Praktikum Algorithmen

Sommersemester 2026

TECHNISCHE UNIVERSITÄT DARMSTADT

Das Praktikum behandelt Themen im Rahmen des MOTIS[1] Projekts und erfordert 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 Anmeldung 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[2]
   - Windows Developer Setup[3]
   - MacOS Developer Setup[4]
3. Setzen Sie entsprechend der Anleitung[5] 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.

---

[1] `https://github.com/motis-project/motis`
[2] `https://github.com/motis-project/motis/blob/master/docs/linux-dev-setup.md`
[3] `https://github.com/motis-project/motis/blob/master/docs/windows-dev-setup.md`
[4] `https://github.com/motis-project/motis/blob/master/docs/macos-dev-setup.md`
[5] `https://github.com/motis-project/motis/blob/master/docs/dev-setup-server.md`

Modul: 20-00-0189 Praktikum Algorithmen, 6 CP

Speed-up Techniken für das Straßenrouting mit OSR: Customizable Contraction Hierarchies

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 (CH) [3] 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 alle Knoten auch der kleinsten Dorfstraße bei Kassel besuchen. Contraction Hierarchies formalisieren diese Intuition in einer Weise, die weiterhin Optimalität garantiert. Konkret, indem in einem Preprocessing-Schritt 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.

Customizable Contraction Hierarchies (CCH) [2] sind eine Weiterentwicklung der CH, die das Preprocessing in zwei Phasen aufteilt: Das Einfügen der Shortcuts auf einem ungewichteten Graphen und anschließend die “Customization” für gegebene Gewichte, die etwa vom betrachteten Verkehrsmittel abhängen können. Da der erste Teil i.d.R. deutlich teurer ist, können auf diese Weise relativ dynamisch neue Gewichte angewendet werden. Für die Abgabe sollen mindestens die OSR-Profile Car und Bus unterstützt werden.

Relevant für die Performance ist, in welcher Reihenfolge die Shortcuts erstellt werden (“Node Ordering” / “Rank”), da man intuitiv die wichtigsten Knoten möglichst spät mit Shortcuts überbrücken will. Die Reihenfolge wird von uns vorgegeben bzw. kann initial als zufällig angenommen werden.

Implementiert werden soll sowohl das CCH-Preprocessing als auch die bidirektionale one-to-one Query, die auf dem präprozessierten Graphen arbeitet.

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

Eine recht gut verständliche Einführung und Übersicht zu CCH gibt [1].

Zeitlicher Ablauf

Datum	Event
15.04.2026 15:00 Uhr	Kick-Off in S2|02 E202
27.04.2026	Fortschrittsbericht / Anmeldung
11.05.2026	Fortschrittsbericht
25.05.2026	Fortschrittsbericht
08.06.2026	Fortschrittsbericht
22.06.2026	Fortschrittsbericht
06.07.2026	Fortschrittsbericht
20.07.2026	Fortschrittsbericht
31.07.2026	1. Milestone Git Patch
10.08.2026	Fortschrittsbericht
24.08.2026	Fortschrittsbericht
bis 11.09.2026	feature-vollständiger Git Patch
ab 12.09.2026	Pull Request stellen
bis 30.09.2026	Finale Abgabe

Kick-Off

Der Kick-Off findet am Mittwoch, den **15.04.2026, von 15:00 bis 16:00 Uhr in Raum S2|02 E202** statt.

Anmeldung

Sie melden sich bei uns an durch Einsenden des ersten Fortschrittsberichts **bis zum 27.04.2026** unter der E-Mailadresse **lab@algo.informatik.tu-darmstadt.de** (siehe Abschnitt *Fortschrittsberichte* unten)

Sprechstunde

Die Sprechstunde findet jeden Mittwoch von 15:00 bis 16:00 Uhr in Raum S2|02 E123 statt. Dafür ist eine vorherige Anmeldung per E-Mail notwendig. Die Deadline hierfür ist jeweils am Montag derselben Woche. Außerhalb davon sind Nachfragen nur per E-Mail möglich.

