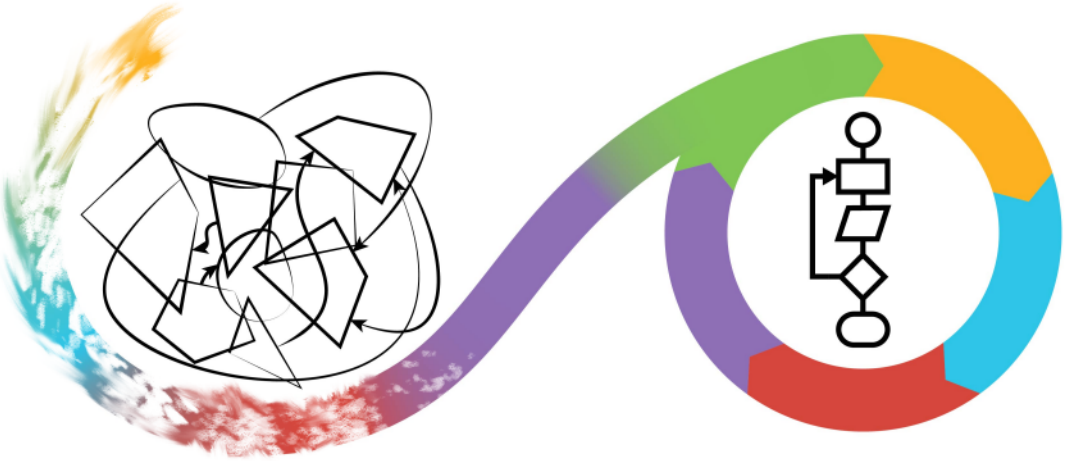
Optimisation

Richèl Bilderbeek

1 The Big Picture 

https://github.com/UPPMAX/programming_formalisms/blob/main/tdd/tdd_lecture/tdd_lecture.qmd



1.1 Breaks

Please take breaks: these are important for learning. Ideally, do something boring (1)!

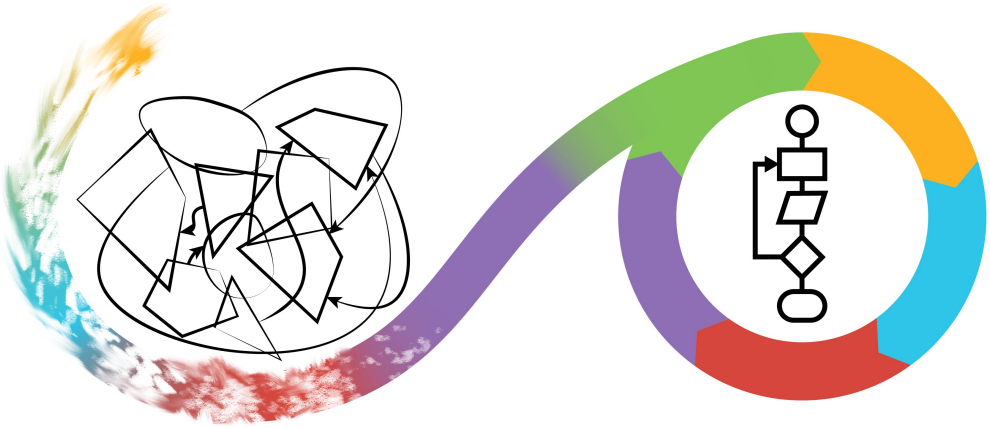
1.2 Schedule

Day	From	To	What
Fri	12:00	13:00	Lunch
Fri	13:00	14:00	Lecture: optimization
Fri	14:00	14:15	Break

Day	From	To	What
Fri	14:15	15:00	Project: write/optimize code
Fri	15:00	15:15	Break
Fri	15:15	15:45	Project: write/optimize code
Fri	15:45	16:00	Reflection

2 Optimisation

https://github.com/UPPMAX/programming_formalisms/blob/main/optimisation/optimisation_lecture/optimisation_lecture.qmd



3 Why optimization?

To improve the runtime speed (or memory use) of a program

Captain Obvious

4 Misconceptions

Q: What would be **bad advice** to improve the run-time speed of an algorithm?

