3 Growth of Functions

$\Theta(n\lg n)$ beats $\Theta(n^2)$?

How's $100n\lg n$ vs $3n^2$ ($n = 2$: $100n\lg n = 200 > 3n^2 = 27$) ?

We say $n \to \infty$, merge sort, $\Theta(n\lg n)$, beats insertion sort, $\Theta(n^2)$.

Overview

- A way to describe behavior of functions in the limit. We're studying asymptotic efficiency.（函数的渐近效率，即极限情况下的函数行为）
- Describe growth of functions（函数增长率的描述）
- Focus on what's important by abstracting away low-order terms and constant factors. （通常忽略低阶项和常数因子）
- How we indicate running times of algorithms.（如何描述算法的运算时间）
- A way to compare "sizes" of funcitons（比较函数大小的方法）

$$o \approx < \;;\; O \approx \leq \;;\; \Theta \approx = \;;\; \Omega \approx \geq \;;\; \omega \approx >$$

3.1 Asymptotic notation

- the asymptotic running time are defined in terms of functions whose domains are the set of natural numbers $N = \{0, 1, 2, \ldots\}$.（运行时间函数的定义域为自然数集）
- **Abuse(“滥用，泛用”)**
  - just for convenient
  - for example, extended to the real numbers domain
- Not misused(误用、错用)
  - We need understand the precise meaning of the notation when it is abused. It is not misused.

3.1.1 Θ-notation: asymptotically tight bound（渐近紧界）

- **What this notation $T(n) = \Theta(n^2)$ means**

For a given function $g(n)$, we denote by $\Theta(g(n))$ the set of functions

$\Theta(g(n)) = \{ f(n)$: there exist positive constants c_1, c_2 , and n_0 such that

$$0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \}.$$

We could write "$f(n) \in \Theta(g(n))$" to indicate that $f(n)$ is a member of $\Theta(g(n))$.

Instead, we will usually write "$f(n) = \Theta(g(n))$" to express the same notion. The abuse may at first appear confusing, but it has advantages.

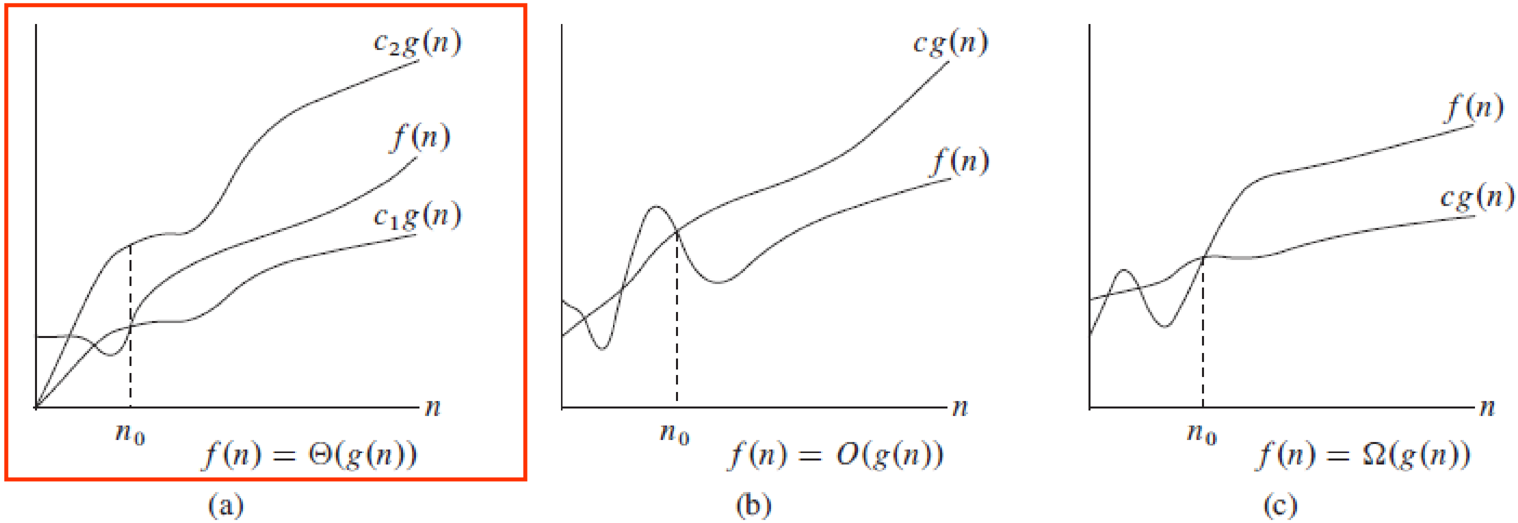
3.1.1 Θ-notation: asymptotically tight bound

