Algorithmen und Programmierung
Übungsblatt 6
WS 2023/24

UNIVERSITÄT BONN

Dr. Felix Jonathan Boes
Adrian Oeyen, Afeef Neiroukh, Amelie Artmann, Anna Höpfner, Carla Münnich, Chris Geron, Felix Roth, Greta Willing, Louis Hänsch, Luisa Haas, Mathilde Schreck, Mathis Kock

Ausgabe: 13.11.2023
Abgabe: Gitlab mit Präsentation in der Übung

Aufgabe 1 (Mergesort)**.** In Ihrem Abgabe-Repository hat Flavius einen Branch mit dem Namen `AlPro_06` angelegt. Implementieren Sie dort, unter Verwendung der Projektstruktur, Mergesort und demonstrieren Sie Ihre Implementierung.

Um das Projekt zu kompilieren, soll (im Stammverzeichnis des Projekts) folgender Compileraufruf verwendet werden. Alternativ verwenden Sie das Buildsystem cmake.

```
clang++ -std=c++17 -I./include \
        src/hilfsfunktionen.cpp src/mergesort.cpp \
        examples/aufg1.cpp -o build/aufg1
```

Aufgabe 2 (Laufzeiten empirisch bestimmen)**.** In Ihrem Abgabe-Repository hat Flavius einen Branch mit dem Namen `AlPro_06` angelegt. Nutzen Sie die vorhandene Projektstruktur um Ihrer Minsort- und Mersortimplementierung sowie der vorhandenen Schnellsortimplementierung zu vermessen. Erheben Sie genügend Laufzeiten um einen asymptotischen Trend zu erkennen und diskutieren Sie diesen.

Um das Projekt zu kompilieren, soll (im Stammverzeichnis des Projekts) folgender Compileraufruf verwendet werden. Alternativ verwenden Sie das Buildsystem cmake.

```
clang++ -std=c++17 -I./include \
        src/hilfsfunktionen.cpp src/schnellsort.cpp src/zeitmesser.cpp \
        examples/aufg2.cpp -o build/aufg2
```

Bonus: Zeigen Sie, dass Minsort Laufzeitaufwand $\mathcal{O}(n^2)$ und Speicheraufwand $\mathcal{O}(1)$ hat.

Aufgabe 3 (Gerichtete Graphen)**.** In Ihrem Abgabe-Repository hat Flavius einen Branch mit dem Namen `AlPro_06` angelegt. Implementieren Sie dort, unter Verwendung der Projektstruktur, die Klasse `SimplerGraph` und demonstrieren Sie Ihre Implementierung. Diese Klasse soll gerichtete Graphen realisieren und folgende Bedingungen berücksichtigen.

- Die Anzahl der Knoten soll zum Instanziierungszeitpunkt festgelegt werden und zur Laufzeut nicht veränderbar sein.
- Die Knoten sollen durch die Indizes 0, 1, ... benannt werden.
- Alle existierenden Knoten und Kanten sollen ausgedruckt werden können.
- Durch die Angabe von Knotenpaaren soll es möglich sein, Kanten hinzuzufügen oder zu entfernen.

Zur Implementierung sollen Sie ein zweidimensionales, quadratisches `bool`-Array verwenden. Dabei wird der Eintrag `(i,j)` genau dann auf **`true`** gesetzt, wenn die Kante $i \to j$ im Graph existiert.

Um das Projekt zu kompilieren, soll (im Stammverzeichnis des Projekts) folgender Compileraufruf verwendet werden. Alternativ verwenden Sie das Buildsystem cmake.

```
clang++ -std=c++17 -I./include \
        src/simpler_graph.cpp \
        examples/aufg3.cpp -o build/aufg3
```

Bonus: Diskutieren Sie, wie Sie die Implementierung ändern würden, um das Löschen von Knoten zuzulassen (ohne bereits vorhandene Knoten neu zu indizieren).