

## Algorithm definition

**New!** An algorithm is a finite sequence of precise instructions for solving a problem.

Algorithms can be expressed in English or in more formalized descriptions like pseudocode or fully executable programs.

Sometimes, we can define algorithms whose output matches the rule for a function we already care about. Consider the (integer) logarithm function

$$\log b : \{b \in \mathbb{Z} \mid b > 1\} \times \mathbb{Z}^+ \rightarrow \mathbb{N}$$

defined by

$$\log b( (b, n) ) = \text{greatest integer } y \text{ so that } b^y \text{ is less than or equal to } n$$

### Calculating integer part of base $b$ logarithm

---

```

1  procedure  $\log b(b, n)$ : positive integers with  $b > 1$ )
2     $i := 0$ 
3    while  $n > b - 1$ 
4       $i := i + 1$ 
5       $n := n \text{ div } b$ 
6    return  $i$  { $i$  holds the integer part of the base  $b$  logarithm of  $n$ }

```

---

Trace this algorithm with inputs  $b = 3$  and  $n = 17$

	$b$	$n$	$i$	$n > b - 1$ ?
Initial value	3	17		
After 1 iteration				
After 2 iterations				
After 3 iterations				

Compare: does the output match the rule for the (integer) logarithm function?

# Base expansion algorithms

Two algorithms for constructing base  $b$  expansion from decimal representation

**Most significant first:** Start with left-most coefficient of expansion (highest value)

*Informally:* Build up to the value we need to represent in “greedy” approach, using units determined by base.

Calculating base  $b$  expansion, from left

---

```
1 procedure baseb1( $n, b$ : positive integers with  $b > 1$ )
2    $v := n$ 
3    $k := 1 + \text{output of } \log b \text{ algorithm with inputs } b \text{ and } n$ 
4   for  $i := 1$  to  $k$ 
5      $a_{k-i} := 0$ 
6     while  $v \geq b^{k-i}$ 
7        $a_{k-i} := a_{k-i} + 1$ 
8        $v := v - b^{k-i}$ 
9   return  $(a_{k-1}, \dots, a_0)\{(a_{k-1} \dots a_0)_b \text{ is the base } b \text{ expansion of } n\}$ 
```

---

---

**Least significant first:** Start with right-most coefficient of expansion (lowest value)

Idea: (when  $k > 1$ )

$$\begin{aligned} n &= a_{k-1}b^{k-1} + \cdots + a_1b + a_0 \\ &= b(a_{k-1}b^{k-2} + \cdots + a_1) + a_0 \end{aligned}$$

so  $a_0 = n \bmod b$  and  $a_{k-1}b^{k-2} + \cdots + a_1 = n \operatorname{div} b$ .

Calculating base  $b$  expansion, from right

---

```
1 procedure baseb2( $n, b$ : positive integers with  $b > 1$ )
2    $q := n$ 
3    $k := 0$ 
4   while  $q \neq 0$ 
5      $a_k := q \bmod b$ 
6      $q := q \operatorname{div} b$ 
7      $k := k + 1$ 
8   return  $(a_{k-1}, \dots, a_0)\{(a_{k-1} \dots a_0)_b \text{ is the base } b \text{ expansion of } n\}$ 
```

---

# Base conversion algorithm

Practice: write an algorithm for converting from base  $b_1$  expansion to base  $b_2$  expansion: