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, $2 \leq p \leq n$.

a) Provide a representative test example.

① Input : $n = 5$

Return should be [3]

② 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:

① main: able to iterate number from 2 to n
(A simple for-loop might do the trick)

② prime number: able to distinguish a prime number
(An input: number, output: boolean function might satisfy the need)

③ $2^k - 1 / 2^k + 1$: able to judge if a number has a distance of 1 to any power of 2
(Similar to ②)

The two functions ② and ③ shall be integrated in the loop ①. when both conditions are fulfilled, we store the number in a array and in the end return the array after iteration.

we store the number in a array and in the end

return 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 ← ∅
    i = 2
    while True:
        if 2^i - 1 > n:
            return A
        else:
            if 2^i + 1 < n:
                A = add_prime(A, 2^i + 1)
            A = add_prime(A, 2^i - 1)

add_prime(A, p):
    for i ← 2 to p-1:
        if p mod i ≡ 0:
            return A
    return A ∪ {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

$$i = \log_2(n+1) \in O(\log_2 n),$$

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

$$O(p) = O(2^i) = O(2^{\log_2(n)}) = O(n)$$

So in combination, the algorithm has a running time in $O(n\log_2 n)$

c) 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 $2, \dots, i-1$.

2. Prove that add-prime returns A if p is composite num, and $A \cup \{p\}$ if p is prime num.

3. Prove the loop invariant in the main algorithm is correct:

   S[i]: at the beginning of each loop, $2^{i-1} - 1 < n$ and A contains prime numbers in a form of $2^k + 1$ or $2^k - 1$ with $1 \le k \le i-1$

4. Prove the main algorithm indeed return the required list.