Introduction to lab 1

The what, when and how + tips & tricks

Agenda

- Intro to function time complexity analysis
  - Demo of general approach
  - Best, worst and average case?
- Requirements
  - Groups
  - Code
  - Hand-in
- Some tips and tricks

What's the purpose of the lab?

- Experimentally determine the time complexity for 5 functions
  - How does the size of the input affect the run time?
  - Measured in big-O
  - Martin will go through this in his lectures
- Crash course:
  - See provided Jupyter notebook for example
  - Can run at ex. https:\\hub.cse.kau.se

What is the best, worst and average case?

- Algorithms may perform differently depending on data
  - You need to find the input data that gives the best, worst and average case time run time

- Example for linear search:
  - When do I need to search through the fewest elements?
  - When do I need to search through the most elements?

1	2	3	4	5	6	7	8	9	10

How to find best, worst and average case?

- Can we do it experimentally?
  - Test every single permutation of input sequence
  - All possible 1000 element long sequences of numbers from 1-100: $100^{1000} = 10^{2000}$
  - Age of the universe: $13.8 * 10^{9}$ years $\approx 4 * 10^{17}$ seconds
  - The sun will have exploded long before you're done…

- Have to derive it from theory
  - Have to figure out best and worst case based on how the algorithm works
  - Go through text books, search through the internet, discuss with each other

- Average case can be approximated with random input

Requirements

- All the requirements are in the lab instructions
  - Read them thoroughly!
  - If anything is unclear – ask a lab supervisor

- You need to both implement the program **AND** write a lab report
- Deadline: See Canvas (this year, 2022-11-22)

Scoring

Implementation

Task	Points
Backend/frontend	1
Bubble sort	0.5
Insertion sort	0.5
Quick sort	1
Linear search	0.5
Binary search	0.5

Lab report

Task	Points
Algorithms	1
Results	0.5
Analysis	0.5
Completness	1

Groups

- Work in groups of two
  - If you are unable to find a lab partner you may work solo
  - Preferably use the same group for all labs

- Please assign yourself to a lab group on Canvas

- Include name + email address in lab report

Code

- We have provided you with a skeleton
  - You **MUST** use the function delaractions in algorithms.{c,h} as provided
  - You are free to add additional functions in algorithms.{c,h}
  - You are free to change everything else

- You **SHOULD** separate backend and frontend (1p)

- We may subtract up to 1p for excessive use of bad coding practices
  - Ex. global variables, magical numbers, overly long functions, lot of repititious code etc.

Report

- Report should be in Swedish or English
- Should contain:
  - Introduction
  - Background
  - Design/implementation
  - Method + Results (from your program)
  - Analysis/discussion/conclusion

- See lab specification for further details

Hand-in

- Zip all files and upload on Canvas
  - Code files (c-and h-files) + Makefile
  - Lab report (preferably in PDF-format)

- We test and grade the labs offline
  - No returns

- Late handins get a 25% penalty to the score
  - 50% if handed in after end of course

Tips and tricks - general

- Read the lab specification (!!!)
- Get started ASAP
  - Deadline for code + report in 2 weeks
- Start writing on the report early
  - You can start with introduction and background before finishing code
  - The report is half of the points!

Tips and tricks - implementation

- Test your algorithms!
  - Sorting algorithms should sort the input
  - Search algorithms should only find elements if they exist
  - Consider the empty case (sequence length 0)
- Run time will vary
  - Run many times and compute average
  - Consider adding a warmup phase
  - Searching is faster than sorting – can use larger input/more iterations for search
- Sorting algorithms modify their input
  - Need to reset input between repeated runs
- Confused by skeleton code?
  - See old introduction slides (section 4.1 – 4.2, slides 14 – 16)

Extra tip: Function pointers

- You can pass functions to other functions
  - Can be helpful to avoid some repetition
  - See my example, or just search on the internet

- Example use (pseudo code):

```
float time_any_function(function_pointer func, int arg) {
        float start = clock();
        func(arg);
        float end = clock();
        return end – start;
}
```

Questions?

Good luck with the lab!