PrimeNumbers Pseudocode

Lucky Mahlangu, Mpinane Mohale, Thulisile Shipyana, Sbusiso Mkhombe

22 August 2018

PrimeNumbers

Algorithm: PrimeNumbers(X)

Input:X, where X is a positive long integer.

Output:Primes[],N, where Primes is an array of prime numbers less than X, and N is the length of Primes.

```
01 ~Primes ←[]
02 ~counter ← 2
03 ~While counter<=X
04 ~~~~isPrime ← True
05 ~~~~For i from 2 to X
06 ~~~~~~~~If mod(counter,i)==0 and (counter not equal to i)
07 ~~~~~~~~~~~~isPrime ← False
09 ~~~~~~~~~~~~break
10 ~~~~If isPrime
11 ~~~~~~~~add counter into Primes
12 ~~~~counter ← counter+1
13 ~N ← size of Primes
14 ~Return Primes and N
```

Proof Correctness of PrimeNumbers

Proof by Induction

We will use proof by induction to show that the algorithm *PrimeNumbers* will return the correct array and the output N for any *valid input*. The formal problem statement is as follows:

Theorem : *PrimeNumbers* computes corretly for any valid input.

Proof.

Base Case:

Let the number inputed be 2. The algorithm first initializes Primes to be an empty array, it then sets counter to be 2, followed by testing the condition of the *while* loop. The condition counter $\leq$ X is tested and passes (2 $\leq$ 2), the algorithm moves to the next line and sets isPrime to True. Then the *for* loop in line 05 is executed, for each pass of the *for* loop the body of the loop is executed, that is, the two conditions of the *if* statement are executed. The first condition checks for mod(counter,i)==0 which passes (mod(2,2)==0), that is, 2 is divisible by 2 but the second condition counter is not equal to i fails because (counter=2=i). Likewise for all i not equal to 2, mod(counter,i)!=0, in conclusion the body of the if statement will never be executed for all passes of the *for* loop thus isPrime will remain True. Thus the *if* statement in line 10 will be passed, next the algorithm will execute line 11 , adding counter=2 into Primes. Line 12 increments counter to 3, going back to line 03 counter=3 $\leq$ X=2 fails, making the *while* loop to stop executing. Then line 13 sets N to be the length of Primes which in this case is 1. Therefore the algorithm returns Primes=[2] and N=1 as expected.

The algorithm works for an input of X=2. The *Base Case holds True.*

Induction Hypothesis:

Assume that *PrimeNumbers* returns the correct answer for an input k, where k $>$ 0.

Induction Step:

We need to show that the algorithm returns the correct answer for an input k+1, after executing the k-th iteration of the algorithm we then make an extra iteration called the (k+1)-th iteration. counter will be having the value k+1, thus the body of the *while* loop in line 03 will be executed. isPrime will be set to True. We therefore consider two cases(Case 1: k+1 is not a prime number, Case 2: k+1 is a prime number), the first case occurs when the *if* statement in line 06 is passed, setting isPrime to False, thus breaking out of the *for* loop, then the condition in line 10 will be false because isPrime is False. Thus k+1 will not be added in the Primes array since it is not a prime number as expected, then by the *Induction Hypothesis* the algorithm will return the correct array. Now considering case two, the *if* statement in line 06 fails, the body of the *if* statement will not be executed. Then line 10 passes since the value of isPrime is still True, then k+1 will be added in the Primes array, since it is a Prime number, by the *Induction Hypothesis* the algorithm will return the correct array.

Thus by the principle of mathematical induction our result is proved.

Therefore, *PrimeNumbers* computes corretly for any valid input.