Geschichte der Programmierung

Modus: Einzelarbeit

Typ: Miniprojekt

Beschreibung

Programmieren Sie zwei unterschiedliche Algorithmen, die feststellen, ob eine Zahl eine Primzahl ist (iterativ, Sieb des Eratosthenes), in jeweils vier unüblichen Programmiersprachen, und führen Sie Ihre Programme in einer geeigneten Umgebung aus. Idealerweise ist zumindest eine der Sprachen schon älter als 45 Jahre (dafür stehen online-Umgebungen zur Verfügung). Schließlich vergleichen Sie Laufzeiten und diskutieren die Unterschiede.

Ablauf

Führen Sie während des gesamten Prozesses ein Forschungstagebuch (siehe Beschreibung im Anhang). Dokumentieren Sie darin die Aktivitäten, Ergebnisse, Hindernisse und Erfolge sämtlicher Schritte Ihrer Arbeit.

Es gibt zwei einfache Wege, eine Zahl als Primzahl zu identifizieren.

Wir können für eine gegebene Zahl *n* feststellen, ob Sie durch eine Zahl, die kleiner ist als *n* teilbar ist; wenn das nicht der Fall ist, dann ist *n* eine Primzahl. Wir nennen diesen Algorithmus im weiteren Verlauf *Brute.*

Wir können zuerst das *Sieb des Eratostenes* anwenden und alle Primzahlen zwischen 1 und der Quadratwurzel aus *n* ermitteln, um dann nur noch die Division durch diese Zahlen zu versuchen. Diesen Algorithmus nennen wir im weiteren *Sieb.*

Beachten Sie, dass beide Algorithmen nicht optimal sind.

1. Suchen Sie vier Programmiersprachen, von denen Sie noch nie zuvor gehört haben, oder die es bereits seit mindestens 1982 gibt. Diese Sprachen müssen die folgenden Eigenschaften erfüllen:

› Die Sprache muss in der *Liste von Programmiersprachen*[1] auf Wikipedia enthalten sein.

› Für die Programmiersprache muss es eine Beschreibung/Dokumentation geben, die ausreicht, um damit einfache Programme zu schreiben.

› Es muss geeignete Umgebung zur Ausführung geben. Für viele sehr alte Programmiersprachen gibt es Online-Emulatoren, oft in Javascript geschrieben; je nachdem, wieviel Erfahrung und Grundkenntnisse Sie mitbringen, können Sie auch Compiler bzw. Interpreter auf Ihrem Computer installieren. Ein weiterer Weg wäre, einen Emulator für einen alten Computer zu verwenden und eine dort vorhandene oder installierbare Programmiersprache zu nutzen.

---

[1] https://de.wikipedia.org/wiki/Liste_von_Programmiersprachen

› Mindestens eine der Programmiersprachen sollte schon vor 1982 existiert haben.

Beschreiben Sie Ihre Suche nach Programmiersprache und Emulator bzw. Laufzeitumgebung im Forschungstagebuch, und dokumentieren Sie das Ergebnis, also die vier Sprachen, jeweils mit einer Quellenangabe für die Dokumentation und eine für Laufzeitumgebung/Emulation/Compiler/etc.

2. Implementieren Sie in diesen vier Programmiersprachen die beiden oben beschriebenen Algorithmen zur Feststellung, ob eine Zahl eine Primzahl ist.

Inkludieren Sie alle 8 Codestücke in Ihrem Forschungstagebuch, und beschreiben Sie Ihre Erfahrungen während der Implementierung. Womit hatten Sie Probleme, und was ist Ihnen leicht gefallen? Welche unerwarteten Sprachelemente oder -strukturen haben Sie gefunden? Wie gefällt Ihnen die Programmiersprache?

3. Testen Sie alle acht Implementierungen mit jeweils denselben vier Primzahlen: 2, 13, 401 und 2003. Stoppen Sie die Laufzeit (so gut als möglich mit der Hand) und notieren Sie die Laufzeiten. Vergleichen Sie die Verhältnisse zwischen den beiden Algorithmen, und spekulieren Sie über eventuelle Unterschiede. Dokumentieren Sie die Ausführung des Codes mit Screenshots.

