Einführung in die Programmierung mit C++

Übungsblatt 3

Graphen mit Zeigern und Referenzen

Sebastian Christodoulou Alexander Fleming Uwe Naumann

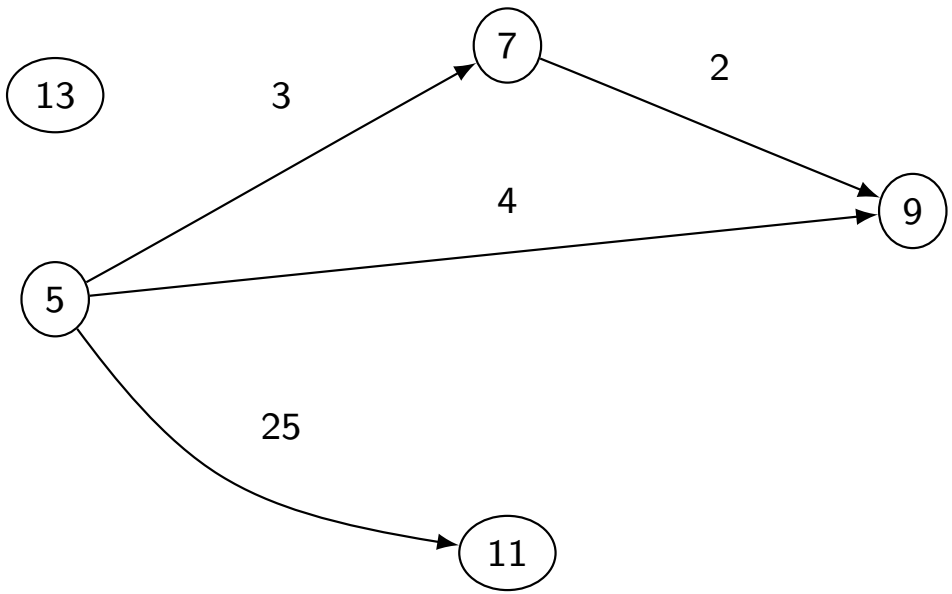
Informatik 12:
Software and Tools for Computational Engineering (STCE)

RWTH Aachen

Übung 3
Graphen und Speicherformat

Ein Graph $G(V, E)$ besteht aus Knoten ($\circ$) und Kanten ($\rightarrow$). Die Knoten V sind durch Zahlen identifiziert und haben auch eine Beschriftung. Der untere Graph aus `my_graph.txt` (links) hat 5 Knoten mit Beschriftungen $[5, 7, 9, 11, 13]$ und 4 Kanten.

```
5
5 7 9 11 13
4
0 1 3
0 2 4
1 2 2
0 3 25
```



Der obere Graph hat 4 Kanten. Kanten beschreiben welcher Knoten über welchen anderen erreichbar ist. Die Kanten sind hier durch die Indizes der Knotenliste gegeben. Zudem haben sie einen Wert.

- $0 \rightarrow 1$ mit Wert 3
- $0 \rightarrow 2$ mit Wert 4
- $1 \rightarrow 2$ mit Wert 2
- $0 \rightarrow 3$ mit Wert 25

Auf den letzten Knoten (mit Beschriftung 13) zeigt keine Kante.

```
5
5 7 9 11 13
4
0 1 3
0 2 4
1 2 2
0 3 25
```

Die Datei `my_graph.txt` hat zwei Abschnitte. Der Erste beschreibt die Knoten, und der Zweite beschreibt die Kanten.

Knotenabschnitt

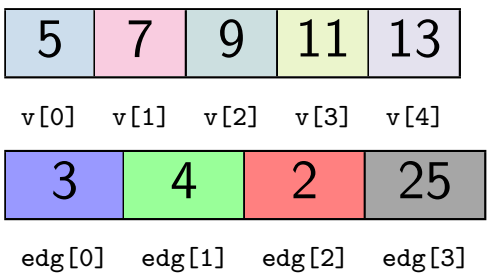
- Die *erste* Zahl (N_v) entspricht der Anzahl der Knoten im Graph.
- Die folgenden N_v Zahlen sind die Beschriftungen der Knotenindizes $[0, \ldots, N_v - 1]$.

