsequences we need to make?
  - Look at the **longest decreasing subsequence**.
  - Every item must be in separate rounds, giving a lower bound.
  - The elements together can form an increasing subsequence covering the whole array, giving an upper bound.
- Apply the same Patience Sort algorithm in reverse order.

Eager Packing

45 correct • solved at: **00:26** by
IMPostors
Imperial College London

Overview

- We pack parcels into vehicles greedily, always putting the largest available in
- There are N vehicles, and P parcels of various given sizes
- How large must the vehicles be if we want to pack all of the parcels?

Eager Packing

Techniques

- Binary search
- Amortisation

Algorithm

- Assuming we’ve fixated on a given size per vehicle,
  - We need to repeatedly find the largest remaining element less than or equal to remaining capacity X.
    - We can use a sorted tree structure (eg. std::map or TreeMap) to keep track of counts, and to implement a nextLessOrEqual function
      - std::upper_bound(x)-1 in C++
      - floorEntry(x) in Java
- Once this is done, binary search on the size per vehicle.

Fungible Growth

139 correct • solved at: **00:07** by

anonymous squids
University of Cambridge

Overview

- Two investment strategies, A, and B.
  - Gains of A = A% growth per year.
  - Gains of B = B% growth per year.
- Which strategy should we use for our investments?
  - Assuming we want to make as much money as possible.

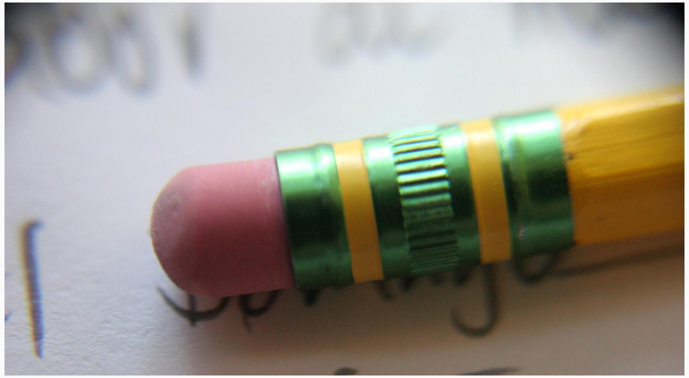
Fungible Growth

Techniques

- Input/Output
- Nitrogen

Algorithm

- Check whether A > B
    - If A > B, print A
    - If B > A, print B
    - It will never be the case that A = B, but in case it does, throw an exception like a good software engineer.



Grapheme Game

9 correct • solved at: **02:59** by

Holland King
University of Cambridge

Overview

- Each letter of the alphabet has a value.
- We start the game with a long word.
- Each player has to remove letters from the start and end of the word to leave behind a new one (or empty set).
  - Then the other player takes their turn, and again, and again, until there are no letters left.
- You receive a score equal to the sum of values of the letters you removed.
- What is the minimum score you can have if your opponent also plays to minimise score?

Grapheme Game - Solution

Techniques

- Aho-Corasick
- Zero-sum games
- Dynamic programming

Algorithm

- Imagine an algorithm where we have 3 choices:
  - delete_start, delete_end: Remove a letter, consider next move
  - pass: (If the word is still valid) give the word to the opponent
- Encode this function as a dynamic programming table:
  - For modifications: score[word] = (value[word]-value[word’]) + score[word’]
  - For handovers: score[word] = (value[word]-score[word]) **if** word **in** dict
- The starting string has O(N^2) possible reductions, with O(N) letters each, so this is too slow.
  - However, only O(N) of the reductions correspond to words.
  - The words are connected by an Aho-Corasick graph.
- For each word, its modified versions are close by in the graph:
  - Its fail-node is the result of removing letters from the start, and its parent is the result of removing a letter from the end.

Hare’s Breadth

26 correct • solved at: **00:23** by

we <3 ronnie
University of Oxford

Overview

- Greyhounds run from positions X[i] metres behind the starting line, at V[i] metres per second.
- Over time, the identities and relative distances of the first and last place greyhounds will change.
- At some time, the distance between first and last place is the minimum possible.
  - What is that distance?

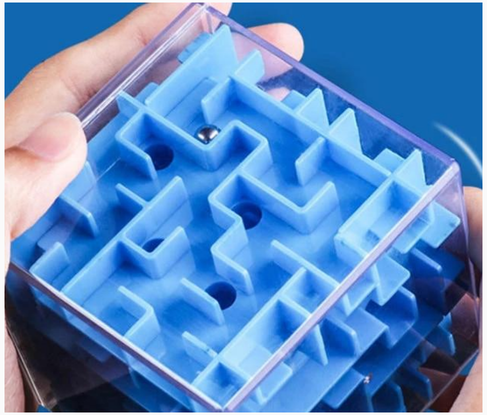
Hare’s Breadth - Solution

Techniques

- Lower Envelope
- Sorting

Algorithm

- The distance of the first-place at any time is on the **upper envelope** of the graph of distances travelled.
    - Likewise, the last-place dog is defined by the **lower envelope.**
    - The upper envelopes can be found with an algorithm similar to convex hull, but dealing with line segments, by sorting the dogs by starting position and inserting them at the end of the envelope polygon one-by-one.
    - Upper and lower envelopes are convex. Usually this means ternary search could give the closest place, however combining two envelopes may give us flat points in the graph.
- Instead, check every interesting X coordinated defined by vertices on the upper or lower and use the envelopes to calculate Y-Y’ fast.



Incline Game

62 correct • solved at: **00:29** by

one more try
University of Edinburgh

Overview

- You can make one move in a maze by tilting in any cardinal direction, and a ball will slide as far as possible in that direction
  - Either into a wall,
  - Or into the target hole.
- How many times do you need to tilt the board to win the game?