- $\boldsymbol{T(n) = \Theta(n^2)}$

For a given function $g(n)$, we denote by $\Theta(g(n))$ the set of functions

$\Theta(g(n)) = \{ f(n)$: there exist positive constants c_1, c_2 , and n_0 such that

$$0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \}.$$



We say that $g(n)$ is an asymptotically tight bound for $f(n)$.

3.1.1 Θ-notation: asymptotically tight bound

$\Theta(g(n)) = \{f(n)$: there exist positive constants c_1, c_2 , and n_0 such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0$ $\}$.

- In this chapter, assume that every asymptotic notations are asymptotically nonnegative.（设所有的渐近符号为渐近非负）
- Example: How to show that $n^2/2-3n = \Theta(n^2)$?

  We must determine positive constants c_1, c_2, and n_0 such that

$$c_1 n^2 \le n^2/2 - 3n \le c_2 n^2$$

$$\Rightarrow \quad c_1 \le 1/2 - 3/n \le c_2$$

  by choosing c_1 =1/14, c_2=1/2, and n_0=7, we can verify that $n^2/2-3n=\Theta(n^2)$
- Other choices for the constants may exist. The key is some choice exists.

3.1.1 Θ-notation: asymptotically tight bound

$\Theta(g(n)) = \{f(n)$: there exist positive constants c_1, c_2, and n_0 such that
$0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0$ $\}$.

- How verify that $6n^3 \neq \Theta(n^2)$?

  Suppose for the purpose of contradiction that c_2 and n_0 exist such that $6n^3 \le c_2 n^2$ for all $n \ge n_0$. But then $n \le c_2 / 6$, which cannot possibly hold for arbitrarily large n, since c_2 is constant.

3.1.1 Θ-notation: asymptotically tight bound

$\Theta(g(n)) = \{f(n)$: there exist positive constants c_1, c_2 , and n_0 such that $0 \le c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0$ }.

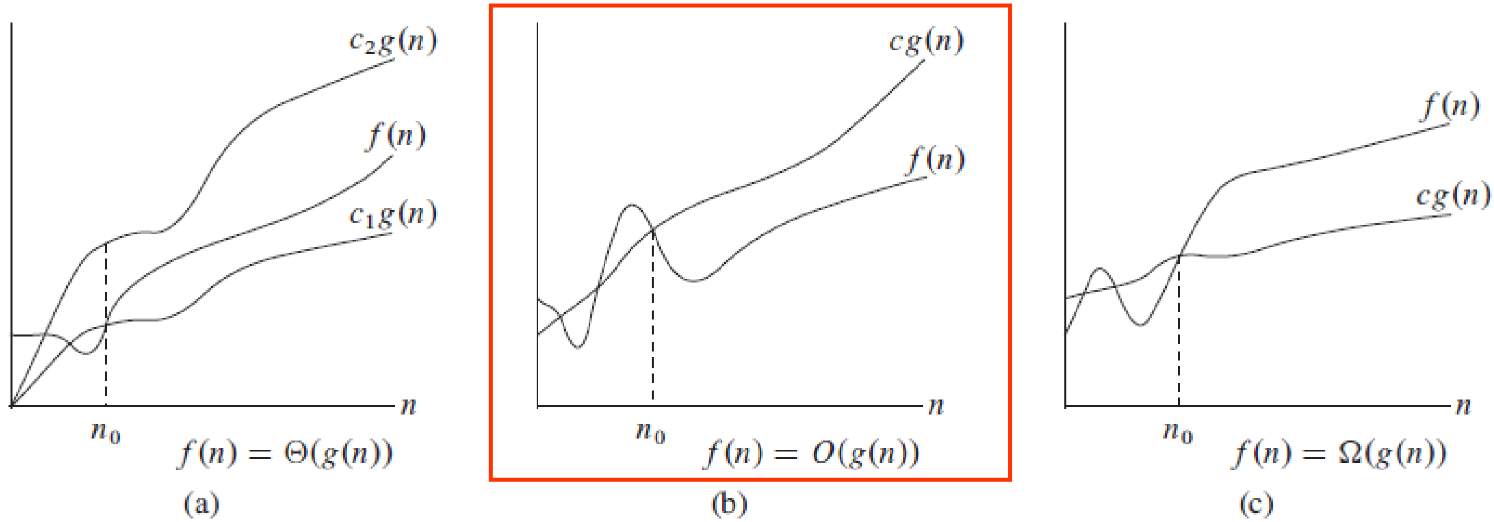
- The lower-order terms, the coefficient of the highest-order term can be ignored.
- Example: $f(n) = an^2 + bn + c$, where $a>0$, b, c are constants.