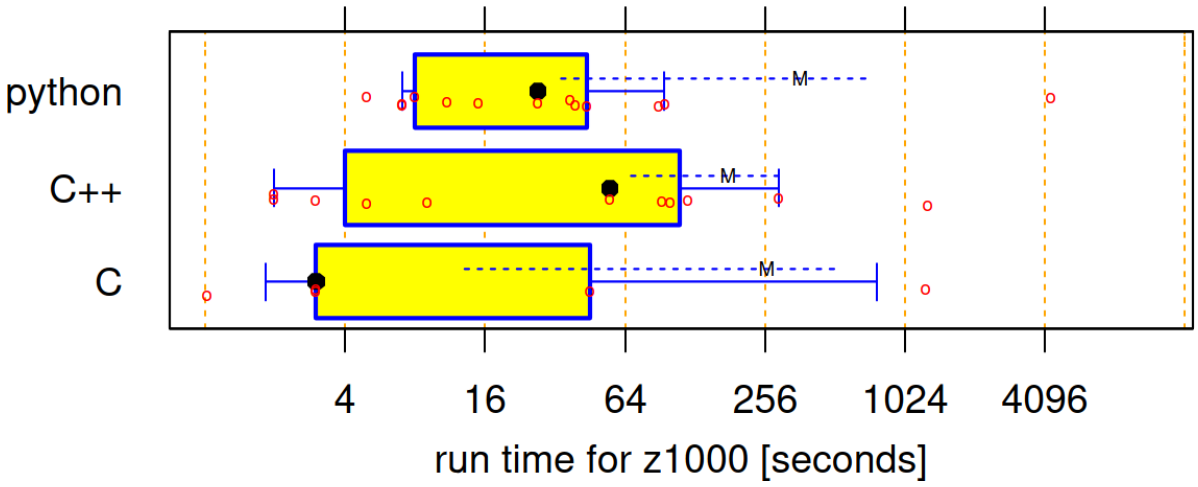
Fill in in the shared document!

4.1 Bad advice 1

'Use C or C++ or Rust'

. . .

Variance within programming languages is bigger than variance between languages (adapted fig 2, from (2))



4.2 Bad advice 2

'No for loops', 'unroll for-loops', any other micro-optimization.

. . .

Premature optimization is the root of all evil. It likely has no measurable effect.

4.3 Bad advice 2

> We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil. Yet we should not pass up our opportunities in that critical 3%.
>
> Donald Knuth
>
> Source: Wikipedia

4.4 Bad advice 3

'Always parallelize'

. . .

- Maximum gain depends on proportion spent in the parallelized part (3)
- Overhead is underestimated
- Hard to debug

