First year project
Project 99: Who's Julia?

Group: 99a
Simon Lehmann Knudsen, simkn15
Sonni Hedelund Jensen, sonje15
Asbjørn Mansa Jensen, asjen15
DM501

May 25, 2016

Contents

1 Introduction **3**
- 1.1 Julia . . . 3
- 1.2 Features of Julia . . . 3
- 1.3 Syntax differences . . . 3

2 Projecteuler 11 **3**

3 Projecteuler 116 **6**

4 Quicksort **6**

5 Statistics **6**

6 Learning / Personal experience **6**

7 Conclusion **6**

8 Appendix (source code) **7**

1 Introduction

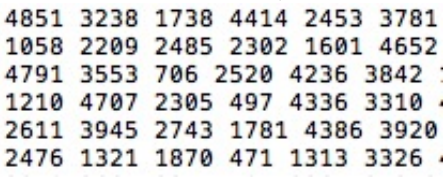
1.1 Julia

1.2 Features of Julia

1.3 Syntax differences

2 Projecteuler 11

In the 20x20 grid below, four numbers along a diagonal line have been marked in red. The product of these numbers is 26 x 63 x 78 x 14 = 1788696. What is the greatest product of four adjacent numbers in the same direction (up, down, left, right, or diagonally) in the 20x20 grid?



The algorithm first reads through a file with the input and makes a matrix. To calculate products in all directions needed in the problem, the algorithm goes through the matrix a total of 3 times. A nested for loop is needed to go though a matrix, lines 1-2 at figure 1. The outer loop runs through each row, and the inner loop runs through the columns. Figure 1 calculates the horizontal and vertical directions, firgure 2 diagonally from left to right(downwards) and figure 3 going diagonally from right to left(downwards). The algorithm starts in the most upper left cell, and iterates through all cells in the matrix. Lets have the first cell as (1,1), first number is row(**i**) and second number is column(**j**). Looking at figure 1, **matLength** is the size of the matrix, size = 100 would mean a matrix of size $100 \cdot 100$. **numProd** is the number of adjacent numbers multiplied together. For matrix with size 100, and multiplying four adjacent numbers, the algorithm would do the following in figure 1: Make visual example of algorithm

Starting at cell (1,1) to the end which is cell (100, 100 - 4)

Line 5-7: Loops over the adjacent cells and multiplies the numbers.

Line 8-10: Sets current product to max product if current is larger than previously max product.

```
for i = 1 : matLength
  for j = 1 : matLength - numProd
    #right/left
    prod = 1
    for k = 0 : numProd - 1
      prod *= mat[i, j + k]
    end
    if prod > maxProd
      maxProd = prod
    end
    #up/down
    prod = 1
    for k = 0 : numProd - 1
      prod *= mat[j + k, i]
    end
    if prod > maxProd
      maxProd = prod
    end
  end
end
```

Figure 1: Horizontal and vertical

Figure 2 shows the loop which iterates over the matrix and calculating the product of the diagonal going downwards from left to right. Figure 3 shows the diagonal product going upwards from left to right. The loop starts in the first row, and starting column is the last, not the first like the previously loops.

```
#diagonal left->right
for i = 1 : matLength - numProd
  for j = 1 : matLength - numProd
    prod = 1
    for k = 0 : numProd - 1
      prod *= mat[i + k, j + k]
    end
    if prod > maxProd
      maxProd = prod
    end
  end
end
```

Figure 2: Diagonal downwards left to right

```
#diagonal right->left
for i = 1 : matLength - numProd
  for j = matLength : -1 : numProd
    prod = 1
    for k = 0 : numProd - 1
      prod *= mat[i + k, j - k]
    end
    if prod > maxProd
      maxProd = prod
    end
  end
end
```

Figure 3: Diagonal upwards left to right

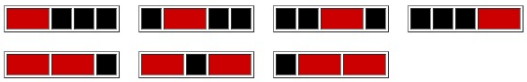
The program was made to run from commandline/terminal in order to be able to test on different inputs and amount of adjacent numbers multiplied. To run the julia version from the terminal: **Julia euler11.jl 100 4**. This would make the program run a matrix of size 100 and calculating product of 4 adjacent numbers. There's been made a matrix generator, found in appendix, which makes a file with data to test. The file would be named

mat100.txt for data to a $100 \cdot 100$ matrix. The numbers in the file ranging from 100-9999, 3-4 digit numbers. The original problem states two digit numbers in the data, but this would make a lot of duplicated numbers when testing with larger inputs, e.g. a $5000 \cdot 5000$ matrix. Some programming languages optimizes code during run time if it can predict what a given result will be. Having many duplicated numbers in the dataset could have an influence of the benchmarking between languages. To avoid this situation, to some extent, the digits have been increased.

3 Projecteuler 116

A row of five black square tiles is to have a number of its tiles replaced with coloured oblong tiles chosen from red (length two), green (length three), or blue (length four).

If red tiles are chosen there are exactly seven ways this can be done.



If green tiles are chosen there are three ways.

And if blue tiles are chosen there are two ways.

Assuming that colours cannot be mixed there are 7 + 3 + 2 = 12 ways of replacing the black tiles in a row measuring five units in length.

How many different ways can the black tiles in a row measuring fifty units in length be replaced if colours cannot be mixed and at least one coloured tile must be used?

4 Quicksort

5 Statistics

6 Learning / Personal experience

7 Conclusion

8 Appendix (source code)