▲ Pro následující třídicí algoritmus na následujících daných vstupních datech ověřte jeho chování a spočítejte počet porovnání a počet volání funkce swap:

```
1 procedure bubblesort (A[1..n])
2 for i := 1 to n - 1 do
3     for j := 1 to n - i do
4         if A[j] > A[j+1] then
5             swap A[j], A[j+1]
```

a) $\mathtt{A} = [5, 3, 1, 6]$

b) $\mathtt{A} = [2, 3, 7, 9]$

□ Zkuste modifikovat algoritmus `bubblesort` tak, aby se zlepšila jeho složitost na příznivých datech (nápověda: nejvíce příznivými daty jsou již setřízené posloupnosti).

▲ Rozhodněte, zda jsou následující tvrzení pravdivá nebo nepravdivá:

$$f(n) = 3n^5 - 16n + 2$$

a) $f \in \mathcal{O}(n^5)$

b) $f \in \mathcal{O}(n)$

c) $f \in \mathcal{O}(n^{17})$

d) $f \in \Omega(n^5)$

e) $f \in \Theta(n^5)$

f) $f \in \Theta(n)$

g) $f \in \Theta(n^{17})$

▲ Pro následující funkce f a g nakreslete jejich graf a určete dvojici konstant c a n_0, která je svědkem toho, že platí $f \in \mathcal{O}(g)$, resp. $g \in \Omega(f)$. Kolik existuje takových dvojic?

a) $f(n) = \frac{1}{2}n + 5$; $g(n) = n^2 - 4n + 7$

b) $f(n) = \log_2 n$; $g(n) = 2^{n-1} - 2$

▲ Dokažte, že platí následující tvrzení:

1. $\log n \in O(n)$
2. $2^{n+1} \in O(\frac{3^n}{n})$

▲ Seřaďte následující funkce podle rychlosti jejich růstu:

$n^2 + \log n$	$7n^5 - n^3 + n$	n^2
$\log \log n$	2^n	$\log n$
$(\frac{3}{2})^n$	$n \cdot \log n$	$n!$
n	n^n	6
$\log(n!)$	$\log^{14} n$	

▲ Pro exaktní řešení problému obchodního cestujícího je znám algoritmus v $\mathcal{O}(2^n)$ (závislost na počtu měst). Při jeho řešení na starém počítači trval výpočet pro 36 měst téměř jeden den. Nyní máme k dispozici nový počítač, který je 1000-krát rychlejší. Určete, kolik měst můžeme zpracovat, aby výpočet nepřesáhl jeden den.

▲ Určete časovou složitost následujících algoritmů vzhledem k velikosti n:

```
1 procedure printer1 (A[1..n])
2 for i := 1 to 100000 do
3     if i < n then print A[i]
```

```
1 procedure printer2 (A[1..n])
2 for i := 1 to n - 1 do
3     for j := i to i + 1 do
4         print A[j]
```

```
1 procedure bubblesort (A[1..n])
2 for i := 1 to n - 1 do
3     for j := 1 to n - i do
4         if A[j] > A[j+1] then
5             swap A[j], A[j+1]
```

▲ Určete časovou složitost následujícího algoritmu, který vrátí součin dvou přirozených čísel y a z.

```
function multiply (y, z)
x := 0
while z > 0 do
    if z is odd then x := x + y
    y := 2 * y
    z := ⌊z/2⌋
return x
```

▲ Co počítají následující dvě funkce? Jaká je jejich složitost?

```
function kralicek(n: integer)
if n < 0 then fail "Chyba"
if n < 2
  then return n
  else
    return kralicek(n-1) + kralicek(n-2)
```

```
function mysicka(n: integer)
if n < 0 then fail "Chyba"
first := 0
second := 1
for i := 1 to n do
    (first,secod)
        := (second,first+second)
return first
```

▲ Tabulka časů výpočtu algoritmů o složitostech $\log n$, n, n^2, 2^n a n^n pro vstup délky 10, 20, 50 a 1000. Předpokládejme, že jedna iterace algoritmu trvá 1μs.

	10	20	50	1000
$\log n$	0,000001s	0,000001s	0,000002s	0,000003s
n	0,00001s	0,00002s	0,00005s	0,001s
n^2	0,0001s	0,0004s	0,0025s	1s
2^n	0,001024s	1,048576s	35,7 let	$3,4 \cdot 10^{287}$ let*
n^n	2,8 hodiny	$3 \cdot 10^{12}$ let*		

* Stáří vesmíru je odhadováno na $13,7.10^9$ let.