▲ Pro následující třídicí algoritmus ověřte jeho chování na následujících vstupních datech:
A = [5,3,1,6]
A = [2,3,7,9]

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

▲ 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.

▲ Rozhodněte, zda jsou následující tvrzení pravdivá nebo nepravdivá:

1. $3n^5 - 16n + 2 \in O(n^5)$
2. $3n^5 - 16n + 2 \in O(n)$
3. $3n^5 - 16n + 2 \in O(n^{17})$
4. $3n^5 - 16n + 2 \in \Omega(n^5)$
5. $3n^5 - 16n + 2 \in \Theta(n^5)$
6. $3n^5 - 16n + 2 \in \Theta(n)$
7. $3n^5 - 16n + 2 \in \Theta(n^{17})$

▲ Seřaďte následující funkce podle rychlosti jejich růstu:

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

▲ Dokažte, že platí následující tvrzení:
$\log n \in O(n)$

▲ Dokažte, že platí následující tvrzení:
$2^{n+1} \in O(\frac{3^n}{n})$

▲ Pro exaktní řešení problému obchodního cestujícího je znám algoritmus v $O(2^n)$. 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 následujícího algoritmu:

```
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)
1.  x := 0;
2.  while z > 0 do
3.    if z is odd then x := x + y;
4.    y := 2y;
5.    z := ⌊z/2⌋;
6.  return (x);
```

▲ Určete časovou složitost následujícího algoritmu pro vyhledání klíče v zadaném poli:

Algoritmus lineárního vyhledávání (vrátí index nalezeného prvku v poli D):

```
function ls (n: Integer; k: Integer): Integer;
  { n je pocet prvku v poli, k je hledaný klíč}
  var index: Integer;
  D = array[1..n] of Integer;

  begin
    index := -1;
    for i := 1 to n do
      if k = D[i] then index := i;
    if (index = -1) then "prvek nenalezen"
      else ls := index;
  end
```

▲ Dokážete výše zmíněný algoritmus (v rámci stejné třídy složitosti) vylepšit?

▲ Určete časovou složitost následujícího algoritmu pro vyhledání klíče v zadaném poli (vstupní posloupnost je setřízená)

Algoritmus binárního vyhledávání (vrátí index nalezeného prvku v poli D):

```
function bs (l, r: Integer; k: Integer): Integer;
  { l, r jsou levý a pravý konec pole, k je hledaný klíč }
  var m, index: Integer;
  D = array[1..999] of Integer;
  begin
    if l > r then index := -1; {nenalezeno}
      else
      begin
        m := (l + r) div 2;
        if k < D[m] then index := bs(l, m-1)
          else if k > D[m] then index := bs(m+1, r)
                           else index := m;
      end
    bs := index;
  end
```

▲ U následujících algoritmů počítajících mocninu čísla určete jejich časovou složitost.

1. verze:

```
function power (z: Real; n: Integer): Real;
  {Předpokládá se z > 0, n ≥ 0 }
  var r: Real;
  begin
    r := 1;
    for i := 1 to n do
      r := r * z;
    power := r;
  end
```

2. verze:

```
function power (z: Real; n: Integer): Real;
  var t: Real;
  var pom: Real; {uchovává mezivýsledky i konečný výsledek }
  begin
    pom := 1;
    t := z;
    if n = 0 then power := 1;
    while n > 0 do
      if n is odd then
        begin
          pom := pom * t;
        end
      begin
        n := n div 2;
        t := t * t;
      end
```

```
    power := pom;
  end
```

▲ Zkuste modifikovat algoritmus bubblesort tak, aby se zlepšila jeho složitost na příznivých datech (nápověda: nejvíce příznivá data jsou zde již setřízená posloupnost).