Ranges

Dhruba Das

Spring Term 2024

Introduction

This assignment is pretty short: to sum it up, we are given seeds, which we have to map to their final locations, using several maps in the middle to map the seed no. to the soil no., and the soil no. to the fertilizer no. and so on, until we final reach the location no. corresponding to that particular seed. The transformations (i.e special rules used to map the seeds from one map to the next) are given as: *destination*, *source* and *range*, 50 98 2 for example.

Parse method

We are given the seeds and maps as one sample text, which we make "workable" by parsing each part separately. This is done by the *parse* method, which first splits the entire text into a list, with the seeds as the first element. We then further extract the seeds no. into a list by splitting the string containing the seeds further, and mapping the seeds to a list after parsing the numbers using $Integer.parse/1$. We do something quite similar for the maps as well, but we don't just use one map; we use a map for each jump, from seed to soil, soil to fertilizer and so on:

```
def parse(text) do
...
maps = Enum.map(maps, fn(map) ->
   [_|transf] = String.split(map, "\n")
    Enum.map(transf, fn(tr) ->
     [d, s, r] = Enum.map(String.split(tr, " "), fn(nr) -> {n,_} =
      Integer.parse(nr); n end)
     {:tr, d, s, r}
    end)
  end)
end
```

We represent a transformation as a tuple which looks like: {:tr, d, s, r} , *d*, *s* and *r* representing the *destination*, *source* and *range* respectively.

Location method

This is the method which is used to move from map to map, given a tuple containing the seeds and the maps. It takes a map, each transformation(from that map), and a number; if the transformation has a range into which the number falls into, we then transform that number to it's corresponding number in that map, otherwise we check the if the next transformation has the desired range. If number doesn't fall into the range of any of the transformations, the number retains it's value and we move to the next map:

```
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
            d + (nr - s)
          else
            nil
          end
   ...
end
```

Final result

After parsing and traversing the maps, we get the final location for the seeds [79, 14, 55, 13] to be : [82, 43, 86, 35]