Praktikum – Beating the Worst Case

Beating the Worst Case

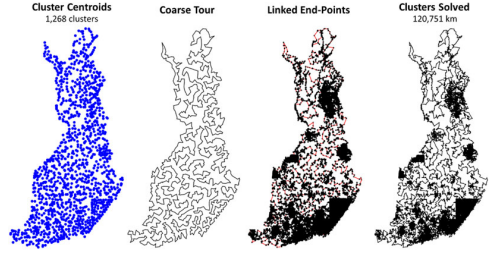
Jean-Pierre von der Heydt und Marcus Wilhelm | 25.10.2023

Motivation

- Das $\mathcal{O}$-Kalkül ist eines der wichtigsten Werkzeuge, um Algorithmen zu analysieren
- Häufig ist die obere Schranke eine sehr pessimistische Abschätzung

Motivation

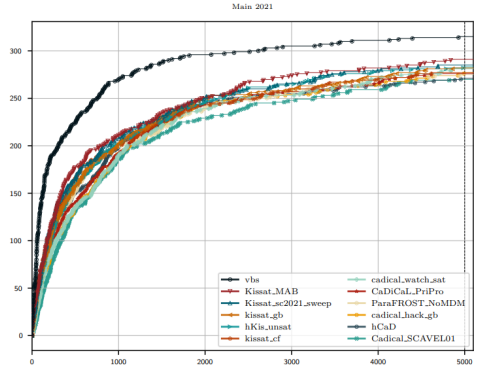
- Das $\mathcal{O}$-Kalkül ist eines der wichtigsten Werkzeuge, um Algorithmen zu analysieren
- Häufig ist die obere Schranke eine sehr pessimistische Abschätzung
- TSP-Solver können optimale Routen für Graphen mit $\sim$1Mio. Knoten finden
- SAT-Solver können Probleme mit $\sim$1Mio. Variablen und noch mehr Klauseln lösen



Source: *www.frontiersin.org/articles/10.3389/frobt.2021.689908/full*

Motivation

- Das $\mathcal{O}$-Kalkül ist eines der wichtigsten Werkzeuge, um Algorithmen zu analysieren
- Häufig ist die obere Schranke eine sehr pessimistische Abschätzung
- TSP-Solver können optimale Routen für Graphen mit ∼1Mio. Knoten finden
- SAT-Solver können Probleme mit ∼1Mio. Variablen und noch mehr Klauseln lösen



Beobachtung

Praktische Laufzeit ist nicht in $2^{\mathcal{O}(n)}$ sondern viel besser. TSP und SAT sind nicht die einzigen Problem.

Motivation

- Zusammenspiel aus Algorithmus und Eingabeverteilung
- Praktische Laufzeit kann asymptotisch besser sein als in der Analyse

Motivation

- Zusammenspiel aus Algorithmus und Eingabeverteilung
- Praktische Laufzeit kann asymptotisch besser sein als in der Analyse

Unser Ziel

- Wir schauen uns ein paar dieser Probleme genauer an:
- Warum sind manche Algorithmen so schnell? Auf welchen Eingaben funktionieren sie nicht?

Motivation

- Zusammenspiel aus Algorithmus und Eingabeverteilung
- Praktische Laufzeit kann asymptotisch besser sein als in der Analyse

Unser Ziel

- Wir schauen uns ein paar dieser Probleme genauer an:
- Warum sind manche Algorithmen so schnell? Auf welchen Eingaben funktionieren sie nicht?
- Wie kann ich ein brauchbares Test-Setup aufbauen?
- Mit welchen Methoden kann ich meine Daten gut visualisieren?

Ablauf

Erster Teil:

- Übungsblatt 0 mit coolem Rätsel (3 Wochen).
- Weitere Übungsblätter (∼6 Wochen).
- Aufbau eines Setups für Experimente, Visualisieren von Daten, Auswerten von Algorithmen

Ablauf

Erster Teil:

- Übungsblatt 0 mit coolem Rätsel (3 Wochen).
- Weitere Übungsblätter (∼6 Wochen).
- Aufbau eines Setups für Experimente, Visualisieren von Daten, Auswerten von Algorithmen

Zweiter Teil:

- Selbstständiges Projekt an einem neuen Thema (∼6 Wochen)
- Planung und Umsetzung eines Versuchsaufbaus
- Anschließende Ausarbeitung und Präsentation

Ablauf

Erster Teil:

- Übungsblatt 0 mit coolem Rätsel (3 Wochen).
- Weitere Übungsblätter (∼6 Wochen).
- Aufbau eines Setups für Experimente, Visualisieren von Daten, Auswerten von Algorithmen

Zweiter Teil:

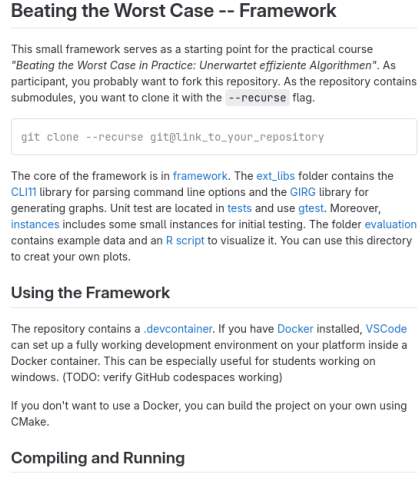
- Selbstständiges Projekt an einem neuen Thema (∼6 Wochen)
- Planung und Umsetzung eines Versuchsaufbaus
- Anschließende Ausarbeitung und Präsentation

Bewertung

Die Bewertung des Praktikums findet nur anhand der Ausarbeitung und der Präsentation statt.

Framework

- Forkt das repository und macht euch mit ihm vertraut. Die READMEs sollten helfen das meiste zu verstehen.



Beating the Worst Case -- Framework

This small framework serves as a starting point for the practical course *"Beating the Worst Case in Practice: Unerwartet effiziente Algorithmen"*. As participant, you probably want to fork this repository. As the repository contains submodules, you want to clone it with the `--recurse` flag.

```
git clone --recurse git@link_to_your_repository
```

The core of the framework is in framework. The ext_libs folder contains the CLI11 library for parsing command line options and the GIRG library for generating graphs. Unit test are located in tests and use gtest. Moreover, instances includes some small instances for initial testing. The folder evaluation contains example data and an R script to visualize it. You can use this directory to creat your own plots.

Using the Framework

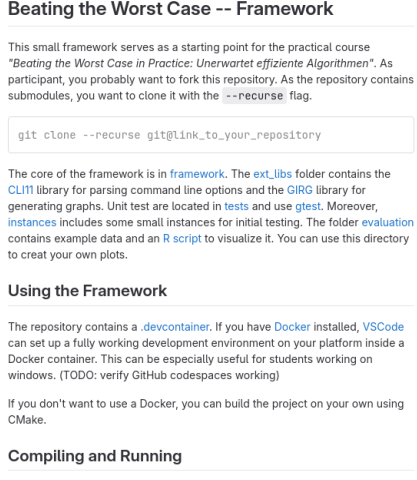
The repository contains a .devcontainer. If you have Docker installed, VSCode can set up a fully working development environment on your platform inside a Docker container. This can be especially useful for students working on windows. (TODO: verify GitHub codespaces working)

If you don't want to use a Docker, you can build the project on your own using CMake.

Compiling and Running

Framework

- Forkt das repository und macht euch mit ihm vertraut. Die READMEs sollten helfen das meiste zu verstehen.
- Graph-I/O und ein paar nützliche Algorithmen sind in `framework`
- Beispiel Command Line Applikationen in `cli`
- Zum Auswerten von Daten empfehlen wir *R* mit Beispielen in `evaluation`



Beating the Worst Case -- Framework

This small framework serves as a starting point for the practical course *"Beating the Worst Case in Practice: Unerwartet effiziente Algorithmen"*. As participant, you probably want to fork this repository. As the repository contains submodules, you want to clone it with the `--recurse` flag.

```
git clone --recurse git@link_to_your_repository
```

The core of the framework is in framework. The ext_libs folder contains the CLI11 library for parsing command line options and the GIRG library for generating graphs. Unit test are located in tests and use gtest. Moreover, instances includes some small instances for initial testing. The folder evaluation contains example data and an R script to visualize it. You can use this directory to creat your own plots.

