Exercise 04

Sunday, November 12, 2023 6:02 PM

Problem 1 to hand in: Algorithm Example

Write an algorithm for the following problem:

Given a positive integer n, return a list that contains all prime numbers p with distance 1 to a power of two, $2 \le p \le n$.

- a) Provide a representative test example.
 - 1 Input: n = 5

Return should be [3]

@ Input: n = 100

Return should be [3, 5, 7, 17, 31]

b) Describe an algorithm that solves this problem intuitively.

Intuitively, the algorithm should suffice 3 requirements:

- O main: able to iterate number from 2 to n

 (A simple for-loop might do the trick)
- prime number: able to distinguish a prime number
 (An input: number, output: boolean function might
 satisfy the need)
- ② $2^{k}-1/2^{k}+1$: able to judge if a number has a distance of 1 to any power of 2 (Similar to ③)

The two functions @ and @ shall be integrated in the loop @. when both conditions are fulfilled, we store the number in a array and in the enclineturn the array after iteration.

c) Formulate the algorithm in Pseudocode. For a better learning effect, use as few Python-specific functions as possible.

Here we suppose the input is correct (positive integer = 2)

main(n):

$$A \leftarrow \emptyset$$

$$i = 2$$

while True:

if
$$2^{i}-1 > n$$
:

return A

else:

if
$$2^i + i < n$$
:

$$A = add-prime(A, 2^{i}+1)$$

$$A = add_prime(A, 2^i - 1)$$

add-prime (A,p):

for $i \in 2$ to p-i:

if $p \mod i \equiv 0$:

return A

add-prime (A, p):

for $i \leftarrow 2$ to $(p-1) \operatorname{div} 2 + 1$

if p mod i ≡ o:

return A

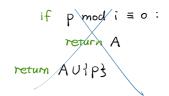
note:

no need to

go through

all the numbers < p

Advanced Programming and Algorithms 1 Page 1



if $p \mod i \equiv 0$:
return A

return AUIP3

no need to go through all the numbers < p

d) Analyse the asymptotic worst-case running time of your algorithm.

In the worst case scenario:

The while-loop in the main algorithm has a running time in $\hat{I} = \log_2(n+1) \in O(\log_2 n),$

the odd-prime algorithm has a running time (suppose the p is a prime number) in

$$O(\rho) \approx O(2^{\frac{1}{2}}) = O(2^{\log_2(n)}) = O(n) \qquad O(\frac{\rho}{2}) \Rightarrow O(n)$$

So in combination, the algorithm has a running time in O(nlog2n)

- e) Provide a proof sketch that the algorithm is correct.
 - 1. Prove that loop invariant in add-prime is correct:

 S[i]: at the beginning of the i-th iteration,

 P is proven not to be divisible by 2,..., i-1.
 - 2. Prove that add-prime returns A if p is composite num, and AUIP's if p is prime num.
 - 3. Prove the loop invariant in the main algorithm is correct: S[i]: at the beginning of each loop, $2^{i-1}-1 < n \text{ and } A \text{ contains prime numbers in}$ a form of 2^R+1 or 2^R-1 with $1 \le k \le i-1$
 - 4. Prove the main algorithm indeed return the required list.