6-Week Plan: Evaluating, Measuring, and Comparing Algorithms

This document outlines a 6-week plan to learn evaluating, measuring, and comparing algorithms, going beyond asymptotic complexity to include empirical behaviour, structural analysis, and how to communicate comparisons.

Week 1 — Foundations of Measuring Algorithms

Goal

Understand what Big-O gives you and what it does not; get a clear feel for algorithm growth.

Study

- **MIT 6.046J: Design and Analysis of Algorithms**
  - Course homepage: `https://ocw.mit.edu/courses/6-046j-design-and-analysis-of-algorithms-spring-2015/`
  - Lecture videos (watch Lecture 1 and Lecture 2): `https://ocw.mit.edu/courses/6-046j-design-and-analysis-of-algorithms-spring-2015/resources/lecture-videos/`
- **Tim Roughgarden — Algorithms Illuminated (Part 1)**
  - Front matter / sample PDF: `https://assets.cambridge.org/97809992/82984/frontmatter/9780999282984_frontmatter.pdf`
  - Skim the chapter that discusses running time vs. practical performance and Big-O basics.

Tiny writing task

Write a short note for yourself (5–10 sentences):

> What aspects of an algorithm matter besides asymptotic complexity?

Week 2 — Practical Algorithm Engineering & Real Engines

Goal

See how real systems people talk about algorithm behaviour and performance.

Study

- **Russ Cox — "Regular Expression Matching Can Be Simple and Fast"**
  - Article:
    `https://swtch.com/~rsc/regexp/regexp1.html`
- **Rust regex engine documentation**
  - Crate documentation:
    `https://docs.rs/regex/latest/regex/`
  - Crate overview (guarantees, discussion):
    `https://crates.io/crates/regex`
- **Algorithm Engineering / Empirical Slides (Meyer)**
  - Slides PDF:
    `https://people.mpi-inf.mpg.de/~mehlhorn/AlgorithmEngineering/ExternalMemorySlides.pdf`

Tiny writing task

Write six bullet points answering:

> What practical problems do real regex engines face, and how might a marked approach help?

Week 3 — Structure-Driven Behaviour (Your Regex World)

Goal

Understand structurally why algorithms blow up (derivatives vs. marks).

Study

- **Your own report (ReportV3)**
  - Re-read the sections on the derivative-based matcher and the marked matcher, focusing on where you discuss size explosion and mark growth.

- **Sulzmann & Lu — POSIX Regular Expression Parsing with Derivatives**
  - Springer chapter:
    `https://link.springer.com/chapter/10.1007/978-3-319-07151-0_13`
  - Preprint PDF:
    `https://arxiv.org/pdf/1604.06644`

Tiny writing task

Create a small table (for yourself) with the columns:

Construct | Growth in derivatives | Growth in marks

Fill it for `SEQ`, `STAR`, `ALT`, and `NTIMES`, noting what structurally causes growth in each representation.

Week 4 — Statistical and Experimental Thinking

Goal

Move from "I ran it once" to systematic empirical evaluation.

Study

- **Intro to performance analysis (lecture notes)**
  - Example notes inspired by Raj Jain:
    `https://www.cs.rice.edu/~johnmc/comp528/lecture-notes/Lecture1.pdf`
- **Raj Jain — The Art of Computer Systems Performance Analysis**
  - Book on archive.org:
    `https://archive.org/details/artofcomputersys0000jain`
  - Skim the early chapters on measurement and experimental design.
- **(Optional) Methodology of Algorithm Engineering**
  - arXiv PDF:
    `https://arxiv.org/pdf/2310.18979`

Tiny writing task

Design a small benchmark for your own matchers and write down:

- Which regular expressions you will test.
- Which input lengths you will use.
- Which metrics you will record (e.g. time, memory, mark count, derivative size).

Week 5 — Learning from Evaluation Sections in Papers

Goal

Copy the style of evaluation sections from real papers in your area.

Study

- **Sulzmann & Lu — POSIX Regular Expression Parsing with Derivatives**
  - Focus on the evaluation / performance discussion:
    `https://link.springer.com/chapter/10.1007/978-3-319-07151-0_13`
- **Tan & Urban — POSIX Lexing with Bitcoded Derivatives**
  - DROPS entry:
    `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITP.2023.27`
  - Direct PDF:
    `https://drops.dagstuhl.de/storage/00lipics/lipics-vol268-itp2023/LIPIcs.ITP.2023.27/LIPIcs.ITP.2023.27.pdf`
- **Urban — POSIX Lexing with Derivatives of Regular Expressions**
  - PDF:
    `https://urbanchr.github.io/Publications/posix.pdf`
- **Might, Darais, Spiewak — Parsing with Derivatives**
  - PDF:
    `https://matt.might.net/papers/might2011derivatives.pdf`

Tiny writing task

For each of the above papers, write three bullet points:

> What do they measure, and how do they explain why the algorithm behaves that way?

Week 6 — Synthesising Theory, Empirics, and Explanation

Goal

Be able to answer viva-style questions such as:

> Why is your algorithm better here?
> What causes slow behaviour in your approach?
> What evidence supports your claims?

Study

No new resources this week; instead:

- Review your notes from Weeks 1–5.
- Review your benchmarks and plots.
- Revisit key parts of the papers above as needed.

Final exercises

- **Four-component comparison.** For both the derivative-based and marked matchers, write:
  - Asymptotic complexity (what is known or conjectured).
  - Structural behaviour (which regex structures cause trouble).
  - Empirical behaviour (what your tests showed).
  - Semantic behaviour (POSIX values, determinism, formalisation).
- **Viva rehearsal.** Practise answering out loud:
  - How did you compare the two algorithms?
  - What do your experiments tell you, and what do they *not* tell you?
  - In which cases could your marked approach also be slow, and why?