Ranges basic: Rigamarole

Praanto Samadder

February 21, 2024

This is the first section of ranges. I will attempt philosophers.

1 Parser

1.1 Parsing the trojan horse

Provided was a CSV file in the format of a txt file. Trivial implementation. Easier solution: split every section into it's own file and load each file individually.

1.2 Parsing lists

We are faced with clues that are separated by new lines where we can split our string to get a list of clues. Further splitting over the commas and converting each element in our list into an integer using `String.to_integer()` and we have the list that we want.

```
defp parseListOfLists filename do
  case File.read(filename) do
    {:ok, data} ->
        Enum.filter(Enum.map(String.split(data, "\n"), fn x ->
    String.split(x, ",") end), fn x -> x != [""] end)
    {:error, msg} -> {:error, msg}
  end
end
```

Listing 1: Splitting the lists with newlines and commas

We will also convert every 'inner list' into a tuple to help us in our pattern matching.

1.3 Generating our map

Generating our map can be done very easily using recursion. We take in a map and a tuple (with our destination, source and range parameters) and recursively add new elements to our maps.

Every iteration, we decrement our range until we have we reach `range == 0`.

```elixir
defp generate_map map, {_, _, 0} do
  map
end

defp generate_map map, {dest, src, f_length} do
  generate_map Map.put(map, src, dest), {dest + 1, src + 1,
    f_length - 1}
end
```

Listing 2: Generating maps

2 Problem solved

At this point, we have already solved our problem.

But what if the element we are looking for does not exist in the map? Elixir's `Map.get/3` function allows us to provide a default value which we can take advantage of in situations where we do not require a mapping to take place.

And then it's a series of pipe operations: getting a map result, passing it to the next map, loading the next map, getting the map result and repeat. Following listing provides a snippet.

```elixir
def rigamarole do
  Parser.parseSeeds() |> Enum.map(fn x -> String.to_integer(x) |> r
    () |> IO.inspect() end)
end

def r x do
  seed2soil = Parser.getSeed2Soil() |> Map.get(x, x)
  soil2fart = Parser.getSoil2Fart() |> Map.get(seed2soil, seed2soil
    )
  fart2water = Parser.getFart2Water() |> Map.get(soil2fart,
    soil2fart)
  water2ljus = Parser.getWater2Ljus() |> Map.get(fart2water,
    fart2water)
  ljus2temp = Parser.getLjus2Temp() |> Map.get(water2ljus,
    water2ljus)
  temp2hum = Parser.getTemp2Hum() |> Map.get(ljus2temp, ljus2temp)
  Parser.getHum2Pos() |> Map.get(temp2hum, temp2hum)
end
```

Listing 3: Printing the entire rigamarole

We can store all the outputs in a list and find the lowest to get the lowest position. Boom, done!