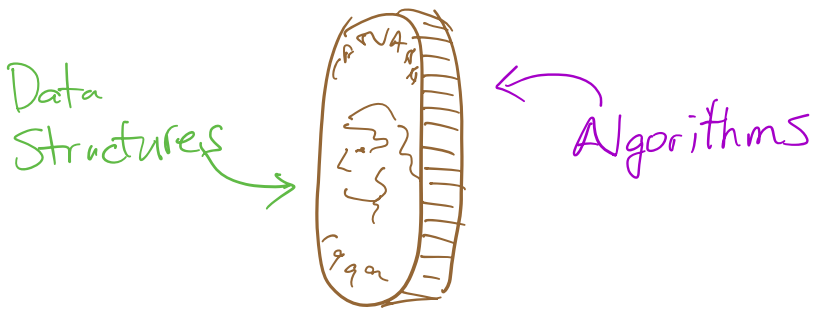
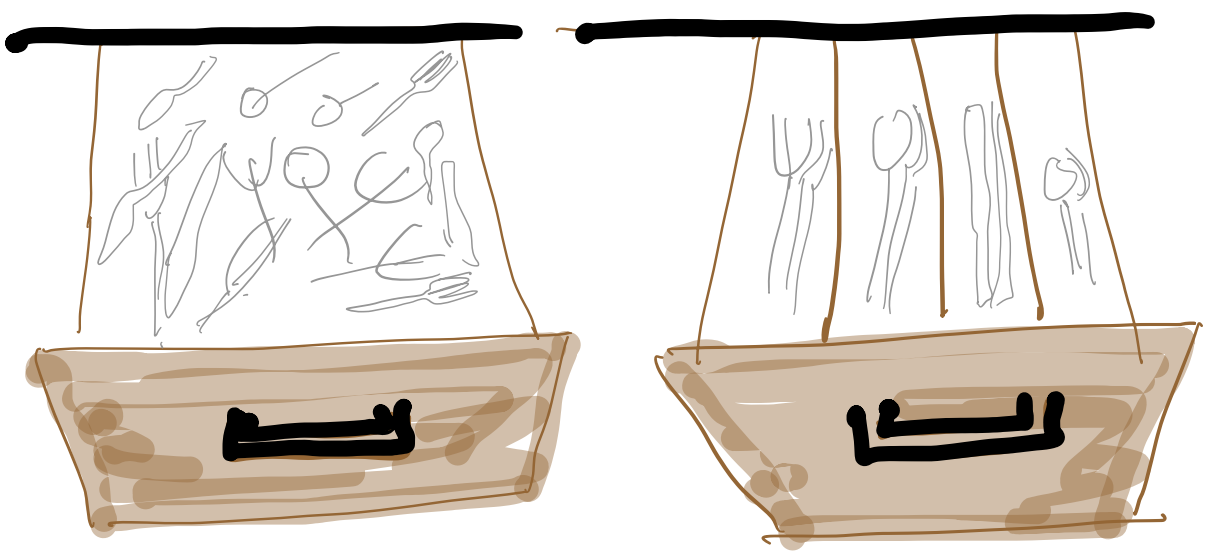
CSCI 260 - Data Structures and Algorithms

Administrivia - See course web pages at
csci.viu.ca/~gpruesse/teaching/260



Eg. cutlery drawer



Sort going in ... or sort coming out
Which is better?

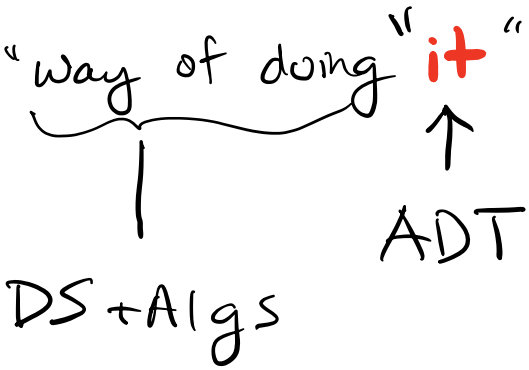
Either way, **what** is done/needs doing is the same:

```
init()

insert(x)

getfork()
get knife()
get spoon()
```

What makes one "way of doing **it**" the best way?



- running time
- memory
- power consumption.

Abstract Data Types (ADT)

- defines what is done by the alg/DS
- ∃ many ways of "getting the job done"
  - implementation is the code that does it
  - algorithm is a mathematical object
    - ... a step-by-step procedure that "gets the job done" (meets reqs of ADT)
    - ... is language independent.

1. We need to select an algorithm before we can implement.
2. DS + Algorithm go hand-in-hand
   - your getSpoon() depends on the DS: messy drawer or sorted?