Using the Framework

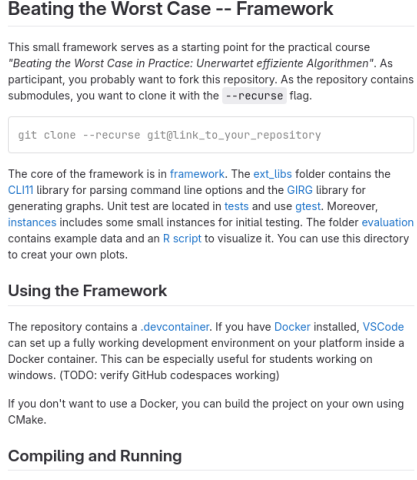
The repository contains a .devcontainer. If you have Docker installed, VSCode can set up a fully working development environment on your platform inside a Docker container. This can be especially useful for students working on windows. (TODO: verify GitHub codespaces working)

If you don't want to use a Docker, you can build the project on your own using CMake.

Compiling and Running

Framework

- Forkt das repository und macht euch mit ihm vertraut. Die READMEs sollten helfen das meiste zu verstehen.
- Graph-I/O und ein paar nützliche Algorithmen sind in `framework`
- Beispiel Command Line Applikationen in `cli`
- Zum Auswerten von Daten empfehlen wir *R* mit Beispielen in `evaluation`
- Versucht das Repository mit cmake zu bauen und das Programm `run` auszuführen



Beating the Worst Case -- Framework

This small framework serves as a starting point for the practical course *"Beating the Worst Case in Practice: Unerwartet effiziente Algorithmen"*. As participant, you probably want to fork this repository. As the repository contains submodules, you want to clone it with the `--recurse` flag.

```
git clone --recurse git@link_to_your_repository
```

The core of the framework is in framework. The ext_libs folder contains the CLI11 library for parsing command line options and the GIRG library for generating graphs. Unit test are located in tests and use gtest. Moreover, instances includes some small instances for initial testing. The folder evaluation contains example data and an R script to visualize it. You can use this directory to creat your own plots.

Using the Framework

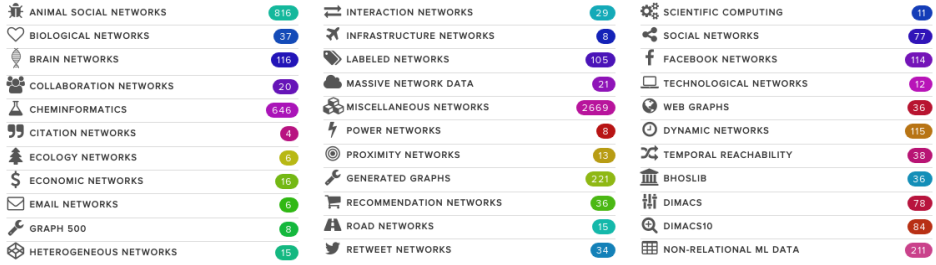
The repository contains a .devcontainer. If you have Docker installed, VSCode can set up a fully working development environment on your platform inside a Docker container. This can be especially useful for students working on windows. (TODO: verify GitHub codespaces working)

If you don't want to use a Docker, you can build the project on your own using CMake.

Compiling and Running

Übungsblatt 0

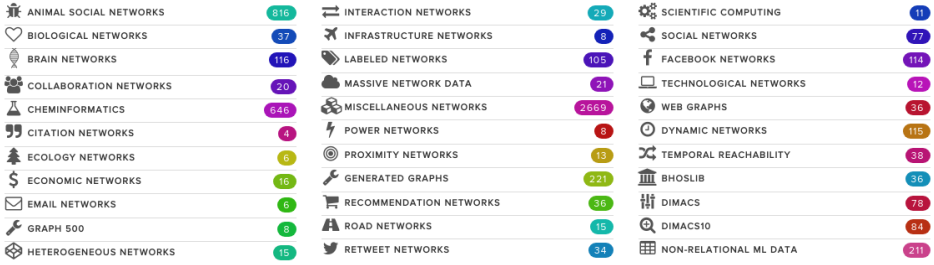
- Ihr müsst **2 Algorithmen** implementieren
- Wir haben **800 Graphen** für euch ausgewählt



Source: *https://networkrepository.com/*

Übungsblatt 0

- Ihr müsst **2 Algorithmen** implementieren
- Wir haben **800 Graphen** für euch ausgewählt