Figure 1: Donald Knuth

4.5 Bad advice 3

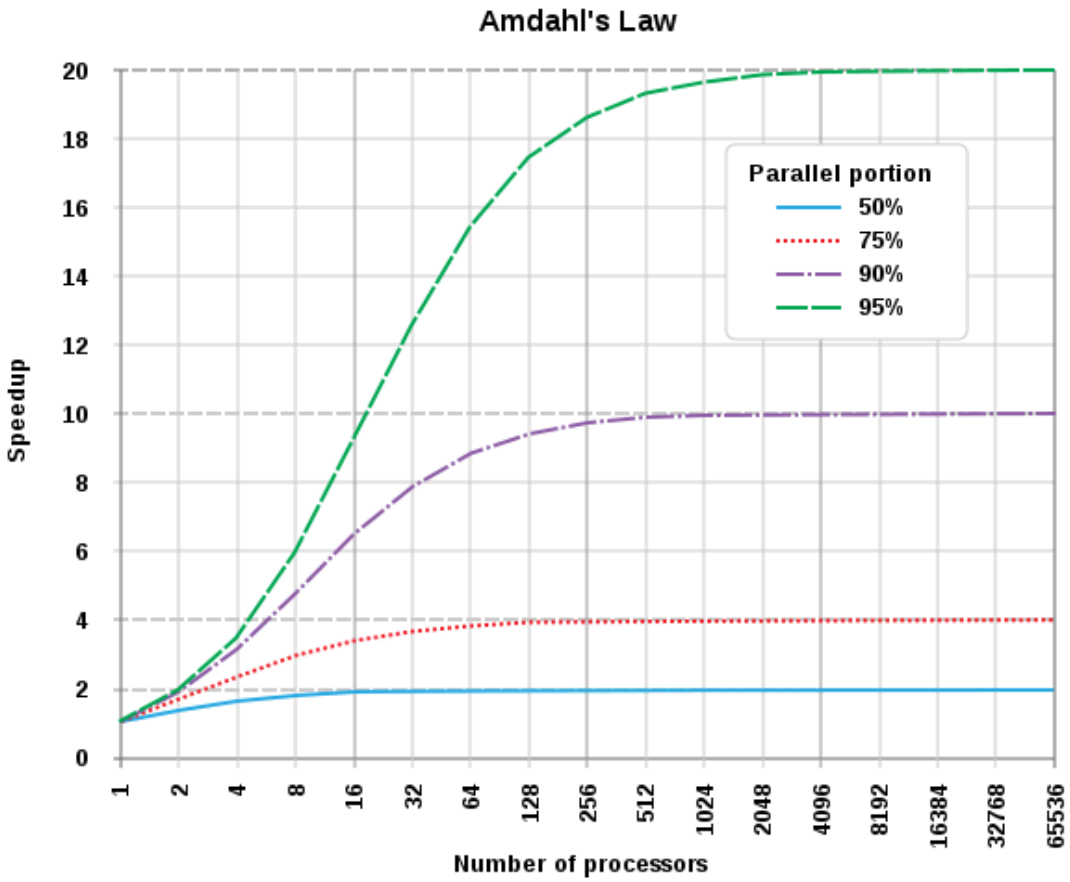


Figure 2: https://en.wikipedia.org/wiki/File:AmdahlsLaw.svg#file

4.6 Bad advice 4

'Optimize the function where you feel the performance problem is'

Developers -also very experienced developers- are known to have a bad intuition (4)

Instead, from (5):

1. finding code program spends most time in
2. measure timing of that code
3. analyze the measured runtimes

4.7 Bad advice 5

'Optimize each function'

- The 90-10 rule: 90% of all time, the program spends in 10% of the code.
- Your working hours can be spent once

5 Proper method

5.1 Problem

Q: When to optimize for speed?

. . .

A:

- C++ Core Guidelines: Per.1: Don't optimize without reason
- C++ Core Guidelines: Per.2: Don't optimize prematurely
- C++ Core Guidelines: Per.3: Don't optimize something that's not performance critical

5.2 Problem

Q: How to improve the run-time speed of an algorithm?

. . .

> Make it work, make it right, make it fast.
>
> Kent Beck

A (simplified):

1. Measure (hard to do (6))
2. Think
3. Change code
4. Measure again

5.3 Problem

Q: How to improve the run-time speed of an algorithm?

A (simplified):

