Praktikum Algorithmen

Sommersemester 2025

TECHNISCHE UNIVERSITÄT DARMSTADT

Bewerbung

Bewerbungen für das Praktikum werden bis zum **28.04.2025** um **08:00** Uhr (MESZ) unter der E-Mail Adresse **lab@algo.informatik.tu-darmstadt.de** angenommen. Danach ist keine Bewerbung mehr möglich. Bewerbungen unter anderen E-Mail Adressen werden ignoriert. In der Bewerbung schreiben Sie uns bitte eine kurze Motivation, warum Sie genau dieses Thema bearbeiten möchten sowie Ihre Vorkenntnisse.

Kick-Off & Treffen

Der Termin für das Kick-Off-Treffen wird nach der Bewerbungsphase bekannt gegeben. Darauffolgende Treffen finden nicht in einem festen Rhythmus statt, sondern werden bei Bedarf von den Teilnehmenden angefragt.

Fortschrittsberichte

Während des Semesters ist die Abgabe **wöchentlicher**, kurzer Fortschrittsberichte verpflichtend. Dafür senden Sie jeweils eine E-Mail mit folgendem Inhalt an den Betreuenden:

- Welche Fortschritte / Erkenntnisse wurden seit dem letzten Bericht erreicht?
- Welche nächsten Schritte sind geplant?
- evtl. Liste mit Fragen - bitte achten Sie darauf, dass die Bewertungskriterien unter anderem selbstständiges Arbeiten berücksichtigen. Hier sollten Sie folglich nur Fragen stellen, deren Beantwortung nicht durch Google bzw. das Lesen von bereitgestelltem Material möglich ist.
- Möchten Sie ein persönliches Treffen zur Beantwortung der Fragen oder Besprechen des weiteren Vorgehens vereinbaren?

Falls ja, nennen Sie bitte Terminvorschläge: Zeitfenster, in denen Ihnen ein Treffen möglich ist

Wird wiederholt kein Fortschrittsbericht abgegeben, ist einer weitere Bearbeitung des Themas ausgeschlossen.
Sollte es Ihnen nicht möglich sein innerhalb einer Woche am Praktikum zu arbeiten, begründen Sie dies bitte kurz im Fortschrittsbericht, mögliche Gründe können z.B. sein: Erkrankung, Workload aus anderen Veranstaltungen oder Lernen für Klausuren.

Abgabe

Die endgültige Abgabe besteht aus:

- Code, der für dieses Praktikum geschrieben wurde (beinhaltet Code, der Auswertungen, Tests oder ähnliches durchführt).
- Einen Praktikumsbericht, in dem Sie beschreiben, welche Probleme Sie wie gelöst haben und wieso Sie diesen konkreten Ansatz gewählt haben.

Folgende Deadlines sind einzuhalten:

- Bis zum **16.09.2025** muss ein feature-vollständiger Stand als Git Patch per E-Mail geschickt werden – spätere Einreichungen werden nicht berücksichtigt.
- Ab dem 17.09.2025 muss ein Pull Request an das `motis-project/osr` Repository gestellt werden. Nun können ggf. noch Änderungen vorgenommen werden, um von der CI festgestellte Probleme zu beheben. Das Ziel ist, dass alle Checks der CI erfolgreich durchlaufen.
- Der Abgabetermin des Praktikumsberichtes und Deadline für letzte Änderungen am PR ist der **30.09.2025**.

Selbsteinschätzung für das Praktikum Algorithmen

Da die Themen des Praktikums im Rahmen des MOTIS[a] Projekts stattfinden, erfordern sie eine selbstständige Einarbeitung in selbiges. Unsere Erfahrungen haben gezeigt, dass Studierende die Komplexität eines modernen, real-world C++ Projekts und den damit verbundenen Arbeitsaufwand häufig unterschätzen.

Führen Sie daher bitte **vor** einer Bewerbung die folgenden Schritte zur Selbsteinschätzung Ihrer Eignung durch.

1. Forken Sie das MOTIS Projekt auf GitHub und klonen Sie Ihren Fork
2. Bauen Sie das Projekt:
   - Linux Developer Setup[b]
   - Windows Developer Setup[c]
   - MacOS Developer Setup[d]
3. Setzen Sie entsprechend der Anleitung[e] einen Motis Server auf
4. Rufen Sie `localhost:8080` mit Ihrem Browser auf und prüfen Sie die Funktionalität mit einigen Testanfragen an das Routingsystem

Uns ist bewusst, dass das erstmalige Aufsetzen des Projekts je nach Vorkenntnissen mit einer steilen Lernkurve verbunden sein kann. Die Selbsteinschätzung dient nicht dazu Studierende von der Teilnahme am Praktikum auszuschließen, sondern soll eine Hilfestellung sein und Sie beim Sicherstellen Ihres Studienerfolges unterstützen.

Das selbstständige Einarbeiten in einen unbekannten, umfangreichen Quellcode anhand bereitgestellter Dokumentation ist eine Schlüsselfähigkeit, die Sie im Rahmen dieser Selbsteinschätzung verbessern können.

