

# *Kapitel 2: Graphen*

*Algorithmen und Datenstrukturen  
WS 2021/22*

**Prof. Dr. Sándor Fekete**

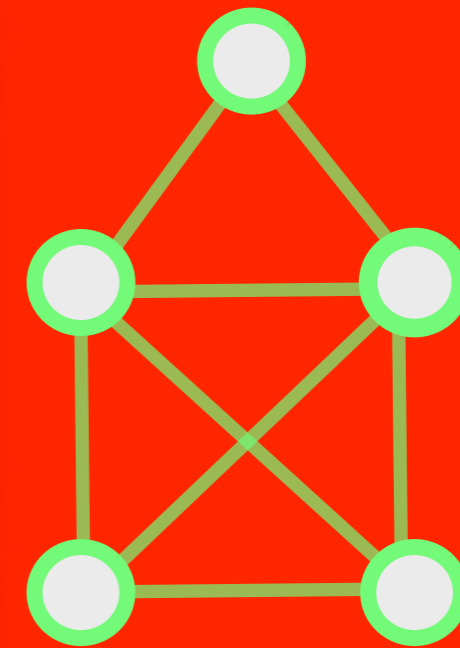
# Gestatten, Graph!



# Gestatten, Graph!

Formal:  
DEFINITION 2.1

- (1) (i) Ein ungerichteter Graph  $G$  ist ein Tripel  
 $(V, E, \Psi)$ , für das
- (a)  $V$  und  $E$  endliche Mengen sind
- (b)  $\Psi: E \rightarrow \{X \subseteq V \mid 1 \leq |X| \leq 2\}$   
 $\uparrow \uparrow$   
Kardinalität von  $X$



Graph

**Graph: Ein Gebilde aus Knoten (Haltestellen)  
und Kanten (Verbindungen)**

**Definition 2.1** (Ungerichteter Graph).

(1)(i) Ein ungerichteter Graph  $G$  ist ein Tripel  $(V, E, \Psi)$ , für das

(a)  $V$  und  $E$  endlichen Mengen sind und

(b)  $\Psi$  eine Funktion mit

$$\Psi : E \rightarrow \{X \subseteq V \mid 1 \leq |X| \leq 2\} \quad (2.1)$$

ist. D. h. jede Kante enthält einen Knoten (Schleife) oder zwei.

