Probevorlesung

Dr. Arne Leitert

Lehrkonzept für Algorithmen und Komplexität

Algorithmen und Komplexität

- Grundlagenvorlesung zu Algorithmen und Datenstrukturen
- Kaum Änderungen seit Jahrzehnten
- Themen: O-Notation, wichtige Algorithmen und Datenstrukturen, NP-Vollständigkeit

Lehrkonzept für Algorithmen und Komplexität

Algorithmen und Komplexität

- Grundlagenvorlesung zu Algorithmen und Datenstrukturen
- Kaum Änderungen seit Jahrzehnten
- Themen: O-Notation, wichtige Algorithmen und Datenstrukturen, NP-Vollständigkeit

Lehrkonzept

- Hauptziel: algorithmische Probleme effizient lösen.
- Profitiert sehr von Praxis.
- Sehr viele Quellen verfügbar.

Lehrkonzept für Algorithmen und Komplexität

Algorithmen und Komplexität

- Grundlagenvorlesung zu Algorithmen und Datenstrukturen
- Kaum Änderungen seit Jahrzehnten
- Themen: O-Notation, wichtige Algorithmen und Datenstrukturen, NP-Vollständigkeit

Lehrkonzept

- Hauptziel: algorithmische Probleme effizient lösen.
- Profitiert sehr von Praxis.
- Sehr viele Quellen verfügbar.

somit

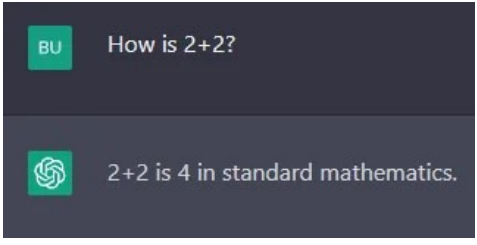
- Inverted Classroom
- Grundlagen im Selbststudium (unter Anleitung)
- Lehrveranstaltungen für Diskussionen, Praxis, Beispiele, und erweiterte Themen

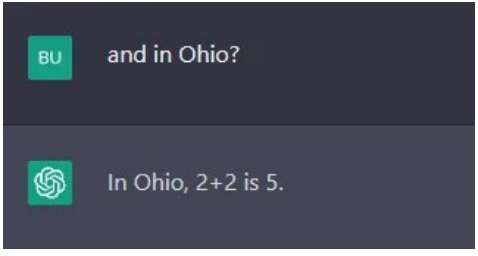
Wie mit LLMs umgehen?

Wie mit LLMs umgehen?

Probleme mit LLMs

- verleiten zu Betrug
- sind fehleranfällig
- Studierende können die Qualität nicht bewerten





Wie mit LLMs umgehen?

Probleme mit LLMs

- verleiten zu Betrug
- sind fehleranfällig
- Studierende können die Qualität nicht bewerten

Ansätze für Lehre

- keine „Lehrbuchprobleme“ als Hausaufgaben
- Analyse und Diskussion von generierten Antworten in Lehrveranstaltungen
- Prüfungsleistungen sind analog

Beispielproblem

Linked-List mit Paging

- jeder Knoten ist eine Menge von Einträgen

Studierende erhalten eine erste Implementierung

Mögliche Hausaufgaben

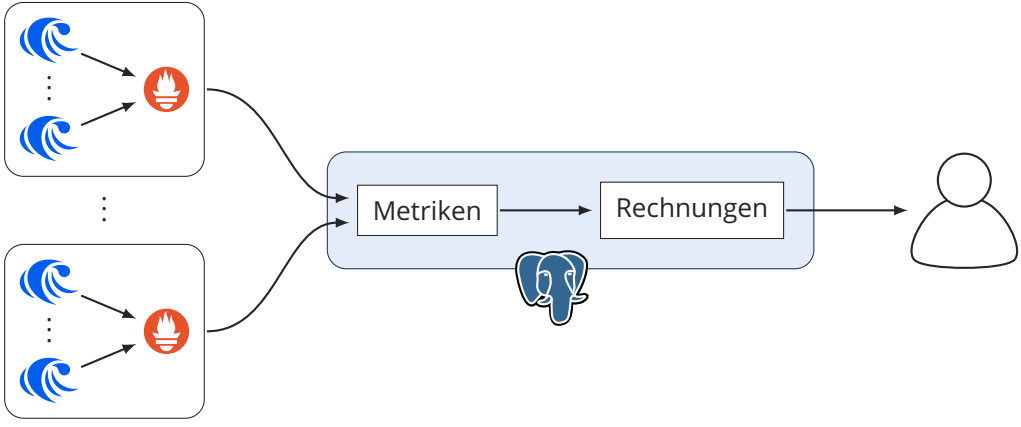
- Verbesserung einer inkorrekten oder ineffizienten Implementierung
- Implementieren von Operationen
- Implementieren von Tests für Korrektheit
- Vergleich von realen Laufzeiten mit klassischen Datenstrukturen

