

# Präsenzblatt 0

Dieses Blatt dient lediglich der persönlichen Vorbereitung. Es wird nicht abgegeben und geht nicht in die Bewertung ein. Die Aufgaben und ihre Lösungen werden in den kleinen Übungen (14.11.-18.11.2022) besprochen.

## Präsenzaufgabe 1 (Begriffe der Graphentheorie):

Bestimme für den in Abbildung 1 dargestellten Graphen  $G = (V, E)$  das Folgende:

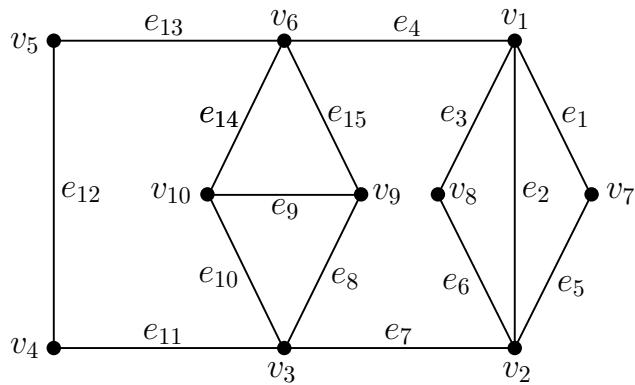


Abbildung 1: Der Graph  $G$

- a) Alle zu  $v_1$  adjazenten Knoten,
- b) alle zu  $v_1$  inzidenten Kanten,
- c) eine Kantenfolge von  $v_1$  nach  $v_4$ , die kein Weg ist,
- d) einen Weg von  $v_1$  nach  $v_4$ , der kein Pfad ist,
- e) einen Pfad von  $v_1$  nach  $v_4$ ,
- f) einen Eulerweg (oder begründe, dass dieser nicht existiert),
- g) einen Hamiltonpfad (oder begründe, dass dieser nicht existiert),
- h) einen Hamiltonkreis (oder begründe, dass dieser nicht existiert).

## Präsenzaufgabe 2 (Fibonacci-Zahlen):

Die Fibonacci-Zahlen sind rekursiv definiert. So berechnet sich eine bestimmte Fibonacci-Zahl aus der Summe der beiden vorherigen Fibonacci-Zahlen. Konkret:  $F(n) := F(n - 1) + F(n - 2)$ , wobei  $F(0) = 0$  und  $F(1) = 1$ . Wir wollen nun zwei Algorithmen (siehe Algorithmus 1) testen, die  $F(n)$  berechnen.

- a) Berechne  $F(5)$  mit FIBONACCI<sub>REK</sub> und gib dabei alle Aufrufe der Funktion an. Wie oft wird eine Summe berechnet? Hat FIBONACCI<sub>REK</sub> alle Eigenschaften eines Algorithmus?
- b) Berechne  $F(5)$  mit FIBONACCI und gib dabei die Werte aller  $f(i)$  aus. Wie viele Summen wurden berechnet? Hat FIBONACCI alle Eigenschaften eines Algorithmus?

<pre> 1: <b>function</b> FIBONACCI<sub>REK</sub>(<i>n</i>) 2:   <b>if</b> <i>n</i> ≤ 1 <b>then</b> 3:     <b>return</b> <i>n</i> 4:   <i>a</i> := FIBONACCI<sub>REK</sub>(<i>n</i> - 1) 5:   <i>b</i> := FIBONACCI<sub>REK</sub>(<i>n</i> - 2) 6:   <b>return</b> <i>a</i> + <i>b</i> </pre>	<pre> 1: <b>function</b> FIBONACCI(<i>n</i>) 2:   <i>f</i>(0) := 0 3:   <i>f</i>(1) := 1 4:   <b>for</b> <i>i</i> <b>in</b> 2, ..., <i>n</i> <b>do</b> 5:     <i>f</i>(<i>i</i>) := <i>f</i>(<i>i</i> - 1) + <i>f</i>(<i>i</i> - 2) 6:   <b>return</b> <i>f</i>(<i>n</i>) </pre>
--	--