↑ ↑  
Kardinalität von  $X$

## Graph

**Graph: Ein Gebilde aus Knoten (Haltestellen)  
und Kanten (Verbindungen)**

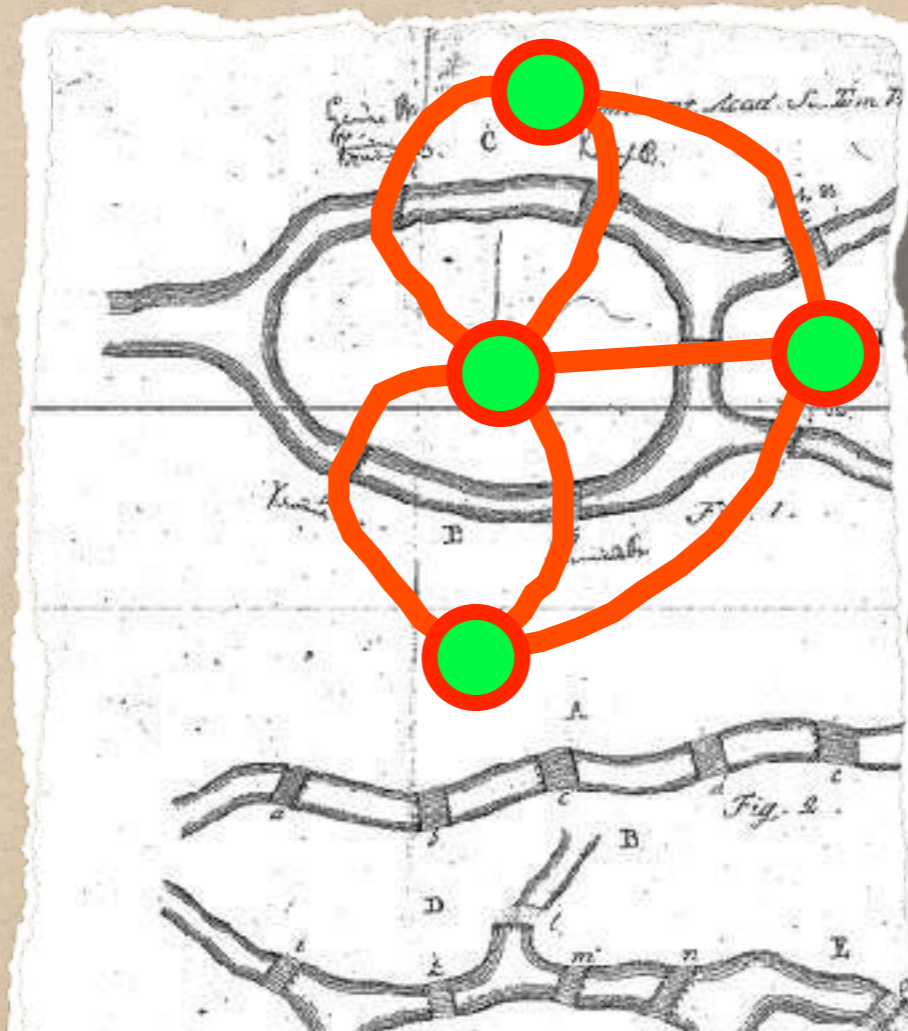
## 2.3 Eulerwege

### Problem 2.3 (Eulerweg)

**Gegeben:** Ein Graph  $G=(V,E)$

**Gesucht:** Ein Eulerweg  $W$  in  $G$  - oder ein Argument, dass kein Eulerweg existiert

## 2.1 Historie



- Alle Knoten sind ungerade?!
- Man müsste an allen anfangen oder aufhören!
- Das geht nicht an einem Stück!

Euler: (1) Das gilt für jede beliebige Instanz: Mit mehr als zwei ungeraden Knoten gibt es keinen solchen Weg.

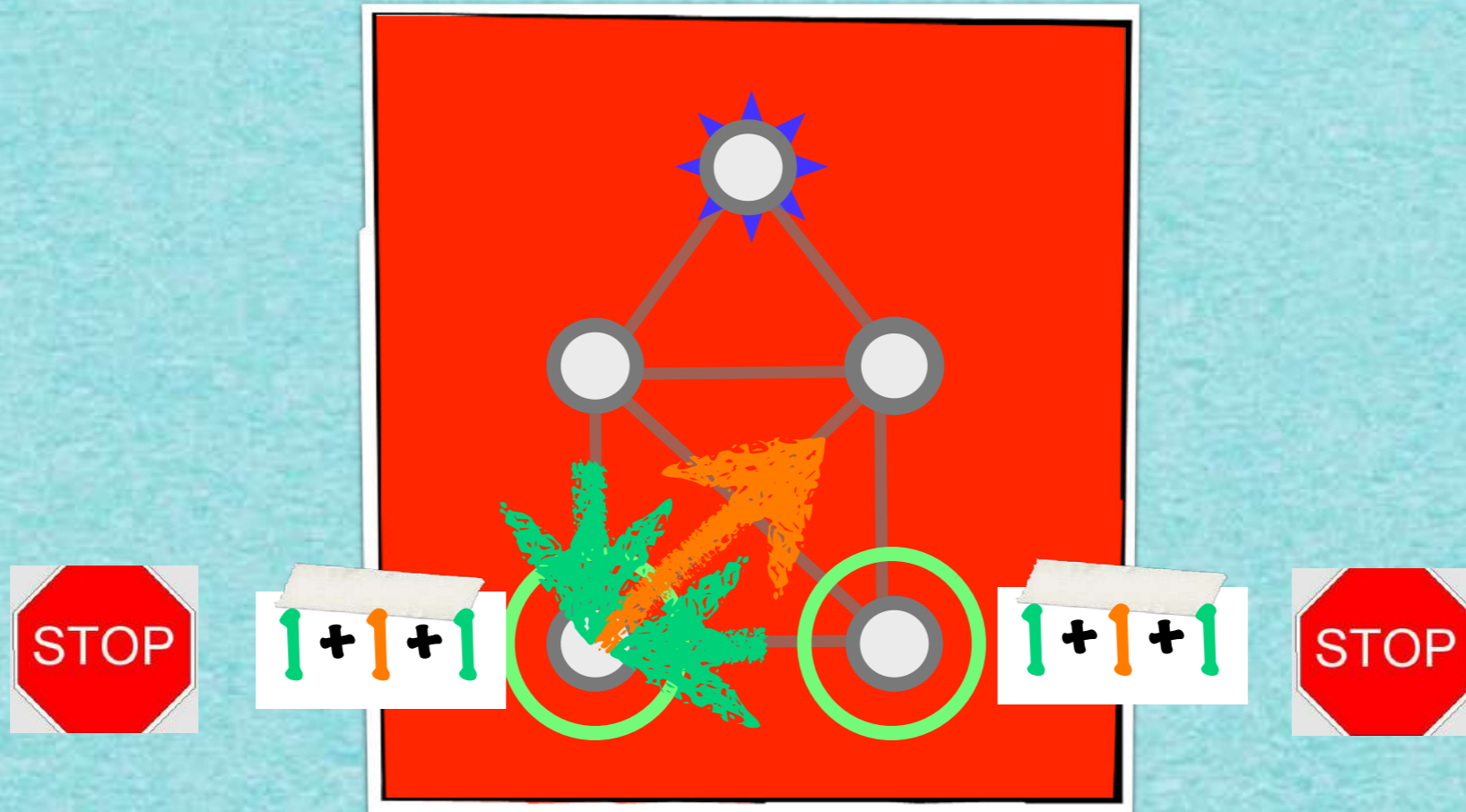
(2) Man kann auch charakterisieren, unter welchen Bedingungen es einen Weg tatsächlich gibt.