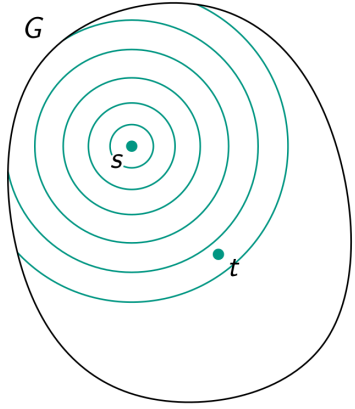
Source: *https://networkrepository.com/*

Euer Ziel

Findet heraus auf welchen Graphen die Algorithmen gut funktionieren und auf welchen nicht

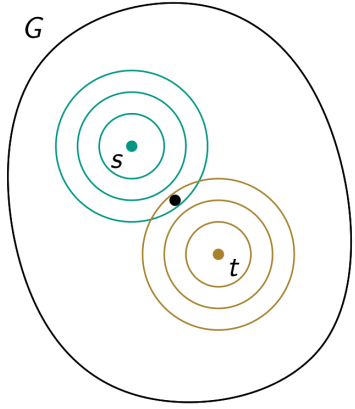
Bidirektionale Breitensuche

- Breitensuche ist einer der einfachsten Algorithmen um kürzeste Wege und ungewichteten Graphen zu finden



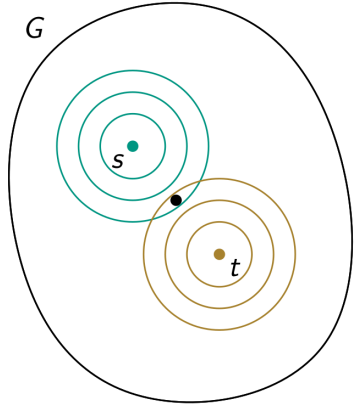
Bidirektionale Breitensuche

- Breitensuche ist einer der einfachsten Algorithmen um kürzeste Wege und ungewichteten Graphen zu finden
- Bidirektionale Breitensuche ist eine der einfachsten Optimierungen
- Dabei wird abwechseln von t und von s aus ein Suchschritt ausgeführt



Bidirektionale Breitensuche

- Breitensuche ist einer der einfachsten Algorithmen um kürzeste Wege und ungewichteten Graphen zu finden
- Bidirektionale Breitensuche ist eine der einfachsten Optimierungen
- Dabei wird abwechseln von t und von s aus ein Suchschritt ausgeführt
- Variante: Schätze die Größe der nächsten Layer und exploriere das kleinere

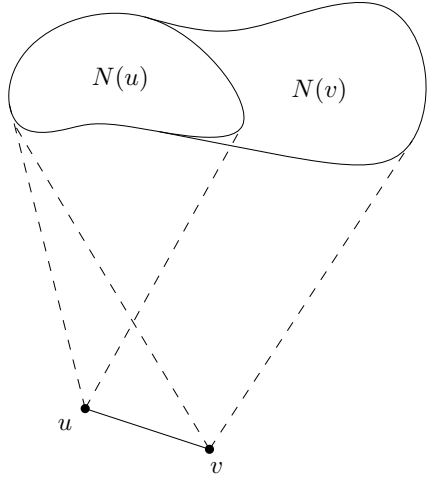


Vertex Cover

- Gegeben einen Graphen $G = (V, E)$
- $X \subseteq V$ ist *Vertex Cover*, falls
  $\forall \{u, v\} \in E : u \in X \vee v \in X$

Vertex Cover

- Gegeben einen Graphen $G = (V, E)$
- $X \subseteq V$ ist *Vertex Cover*, falls $\forall \{u, v\} \in E : u \in X \vee v \in X$
- Für $u, v \in V$: v *dominiert* u, wenn $N[u] \subseteq N[v]$



Vertex Cover

- Gegeben einen Graphen $G = (V, E)$
- $X \subseteq V$ ist *Vertex Cover*, falls $\forall \{u, v\} \in E : u \in X \vee v \in X$
- Für $u, v \in V$: v *dominiert* u, wenn $N[u] \subseteq N[v]$
- **Dominanzregel**: Ein dominierender Knoten kann ins Vertex Cover aufgenommen werden