Ranges

Ayda Ghalkhanbaz

Spring Term 2024

Introduction

This report briefly covers the solution for the first task of day 5 of the advent of code 2023, where we have to transform a seed to soil, fertiliser etc. until we find the location number for that seed. We will delve into the implementation details and finally show some run tests indicating the program's functionality.

Method

The first task consists of two main parts; parsing the input data and the transformation. We'll look into each part in detail in the upcoming subsections.

Parse

Since the input data has a special way of formatting, the `parse/1` function is specially designed to handle this data. First the input is splitted after double newlines and divided after seeds and maps. Parsing the seeds list was quite straightforward since we only had to split it after a space and then parse every number to an integer.

Although, parsing the maps wasn't challenging, it took a while to understand how to split them as splitting each map lead to nested lists and the structure became more complicated. Finally the `parse/1` returns a seed list and another list containing all the mappings.

Transform

The transformation of seeds is divided into four functions. The first function `transform/2` is called by user/test function and takes a seeds list and maps lists. This function then calls `transformAllSeeds/3` that in addition to the two mentioned lists, takes an empty list as an accumulator. This accumulator is used for keeping the final number of a transformed seed.

The `transformAllSeeds/3` has the job of exploring all seed numbers existing in the seeds list. It employs the function `transformOneSeed/3` and sends the 3 mentioned arguments as parameters for the next function. `TransformOneSeed/3` traverses all of the mappings and applies the relevant transformation on one single seed. In order to find the appropriate transformation mapping and calculations, the `transformOneSeed/3` function calls the last function for this part, i.e. `search/2`.

The `search/2` function only checks if the seed number matches one of the mappings in each transformation part. You'll find the implementation of these 4 functions in the code snippet below.

```
def transform(seeds, maps) do
transformAllSeeds(seeds, maps, [])
end
def transformAllSeeds([], _, acc) do Enum.reverse(acc) end
def transformAllSeeds([seed|seeds], maps, acc) do
  newAcc = transformOneSeed(seed, maps, acc)
  transformAllSeeds(seeds, maps, newAcc)
end
def transformOneSeed(seed, [], acc) do [seed|acc] end
 def transformOneSeed(seed, [map|maps], acc) do
  case search(seed, map) do
    {:ok, diff} ->
      newSeed = diff + seed
      transformOneSeed(newSeed, maps, acc)
    {:unchanged, seed} -> transformOneSeed(seed, maps, acc)
  end
 end

 def search(seed, []) do {:unchanged, seed} end
 def search(seed, [[to, from, len]|map]) do
  if (seed >= from and seed <= from+len) do
    diff = to - from
    {:ok, diff}
  else
    search(seed, map)
  end
 end
```

Result

The program has been tested with two input data; the sample data given through the task and a larger dataset which was provided as a csv file along with the assignment. Running the program on these files resulted in 35 and 389056265 as the lowest location number for the sample data and the larger data, respectively.