4. Recherchieren Sie abschließend über die Bedeutung von Primzahlen, insbesondere große Primzahlen, für die Informatik, und halten Sie die Ergebnisse im Forschungstagebuch fest.

Abgabe

5. Ihre Abgabe besteht aus Ihrem Forschungstagebuch, eventuell bereinigt um persönliche Einträge, die Sie nicht preisgeben wollen. Gliedern Sie dieses Dokument bitte sinnvoll, und bemühen Sie sich, ein gut lesbares Layout zu gestalten. Erzeugen Sie dann daraus ein PDF[2] und geben Sie dieses im entsprechenden Abschnitt in TUWEL ab.

Bitte beachten Sie, dass Aufgaben dieses Typs spätestens 2 Wochen nach der Verfügbarkeit dieser Beschreibung abgegeben werden müssen, und dann noch eine Review-Phase (1 Woche) durchlaufen. Ihr selbst gewählter Termin gilt erst für die Endabgabe!

Zusatz für Endabgabe

Ein wesentlicher Teil Ihrer Endabgabe ist der Abschnitt *Reflexion & Feedback*. Beantworten Sie dabei die folgenden Fragen für die finale Abgabe, also nachdem Sie die Reviews geschrieben/bekommen haben, und ergänzen Sie Ihr PDF um einen entsprechenden Abschnitt:

- Wurde Ihr Verständnis der gewählten Denkweise durch diese Übungsarbeit verändert?
- Glauben Sie, ein nachhaltiges Verständnis der gewählten Denkweise wird Ihnen im Studium oder danach im Beruf helfen?
- Welche Teile dieser Arbeit fanden Sie besonders schwer, welche zu einfach?
- Welche Aspekte dieser Arbeit haben Ihnen gut gefallen, welche würden Sie ändern?
- Was haben Sie bei dieser Arbeit gelernt? Ist diese Art von Übungsformat Ihrer Meinung nach sinnvoll?
- Hat das Schreiben der Reviews geholfen, Ihre eigene Arbeit zu verbessern? Falls ja: wie?

---

[2] Beachten Sie bitte, dass inzwischen alle aktuellen Betriebssysteme die Erzeugung von PDFs ohne zusätzliche Software erlauben. Geben Sie keine PDFs ab, bei denen Werbung oder Wasserzeichen von Gratis-Software eingebettet ist. Für Unterstützung befragen Sie bitte die allwissende Müllhalde (das Internet) bzw. https://www.wikihow.com/Convert-a-File-Into-PDF

- Haben die Reviews, die sie bekommen haben geholfen, Ihre eigene Arbeit zu verbessern? Falls ja: wie?
- Sind Sie mit Ihrer Arbeit zufrieden?

Beachten Sie: Die Antworten auf die Fragen im Abschnitt *Reflexion und Feedback* gehen **nicht** in die Beurteilung Ihrer Arbeit ein!

Beachten Sie bitte die Richtlinie zur Verwendung von generativer AI, die im PDF »Denkweisen der Informatik 2023« zu finden ist. Wesentliche Teile der Arbeit dürfen nicht durch generative AI-Systeme verfasst werden!

Anhang: Forschungstagebuch

Ein Forschungstagebuch ist ein (physisches oder digitales) Medium, in dem Sie den Fortschritt Ihrer Arbeit und Ihre Gedanken dazu bzw. Probleme damit schriftlich festhalten. Damit Ihr Forschungstagebuch dabei helfen kann, zufällige Ideen oder plötzliche Inspirationen notieren können, sollten Sie es immer bei sich haben (das spricht stark für ein digitales Forschungstagebuch). Für die Zwecke dieser Arbeit genügt eine einfache Text-Datei. Jeder Eintrag ist mit Datum und Uhrzeit versehen.