  Throwing away the lower-order terms and ignoring the constant yields $f(n) = \Theta(n^2)$
- In general, for any polynomial $p(n) = \sum_{i=0}^{d} a_i n^i$, where the a_i are constants and $a_d>0$, we have $p(n) = \Theta(n^d)$.
- We can express any constant function as $\Theta(n^0)$ or $\Theta(1)$.

  $\Theta(1)$ often mean either a constant or a constant function.

3.1.2 *O*-notation: asymptotic upper bound（渐近上界）

- ***O* – notation:** For a given function $g(n)$, we denote by $O(g(n))$ the set of functions $O(g(n)) = \{f(n)$: there exist positive constants c and n_0 such that $0 \le f(n) \le c\, g(n)$ for all $n \ge n_0\}$.



- $"f(n) = O(g(n))"$ indicates $"f(n) \in O(g(n))"$
- $f(n) = \Theta(g(n)) \Rightarrow f(n) = O(g(n))$

$$\Rightarrow \Theta(g(n)) \subseteq O(g(n))$$

3.1.2 *O*-notation: asymptotic upper bound

- ***O* – notation:** For a given function $g(n)$, we denote by $O(g(n))$ the set of functions $O(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that $0 \le f(n) \le c\, g(n)$ for all $n \ge n_0 \}$.

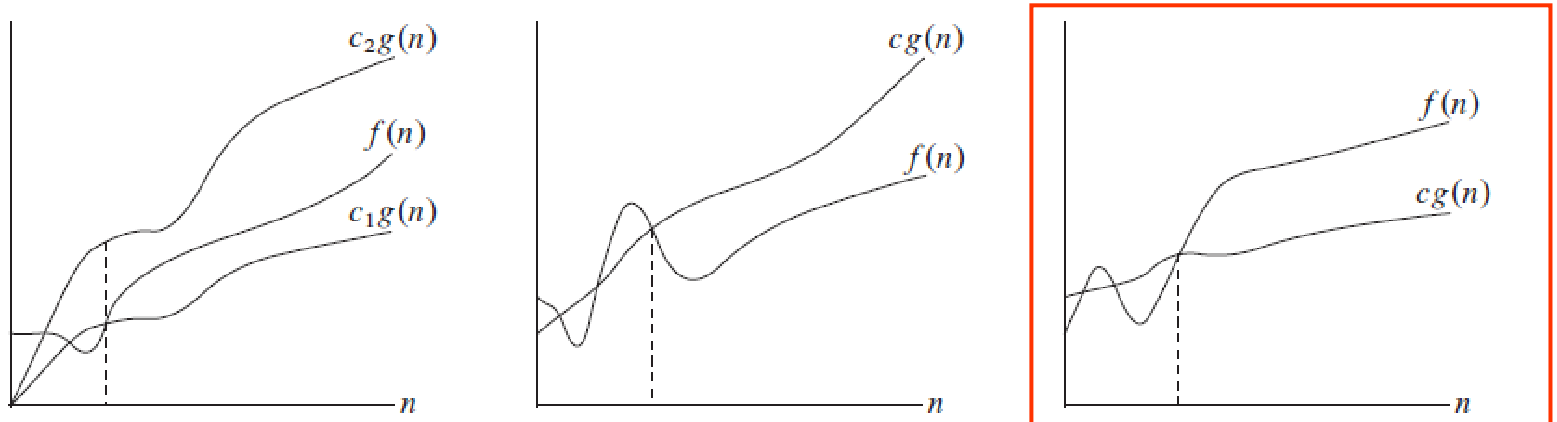
- Example: $2n^2 = O(n^3)$, with $c=1$ and $n_0=2$
- Example of functions in $O(n^2)$

