Advent of Code Day 1

Emil Göransson

Spring Term 2023

Introduction

This report will cover the solution to the problem of day 1 in Advent of code 2022.

Explaining the problem

In day 1 of advent of code we are given a text file containing multiple chunks/bits of integers grouped/separated by an empty line. The problem comes down to summing all the entries in each chunk, and comparing them to each other (size wise). There are two parts to this problem, 1) Finding the largest chunk of them all & 2) Summing the three largest chunks of data.

Basically given the following entries:

```
1000
2000
3000

4000

5000
6000
```

The solution to Part 1 is: 5000+6000, and Part 2): 1000+2000+3000 + 4000 + 5000+6000

Solving day 1

The first issue is deciding on how I would go about handling/saving the data. I found it the most straight forward to save the data inside a large list by first reading from the text file and removing every newline.

```
{:ok, data} = File.read("input.txt")
foo = String.split(data, "\r\n")
```

A second issue that occurred because of this approach was that the numbers where actually strings. To solve this I iterated though the list, calling Integer.parse(x) on every element.

```
Enum.map(foo, fn x ->
    case Integer.parse(x) do
      :error -> ""
      {val, _} -> val
    end
  end)
```

In the case of an error (if x is "") we "replace" it by another "". Otherwise we replace the string-integer by a real Integer.

Recursively adding the elements

Now that the data is in the correct format it comes down to adding the numbers. Basically my recursive function boils down to starting of in index 0 of the list . and recursively summing the head of the list to the tails head until a element containing "" is reached. When "" is reached we can finally sum the first chunk of data and start summing the next chunk. Once the tail is empty, the recursion will reach the basecase and stop. This is archived with the following code.

```
  def sumList([]) do
  IO.puts(:STOP)
end
def sumList([h | ["" | t2]]) do
  [h | sumList(t2)]
end
def sumList([h | [h2 | t2]]) do
  test([h + h2 | t2])
end
def sumList([h | nil]) do
  [h]
end
def sumList(h) do
  h
end
```

The finale

Now that sumList will return a list containing every chunk of data I used

```
sortedList = Enum.sort(sumList(foo2), &(&1 >= &2))
```

to sort the data so that we can find the solution to the first part with the following code:

```
solution1 = Enum.at(sortedList, 0)
```

and the solution to part 2 by adding the first 3 entries with the following snippet of code:

```
solution2 = Enum.at(sortedList, 0) + Enum.at(sortedList, 1) +
    Enum.at(sortedList, 2)
```