Complexity types

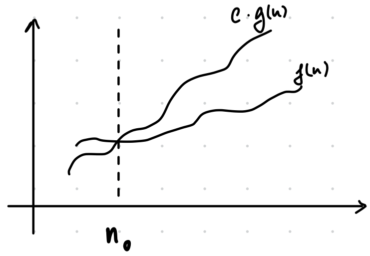
constant	:	1
logarithmic	:	$\log_2 n$
linear	:	n
quadratic	:	n^2
polinomial	:	$n, n^2, n^3 \dots$
exponential	:	$2^n, e^n, n!, n^n$

$\underbrace{f(n)}_{\text{all functions}} \in O(g(n))$

in here

$\exists\, n_0 \in \mathbb{N},\ c \in \mathbb{R}_+^*$

$\forall n \geq n_0 : f(n) \leq c \cdot g(n)$



$\Leftrightarrow \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$

$\hookrightarrow \{$ const. / 0

$f(n) = n^2 + 2n + 1$

$?\ f(n) \in \Theta(n^2)$ T

justification $\lim_{n\to\infty} \frac{n^2+2n+1}{n^2} = 1$

$O(n) + \Theta(n^2) = \Theta(n^2)$

$O \rightarrow$ greater than

$\Omega \rightarrow$ lower than

$f_1(n)$ — $f_2(n)$

$O(n^2)$ — $O(n^2)$

$\Omega(n)$ — $\Omega(n^2)$

Prove that

① $\forall f_1(n) \in O(n)$

② $\forall f_2(n) \in \Theta(n^2)$ $\quad ?\ f_1(n) + f_2(n) \in \Theta(n^2)$

① $\exists\, n_{0_1} \in \mathbb{N},\ c_1 \in \mathbb{R}_+^*,\ \forall n \geq n_{0_1} : f_1(n) \leq c_1 \cdot n$

② $\exists\, n_{0_2} \in \mathbb{N},\ c_2 \in \mathbb{R}_+^*,\ \forall n \geq n_{0_2} : f_2(n) \leq c_1 \cdot n^2$

Take $n_0 = \max(n_{0_1}, n_{0_2})$
$c = \max(c_{0_1}, c_{0_2})$

$\Rightarrow \forall n \geq n_0 \quad f_1(n) + f_2(n) \leq c_1 \cdot n + c_2 \cdot n^2 \leq (n + n^2) \cdot c \in O(n^2)$

or: $\lim_{n\to\infty} \frac{f_1(n) + f_2(n)}{n^2} = \lim_{n\to\infty} \frac{f_1(n)}{n^2} + \lim_{n\to\infty} \frac{f_2(n)}{n^2}$

$O(n) + O(n^2) \neq \Theta(n^2)$

since $n \in O(n)$, $n \in O(n^2)$ but $2n \notin \Theta(n^2)$

$f(n) + g(n) \leq 2 \max(f(n), g(n))$

Seminar : Complexity (Algorithm Analysis)

$2^n = 2 \cdot 2 \cdot \ldots \cdot 2$

$n! = 1 \cdot 2 \cdot \ldots \cdot n$

asimptotical lowerbound

$\lim_{n \to \infty} \frac{n^2}{n^3} = 0$

1. TRUE or FALSE?

p) $n^2 \in O(n^3)$ T	w) $O(n) + \Theta(n^2) = \Theta(n^2)$ T
q) $n^3 \in O(n^2)$ F	x) $\Theta(n) + O(n^2) = O(n^2)$ T
r) $2^{n+1} \in \Theta(2^n)$ T	y) $O(n) + O(n^2) = O(n^2)$ F
s) $2^{2n} \in \Theta(2^n)$ $(2^n)^2 \Rightarrow$ F	z) $O(f) + O(g) = O(\max\{f,g\})$ T
t) $n^2 \in \Theta(n^3)$ F	aa) $O(n) + \Theta(n) = O(n)$ T
u) $2^n \in O(n!)$ T	bb) $(n+m)^2 \in O(n^2 + m^2)$
v) $\log_{10} n \in \Theta(\log_2 n)$ T	cc) $3^n \in O(2^n)$ F
$\log_a n = \frac{\log_b n}{\log_b a}$ const.	dd) $\log_2 3^n \in O(\log_2 2^n)$ T

$O(n) + \Theta(n) = \Theta(n) <^{O}_{\Omega}$

$\longrightarrow (n+m)^2 = n^2 + m^2 + 2mn \le n^2 + m^2 + 2m^2 + 2n^2 = 3(n^2 + m^2)$

$n \cdot \log_2 3 = n \cdot c$

2. Complexity of search and sorting algorithms

first el   last el   O(worst case)

Algorithm	Time Complexity				Extra Space Complexity
	Best C.	W C.	Avg C.	Total	
Linear Search	Θ(1)	Θ(n)	Θ(n)	O(n)	Θ(1)
Binary Search	$\Theta(1)$	$\Theta(\log n)$	$\Theta(\log n)$	$O(\log n)$	
Selection Sort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	Θ(1) – in place
Insertion Sort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	$O(n^2)$	
Bubble Sort	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n^2)$	$O(n^2)$	
Quick Sort	$\Theta(n \log n)$	$\Theta(n^2)$	$\Theta(n \log n)$	$O(n^2)$	
Merge Sort	$\Theta(n \cdot \log n)$	$\Theta(n \cdot \log n)$	$\Theta(n \cdot \log n)$	$\Theta(n \cdot \log n)$	Θ(n)- out of place