1. Measure big-O
2. Measure speed profile
3. Think
4. Change code
5. Measure again

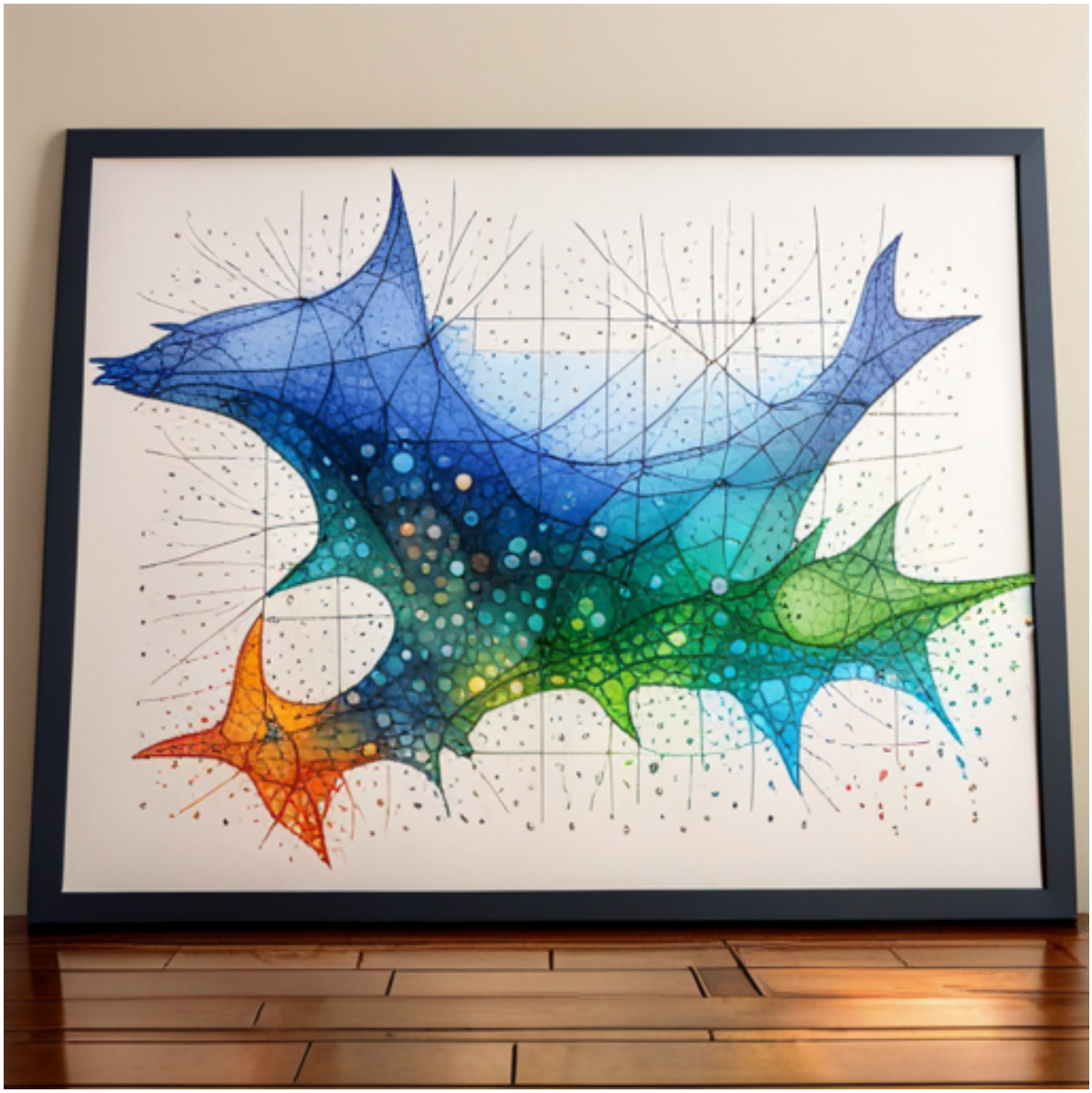
5.4 Measurement 1: big-O

How your (combination of) algorithms scales with more complex input.

- Counting the words in a book: O(n)
- Looking up a word in a dictionary: O(log2(n))

Do measure big-O in release mode!

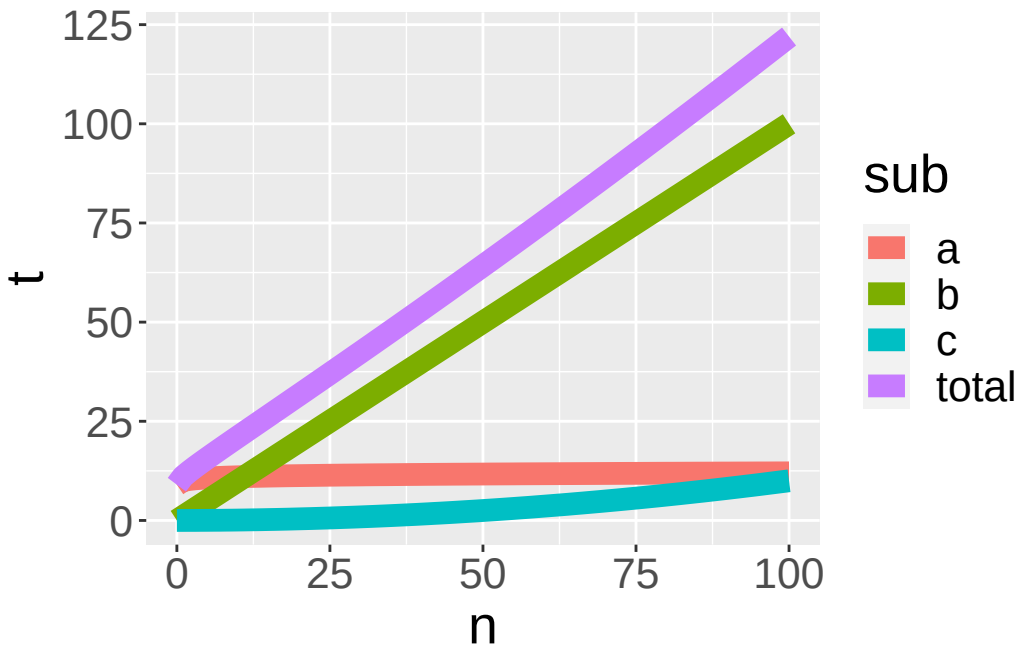
5.5 Your algorithm



5.6 Example

```
create_big_o_example <- function(n = seq(0, 100)) {
  t_wide <- tibble::tibble(n = n)
  t_wide$a <- 10 + log10(t_wide$n + 0.1)
  t_wide$b <- t_wide$n
  t_wide$c <- 0.001 * (t_wide$n ^ 2)
  t_wide$total <- t_wide$a + t_wide$b + t_wide$c
  t <- tidyr::pivot_longer(t_wide, cols = c("a", "b", "c", "total"))
  colnames(t) <- c("n", "sub", "t")
  t
}
t <- create_big_o_example(n = seq(0, 100))
ggplot2::ggplot(t, ggplot2::aes(x = n, y = t, color = sub)) +
  ggplot2::geom_line(size = 4) +
  ggplot2::theme(text = ggplot2::element_text(size = 20))
```

