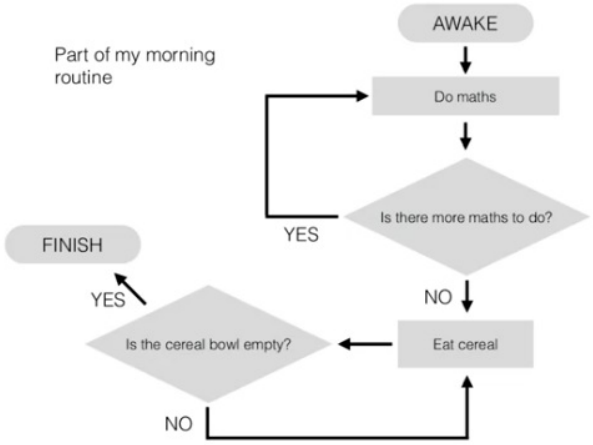
Cheatsheet - Pseudocode, Data Structures & Time Complexity

Fabio Lama – fabio.lama@pm.me

1. General Pseudocode Syntax (Example)

Pseudocode describes the logical steps of an algorithm or process in (mostly) plain language.

1. **function** Morning(`conscious, done, cereal`)
2. **if** `conscious`$=$TRUE **then**
3. `maths` $\leftarrow 0$
4. **for** $1 \leq i \leq$ `done` **do**
5. `maths` $\leftarrow$ `maths` $+ 1$
6. **end for**
7. **while** `cereal` > 0 **do**
8. `cereal` $\leftarrow$ `cereal` $- 1$
9. **end while**
10. **end if**
11. **return** ready!
12. **end function**



We can now call the Morning function:

1. `status` $\leftarrow$ Morning(TRUE, 10, 5)

2. Data Structures

2.1. Vector

A **vector** is a sequentially orderer collection of elements (like an ordered set). For example, the following vector v is of size *three* and contains the elements A, B and C:

$$\text{new Vector} \quad v(3) = (A, B, C)$$

The size of the vector is **fixed**, meaning one cannot add or remove items from it (only replace individual items).

> NOTE | In some programming languages, "vector" refers to a *growable* collection of data, which is very different and not the case here.

2.1.1. Operations

We can do operations on vectors:

- $\text{LENGTH}[v]$: Returns the size of vector v.
- $v[k]$: Returns the element of the vector v at index k.
- $v[k] \leftarrow o$: Stores the element o into the vector v at index k.

For example, given vector:

$$\text{new Vector} \quad v(4) = (A, B, C, D)$$

Then:

$$\text{LENGTH}[v] = 4$$
$$v[2] = B$$
$$v[2] \leftarrow Z$$
$$v[2] = Z$$
$$v = (A, Z, C, D)$$

> NOTE | In programming languages, the index usually starts at 0, respectively the first element of the set is indexed at 0, followed by 1, 2, ... In our case, we start the index at 1.

2.2. Queue

A **queue** is a data structure where elements need to "wait" before they get processed. Elements the get processed in the order the elements were added, respectively "first in first out" (FIFO). Queues are not fixed sized. Elements get added to the "tail" and come out at the "head".

$$\text{new Queue} \quad q = (A, B, C)$$

2.2.1. Operations

We can do operations on queues:

- $\text{HEAD}[q]$: Returns the element at the head of the queue.
- $\text{DEQUEUE}[q]$: Returns the element at the head of the queue and removes that element from the queue.
- $\text{ENQUEUE}[o, q]$: Adds the element o to the tail of the queue.
- $\text{EMPTY}[q]$: Returns *true* if the queue is empty or *false* if otherwise.

For example, given queue:

$$\text{new Queue} \quad q = (A, B, C, D)$$

Then:

$$\text{HEAD}[q] = D$$
$$q = (A, B, C, D)$$
$$\text{DEQUEUE}[q] = D$$
$$q = (A, B, C)$$
$$\text{ENQUEUE}[Z, q]$$
$$q = (Z, A, B, C)$$
$$\text{EMPTY}[q] = \text{false}$$

2.3. Stack

A stack is like a queue, but elements are processed in the "last in first out" (LIFO) order.

2.3.1. Operations

- $\text{PUSH}[o, s]$: Adds element o to the stack.
- $\text{TOP}[s]$: Returns the last inserted element from the stack.
- $\text{POP}[s]$: Returns the last inserted element from the stack and removes that element from the stack.
- $\text{EMPTY}[s]$: Returns *true* if the stack is empty or *false* if otherwise.

For example, given stack:

$$\text{new Stack} \quad s = (A, B, C)$$

Then:

$$\text{PUSH}[Z, s]$$
$$s = (A, B, C, Z)$$
$$\text{TOP}[s] = Z$$
$$s = (A, B, C, Z)$$
$$\text{POP}[s] = Z$$
$$s = (A, B, C)$$
$$\text{EMPTY}[s] = \text{false}$$

3. Time Complexity

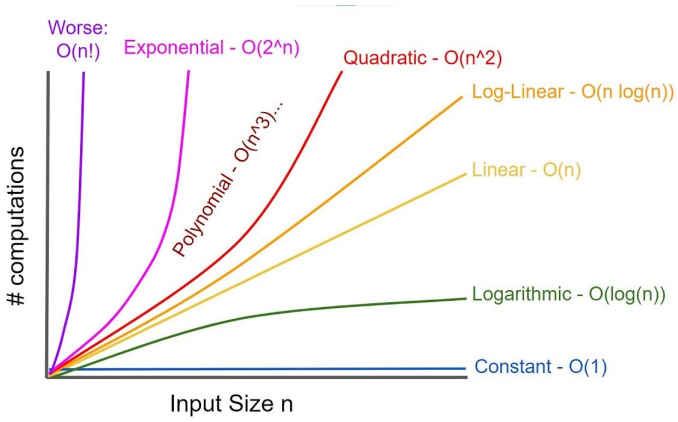


Figure 1. Source: https://youtu.be/47GRtdHOKMg

Last updated 2023-02-27 17:07:09 UTC