n^2	n
$n^2 + n$	$n / 3000$
$n^2 + 2000n$	$n^{1.99999}$
$500n^2 + 1000n$	$n^2 / \lg\lg\lg n$

3.1.3 Ω-notation: asymptotic lower bound（渐近下界）

- **Ω – notation:** For a given function $g(n)$, we denote by $\Omega(g(n))$ the set of functions $\Omega(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that

$$0 \le c\, g(n) \le f(n) \text{ for all } n \ge n_0 \}.$$



$f(n) = \Theta(g(n))$
(a)

$f(n) = O(g(n))$
(b)

$f(n) = \Omega(g(n))$
(c)

3.1.3 Ω-notation: asymptotic lower bound

- Ω – **notation:** For a given function $g(n)$, we denote by $\Omega(g(n))$ the set of functions $\Omega(g(n)) = \{ f(n)$: there exist positive constants c and n_0 such that

$$0 \le c\, g(n) \le f(n) \text{ for all } n \ge n_0 \}.$$

- Example of functions in $\Omega(n^2)$

n^2

$n^2 + n$

$n^2 + 2000n$

$500n^2 - 1000n$

n^3

$n^{2.0000001}$

$n^2 \lg \lg \lg n$

3.1.3 Ω-notation: asymptotic lower bound

- **Theorem** 3.1

  For any two functions $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

  Prove: $\Rightarrow$: $f(n) = \Theta(g(n))$, then $\exists c_1 > 0, c_2 > 0, n_0 > 0$,

  s.t. $n \geq n_0$, $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$

  then $n \geq n_0$, $0 \leq f(n) \leq c_2 g(n) \Rightarrow f(n) = O(g(n))$

  then $n \geq n_0$, $0 \leq c_1 g(n) \leq f(n) \Rightarrow f(n) = \Omega(g(n))$

  $\Leftarrow$: $f(n) = O(g(n))$, then $\exists c_2 > 0, n_{20} > 0$,

  s.t. $n \geq n_{20}$, $0 \leq f(n) \leq c_{20} g(n)$

  $f(n) = \Omega(g(n))$, then $\exists c_{10} > 0, n_{10} > 0$,

  s.t. $n \geq n_{10}$, $0 \leq c_{10} g(n) \leq f(n)$

  let $n_0 = \max\{n_{10}, n_{20}\}$, then $n \geq n_0$,

  $0 \leq c_{10} g(n) \leq f(n) \leq c_{20} g(n)$, that is $f(n) = \Theta(g(n))$.

3.1.3 Ω-notation: asymptotic lower bound

- **Theorem** 3.1

  For any two functions $f(n)$ and $g(n)$, we have $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

- In practice, rather than using the theorem to obtain asymptotic upper and lower bounds from asymptotically tight bounds, we usually use it to prove asymptotically tight bounds from asymptotic upper and tower bounds.

（定理作用：实际中，通常根据渐近上界和渐近下界来证明渐近紧界，而不是根据渐近紧界来得到渐近上界和渐近下界。）

3.1.3 Ω-notation: asymptotic lower bound

- The running time of insertion sort falls between $\Omega(n)$ and $O(n^2)$, the bounds are asymptotically tight.
- The running time of insertion sort is not $\Omega(n^2)$. Why?
- It is not contradictory to say that the **worst-case running time** of insertion sort is $\Omega(n^2)$. Why?
- The running time of an algorithm is $\Omega(g(n))$, we mean that no matter what particular input of size n is chosen for each value of n, the running time on that input is at least a constant times $g(n)$, for large n.

