1. Sort the following functions in increasing order of asymptotic (big-*O*) complexity and explain why:

$f_1(n) = n^{\sqrt{n}}$

$f_2(n) = \log(n^{100000})$

$f_3(n) = 2^{2^{100000}}$

$f_4(n) = 2^n$

$f_5(n) = n^{10} \cdot 2^{n/2}$

$f_6(n) = n\sqrt{n}$

$f_7(n) = \sum_{i=1}^{n} i + 1$

$f_8(n) = 100000n$

Answer:

$f_1(n) = n^{\sqrt{n}} = (2^{\lg n})^{\sqrt{n}} = 2^{\sqrt{n} \cdot \lg n}$

$f_2(n) = \log(n^{100000}) = 100000 \log n = \theta(\log n)$

$f_3(n) = 2^{2^{100000}} = \theta(1)$

$f_4(n) = 2^n = 2^{\sqrt{n} \cdot \sqrt{n}}$

$f_5(n) = n^{10} \cdot 2^{n/2} = 2^{\lg(n^{10})} \cdot 2^{n/2} = 2^{n/2+10\lg n} = 2^{\sqrt{n} \cdot \sqrt{n}/2+10\lg n}$

$f_6(n) = n\sqrt{n} = n^{1.5}$

$f_7(n) = \sum_{i=1}^{n} i + 1 = n + (n-1) + \ldots + 1 + 1 = \theta(n^2)$

$f_8(n) = 100000n = \theta(n)$

so, $f_3(n) < f_2(n) < f_8(n) < f_6(n) < f_7(n) < f_1(n) < f_5(n) < f_4(n)$

2. Solve the following recurrence relations and give a θ bound for each of them.

$(1) T(n) = 9T(n/3) + n$

$$(2)\begin{cases} T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + cn \\ T(1) = 1 \end{cases}$$

$(3) T(n) = T(n-1) + \log n$

Answer:

(1) Using the master theorem:

***a*=9, *b*=3, *d*=1, $\log_3 9 > 1$，so** $T(n) = O(n^{\log_3 9}) = O(n^2)$

(2) Using the recursive tree method:

$$T(n) = cn + \frac{3cn}{4} + \left(\frac{3}{4}\right)^2 cn + \ldots = \left[1 + \left(\frac{3}{4}\right) + \left(\frac{3}{4}\right)^2 + \ldots\right] cn = \theta(n)$$

(3) Using the iterative method:

$$T(n) = \log n + \log(n-1) + \ldots + \log 2 + \log 1 + T(0) = \log(n!) + T(0) = \theta(n \log n)$$

3. Divide a positive integer *N* into the sum of several positive integers. How many non-repetitive splitting schemes there are? Give an algorithm with the time complexity $\underline{O(N^2)}$ to do this.

For example, *N* = 5 has seven splitting schemes:

1) 5=5
2) 5=4+1
3) 5=3+2
4) 5=3+1+1
5) 5=2+2+1
6) 5=2+1+1+1
7) 5=1+1+1+1+1

Answer:

Let the splitting fraction be sorted from small to large. *g*(*i*, *j*) is used to express the number of schemes of splitting *i* into several numbers and the maximum number (i.e., the last number) does not exceed *j*. There are two cases. First, if the last number does not exceed *j*-1, the number of schemes is *g*(*i*, *j*-1). Second, if the last number is *j*, the number of schemes is *g*(*i*-*j*, *j*). So *g*(*i*, *j*) = *g*(*i*, *j*-1) + *g*(*i*-*j*, *j*). Finally, *g*(*N*, *N*) is the answer, and the time complexity is $O(N^2)$.

***g*(*n*, *m*):**
if (*n*=1 or *m*=1) return 1
if (*m*≥*n*) return 1+*g*(*n*, *n*-1)
if (*m*<*n*) return *g*(*n*, *m*-1)+*g*(*n*-*m*, *m*)