## 2.3 Eulerwege

### Satz 2.4 (Euler)

- (1) Ein Graph  $G=(V,E)$  kann nur dann einen Eulerweg haben, wenn es höchstens zwei Knoten mit ungeradem Grad gibt.
- (2) Ein Graph  $G=(V,E)$  kann nur dann einen geschlossenen Eulerweg haben, wenn alle Knoten geraden Grad haben.

# Das Haus des Nikolaus



*Beweis.* Seien  $G = (V, E)$  ein Graph und  $v \in V$  ein Knoten mit ungeradem Grad  $\delta(v)$ . Dann kann die Zahl der in einem Eulerweg  $W$  zu  $v$  hinführenden Kanten nicht gleich der von  $v$  wegführenden Kanten sein. Also muss  $W$  in  $v$  beginnen oder enden. Damit:

- (1) Es gibt in einem Eulerweg nur einen Start- und Endknoten.
- (2) Bei einem geschlossenen Eulerweg gibt es für den Start- und Endknoten  $w$  gleich viele hin- und wegführende Kanten. Also ist auch  $\delta(w)$  gerade.  $\square$



## 2.3 Eulerwege

### Fragen:

- (I) Was ist mit Graphen, in denen nur ein Knoten ungeraden Grad hat?
- (II) Die Bedingungen oben sind *notwendig*, d.h. sie müssen auf jeden Fall erfüllt werden, wenn es eine Chance auf einen Eulerweg geben soll. Sind sie auch *hinreichend*, d.h. gibt es bei Erfüllung auch wirklich einen Eulerweg?
- (III) Wie findet man einen Eulerweg?
- (III') Wie sieht ein Algorithmus dafür aus?

## 2.3 Eulerwege

Zu (I):

### Satz 2.5 (Handshake-Lemma)

Für jeden einfachen Graphen ist die Zahl der Knoten mit ungeradem Grad eine gerade Zahl.

## 2.3 Eulerwege

**Zu (II):**

**Gegeben ein Graph  $G$**

- mit nur zwei ungeraden Knoten. Hat er einen Eulerweg?
- mit nur geraden Knoten. Hat er eine Eulertour?

