Problem Statement

The four adjacent digits in the 1000-digit number that have the greatest product are $9 \times 9 \times 8 \times 9 = 5832$.

7316717653133062491922511967442657474235534919493496983520312774506326239578318016984801869478851843858615607891129494954595017379583319528532088055115125406987471585238630507156932909632952274430435576689664895044524452316173185640309871112172238311362229893423380308135336276614282806444486645238749303589072962904915604407723907138105158593079608667017242712188399879790879227492190169972088809377665727333001053367881220235421809751254540594752243525849077116705560136048395864467062441572215539753697817977846174064955149290862569321978468622482839722413756570560574902614079729686524145351004748216637048440319989000889524345065854122758866688116427171479924442928230863465674813919123162824586178664583591245665294765456828489128831426076900422421902267105562632111110937054421750694165896040807198403850962455444362981230987879927244284909188845801561660979191338754992005240636899125607176060588611646710940507754100225698315520005593572972571636269561882670428252483600823257530420752963450

Find the thirteen adjacent digits in the 1000-digit number that have the greatest product. What is the value of this product?

Solution

Honestly, overall not a very interesting problem. There are two complications in this problem: the number and product might exceed the maximum number of digits the data types can hold and divide by zero errors. Nominally, though, I would implement a class that would handle the representation as well as basic arithmetic. Python addresses the first one with it's own internal way of handling integer representation. However, for fun, I implemented a version of this code in python. Note, in languages like FORTRAN, you can create data types of arbitrary size, also avoiding this problem. I'm not completely up-to-date on what languages like Java or C++ can do, but there is likely already a library that exists to solve this problem.

As for the second complication, the naive algorithm of multiplying by the next number and dividing by the previous number to create the product breaks due to the inclusion of zero. In the code, each product is calculated from scratch at each index giving an order nk algorithm, where n is the number of adjacent

digits of interest and k is the number of digits in the number. In this case $n = 13$ and $k = 1000$.

The solution then is:

$$5 \times 5 \times 7 \times 6 \times 6 \times 8 \times 9 \times 6 \times 6 \times 4 \times 8 \times 9 \times 5 = 23514624000 \qquad (1)$$