Fortschrittsberichte

Während des Semesters ist die Abgabe kurzer Fortschrittsberichte verpflichtend. Die Fälligkeiten entnehmen Sie bitte der Tabelle über den zeitlichen Ablauf. Dafür senden Sie jeweils eine E-Mail mit folgendem Inhalt an die E-Mailadresse lab@algo.informatik.tu-darmstadt.de:

- 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 am Mittwoch die Sprechstunde zur Beantwortung der Fragen oder Besprechen des weiteren Vorgehens besuchen?

Wird wiederholt kein Fortschrittsbericht abgegeben, ist eine weitere Bearbeitung des Praktikums ausgeschlossen.
Sollte es Ihnen nicht möglich gewesen sein 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.

1. Milestone

Als 1. Milestone soll eine vollständige Implementierung des beschriebenen Ansatzes abgegeben werden. Die Implementierung muss noch nicht vollständig korrekt sein. Sie werden im Verlauf des Praktikums noch weiter daran arbeiten, um sie auf real-world Daten zu testen, Fehler zu beheben und die Performance zu verbessern. Die Abgabe erfolgt in Form eines Git Patches per E-Mail an **lab@algo.informatik.tu-darmstadt.de**. Die Deadline für die Abgabe des 1. Milestones ist:

31.07.2026 23:59:59 Uhr (Ortszeit Darmstadt)

Spätere Abgaben werden nicht berücksichtigt.

Finale Abgabe

Zusätzlich zum 1. Milestone soll die finale Abgabe Folgendes enthalten.

- Einen Praktikumsbericht, in dem Sie beschreiben, auf welche Probleme Sie während der Implementierung gestoßen sind und wie Sie diese gelöst haben. Erläutern Sie Ihre Ergebnisse aus der der Validierung und führen Sie Messungen der Performance Ihrer Implementierung durch. Der Umfang soll 3-4 Seiten betragen.

Folgende Deadlines sind einzuhalten:

11.09.2026 23:59:59 Uhr (Ortszeit Darmstadt) Bis zu dieser Deadline muss ein featurevollständiger Stand als Git Patch per E-Mail geschickt werden – spätere Einreichungen werden nicht berücksichtigt.

12.09.2026 00:00:00 Uhr (Ortszeit Darmstadt) Ab dieser Deadline darf ein Pull Request an das *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.

30.09.2026 23:59:59 Uhr (Ortszeit Darmstadt) Der Abgabetermin des Praktikumsberichtes per E-Mail und Deadline für letzte Änderungen am PR - spätere Einreichungen werden nicht berücksichtigt.

Hinweise

- Eine Bearbeitung des Praktikums in Gruppen ist nicht möglich.
- Obige Aufgabenstellung betrifft das Modul 20-00-0189 Praktikum Algorithmen. Nach erfolgreichem Abschluss ist es möglich in einem späteren Semester das Modul 20-00-0276 Praktikum Algorithmen II (Vertiefung) zu belegen. Hierbei werden mit den Studierenden individuelle Aufgabenstellungen erarbeitet.
- Das Modul 20-00-1029 Projektpraktikum Algorithmik kann nur nach erfolgreichem Abschluss des Moduls 20-00-0189 Praktikum Algorithmen, oder alternativ einer Bachelor-/Masterarbeit am Fachgebiet Algorithmik, belegt werden. Hierbei werden mit den Studierenden individuelle Aufgabenstellungen erarbeitet.
- Planen Sie bitte zusätzlichen Aufwand ein, falls die genannten Programmiersprachen oder verwendeten Technologien (z.B. Git, CMake, etc.) neu für Sie sind.
- 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`

Literatur

[1] Thomas Bläsius u. a. „Customizable Contraction Hierarchies – A Survey“. In: (Feb. 2025). DOI: 10.48550/ARXIV.2502.10519. arXiv: 2502.10519 [cs.DS].

[2] Julian Dibbelt, Ben Strasser und Dorothea Wagner. „Customizable Contraction Hierarchies“. In: *ACM Journal of Experimental Algorithmics* 21 (Apr. 2016), S. 1–49. ISSN: 1084-6654. DOI: 10.1145/2886843.

[3] 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.