**Nein!**



**Der Graph muss zusammenhängend sein!**

## 2.3 Eulerwege

Zu (II):

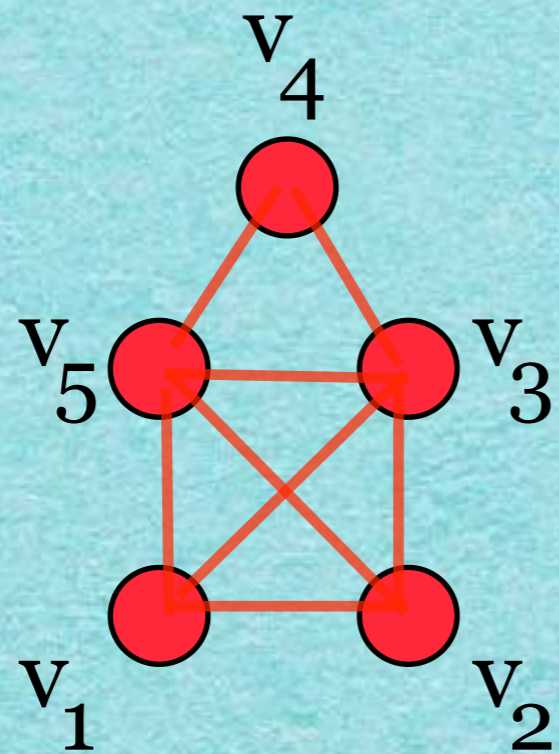
Gegeben ein *zusammenhängender* Graph  $G$

- mit nur zwei ungeraden Knoten. Hat er einen Eulerweg?
- mit nur geraden Knoten. Hat er eine Eulertour?

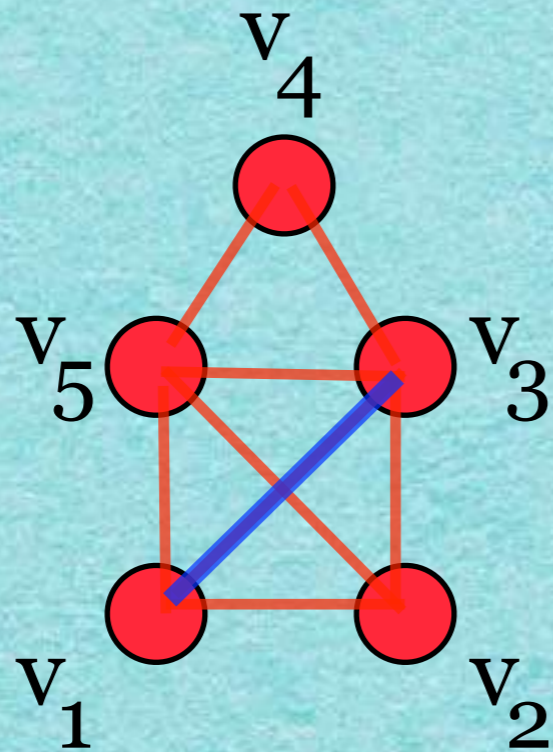
Grundtechnik:

Um zu sehen, wo man etwas richtig machen muss, überlegt man sich, wo man etwas falsch machen kann!