Einträge im Forschungstagebuch werden zB. zu folgenden Anlässen gemacht:

- Artikel gelesen (mit kurzer Anmerkung der Relevanz für Ihr Thema, Auflistung für SIe wesentlicher Punkte)
- Gute Suchbegriffe für Ihr Thema
- In einem Gespräch etwas relevantes gehört, mit Ideen, wie Sie das weiterverfolgen könnten
- Teil der Arbeit geschrieben, mit Einschätzung der Qualität

Sie können auch persönliche Dinge im Forschungstagebuch festhalten, also erfreuliche (zB. Gute Quelle gefunden!) wie unerfreuliche (zB. heute gar nichts weitergegangen, sehr frustrierend). Für die Abgabe des Forschungstagebuchs können Sie Teile, die Sie nicht preisgeben wollen, entfernen.

Anhang: Qualität von Quellen

Ein wesentlicher Teil der Recheche im Internet ist die Einschätzung der Qualität von Quellen. Dazu gibt es, nicht ganz unironisch, viele Hilfestellungen im Internet. Wir haben einige davon für Sie zusammengestellt, denen wir vertrauen:

- Saferinternet, Quellen richtig beurteilen – https://www.saferinternet.at/news-detail/online-quellen-richtig-beurteilen-aber-wie
- Lehrerfortbildung Baden-Württemberg, Arbeitstechnik 2: Überprüfung von Quellen im Internet – https://lehrerfortbildung-bw.de/u_gewi/gk/gym/bp2016/fb5/2_komp/6_vorlagen/3_methode/02_technik2/
- Wer es ganz genau will: Qualitätskriterien für wissenschaftliches Arbeiten – https://soztheo.de/forschung/qualitaetskriterien-fuer-wissenschaftliches-arbeiten/

Anhang: wie man einen wissenschaftlichen Artikel liest

Wissenschaftliche Artikel sind meistens nicht dafür geschrieben, von vorne bis hinten gelesen zu werden. In Ihrem Studium werden Sie aber viele wiss. Publikationen lesen. Da hilft es oft, eine klare Strategie zu haben, wie man das angeht.

Ich habe hier für Sie die Ultrakurzversion zusammengeschrieben. Sie finden nach diesem kurzen Guide einige Links zu längeren Versionen. Dieser Guide gilt für »typische« wissenschaftliche Texte, also solche, die dem üblichen Aufbau folgen.

1. Überfliegen Sie das Abstract. Sie werden dann verstehen, um was es im Artikel geht, warum die Arbeit verfasst wurde, und in wenigen Worten üblicherweise auch, was das Ergebnis der Arbeit war. Das hilft Ihnen, den Rest besser einordnen zu können.

2. Lesen Sie jetzt den letzten Abschnitt des Papers, üblicherweise »Conclusions« oder »Discussion« genannt. Damit sollten Sie jetzt wissen, was die Autor_innen gemacht haben, und warum Sie es gemacht haben. Sie wissen auch, was dabei herausgekommen ist.

3. Der Abschnitt vor den Schlussfolgerungen sind üblicherweise »Results«. Überfliegen Sie diesen Teil, um zu sehen, wie relevant er für Sie ist.

4. Sehen Sie sich die Abbildungen an. In groben Zügen können Sie jetzt verstehen, um was es in diesem Paper geht, und was die Autor_innen gemacht haben. Zugegeben, das wird einfacher, je öfter Sie es machen.

5. Es sollte einen Abschnitt geben, der die Methodologie beschreibt, meistens »Methods« o.ä. Versuchen Sie grob zu verstehen, wie die Autor_innen gearbeitet haben (qualitativ, quantitativ, etc.).

Sie haben jetzt ein gutes Bild davon, um was es geht, und können entscheiden, ob Sie den Rest des Papers auch lesen wollen (zB. weil es relevant oder interessant ist). Eventuell ist aber auch nur noch der Abschnitt »Related Work« (o.ä.) für Sie spannend, weil Sie dort weitere Papers finden, die sich mit derselben oder einer ähnlichen Fragestellung beschäftigen – und vielleicht suchen Sie ja genau solche Arbeiten.

Weitere Guides:

- https://drewdennis.medium.com/how-to-read-scientific-papers-quickly-efficiently-e7030c4018fa
- https://www.bmj.com/about-bmj/resources-readers/publications/how-read-paper
- https://paperpile.com/g/read-scientific-paper/