（算法的运行时间为$\Omega(g(n))$意味着对足够大的n，对输入规模为 n 的任意输入,其运算时间至少是$g(n)$的一个常数倍。）

3.1.4 *o*-notation: upper bound but not asymptotically tight

- The bound provided by O-notation may or may not be asymptotically tight.
- The bound $2n^2=O(n^2)$ is asymptotically tight, but the bound $2n=O(n^2)$ is not.
- The o-notation denotes an upper bound that is not asymptotically tight. Formally, define $o(g(n))$ as the set
（非渐近紧的上界）

$$o(g(n)) = \{f(n): \text{for any positive constants } c>0, \text{ there exits a constant } n_0>0 \text{ such that } 0\le f(n)<c\, g(n) \text{ for all } n\ge n_0\}.$$

For example, $2n=o(n^2)$, but $2n^2 \neq o(n^2)$.

3.1.4 *o*-notation: upper bound but not asymptotically tight

$o(g(n))$ = {$f(n)$: for any positive constants $c>0$, there exits a constant $n_0>0$ such that $0 \le f(n) < c\, g(n)$ for all $n \ge n_0$ }.

- The definitions of O-notation and o-notation are similar.
- The main difference
  - In $f(n)=O(g(n))$, the bound $0 \le f(n) \le c\, g(n)$ holds for ***some*** constant $c>0$
  - In $f(n)=o(g(n))$, the bound $0 \le f(n) < c\, g(n)$ holds for ***all*** constant $c>0$
- Intuitively, in the o-notation, the function $f(n)$ becomes insignificant relative to $g(n)$ as n approaches infinity; that is

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

3.1.5 ω-notation: lower bound but not asymptotically tight

- ω-notation is to Ω-notation as o-notation is to O-notation.
- The ω-notation denotes an lower bound that is not asymptotically tight. Formally, define $\omega(g(n))$ as the set

$\omega(g(n))$ = { $f(n)$: for any positive constants c>0, there exits a constant n_0>0 such that $0 \le c\, g(n) < f(n)$ for all $n \ge n_0$}.

One way to define it is by

$f(n) \in \omega(g(n))$ if and only if $g(n) \in o(f(n))$

For example, $n^2/2=\omega(n)$, but $n^2/2 \neq \omega(n^2)$.

- The relation $f(n)=\omega(g(n))$ implies that

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty, \text{ if the limit exists.}$$

3.1.6 Comparison of functions

- Many of the relational properties of real number apply to asymptotic comparisons.

  For the following, Assume that $f(n)$ and $g(n)$ are asymptotically positive.

- **Transitivity（传递性）**

$$f(n) = \Theta(g(n)) \text{ and } g(n) = \Theta(h(n)) \text{ imply } f(n) = \Theta(h(n)),$$

$$f(n) = O(g(n)) \text{ and } g(n) = O(h(n)) \text{ imply } f(n) = O(h(n)),$$

$$f(n) = \Omega(g(n)) \text{ and } g(n) = \Omega(h(n)) \text{ imply } f(n) = \Omega(h(n)),$$

$$f(n) = o(g(n)) \text{ and } g(n) = o(h(n)) \text{ imply } f(n) = o(h(n)),$$

$$f(n) = \omega(g(n)) \text{ and } g(n) = \omega(h(n)) \text{ imply } f(n) = \omega(h(n)).$$

3.1.6 Comparison of functions

- **Reflexivity（自反性）**

$$f(n) = \Theta(f(n)),$$
$$f(n) = O(f(n)),$$
$$f(n) = \Omega(f(n)).$$

- **Symmetry（对称性）**

$$f(n) = \Theta(g(n)) \text{ if and only if } g(n) = \Theta(f(n)).$$

- **Transpose symmetry（反对称性）**

$$f(n) = O(g(n)) \text{ if and only if } g(n) = \Omega(f(n)),$$
$$f(n) = o(g(n)) \text{ if and only if } g(n) = \omega(f(n)).$$