ADT

- captures the behavior (job it does)

but not:

- running time
- space

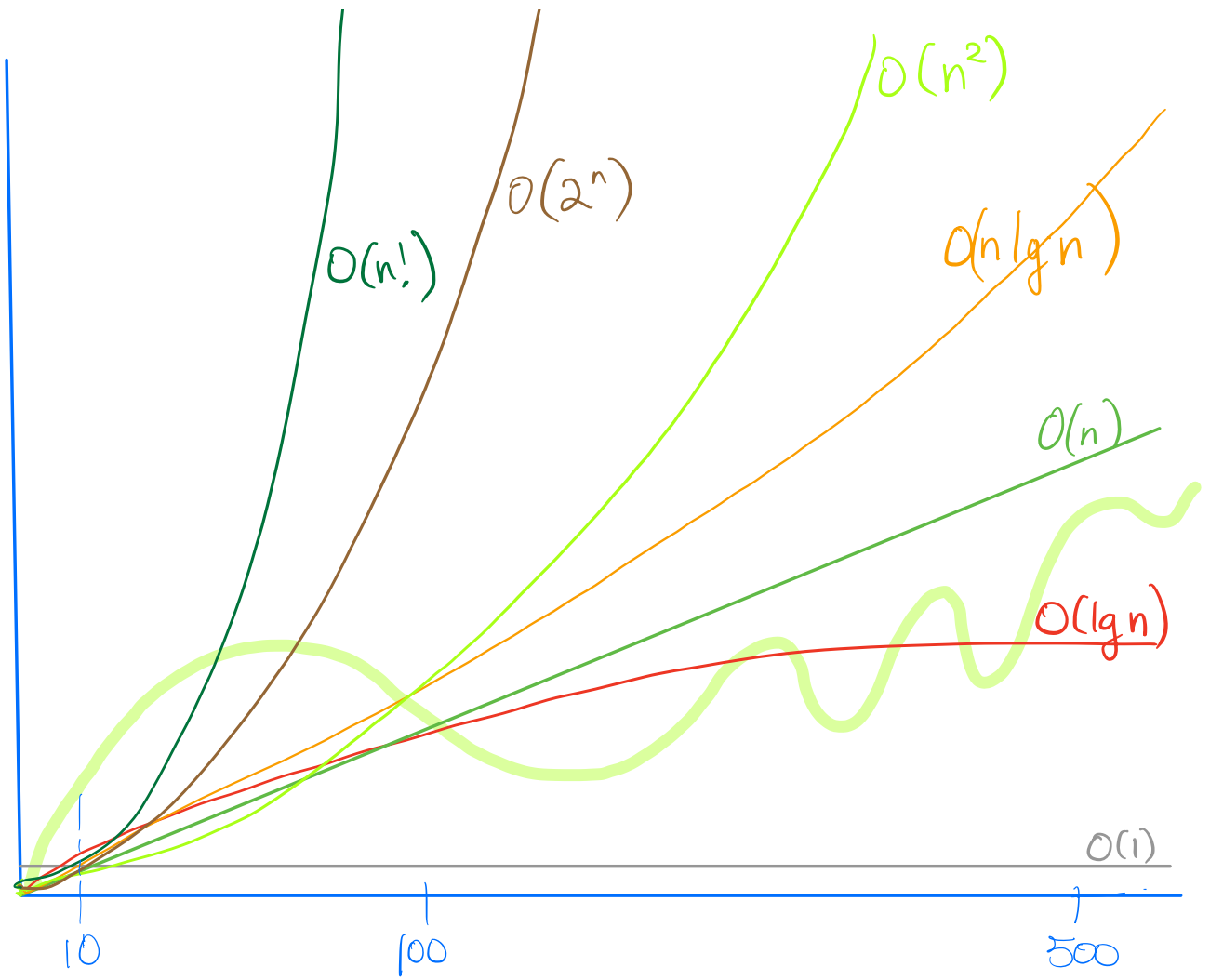
Alg + DS

- need to be correct ...
- have related running times (and space).
- We need to analyze algs

- We need measures of how much time an algorithm takes that is "independent of machine"
  - on machine X, MULT could be 100x as many clock ticks as on machine Y
  - $16n^2 + 190n - 500$

    $n^2 + 20n + 1000$

    $5000n$



What happens when you add two functions?

What happens when you multiply by a constant?

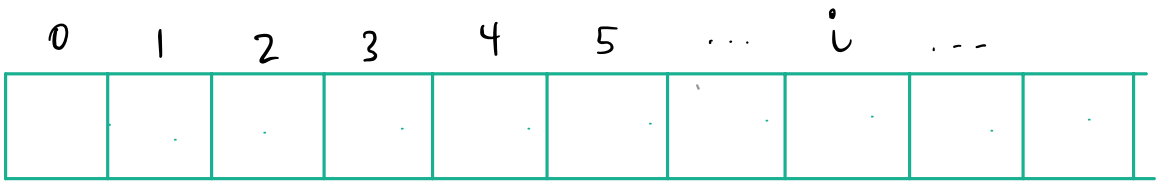
ADT Array

Operations are:

`init()` – initializes a new array. Old one is destroyed. New one has no elements in it.

`set(int i, float val)` – next "get(i)" will return val

`float get(int i)`
- if there has been no "set(i, val)" since last "init()", returns Sentinal value UNDEF
- else returns val of most recent "set(i, val)"



Is it possible to come up with ...

All-operations Constant-time (exact (not "expected")) implementation of

ADT Array

Operations are:

`init()` - initializes a new array. Old one is destroyed. New one has no elements in it.

`set(int i, float val)` - next "get(i)" will return `val`

`float get(int i)`
- if there has been no "set(i, val)" since last "init()", returns sentinal value `UNDEF`
- else returns `val` of most recent "set(i, val)"

Assume: init can allocate an arbitrarily large amount of space (hint: for multiple C++ arrays) in constant time
- such space will be full of garbage
- We will assume the space allocated is sufficient.

Algorithms + DS

- can compute an output for a given input

- can be a DS with a lifetime
  operations can change it, or run
  algorithms on it.

Array ADT

- obvious implementation (init is not O(1))
  - use a sentinel.

- don't use a sentinel.