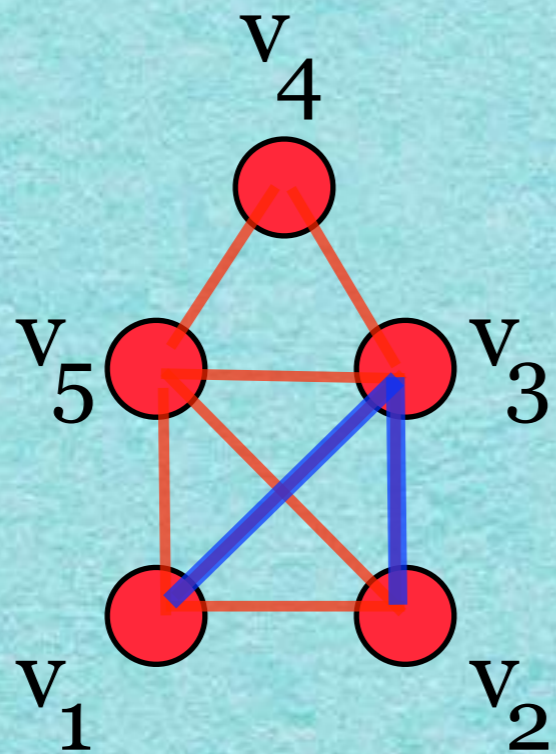
## 2.3 Eulerwege



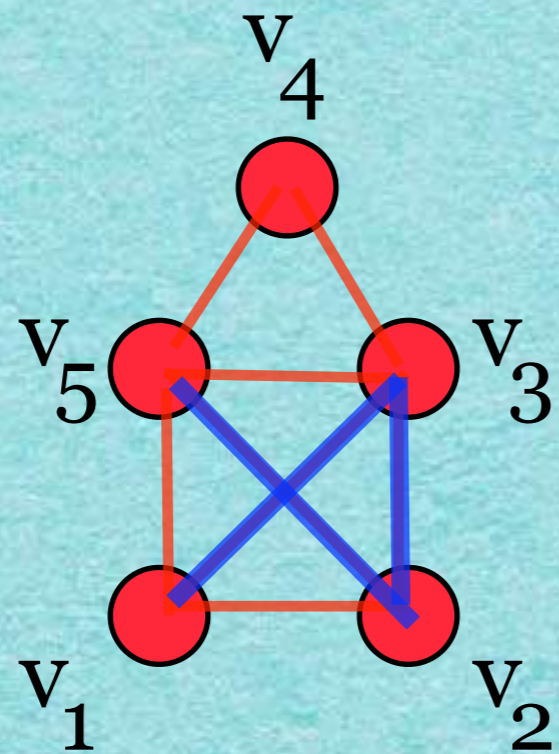
## 2.3 Eulerwege



## 2.3 Eulerwege

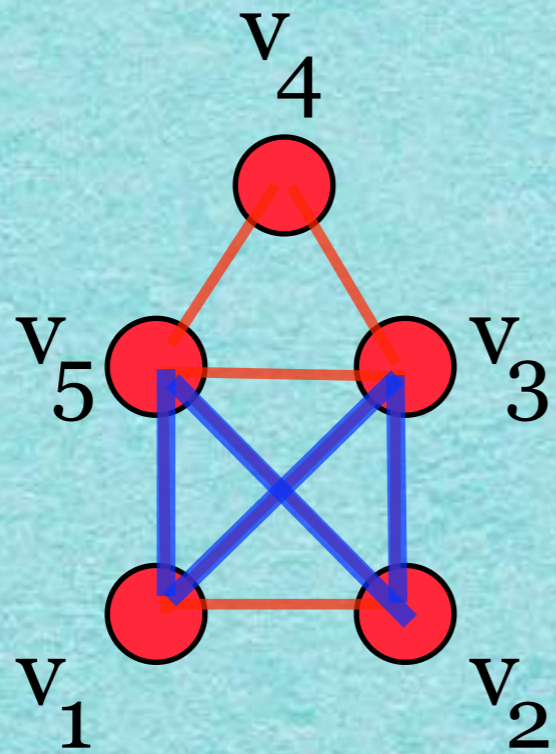


