Logbook AoC 2024

Notes for each submitted solution, extracted from `git-log`s.

Solution: Day 25 (`git diff e570204`)

Thats all for this year! Not much to say about today, it's fast enough to just "brute force".

I'm happy with my effort in this year's AoC! Although I didn't reach my goal of averaging sub 1000 (looking at days I did at launch or had the time in the morning to do; on average 1600) I am still happy that I was able to do each challenge. I'm also happy about these logs I've been writing throughout, they have been a great source of reflection and helped me to get over not getting the ranking I was aiming for and to rather focus on the learning experience. I found while reading them back that I had a tendency to blame a bad solution or ranking on "silly" mistakes I made rather than taking ownership of them. Making mistakes are part of your (lack of) skill and shouldn't be observed as an exception.

I REALLY hope this wasn't the last entry in AoC. The reason I have my doubts about this tradition is the abundance of llm-solutions in the leaderboard (how many is up for debate, but some accounts hade public repo's showing system prompts for AoC) and the way that it spoils the competition (between humans). I would be discuraged to have a creation I've made be used as a benchmarking tool so I'm hoping Eric still has the motivation to keep doing this, next year I'll have my sub 1000!

Solution: Day 24 (`git diff 8ff4b07`)

Every year there is allways that one problem which is both feasable and easier to do manually and I think today hosted that problem. Together with the fact that it's Christmas Eve, I've decided to not write a general solution today.

To get my solution, I looked at a rendering of the graph looked for irregularities *in the shape of the graph* that I used to determine some regular patterns. Then I just wrote a simple program to output any deviation from these patterns and manually checked for a possible swaps.

However, I do think a general solution would be outlined as; use the regular structure for both `zxx` and the OR operation as an expected pattern and with a DFT output all errors that exist and associate them to the OR operation or the z-output. Find overlap of these errors and hopefully it will even out to 4 swaps! And then some edge cases...

Solution: Day 23 (`git diff 953478c`)

This should have been easy, but I just could'nt figure out the algorithm for the last part for the longest time. I knew it was something close to what I had in the end, but I just kept making mistakes and rewrote it completely 3 or 4 times. I also didn't read the problem statment properly and solved for interconnections instead of direct connections. And guess what? I made that mistake for BOTH parts...

I'm not as certain anymore with graphs it seems, but if there are any more graph-problems, at least I'll be warmed up!

Solution: Day 22 (`git diff a43886d`)

The change in the PRICE, not the secret number... Oh well that's what I get for not reading the problem thoroughly. Other than that, nice and simple problem, was easy to solve with the right datastruct. All that's left to see is how fast I can get it to run in rust...

Solution: Day 21 (`git diff 2908a62`)

Amazing. Loved todays challange, it took some time to figure out aspects like how it's allways best to step many steps in the same direction consecutively as well as the fact that every step had to press the A button means that every subsequent robot need only to look at inputs leading up to a A-button press. This led me to the dynamic programming solution for part 2.

I also ran into one of the worst "bugs" I've ever had to deal with; due to it not being very unlikely that any of the shortest paths at any stage is also a component of the total solution, I made the mistake to assume after a SINGLE test that I only needed to look at one of the shortest paths at any layer. However, this is only true for the last layer. Tragically, sometimes the seed used for the hash-set ordered one of the correct paths of the keypad first when picking the shortest, and this led to me making the first incorrect assumption and also messed with my head when trying out different resolutions. Although I picked up on the random-ness of it and correctly assumed it had to do with hashing, I made the second (major-)blunder of thinking it was a hash-collision in the cashed recursion-function. It was a huge relief to finally detirmine that the hash-set being used for the keypad-layer was the culprit.

Again, amazing puzzle.

Solution: Day 20 (`git diff b0579f9`)

Phew, this one kicked my ass. Didn't finish it in time before leaving for school so had to do pick it up once I got back. I first tried exploring the maze with added rules to handle passing through walls but I ended up with some wierd off-by-one error. Solution was finding pairs of points some distance from both

the start and finish and let the manhattan distance determine if there exists a time save by cheating. There are probably some nice criteria that would reduce the amount of pairs of points that need to be considered, but the the current solution "only" takes around half a minute so I'm not gonna bother improving it.

Solution: Day 19 (`git diff 80dbbcc`)

Nice and easy problem, messed up a bit on part one by not using a visited set for each trial. It was hard (for me) to detect the error as well as I thought the error was that `startswith` had some extra functionality that I wasn't taking into account when comparing strings. But it was simply that I was brute forcing too hard. I liked the dynamic programming approach in part 2, nice and simple.

Solution: Day 18 (`git diff 0c91c22`)

Feels strange to have a simple maze-solver so far into AoC but he's probably just pacing the difficulty in preparation for tomorrow... But yeah, just a BFS and then brute forcing adding walls (currupted bytes) until there's no solution.

Solution: Day 17 (`git diff fa474f7`)

Good things come to those that wait, today was GREAT. Classic Eric move to do a Synacore esque challange where you have to dissassemble the input, f***ing love those. I could have gotten an even better placement had I realized that the example input was a red herring, but it was good to use it to realize that ANY output could be achieved since the program is a decoder, it was just that hade the solution for around 15 minutes before I tried to submit an answer. Hopefully I'm not breaking rules by committing my dissassembly, since you could theoretically retrieve the input I had, with one placement off.

Solution: Day 16 (`git diff b41a6aa`)

This was the worse day yet, part 1 I landed a leaderboard position of around 400 but then I for part 2 I could not for the life of me figure out how to do it without brute forcing. There is some serious gap of knowledge in my understanding of shortest path algorithms.