[a] `https://github.com/motis-project/motis`
[b] `https://github.com/motis-project/motis/blob/master/docs/linux-dev-setup.md`
[c] `https://github.com/motis-project/motis/blob/master/docs/windows-dev-setup.md`
[d] `https://github.com/motis-project/motis/blob/master/docs/macos-dev-setup.md`
[e] `https://github.com/motis-project/motis/blob/master/docs/dev-setup-server.md`

Hinweise:

- Planen Sie bitte zusätzlichen Aufwand ein, falls die genannten Programmiersprachen (C++, Typescript, Javascript, CUDA) oder verwendeten Technologien (z.B. Git, CMake, React, etc.) neu für Sie sind.
- Betrifft alle C++ Praktika: Als Referenz für die Komplexität des Projektes, das im Praktikum erweitert wird, dient MOTIS.
  Bitte überlegen Sie sich genau, ob Sie mit einer großen real-world Modern-C++ Code Basis klarkommen. Bisherige Erfahrungen haben gezeigt, dass Studierende den Einarbeitungsaufwand und die Komplexität der Programmiersprache unterschätzen.
- Einen Crash-Kurs in die MOTIS-typische C++-Programmierung gibt es hier: `https://www.algo.informatik.tu-darmstadt.de/dl/cpp.pdf`.
- MOTIS C++ Style Guide: `https://github.com/motis-project/motis/blob/master/docs/STYLE.md`
- Falls Themen in Gruppen bearbeitet werden, steigt der Aufwand linear mit der Summe der CPs der Gruppenmitglieder. Die unten genannten Themen sind auf Einzelpersonen ausgelegt. Bei einer Bearbeitung in der Gruppe (nur bei Themen mit **G** möglich!) wird der Umfang entsprechend erweitert.
- Um die Integration der Praktikumsergebnisse in das quelloffene Projekt MOTIS zu ermöglichen, müssen alle Teilnehmer Ihre Abgaben unter die MIT / Apache2 Lizenz stellen. Nach dem Kick-Off senden Sie dem Betreuer des Themas eine unterschriebene Lizenzvereinbarung (MIT / Apache2) und falls nötig eine unterschriebene Vertraulichkeitserkärung für die für das Praktikum bereitgestellten Daten. Dies wird gleichzeitig als verbindliche Zusage zur Bearbeitung des Themas angesehen.

Markierungen der Themen:

A Das Thema kann als Praktikum Algorithmen (6CP) oder Vertiefungspraktikum Algorithmen (6CP) bearbeitet werden.

P Das Thema kann als Projektpraktikum (9CP) bearbeitet werden.

E Das Thema kann einzeln bearbeitet werden.

G Das Thema kann auch als Gruppe bearbeitet werden.

Der Umfang des Themas wird den Umständen entsprechen angepasst. Die Praktikumsform 'Projektpraktikum', als auch Gruppenarbeit, erhöhen selbstverständlich den Arbeitsumfang.

Speed-up Techniken für das Straßenrouting mit OSR

OSR (Open Street Router) ist ein speichereffizienter Straßenrouter, der verschiedene Nutzerprofile (z.B. Fußgänger, Fahrrad, Auto) unterstützt. Als Datengrundlage dienen Daten von OpenStreetMap.

Das OSR Repository auf GitHub: `https://github.com/motis-project/osr`

Contraction Hierarchies [1] [A, E]

Contraction Hierarchies (CH) ist eine Preprocessing-Technik verbunden mit einer Erweiterung von bidirektionalem Dijkstra für das Shortest Path Problem. Sie eignet sich insbesondere auch für Straßengraphen und kann die Berechnung gegenüber naivem Dijkstra um mehrere Größenordnungen beschleunigen.

Bei einer längeren Autoreise, z.B. von Hamburg nach München, ist intuitiv klar, dass der meiste Weg dazwischen über “wichtigere” Straßen, d.h. Autobahnen zurückgelegt wird. Naiver Dijkstra würde aber natürlich alle Knoten auch der kleinsten Dorfstraße bei Kassel besuchen. Contraction Hierarchies formalisieren diese Intuition in einer Weise, die weiterhin Optimalität garantiert. Konkret, indem durch das Einfügen von “Shortcuts” erreicht wird, dass vom Startknoten aus und ebenso in der Rückwärtssuche vom Endknoten aus immer nur zunehmend “wichtigere” Knoten besucht werden müssen und Kanten, die zu “unwichtigeren” Knoten führen, ignoriert werden können.

Implementiert werden soll sowohl das CH-Preprocessing als auch der bidirektionale Dijkstra, der auf präprozessierten Graphen arbeitet (vgl. [1] und potenziell Weiterentwicklungen). Für unseren Anwendungsfall in OSR sollen folgende Routingvarianten implmentiert werden:

- One-to-one Routing
- One-to-many Routing
- Many-to-many Routing (“Table Routing”)

Die Mindestanforderung zum Node Ordering ist eine zufällige Reihenfolge der Nodes. Sie können weitere Heuristiken zum Node Ordering Ihrer Wahl implementieren.

Zur Überprüfung der Korrektheit und der Performance im Vergleich zum vorhandenen naiven Dijkstra stellen wir Ihnen Tests zur Verfügung.

Literatur

[1] Robert Geisberger u. a. „Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks“. In: *Experimental Algorithms*. Hrsg. von Catherine C. McGeoch. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, S. 319–333. ISBN: 978-3-540-68552-4. DOI: 10.1007/978-3-540-68552-4_24.