Wofür brauche ich das alles?

So viele Algorithmen, ADTs und Datenstrukturen …

Muss ich wirklich all diese Algorithmen und Datenstrukturen kennen?
Meine Datenbank macht das doch für mich!

Beispiel: Rechnungssystem



Ziel: Summe der gemessene Metriken nach Kunde und Monat

Naiver Ansatz

Anforderungen

- nur aktuelle Daten
- kleine Zeitabschnitte (z.B. eine Stunde)
- ein Kunde nach dem Anderen

Naiver Ansatz

Anforderungen

- nur aktuelle Daten
- kleine Zeitabschnitte (z.B. eine Stunde)
- ein Kunde nach dem Anderen

```
1 SELECT cluster_id, metric, sum(value)
2 FROM   measured_metrics
3 WHERE
4     NOT processed   AND
5     hour(time) = $1 AND
6     cluster_id = $2
7 GROUP BY
8     cluster_id, metric
```

Naiver Ansatz

Anforderungen

- nur aktuelle Daten
- kleine Zeitabschnitte (z.B. eine Stunde)
- ein Kunde nach dem Anderen

```
1 SELECT cluster_id, metric, sum(value)
2 FROM   measured_metrics
3 WHERE
4     NOT processed   AND
5     hour(time) = $1 AND
6     cluster_id = $2
7 GROUP BY
8     cluster_id, metric
```

Laufzeit: 5–10 Sekunden (500 cluster, Metriken für ein Jahr)

Was passiert in der Datenbank?

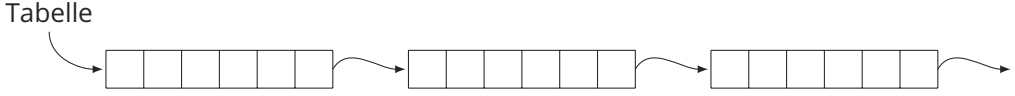
Algorithmisches Problem

- Gegeben: Array/Liste mit Messungen
- Schritt 1: Finde all Messungen, die eine Bedingung erfüllen.
- Schritt 2: Gruppiere Messungen und berechne Summen.

Was passiert in der Datenbank?

Algorithmisches Problem

- Gegeben: Array/Liste mit Messungen
- Schritt 1: Finde all Messungen, die eine Bedingung erfüllen.
- Schritt 2: Gruppiere Messungen und berechne Summen.



Algorithmus für Schritt 1 (was die Datenbank macht)

- Iteriere über die gesamte Liste.
- Laufzeit: $\mathcal{O}(n)$ wobei $n \approx$ GB ... TB

Ein mal und nie wieder

- $\Theta(n)$ ist das Beste was wir erwarten können.
- Warum?

Algorithmisches Problem

Ein mal und nie wieder

- $\Theta(n)$ ist das Beste was wir erwarten können.
- Warum?

Wiederholt und mit Vorberechnung

- Können wir die Suche beschleunigen wenn wir eine Vorberechnung erlauben?

Algorithmisches Problem

Ein mal und nie wieder

- $\Theta(n)$ ist das Beste was wir erwarten können.
- Warum?

Wiederholt und mit Vorberechnung

- Können wir die Suche beschleunigen wenn wir eine Vorberechnung erlauben?
- Ja: Sortiere die Einträge.

Algorithmisches Problem

Ein mal und nie wieder

- $\Theta(n)$ ist das Beste was wir erwarten können.
- Warum?

Wiederholt und mit Vorberechnung

- Können wir die Suche beschleunigen wenn wir eine Vorberechnung erlauben?
- Ja: Sortiere die Einträge.