3. Analyze the time complexity of the following subalgorithms:

```
subalgorithm s1(n) is:
    for i ← 1, n execute
        j ← n
        while j ≠ 0 execute
            j ← [j/2]
        end-while
    end-for
end-subalgorithm

subalgorithm s2(n) is:
    for i ← 1, n execute
        j ← i
        while j ≠ 0 execute
            j ← [j/2]
        end-while
    end-for
end-subalgorithm

subalgorithm s3(x, n, a) is:
    found ← false
    for i ← 1, n execute
        if xᵢ = a then
            found ← true
        end-if
    end-for
end-subalgorithm

subalgorithm s4(x, n, a) is:
    found ← false
    i ← 1
    while found = false and i <= n execute
        if xᵢ = a then
            found ← true
        end-if
        i ← i + 1
    end-while
end-subalgoritm

Subalgorithm s5(x, n) is:
    k← 0
    for i ← 1, n execute
        for j ← 1, xᵢ execute
            k ← k + xⱼ
        end-for
    end-for
end-subalgorithm

subalgorithm s6(n) is:
    for i ← 1,n execute
        @elementary operation
    end-for
    i ← 1
    k ← true
    while i <= n - 1 and k execute
        j ← i
        k₁ ← true
        while j <= n and k₁ execute
            @ elementary operation
                (k₁ can be modified)
            j ← j + 1
        end-while
        i ← i + 1
        @elementary operation
            (k can be modified)
    end-while
end-subalgorithm

subalgorithm p(x,s,d) is:
    if s < d then
        m ← [(s+d)/2]
        for i ← s, d-1, execute
            @elementary operation
        end-for
        for i ← 1,2 execute
            p(x, s, m)
        end-for
    end-if
end-subalgorithm

Initial call for the subalgorithm: p(x, 1, n)

Subalgorithm s7(n) is:
    s ← 0
    for i ← 1, n² execute
        j ← i
        while j ≠ 0 execute
            s ← s + j
            j ← j - 1
        end-while
    end-for
end-subalgorithm

Subalgorithm s8(n) is:
    s ← 0
    for i ← 1, n² execute
        j ← i
        while j ≠ 0 execute
            s ← s + j - 10 * [j/10]
            j ← [j/10]
        end-while
    end-for
end-subalgorithm
```

$$1+2+\dots+n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$1^2+2^2+\dots+n^2 = \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Stirling's approximation

$$\log_2 n! = \Theta(n \log_2 n)$$

S1) while : $\log_2 n$

$$T(n) = \sum_{i=1}^{n} \log_2 n = n \log_2 n \in \Theta(n \log_2 n)$$

$BC = WC = AC$: $T(n) \in \Theta(n \log_2 n)$

S2) while : $\log_2 i$

$$T(n) = \sum_{i=1}^{n} \log_2 i = \log_2 1 + \log_2 2 + \dots + \log_2 n = \log_2 n! = \Theta(n \log_2 n) = BC = WC = AC$$

S3)
$$T(n) = \sum_{i=1}^{n} 1 = n = \Theta(n) = AC = BC = WC$$

S4)
$$\left.\begin{array}{l} WC = \Theta(n) \\ BC = \Theta(1) \\ AC = \Theta(n) \end{array}\right\} \Rightarrow O(n)$$

AC: $T(n) = \sum_{I \in D} P(I) \cdot E(I)$

cases	no. of steps	description of case
1	1	$x_1 = a$
2	2	$x_1 \neq a$ & $x_2 = a$
3	3	$a \notin \{x_1, x_2\}$ $x_3 = a$
	....	
n	n	$a \notin \{x_1, \dots x_{n-1}\}$ $x_n = a$
n+1	n	$a \notin \{x_1, \dots, x_n\}$

assume that all cases are equally prob.

$$P = \frac{1}{n+1}$$

$$T(n) = \frac{1}{n+1} \cdot 1 + \frac{1}{n+1} \cdot 2 + \dots + \frac{1}{n+1} n + \frac{1}{n+1} \cdot n$$

$$= \frac{1}{\cancel{n+1}} \cdot \frac{n\cancel{(n+1)}}{2} + \frac{1}{n+1} \cdot n = \frac{n}{2} + \frac{n}{n+1}$$

$\in \Theta(n)$

S5) $\sum_{i=1}^{n} x_i = s$ → assumption

$T(n) = s \quad \in \Theta(s)$

✓ $x_i \in N^*$

? $x_i \in N$

? $x_i = 0 \ \forall i \in N \quad T(n) = n \ \in \Theta(n)$

$T(n) = n + s \quad \in \Theta(n+s)$

$? \in \Theta(\max(n,s))$

$x_i = \begin{cases} 1 & \text{if } i \text{ is perfect square} \\ 0 & \text{otherwise} \end{cases} \Rightarrow \Theta(\sqrt{n})$ since $s = \sqrt{n}$

56) $BC = T(n) \leq \Theta(n)$

$WC = T(n) \leq \Theta(n^2)$

AC $\longrightarrow$ inner while | fixed i

steps	j
1	i
2	$i, i+1$
$\vdots$	
$n-i+1$	$i, \dots, n$

$\Rightarrow P(I) = \frac{1}{n-i+1}$

$T_{in}(n,i) = \frac{1}{\cancel{n-i+1}} \cdot \frac{(n-i)\cancel{(n-i+1)}}{2} = \frac{n-i}{2}$

outer while

steps	i	
1	1	$T_{in}(n,1)$
2	1, 2	$T_{in}(n,1) + T_{in}(n,2)$
$\vdots$		
$n-1$	$1, \dots, n-1$	$\sum_{i=1}^{n-1} T_{in}(n,i)$

$P(I) = \frac{1}{n-1}$

$T_{out\,w}(n) = p \cdot \left[(n-1) T_{in}(n,1) + \dots + T_{in}(n, n-1)\right]$

$= \frac{1}{n-1} \sum_{i=1}^{n-1} (n-i) T_{in}(n,i) =$