Ranges: Task I

Ruxandra-Stefania Tudose

Spring Term 2024

Introduction

This report aims to open a window onto the implementation of the first task of the Ranges assignment, which corresponds to Day 5 of the Advent of Code 2023. In the end, the goal is to find the smallest mapped location out of all the given seeds.

General Approach

The general approach consists in first and foremost, parsing through the file and storing the list of the seeds and the lists of the maps under the form of tuple. In order to do so, the *parse/1* method has been implemented and the Enum module has been used in order to apply different functions to the enumerable elements.

```
def begin() do
   str = File.read!("input.txt")
   {seeds, maps} = parse(str) #return the seeds and the maps
   [h | t] = location({seeds, maps}) #return the list of all the seed locations
   mini([h | t], h) #return and find the lowest possible location
 end
```

The next step is to set up a method that finds the valid location for each seed. In other other words, we have to once again run through the seeds and apply the rules as indicated in the task. Therefore, the condition in the if statement is as follows: if the seed number is greater or equal than the source and it is less than the highest number in the range, than we map it accordingly. If the seed cannot be mapped in one of the situation, the result is the seed itself.

Last but not least, in order to find the smallest possible location out of all the given seeds, the *mini/2* method has been created. As parameters, we send the list of the seed locations and the list's head. If while going through

the list, we encounter a smaller number, we go to the next recursive step with it. The base case takes place when the list becomes empty and the final result is, therefore returned. The function is also tail recursive.

```
  def location({seeds, maps}) do
  Enum.map(seeds, fn(seed) ->
    Enum.reduce(maps, seed, fn(map, nr) ->
      Enum.find_value(map, nr, fn({tr,d,s,r}) ->
        if((s <= nr) and (nr < (s + r))) do #the mapping condition
          d + (nr - s)
        else
          :nil
        end
      end)
    end)
  end)
  end

 def mini([], n) do n end
  def mini([h | t], n) do
    if(h <= n) do
      mini(t, h)
    else
      mini(t, n)
    end
  end
```

Conclusion

After having tested that the algorithm properly works for the small provided example, I have tested it using the input provided file, for which I have obtained 84470622 as being the smallest possible location out of all the given seeds.