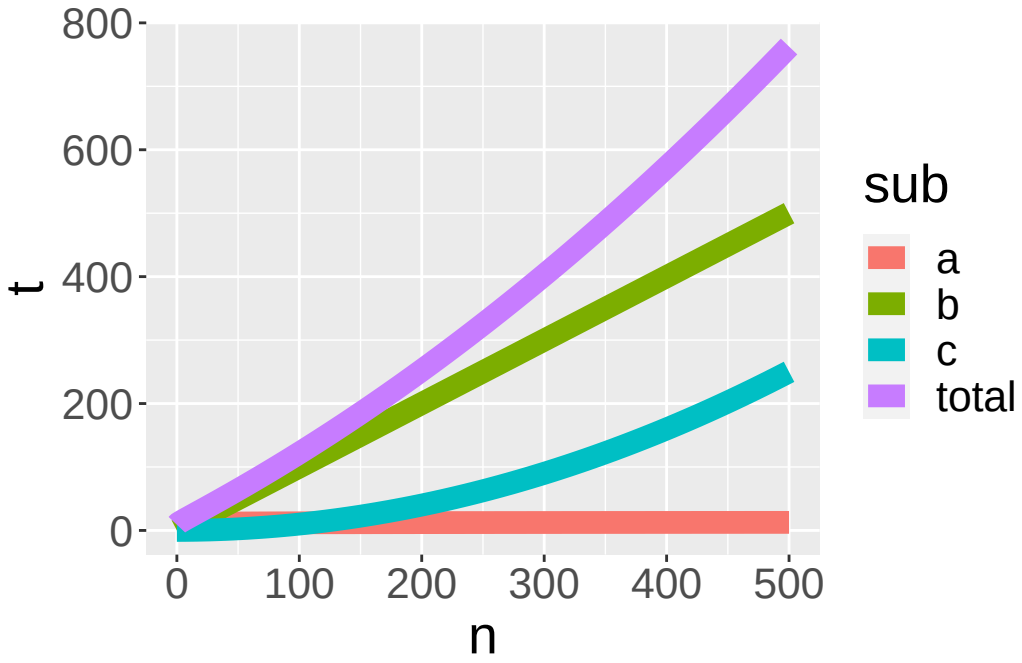
```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```



Work on B?

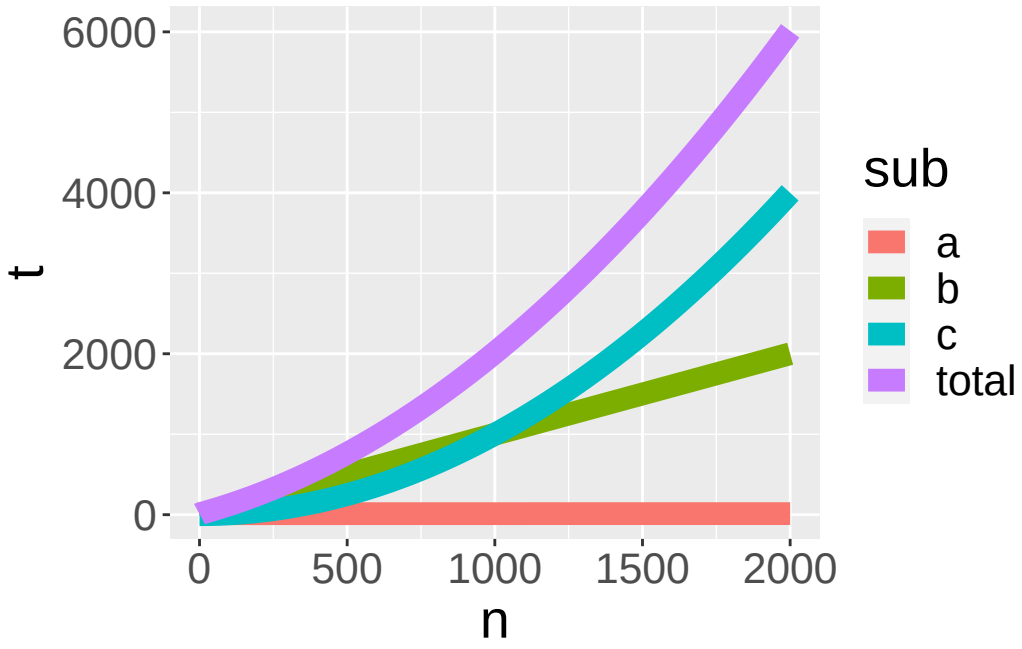
5.7 Example