Incline Game

Techniques

- Shortest Paths
- Generated Graphs

Algorithm

- Split each cell (X, Y) into four:
  - (X, Y, West = travelling left
  - (X, Y, North = travelling up
  - (X, Y East = travelling right
  - (X, Y, South) = travelling down
- If we are already travelling in direction D, then the cost of continuing to move from point (X, Y, D) to (X+x[D], Y+y[D], D) is zero assuming the cell is empty.
  - If the cell is not empty, then the cost of changing direction is exactly one. Add edges between (X, Y, D) and (X, Y, W/N/E/S) if D leads towards an occupied cell.
- Run Dijkstra's algorithm, or a hybrid depth/breadth-first-search, to find the shortest path on this graph.

Joinery

73 correct • solved at: **00:04** by

ACMepted
University of Cambridge

Overview

- Find two pieces of wood which, when laid lengthways, will have a total length of at least T.
    - If there are multiple, pick the pair with the smallest total.

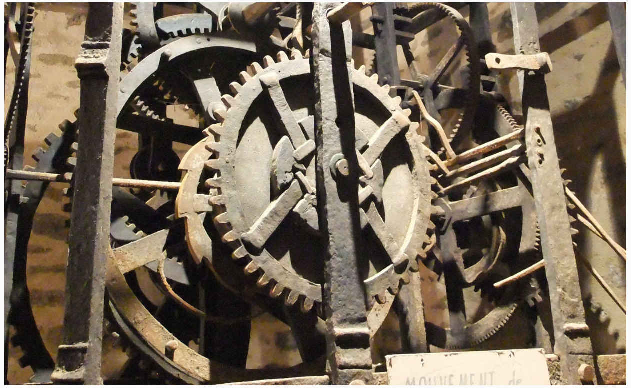
Joinery

Techniques

- Sorting
- Two Pointers

Algorithm

- Really, nodes of the same kind are **not** the same, we just need to go to any of them at some time T.
- So we can make a table of cost_to_visit[T][NodeId] and only fill it in for the relevant kinds of node at time T.
    - Iterate through T in increasing order and do an all-pairs comparison to find if:
        - Station at T is valid to leave from
        - Station at T+1 is valid to go to.
- Read off the minimum number in row T of the matrix at the end.



Keeping Time

36 correct • solved at: **01:30** by

we <3 ronnie
University of Oxford

Overview

- Gears turn against other gears, imparting a spin in the opposite direction.
  - Sometimes, two gears share an axle and turn at exactly the same speed
  - Sometimes, a loop of gears forms and either matches or breaks an existing set of relations.
- Are all of the gears in a given configuration able to turn freely?

Keeping Time

Techniques

- DFS
- Integer Fields
- Chinese Remainder Theorem

Algorithm

- If two gears are connected on the same axle or by meshing, the speed of one is a simple multiple of the other:
    - Same axle: v[b] = v[a]
    - Meshing: v[b] = v[a] * -(r[a] / [r[b])
- In short, we can run a DFS keeping track of the velocity of each gear relative to the first gear found on the DFS.
    - In case of visiting the same gear more than once in a DFS, check that the newly-calculated speed is the same as the old one.
- Problem: the constraints mean that the numbers involved can be huge, or tiny, and we need perfect accuracy.
    - Solution: store speeds modulo multiple large randomly-chosen primes, a la Chinese Remainder Theorem.

Lasso

1 correct • solved at: **04:31** by
Trinity's Trinity
University of Cambridge

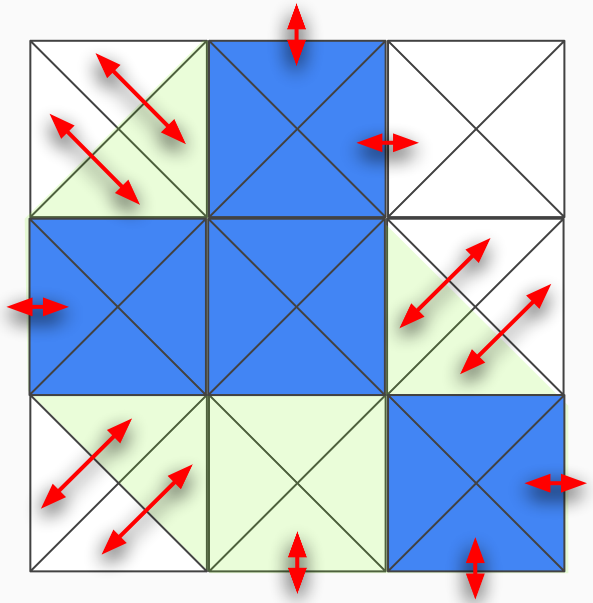
Overview

- Create one or more hulls fully containing N cells of a grid so that they can't get out of the grid.
    - The line segments of the hulls can be 0, or 45, or 90 degrees from the horizontal.

Lasso

Techniques

- Maximum Flow



Algorithm

- For the pure axis-aligned version of this problem, we're looking for the **minimum cut** separating the outside edges from the sheep fields.
    - Draw a 1-edge from every field to its 4 neighbours.
    - Draw an ∞-edge from the source to each border edge.
    - Draw an ∞-edge from each sheep field to the sink.
- The edges represent the necessity of removing every possible path from the border to the sheep fields, without removing the sheep fields or the borders themselves.
- With 0/45/90 degrees the problem is similar. Split each field into 4 parts, diagonal edges cost=√½ and straight edges cost=1.
    - With some more geometry and time complexity, this can be adapted for any angles since only O(N^2) lines are special.

Final Standings

http://ukiepc2022.cloudcontest.org/