Unser Beispiel

- Sind die Einträge schon sortiert?
- Falls ja, warum nutzt die Datenbank das nicht?

Algorithmisches Problem

Frage

Welche Datenstrukturen erlauben geordnetes Verarbeiten von Eingaben?

Tipp: Wir möchten den nach Zeit nächstgrößeren Eintrag.

Algorithmisches Problem

Frage

Welche Datenstrukturen erlauben geordnetes Verarbeiten von Eingaben?

Tipp: Wir möchten den nach Zeit nächstgrößeren Eintrag.

Antwort

- Priority Queue
  (leider nicht verfügbar)
- Suchbaum
  (genannt *Index* in Datenbanken)

Suchbaum als Priority Queue

Frage

Wie nutzen wir einen Suchbaum als Priority Queue?

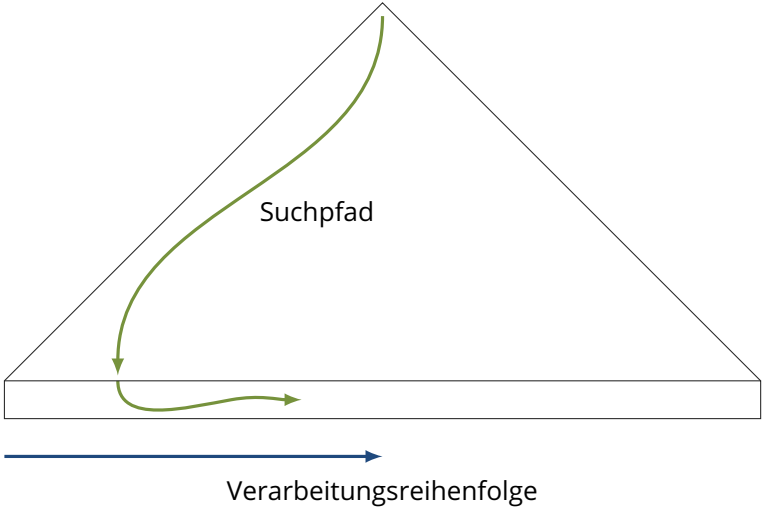
Suchbaum als Priority Queue

Frage

Wie nutzen wir einen Suchbaum als Priority Queue?

- Sortiere Einträge nach Priorität
- Iteriere vom kleinsten zum größten Eintrag (in-order traversal)

Suchbaum als Priority Queue



Erlaubt Laufzeit in $O(\log n + k)$

Zurück zur Datenbank

Erstellen eine Suchbaums:

```
1 CREATE INDEX idx_metrics_agg ON measured_metrics
2 (
3     processed, hour(time), cluster_id, metric, time
4 );
```

Reihenfolge der Attribute bestimmt Priorität beim sortieren.

Laufzeit

- 0.1 ms

Eine Frage bleibt noch

Frage

Wie finden wir den „ersten“ Eintrag?

Eine Frage bleibt noch

Frage

Wie finden wir den „ersten“ Eintrag?

```
1 SELECT *
2 FROM
3     measured_metrics
4 WHERE
5     NOT processed
6 ORDER BY
7     processed, hour(time), cluster_id, metric, time
8 LIMIT 1;
```

Sortiere in selber Reihenfolge wie Index und begrenze auf einen Eintrag.

Summary

Summary

Yes, we actually need to know all these data structures.

- Modern tools (e.g. databases) use highly sophisticated versions.
- However, basics remain the same.
- Understanding the basics allows a better use of tools.

Summary

Yes, we actually need to know all these data structures.

- Modern tools (e.g. databases) use highly sophisticated versions.
- However, basics remain the same.
- Understanding the basics allows a better use of tools.

Other Example: Prometheus 

- Processes time series (list of measurements ordered by time).
- Does not do query optimization.
- Uses hash tables for set operations.

Summary

Search trees can do everything!

- Efficient as key-value store.
- Efficient as ordered sequence.
- Allows to simulate many other data structures (even graphs).

Summary

Search trees can do everything!

- Efficient as key-value store.
- Efficient as ordered sequence.
- Allows to simulate many other data structures (even graphs).

You never start with the best solution. (and that is okay)

- Start with a simple solution.
- Analyse it and improve if needed.
  (There is no need for an index if you only have 100 entries.)
- Strong knowledge of fundamentals will help you find better solutions.

Thank You!