The brute force solution in the end is just looking back at each turn I've made and redoing the calculation from there, but this ended up with a massive amount of trial and error that took around 30s the first time I got it fast enough.

Solution: Day 15 (`git diff f574457`)

Today was a nice problem, I was fearing it would be one of those "now repeat the moves until the boxes return to a previous position" so I was glad that it was just a more involved algorithm. I thought I was doing really bad on time

but even the top 100 leaderboard struggled for over 15 minutes so it seems it was just a problem without shortcuts. I want more problems like these for the next few days, it's nice knowing that it's more about problem solving than being a fast typer.

Solution: Day 14 (`git diff ceb2bd1`)

Fun with a unique problem, should have picked up quicker on the "congregation" every 103 seconds after x seconds but I thought it was a pointless artifact. Given the first occurance was more than 2 hours in, it was a good thing that after I "only" 10 minutes gave up on watching everything at 2x speed and looked for patterns.

It was also funny to look at the leaderboards today, easy to pick out the proompt "engineers" as they were under half a minute on part one but then having no placement on part 2. It's almost like they're not capable of image recognition...

Solution: Day 13 (`git diff 06e8e59`)

The math aint' math'in. Messed up a single step in the linear equation solving, `- a * b + a * c != - (b + c)` which gave me a wrong result and I thus abandoned the idea thinking that because the the base vectors have different costs. I therefore thought that maybe part 2 had a different algorithm instead of just being longer and was happy about my in hindsight wrong conclusion. Should have just dubble checked my equation solving or just used `linalg.solve(a, b)`.

Solution: Day 12 (`git diff bd2da50`)

Felt a bit slow today even though I got better ranking than yesterday. It was one of those problems where the trap is to assume every case is present in the example input, which it wasn't (holes). I didn't immidiatelly spot it either in the real input and kept verifying cases that I already handled until I spotted a hole. I should start "thinking" about the problems rather than just looking through the examples, since today this misled me.

Judging the last couple of days, I seem to be stabalizing around 1000-1500 in ranking, which should be ok but I can't pretend and say I'm not dissapointed.

Solution: Day 11 (`git diff 88327ce`)

I don't know why I'm so mad about todays ranking. I think it is because I knew the solution right away but I wrote `10 % (e - 1)` instead of `10 % (e // 2)` to split the numbers and had to whip out a repl just to realize something so **simple**. But I've played AoC enough to know the oldest trick Eric has in his book which is why I jumped around 5000 from part 1 to part 2 by writing the algorithm so that it would anticipate part 2.

Solution: Day 10 (`git diff 36c85d4`)

There isn't something perticular to say about todays challenge, just a straight-to-the-point depth-first search. Ended up writing the algorithm that was correct for part 2 while writing the one for part 1, so solving part 2 was literally a "Undo", "Copy", "Redo", "Paste".

Solution: Day 9 (`git diff dc6bf49`)

I think I made it harder for myself in the first part, thought I was doing something clever but of course hade to pay the time cost of implementing it and it wasnt *that* fast either. And since part 2 couldn't work with that solution I had to rewrite it and while doing to messed up which index I was inserting spaces into which led me down a lengthy debugging jurney. It would be fun trying to find an effective solution for part 2.

Solution: Day 8 (`git diff c288acd`)

Ok, wasted time before properly realizing the meaning of ". . . so [the antennas] are all also antinodes!", ie. counting the antennas positions alongside the antinodes. But otherwise great success!

Solution: Day 7 (`git diff dbf3090`)

First one not done during release. Made a brute force version first but it was not fast *enough*. I'm satisfied that the depth first solution is as quick as it is.

Solution: Day 6 (`git diff f6194e3`)

I don't think it could have gone worse. That example input was basically ADVERSARIAL, nothing about it related to my input and I was too shortsighted to look beyond it. I think it's best at this point to abandon the idea of reaching top 1000. Unless this was an outlier, I don't have the skills required.

Solution: Day 5 (`git diff d3b91f1`)

Felt that I should have realised that a stable sort would do the trick, didn't realize it until way into part 2, wasted time writing a botched "insertion algoritm". Oh well. . .

Solution: Day 4 (`git diff cb7fdaa`)

As is evident by the amount of code, I could not come up with a way of getting the diagonals effectively in part 1. Maybe a better strategy would have been to do more like part 2 in part 1, going trough each coordinate and expanding the search from that point. At least I reversed the trend of dropping in ranking from doing part 2, but it's pretty clear why. . .

Solution: Day 3 (`git diff 6da4dfa`)

Today was a good day, I funnily enough predicted the solution would use regex and had pre-opened `findall` in the documentation. Didn't need it but still I called it! The drop in score was due to me forgetting that *don't* is not spelled *dont'*... Also that the `mul` instruction starts out as active. But still, best score I've gotten **so far**!

Yesterday's bad score is all in the past.

Solution: Day 2 (`git diff 709d1be`)

A bad run, the server lagged temporarily and I then used an invalid link for some time which cost a few minutes. I also (re-)defined a variable inside a loop, why can't python have scoped variables??? The second part went awful for the simple reason that I could not come up with a simple strategy and rewrote it 3 times and in that process messed up the logic borrowed from part 1. I allways got the same answer as the test-input as well which misled me for a while. Next day will be better.

Solution: day 1 (`git diff 57ec9e8`)

First part went very well, but I messed up on the last part by not reading the entire problem spec and missed the "multiply by the value" part and did therefore not get a good position in the end.