```
t <- create_big_o_example(n = seq(0, 500))
ggplot2::ggplot(t, ggplot2::aes(x = n, y = t, color = sub)) +
  ggplot2::geom_line(size = 4) +
  ggplot2::theme(text = ggplot2::element_text(size = 20))
```



5.8 Example

```
t <- create_big_o_example(n = seq(0, 2000))
ggplot2::ggplot(t, ggplot2::aes(x = n, y = t, color = sub)) +
  ggplot2::geom_line(size = 4) +
  ggplot2::theme(text = ggplot2::element_text(size = 20))
```



No, work on C instead

5.9 Conclusions

Big-O helps to:

- find algorithm to profile
- make predictions

5.10 Exercise 1

- Measure big-O complexity of https://www.pythonpool.com/check-if-number-is-prime-in-python/

```
def isprime(num):
  for n in range(
    2, int(num**0.5)+1
  ):
    if num%n==0:
```

```
      return False
  return True
```

```
def isprime(num):
    if num> 1:
        for n in range(2,num):
            if (num % n) == 0:
                return False
        return True
    else:
        return False
```

5.11 Exercise 1

- Measure big-O complexity of https://www.pythonpool.com/check-if-number-is-prime-in-python/

```
def isprime(num):
  for n in range(
    2, int(num**0.5)+1
  ):
    if num%n==0:
      return False
  return True
```

```
def Prime(no, i = 2):
    if no == i:
        return True
    elif no % i == 0:
        return False
    return Prime(no, i + 1)
```

5.12 Exercise 2

- Measure big-O complexity of DNA alignment at https://johnlekberg.com/blog/2020-10-25-seq-align.html

```
ACGTACGTACGTACGTACGTACGT
ACGTACGTACGTCGTACGTACGT

ACGTACGTACGTACGTACGTACGT
ACGTACGTACGT-CGTACGTACGT
```

5.13 Measurement 2: Run-time speed profile

- See which code is spent most time in
- Use an input of suitable complexity
  - Note to self: next example should take at least 10 seconds!
- Consider using CI to obtain a speed profile every push!

5.14 Run-time speed profile: code

```
def is_sorted(data): return data == sorted(data)
def richel_sort(data):
    while not is_sorted(data):
        from random import shuffle
        shuffle(data)
    return data

def create(): return list(range(0, 10))
def reverse(x):
    x.reverse()
    return x
def sort(x): return richel_sort(x)

def my_function(): return sort(reverse(create()))

import cProfile
cProfile.run("my_function()")
```

5.15 Run-time speed profile: results

```
         149384793 function calls (149384751 primitive calls) in 44.759 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:100(acquire)
      6/1    0.000    0.000    0.002    0.002 <frozen importlib._bootstrap>:1022(_find_and_lo
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:125(release)
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:165(__init__)
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:169(__enter__)
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:173(__exit__)
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:179(_get_module_l
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:198(cb)
     10/1    0.000    0.000    0.002    0.002 <frozen importlib._bootstrap>:233(_call_with_fr
      140    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:244(_verbose_mess
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:357(__init__)
        8    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:391(cached)
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:404(parent)
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:412(has_location)
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:48(_new_module)
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:492(_init_module_
        6    0.000    0.000    0.001    0.000 <frozen importlib._bootstrap>:564(module_from_s
      6/1    0.000    0.000    0.002    0.002 <frozen importlib._bootstrap>:664(_load_unlocke
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:71(__init__)
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:746(find_spec)
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:826(find_spec)
       24    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:893(__enter__)
       24    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap>:897(__exit__)
        6    0.000    0.000    0.001    0.000 <frozen importlib._bootstrap>:921(_find_spec)
      6/1    0.000    0.000    0.002    0.002 <frozen importlib._bootstrap>:987(_find_and_loa
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:1040(__i
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:1065(get
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:1070(get
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:1089(pat
        4    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:1163(__i
        4    0.000    0.000    0.001    0.000 <frozen importlib._bootstrap_external>:1174(cre
        4    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:1182(exe
      126    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:126(_pat
      126    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:128(<lis
        4    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:132(_pat
```