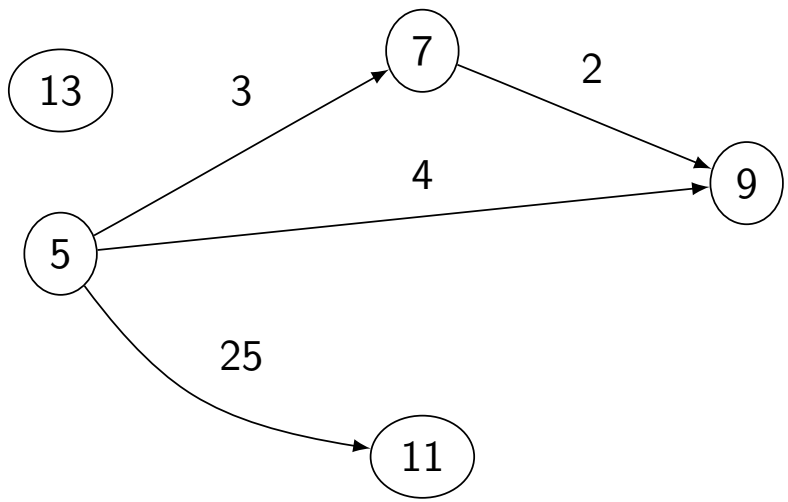
Kantenabschnitt

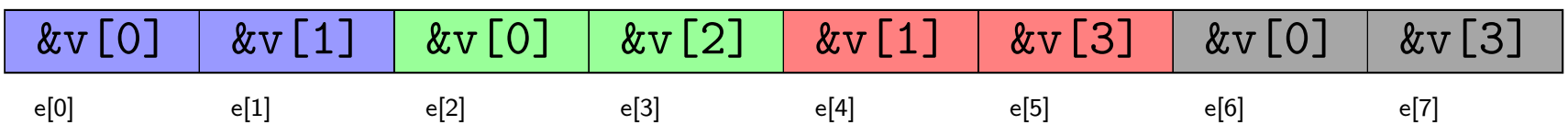
- Die *nächste* Zahl (N_e) entspricht der Anzahl der Kanten im Graph.
- Die folgenden N_e Dreiergruppen sind die Kanten:
  - Jede Dreiergruppe (a, b, c) beschreibt eine Kante zwischen Knotenindizes $a \rightarrow b$ mit Wert c.

Wir kodieren eine Graph im obigen Speicherformat in drei Variablen:

- Eine `std::vector<int> v` für die Knoten-Beschriftungen.
- Eine `std::vector<int> edg` für die Kanten-Beschriftungen.
- Eine `std::vector<int *> e` für die Kanten.







1. Öffnen Sie `my_graph.txt` mittels `std::ifstream`, und lesen Sie deren Knoten und Kanten wie beschrieben in
   - einen `std::vector<int> v`
   - einen `std::vector<int> edg` und
   - einen `std::vector<int *> e`

   von passender Größe aus.
2. Geben Sie alle Kanten mit deren Werten auf der Kommandozeile durch einen `for`-Schleife aus.
3. Ergänzen Sie die Funktion `reverse_edges`, welche die Kanten-Liste als In-Out Parameter `std::vector<int>& e` annimmt.
   - Diese tauscht jeden $2i$-ten Eintrag von `e` mit dem $2i + 1$-ten Eintrag.
4. Rufen Sie die Funktion `graph_to_dot` auf `v`, `edg`, und `e` auf. Diese erstellt, wie letzte Woche, eine Datei `graph.dot`.
5. Der Befehl

   `$ dot -Tpdf graph.dot > graph.pdf`

   erzeugt ein pdf, das den Graph.
6. Öffnen Sie `graph.pdf`. Vergleichen Sie das Ergebnis mit dem Graph auf Seite 4.

Abgaben

- main.cpp
- graph.pdf