Thực hành 3

@March 13, 2023

II. Giải công thức đệ quy xác định độ phức tạp thuật toán

1. $T(n) = \begin{cases} 1 \text{ if } & n = 0, n = 1 \\ T(n-2) + 2 \text{ if } & n > 1 \end{cases}$

$$
\begin{aligned}
T(n) &= T(n-2) + 2 && (1)\\
&= T(n-4) + 4 && (2)\\
&= T(n-8) + 6 && (3)\\
&= \dots && (4)\\
&= T(n - k*2) + 2*k && (5)\\
&= T(n\%2) + n && (6)
\end{aligned}
$$

$$\rightarrow O(n)$$

2. $T(n) = \begin{cases} aT(n/b) + n^c & \text{if } \ n > 0 \\ d & \text{if } \ n = 1 \end{cases}$

III. Viết chương trình cho bài toán

(Sử dụng giải thuật đệ quy thực hiện ít nhất 3 bài trong các bài sau)

1. Tổng các chữ số của một số

```
# 1. Liệt kê các số nguyên tố nhỏ hơn n
def sum_digit(n):
    if n < 0: n = -n
    if n < 10:
        return n
```

```
    else:
        return n%10 + sum_digit(n//10)
print(sum_digit(-99000))
```

2. Liệt kê các số nguyên tố nhỏ hơn n

```
# 2. Liệt kê các số nguyên tố nhỏ hơn n
def check_prime(n):
    if n < 2:
        return False
    else:
        if n == 2: return True
    for i in range(2, int(math.sqrt(n))+1):
        if n % 2 == 0: return False
    return True
def prime_recursive(n):
    answer = []
    if n < 1:
        return answer
    if check_prime(n):
        answer = answer + [n]
    answer += prime_recursive(n-1)
    return answer
print(prime_recursive(10))
```

3. Liệt kê n số nguyên tố đầu tiên

```
# 3. Liệt kê n số nguyên tố đầu tiên

def check_prime(x, lst):
    for i in lst:
        if x % i == 0:
            return False
    return True
def first_n_primes(cur, n, lst):
    if cur == 1:
        lst = [2]
    elif cur == 2:
        lst.append(3)
    elif cur == n+1:
        return lst
    else:
        x = lst[-1] + 2
        while not check_prime(x, lst):
            x = x + 2
        lst.append(x)
    return first_n_primes(cur+1, n, lst)
lst = []
```

```
lst = first_n_primes(1, 10, lst)
print("first n primes")
print(lst)
```

4. Đảo ngược một chuỗi (Reverse a string)

```
# 4. Đảo ngược một chuỗi (Reverse a string)
def reverse_string(x):
    if not x:
        return ""
    else:
        return x[-1] + reverse_string(x[:-1])
print(reverse_string("hihihaha"))
```

5. In ra tam giác Pascal có n tầng

```
# 5. In ra tam giác Pascal có n tầng
def pascal(curr, n, result):
    if curr == n+1:
        return

    if len(result) > 1:
        old = result[0]
        for i in range(1, curr-1):
            tmp = old + result[i]
            old = result[i]
            result[i] = tmp
    result.append(1)
    print(result)
    return pascal(curr+1, n, result)
res = []
pascal(1, 5, res)
```