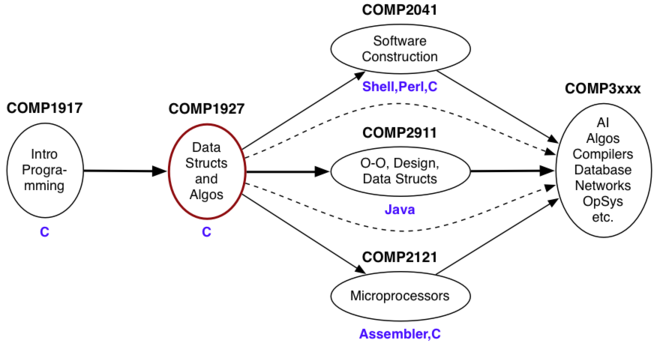
COMP1927

* Theoretical strand
  + Asymptotic complexities (as the data grows larger, how much slower does the program become)
    - Certain kind of programs become infeasible as soon as the data becomes really big.
  + Analysing algorithms and data structures
* Practical strand
  + How long does a program take to run?
* Once understanding both strands, you can figure out if this data structure or algorithm will solve this problem / will it work well.

Think like a scientist

* Observe 🡪
  + Look at a program. If the program is too slow, then the observation is that the program is too slow.
* Hypothesize 🡪
  + Why is the program too slow? Maybe I have chosen a rubbish algorithm.
* Experiment 🡪
  + Do an experiment to analyse the performance / gather some timing data.
* Analyse 🡪
  + Work out which kind of inputs is the algorithm is slow for.
  + Maybe the algorithm is great for random inputs, but not for sorted inputs.
* Repeat
  + Once all the analysis is done, you can go back and modify the program and observe again etc.

Course Context



Comp1927 Data Structures

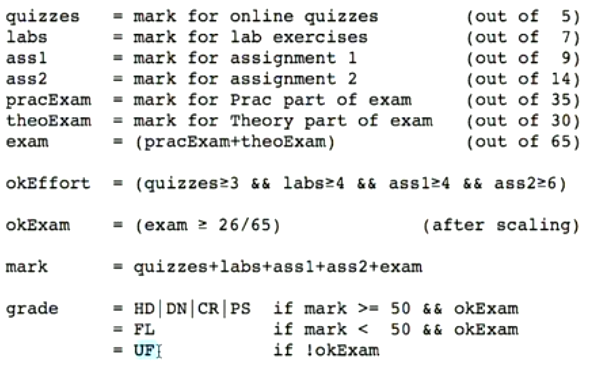
* The course mainly looks at linked data structures
* Branching data (tree) structures
  + Important for searching, parsing
* Cyclic data structures (graphic)
  + Can represent a whole range of real world things
  + E.g. road networks, the web (each node in the web is a page and the arcs between the nodes are hyperlinks)

Comp1927 Themes

1. Analysis: correctness, performance, usability
2. ADTs: sets, lists, trees, graphs, dictionaries
3. Operations: building, sorting, searching, traversing

* For data types: alternative implementation of operations
* For algorithms: complexity analysis, performance analysis

Course Assessment



* Need to pass finals to pass the course
* There is a secondary prac exam ONLY if you got (exam <= 50 && theory >= 50)

Switch Statements

* A **switch** encapsulates a common selection
* Similar to doing a bunch of IF ELSE statements.
* *Default* is the same as using ELSE
* *Break*;
  + Evaluate v
  + If v has the value C1, then execute S1, then exit the switch statement
* If *Break;* is not used:
  + If v is equal to C1, then execute S1, then fall through to the next case and then execute S2
  + And if there is still no *Break;* then all the cases will be executed until you reach default.

**COMP1927 L2**

Check out toptal.com for sorting algorithm diagrams.

Exercise 1

* Finding maximum value in “int a[N]” (int array)
* Pre-condition: defined(integer, a[0 … N-1]) && N > 0)
* Post: for all I [0 … N-1] max >= a[i]
* Test cases
  + Elements 1, 2, 3, 100, 1000, …
  + Duplicates: max occurs > 1, all values same
  + Negative values: some, all.

Exercise 2: Sequence program, using for