Themen für das Praktikum Algorithmen

TECHNISCHE UNIVERSITÄT DARMSTADT

Wintersemester 2023

- Bewerbungen auf Praktikumsthemen werden bis zum **23.10.2023** um **08:00** 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. Jede Bewerbung muss drei Themen mit Prioritäten (Erst-, Zweit- und Drittwunsch) enthalten. Zur Bewerbung auf ein Thema schreiben Sie uns bitte eine kurze Motivation, warum Sie genau dieses Thema bearbeiten möchten sowie Ihre Vorkenntnisse.
- Das Kick-Off Treffen findet nach Vereinbarung für jedes Thema separat statt.
- Um die Integration der Praktikumsergebnisse in die quelloffenen Projekte SORO-S/O bzw. 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.
- In einem zweiwöchigen Rythmus finden regelmäßige verpflichtende Treffen statt. Die genauen Termine werden beim Kick-Off Treffen vereinbart. Bei jedem Termin stellt jeder Teilnehmer einen Foliensatz mit 3 Folien vor (max. 5 min):
  - Welche Fortschritte / Erkenntnisse wurden seit dem letzten Treffen erreicht?
  - Welche nächsten Schritte sind geplant?
  - 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.
- 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.

Es gelten folgende Markierungen:

- **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 Abgabetermin des Praktikums ist der 30.04.2024.

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, können Sie MOTIS[1] anschauen. Das SORO Projekt aus den Themen 2.x ist ähnlich komplex.
  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/SORO-typische C++-Programmierung gibt es hier: `https://www.algo.informatik.tu-darmstadt.de/dl/cpp.pdf`.
- 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.

[1] `https://github.com/motis-project/motis`

Thema 1: Maintaining a Topological Order after Batch Insertion [A, P, E, G]

Zu einem gerichteten, azyklischen Graphen (DAG) G kann eine topologische Sortierung der Knoten ermittelt werden. Bei einer topologischen Sortierung handelt es sich um eine Reihenfolge der Knoten, sodass für jede Kante (u, v) aus gilt: u erscheint vor v in der Sortierung. Offensichtlich bleibt eine gegebene topologische Sortierung gültig, falls man Kanten aus dem Graphen entfernt. Anders verhält es sich beim Einfügen von neuen Kanten. Durch Zykelschluss kann das Ermitteln einer topologischen Sortierung unmöglich werden oder eine bestehende topologische Sortierung ungültig werden. Konkrete Problemstellung des Praktikums ist das möglicherweise notwendige Reparieren einer topologischen Sortierung nach Einfügen einer Menge neuer Kanten (Batch) in einen DAG.

Aufgabe des Praktikums ist Literaturrecherche zum Vergleichen von bereits existierenden Algorithmen und Implementieren eines Algorithmus inklusive ausführlichem Testen. Bei der Implementierung soll die Performanz ausführlich ausgewertet werden.

Programmiersprache: C++20

Thema 2: Energieeffiziente Fahrstrategie eines Zuges [A, E, G]

In diesem Projekt soll SORO-S um einen Algorithmus zur Berechnung einer energieeffizienten Fahrstrategie eines Zuges erweitert werden. Des Weiteren sollen im Rahmen des Projekts zwei Auswertungsmöglichkeiten in SORO-S integriert werden.

1. Auswertung und Vergleich eines beliebigen Algorithmus zur Berechnung einer energieeffizienten Fahrstrategie mit der technisch-kürzesten Fahrzeit bzgl. Energieverbrauch, Fahrzeit, Berechnungszeit, etc.
2. Auswertung und Vergleich zweier beliebiger Algorithmen zur Berechnung beliebiger Fahrstrategien bzgl. Energieverbrauch, Fahrzeit, Berechnungszeit, etc.

Programmiersprache: C++20

Thema 3: Effiziente Vorberechnung von Fahrstrategien in SORO-S [A, E]

Mit Hilfe der Bahnbetriebssimulation SORO-S lassen sich verschiedene Algorithmen zur Berechnung von Fahrstrategien von Zügen betrachten. Um diese Algorithmen in weiterführenden Problemstellungen, wie z.B. der Dispositionsplanung oder der Fahrplanoptimierung, verwenden zu können, ist eine Vorberechnung der Fahrstrategie beliebiger Züge in verschiedensten Szenarien notwendig, so dass diese Problemstellungen effizient

(d.h. ohne zusätzliche Simulation einer Zugfahrt) gelöst werden können. In Abhängigkeit der berücksichtigten Parameter ist die Vorberechnung jedoch selbst bei einem effizienten Algorithmus (zur Berechnung einer Fahrstrategie) nicht effizient möglich.

Ziel dieses Projekts ist die Entwicklung einer Datenstruktur zur effizienten Verwaltung von Parameterkombinationen und ihrer Simulationsergebnisse sowie die Implementierung eines Algorithmus zur Reduktion der in der Vorberechnung betrachteten Parameterkombinationen. Zusätzlich soll die Funktionalität der Datenstruktur in Kombination mit dem Algorithmus durch Anwendung einer Vorberechnung gezeigt werden. Bei der Implementierung soll die Performanz - insbesondere der zusätzliche Aufwand durch Nutzung der neuen Datenstruktur - ausführlich ausgewertet werden.

Programmiersprache: C++20