```
        8    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:134(<ge
       34    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:1356(_pa
        6    0.000    0.000    0.001    0.000 <frozen importlib._bootstrap_external>:1399(_ge
       36    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:140(_pat
        6    0.000    0.000    0.001    0.000 <frozen importlib._bootstrap_external>:1431(fin
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:150(_pat
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:1531(_ge
       28    0.000    0.000    0.001    0.000 <frozen importlib._bootstrap_external>:1536(fin
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:159(_pat
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:180(_pat
        4    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:380(cach
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:510(_get
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:542(_che
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:585(_cla
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:618(_val
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:670(_com
       28    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:71(_rela
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:721(spec
        6    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:84(_unpa
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:874(crea
      2/1    0.000    0.000    0.002    0.002 <frozen importlib._bootstrap_external>:877(exec
        2    0.000    0.000    0.000    0.000 <frozen importlib._bootstrap_external>:950(get_
        1    0.000    0.000   44.759   44.759 <string>:1(<module>)
  4060761    0.975    0.000    2.754    0.000 <string>:1(is_sorted)
        1    0.000    0.000    0.000    0.000 <string>:1(reverse)
        1    0.000    0.000   44.759   44.759 <string>:1(sort)
        1    4.605    4.605   44.759   44.759 <string>:1(stupid_sort)
        1    0.000    0.000    0.000    0.000 <string>:2(create)
        1    0.000    0.000   44.759   44.759 <string>:2(my_function)
        6    0.000    0.000    0.000    0.000 __init__.py:89(find_spec)
        6    0.000    0.000    0.000    0.000 __init__.py:96(<lambda>)
        1    0.000    0.000    0.000    0.000 bisect.py:1(<module>)
        6    0.000    0.000    0.000    0.000 call.py:5(python_function)
      6/1    0.000    0.000    0.003    0.003 loader.py:113(_find_and_load_hook)
      6/1    0.000    0.000    0.002    0.002 loader.py:115(_hook)
        6    0.000    0.000    0.000    0.000 loader.py:40(_maybe_run_hooks)
      6/1    0.000    0.000    0.003    0.003 loader.py:82(_run_hook)
        1    0.000    0.000    0.002    0.002 random.py:1(<module>)
        1    0.000    0.000    0.000    0.000 random.py:103(Random)
        1    0.000    0.000    0.000    0.000 random.py:119(__init__)
        1    0.000    0.000    0.000    0.000 random.py:128(seed)
        1    0.000    0.000    0.000    0.000 random.py:219(__init_subclass__)
 36546840   18.451    0.000   23.971    0.000 random.py:239(_randbelow_with_getrandbits)
```

```
 4060760   13.122    0.000   37.397    0.000 random.py:380(shuffle)
       1    0.000    0.000    0.000    0.000 random.py:813(SystemRandom)
       1    0.000    0.000    0.000    0.000 threading.py:1430(current_thread)
       2    0.000    0.000    0.000    0.000 {built-in method _imp._fix_co_filename}
      36    0.000    0.000    0.000    0.000 {built-in method _imp.acquire_lock}
       4    0.001    0.000    0.001    0.000 {built-in method _imp.create_dynamic}
       4    0.000    0.000    0.000    0.000 {built-in method _imp.exec_dynamic}
       6    0.000    0.000    0.000    0.000 {built-in method _imp.is_builtin}
       6    0.000    0.000    0.000    0.000 {built-in method _imp.is_frozen}
      36    0.000    0.000    0.000    0.000 {built-in method _imp.release_lock}
      12    0.000    0.000    0.000    0.000 {built-in method _thread.allocate_lock}
      13    0.000    0.000    0.000    0.000 {built-in method _thread.get_ident}
       2    0.000    0.000    0.000    0.000 {built-in method builtins.__build_class__}
     3/1    0.000    0.000   44.759   44.759 {built-in method builtins.exec}
      42    0.000    0.000    0.000    0.000 {built-in method builtins.getattr}
      41    0.000    0.000    0.000    0.000 {built-in method builtins.hasattr}
      41    0.000    0.000    0.000    0.000 {built-in method builtins.isinstance}
 4060768    0.304    0.000    0.304    0.000 {built-in method builtins.len}
       6    0.000    0.000    0.000    0.000 {built-in method builtins.locals}
       4    0.000    0.000    0.000    0.000 {built-in method builtins.max}
 4060761    1.779    0.000    1.779    0.000 {built-in method builtins.sorted}
       6    0.000    0.000    0.000    0.000 {built-in method from_bytes}
       2    0.000    0.000    0.000    0.000 {built-in method io.open_code}
       2    0.000    0.000    0.000    0.000 {built-in method marshal.loads}
       1    0.000    0.000    0.000    0.000 {built-in method math.exp}
       2    0.000    0.000    0.000    0.000 {built-in method math.log}
       1    0.000    0.000    0.000    0.000 {built-in method math.sqrt}
      10    0.000    0.000    0.000    0.000 {built-in method posix.fspath}
       6    0.000    0.000    0.000    0.000 {built-in method posix.getcwd}
       1    0.000    0.000    0.000    0.000 {built-in method posix.register_at_fork}
      36    0.000    0.000    0.000    0.000 {built-in method posix.stat}
       6    0.000    0.000    0.000    0.000 {built-in method rpycall.call_r_function}
       1    0.000    0.000    0.000    0.000 {function Random.seed at 0x7ffb9a2870a0}
       2    0.000    0.000    0.000    0.000 {method '__exit__' of '_io._IOBase' objects}
      12    0.000    0.000    0.000    0.000 {method '__exit__' of '_thread.lock' objects}
       6    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
36546840    2.027    0.000    2.027    0.000 {method 'bit_length' of 'int' objects}
       1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects
      10    0.000    0.000    0.000    0.000 {method 'endswith' of 'str' objects}
       6    0.000    0.000    0.000    0.000 {method 'format' of 'str' objects}
      12    0.000    0.000    0.000    0.000 {method 'get' of 'dict' objects}
60046373    3.493    0.000    3.493    0.000 {method 'getrandbits' of '_random.Random' objec
     130    0.000    0.000    0.000    0.000 {method 'join' of 'str' objects}
```