**Algorithmus 1:** Zwei Algorithmen für  $F(n)$ : Rekursiv (links) und „mit Gedächtnis“ (rechts)

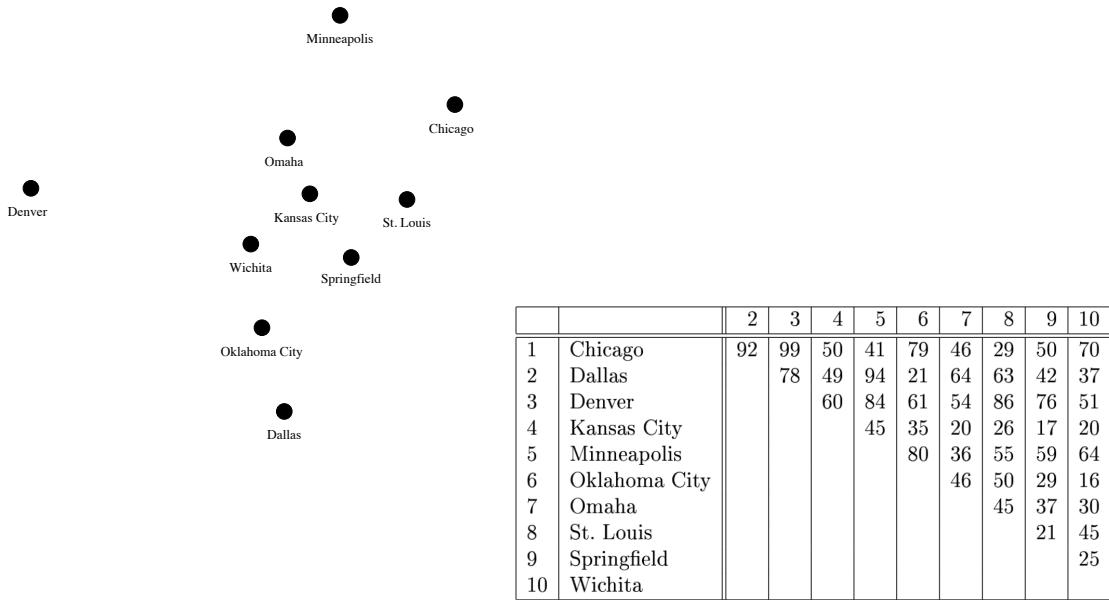
### Präsenzaufgabe 3 (kürzeste Pfade):

Sei  $G = (V, E)$  ein einfacher Graph. Für einen Pfad  $P = (v_1, \dots, v_i)$  ist die Länge von  $P$  die Anzahl der Kanten auf diesem Pfad. Sei nun  $P$  ein kürzester Pfad von Knoten  $v_1$  nach  $v_i$ .

Zeige oder widerlege: Jeder Teilstieg  $P' = (v_k, \dots, v_\ell)$  von  $P$  mit  $1 \leq k \leq \ell \leq i$  ist ein kürzester Pfad zwischen  $v_k$  und  $v_\ell$ .

### Präsenzaufgabe 4 (Rundreise):

Abbildung 2 zeigt Städte der USA und eine Distanzmatrix. Die Zahlen geben die Entfernung pro 10 Meilen an. Beispielsweise ist die Distanz von Chicago nach Dallas 920 Meilen. Ein Handelsreisender möchte nun alle Städte ablaufen und dabei möglichst wenig Meilen zurücklegen.



**Abbildung 2:** Städte der USA (links) und eine Distanzmatrix (rechts, je 10 Meilen)

- Wie viele Kanten braucht man für eine Tour? (Hinweis: Diese Frage bezieht sich nur auf die Anzahl, sie ist unabhängig von den zurückgelegten Distanzen.)
- Finde eine möglichst kurze Tour, d.h. eine für die die Summe der Entfernungen der ausgewählten Kanten möglichst klein ist.
- Wie gut ist Deine Tour? Gib eine Mindestdistanz für *jede mögliche* Städterundreise an und vergleiche diese mit der Länge der von Dir gefundenen Tour.