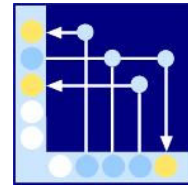




Hochschule Aalen

*Fakultät Elektronik und Informatik
Studienbereich Informatik*



Algorithmen und Datenstrukturen 2

Vorlesung im Wintersemester 2025/2026

Prof. Dr. habil. Christian Heinlein

5. Praktikumsaufgabe (15. – 29. Januar 2026)

Aufgabe 5: Algorithmen auf gewichteten Graphen

Implementieren Sie folgende Graphalgorithmen in C++, indem Sie den Code, der in der Datei `wgraph.h` auf der Vorlesungswebseite vorgegeben ist, entsprechend ergänzen:

- Bestimmung minimaler Gerüste nach Prim
- Bestimmung kürzester Wege nach Bellman-Ford und Dijkstra

Ein gewichteter Graph kann hierbei ein Objekt eines prinzipiell beliebigen Typs sein, der, neben den bei Aufgabe 4 genannten Elementfunktionen `vertices` und `successors`, eine weitere Elementfunktion `weight` besitzt, sodass `g.weight(u, v)` für einen Knoten `u` des Graphen `g` und einen Nachfolger `v` von `u` das Gewicht der Kante von `u` nach `v` als `double`-Wert liefert. (Wenn `u` kein Knoten des Graphen `g` ist oder `v` kein Nachfolger von `u` ist, darf das Verhalten von `g.weight(u, v)` undefiniert sein.)

Die vorgegebene Klasse `WeightedGraph` erfüllt diese Anforderungen und kann wie folgt zur Erzeugung gewichteter Testgraphen verwendet werden: Für einen prinzipiell beliebigen Knotentyp `V` kann ein Objekt des Typs `WeightedGraph<V>` mit der um Kantengewichte erweiterten Adjazenzlistendarstellung eines Graphen initialisiert werden, in der die „inneren“ Initialisiererlisten anstelle von Nachfolgern wiederum Paare in geschweiften Klammern enthalten, die jeweils aus einem Nachfolger und dem zugehörigen Kantengewicht mit Typ `double` bestehen, zum Beispiel:

```
WeightedGraph<string> wg({
    { "A", { { "B", 2 }, { "C", 3 } } },
    { "B", { } },
    { "C", { { "C", 4 } } }
});
```

Für diesen Graphen `wg` liefert:

- `wg.vertices()` einen Container mit den Elementen "A", "B" und "C",
- `wg.successors("A")` einen Container mit den Elementen "B" und "C",
- `wg.successors("B")` einen leeren Container,
- `wg.successors("C")` einen Container mit dem Element "C",
- `wg.weight("A", "B")` den `double`-Wert 2,
- `wg.weight("A", "C")` den `double`-Wert 3,
- `wg.weight("C", "C")` den `double`-Wert 4.

Alle Algorithmen müssen aber auch für andere Typen G funktionieren, sofern sie die oben genannten Anforderungen erfüllen.

Wie bei Aufgabe 4, wird jeder Algorithmus durch eine Funktionsschablone (function template) mit Typparametern V (Typ der Knoten) und G (Typ des Graphen) implementiert, die als ersten Parameter einen Graphen g mit Typ G und als zweiten Parameter einen Startknoten s mit Typ V erhält. Der letzte Parameter, der per Referenz übergeben wird, ist immer eine geeignete Datenstruktur, in der die Ergebnisse des Algorithmus gespeichert werden.

Für Algorithmen, die eine Vorrangwarteschlange benötigen, verwenden Sie die auf der Vorlesungswebseite in der Datei `prioqueue.h` bereitgestellte Klasse `PrioQueue` mit derselben Funktionalität wie `BinHeap` aus Aufgabe 3! (Die Bibliotheksklasse `std::priority_queue` ist nicht geeignet, weil sie keine Möglichkeit bietet, die Priorität eines Eintrags nachträglich zu verändern.)

Testen Sie Ihre Implementierung mit unterschiedlichen Graphen und unterschiedlichen Startknoten sorgfältig und ausführlich!

Hierfür kann das bereits für Aufgabe 4 auf der Vorlesungswebseite bereitgestellte Testprogramm `graphtest.cxx` verwendet werden, wenn beim Übersetzen das Makro `WGRAPH` definiert wird.

Beim Algorithmus von Prim ist darauf zu achten, dass der jeweilige Testgraph ungerichtet sein muss, d. h. er muss zu jeder Kante auch die entgegengesetzte Kante mit dem gleichen Gewicht enthalten.

Um automatisierte Tests der Implementierungen zu ermöglichen, dürfen die vorgegebenen Klassennamen und die Signaturen von Funktionen nicht verändert werden, und es dürfen keine Diagnoseausgaben produziert werden.

Abgesehen von den in `graph.h` und `wgraph.h` bereits verwendeten Bibliotheksklassen (`numeric_limits`, `list`, `map` und `pair`) dürfen keine Bestandteile der C++-Standardbibliothek oder anderer Bibliotheken verwendet werden.

Abzugeben ist die durch Ihren Code erweiterte Datei `wgraph.h`.

Die E-Mail mit der Abgabe muss als Betreff `Algo2 Gruppe NN` mit zweistelliger Gruppennummer `NN` (z. B. 05 oder 12) und als Anhang die abzugebende Datei haben. Der Nachrichtentext muss für jedes Gruppenmitglied aus einer Zeile der Art `Vorname,Nachname,Matrikelnummer` bestehen.