```
     6    0.000    0.000    0.000    0.000 {method 'pop' of 'dict' objects}
     2    0.000    0.000    0.000    0.000 {method 'read' of '_io.BufferedReader' objects
     1    0.000    0.000    0.000    0.000 {method 'reverse' of 'list' objects}
     4    0.000    0.000    0.000    0.000 {method 'rfind' of 'str' objects}
    44    0.000    0.000    0.000    0.000 {method 'rpartition' of 'str' objects}
   256    0.000    0.000    0.000    0.000 {method 'rstrip' of 'str' objects}
     6    0.000    0.000    0.000    0.000 {method 'startswith' of 'str' objects}
```

5.16 Myth 1

```
def slow_tmp_swap(x, y):
    tmp = x
    x = y
    y = tmp
    return x, y

def superfast_xor_swap(x, y):
    x ^= y
    y ^= x
    x ^= y
    return x, y
```

. . .

- C++ Core Guidelines: Per.4: Don't assume that complicated code is necessarily faster than simple code
- C++ Core Guidelines: Per.5: Don't assume that low-level code is necessarily faster than high-level code

5.17 Exercise

Create speed profile of https://www.pythonpool.com/check-if-number-is-prime-in-python/

Create speed profile of DNA alignment

5.18 Step 3: Think

- How to achieve the same with less calculations?
  - Aim to change big-O, not some micro-optimization
  - For example, store earlier results in a sorted look-up table

Feynman Problem Solving Algorithm:

1. Write down the problem.
2. Think very hard.
3. Write down the answer

5.19 Step 4: Measure again

In TDD, this test would have been present already:

```
assert 10.0 * get_t_runtime_b() < get_t_runtime_a()
```

Adapt the constant to reality.

- C++ Core Guidelines: Per.6: Don't make claims about performance without measurements

5.20 Recap quote

> It is far, far easier to make a correct program fast, than it is to make a fast program correct.
>
> Herb Sutter
>
> Source Wikimedia

5.21 Recap

- Be critical on speed optimization solutions
- Tested and clean code always comes first
- Measure correctly, at the right complexity, before and after
- Prefer changing big-O over micro-optimizations (but see first point!)

5.22 The End

5.23 Links

- Lecture of 2022: here:

Figure 3: Herb Sutter

1. Newport C. Deep work: Rules for focused success in a distracted world. Hachette UK; 2016.
2. Prechelt L. An empirical comparison of c, c++, java, perl, python, rexx and tcl. IEEE Computer. 2000;33(10):23–9.
3. Rodgers DP. Improvements in multiprocessor system design. ACM SIGARCH Computer Architecture News. 1985;13(3):225–31.
4. Sutter H, Alexandrescu A. C++ coding standards: 101 rules, guidelines, and best practices. Pearson Education; 2004.
5. Chellappa S, Franchetti F, Püschel M. How to write fast numerical code: A small introduction. Generative and Transformational Techniques in Software Engineering II: International Summer School, GTTSE 2007, Braga, Portugal, July 2-7, 2007 Revised Papers. 2008;196–259.
6. Bartz-Beielstein T, Doerr C, Berg D van den, Bossek J, Chandrasekaran S, Eftimov T, et al. Benchmarking in optimization: Best practice and open issues. arXiv preprint arXiv:200703488. 2020;