3.1.6 Comparison of functions

- **An analogy between the asymptotic comparison of two functions and the comparison of two real numbers**
  （函数渐近性比较与实数比较的类比）

$$
\begin{aligned}
f(n) = o(g(n)) &\leftrightarrow f(n) < g(n) \leftrightarrow a < b,\\
f(n) = O(g(n)) &\leftrightarrow f(n) \le g(n) \leftrightarrow a \le b,\\
f(n) = \Theta(g(n)) &\leftrightarrow f(n) = g(n) \leftrightarrow a = b,\\
f(n) = \Omega(g(n)) &\leftrightarrow f(n) \ge g(n) \leftrightarrow a \ge b,\\
f(n) = \omega(g(n)) &\leftrightarrow f(n) > g(n) \leftrightarrow a > b.
\end{aligned}
$$

3.1.6 Comparison of functions

- **One property of real numbers, does not carry over to asymptotic notation**
  - **Trichotomy**（三分法）: any two real numbers a and b, one of the following must holds: $a < b$, $a = b$, or $a > b$.
  - Not all functions are asymptotically comparable. That is, for two functions $f(n)$ and $g(n)$, it may be the case that neither $f(n) = O(g(n))$ nor $f(n) = \Omega(g(n))$ holds.

    For example, the functions n and $n^{1+\sin n}$ cannot be compared using asymptotic notation.

$$-1 \le \sin n \le 1 \Rightarrow n^0 \le n^{1+\sin n} \le n^2$$

$$n^{1+\sin n} \le n \le n^{1+\sin n} \quad ???$$

** 3.2 Standard notation and common function

(Self-study for these parts)

- **Monotonicity**（单调性）
- **Floors and ceilings** $x-1<\lfloor x \rfloor \le x \le \lceil x \rceil < x+1$
- **Modular arithmetic (remainder or residue)** （模运算）

$$a \bmod n = a - \lfloor a/n \rfloor n$$

- **Polynomials**（多项式）

$$p(n)=\sum_{i=0}^{d} a_i n^i$$

- **Exponentials**（指数）
- **Logarithms**（对数）
- **Factorials**（阶乘）

** 3.2 Standard notation and common function

- **Functional iteration（迭代函数）**

We use the notation $f^{(i)}(n)$ to denote the function $f(n)$ iteratively applied i times to an initial value of n. For non-negative integers i, we recursively define

$$f^{(i)}(n)=\begin{cases} n & \text{if } i=0 \\ f(f^{(i-1)}(n)) & \text{if } i>0 \end{cases}$$

For example, if $f(n)=2n$, then

$$f^{(2)}(n)=f(f(n))=f(2n)=2(2n)=2^2 n$$

$$\cdots$$

$$f^{(i)}(n)=2^i n$$

** 3.2 Standard notation and common function

- The iterated logarithm function

  We use the notation $\lg^* n$ to denote the iterated logarithm. Let $\lg^{(i)} n$ be iterated function, with $f(n)=\lg n$, that is

  $\lg^{(i)}(n)=\lg(\lg^{(i-1)}(n))$. $\lg^{(i)} n$ is defined only if $\lg^{(i-1)} n>0$. Be sure to distinguish $\lg^{(i)} n$ from $\lg^i n$. $\lg^* n$ is defined as

  $$\lg^* n = \min\{i \geq 0 : \lg^{(i)} n \leq 1\}$$

  The iterated logarithms is a very slowly growing function:

  $$\lg^* 2 = 1, \qquad \lg^* 4 = 2, \qquad \lg^* 16 = 3,$$

  $$\lg^* 65536 = 4, \quad \lg^* 2^{65536} = 5.$$

  $2^{65536} >> 10^{80}$. Rarely encounter an input size n such that $\lg^* n>5$.

- Fibonacci numbers (self-study)

Exercises and problems

Show that for any real constants a and b , where $b > 0$, $(n + a)^b = \Theta(n^b)$.

Is $2^{n+1} = O(2^n)$? Is $2^{2n} = O(2^n)$?

- **Exercises**

  All

- **Problems**

  All