## 2.3 Eulerwege

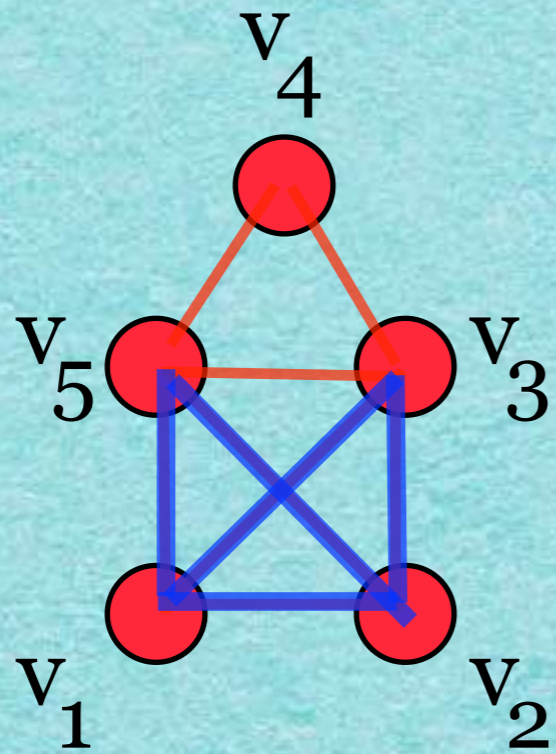




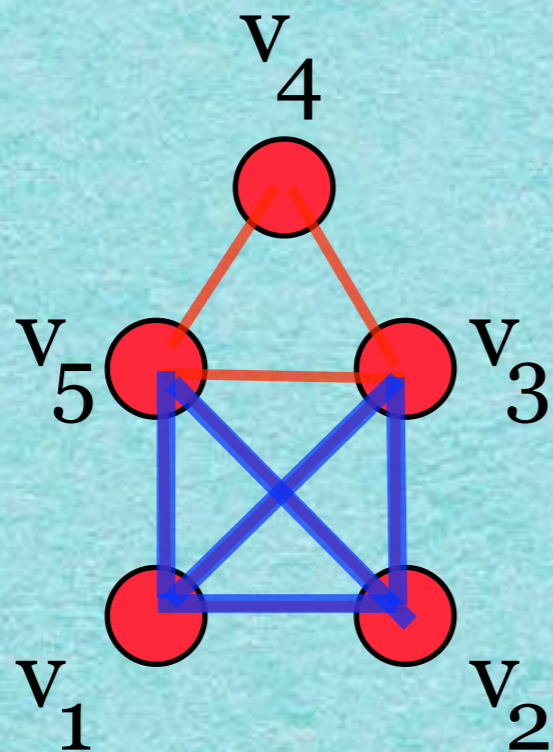
## 2.3 Eulerwege



## 2.3 Eulerwege



## 2.3 Eulerwege

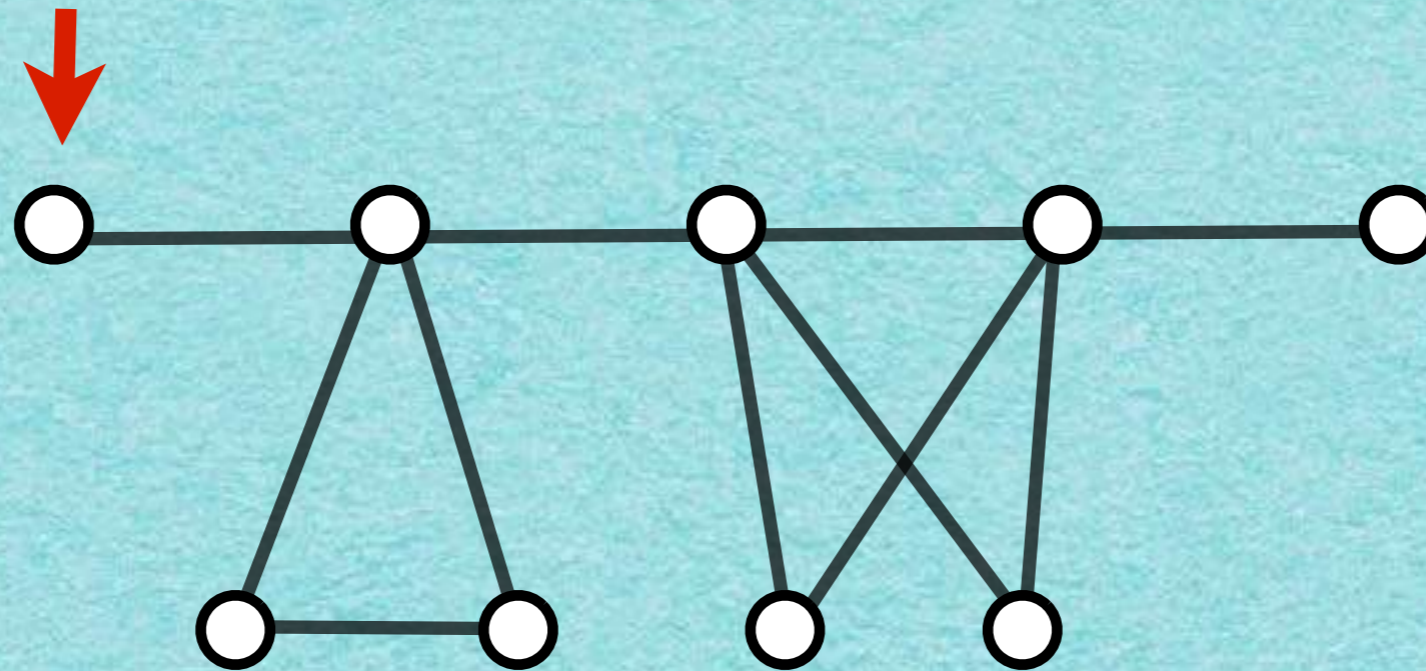


**Wir hinterlassen Kanten?!**

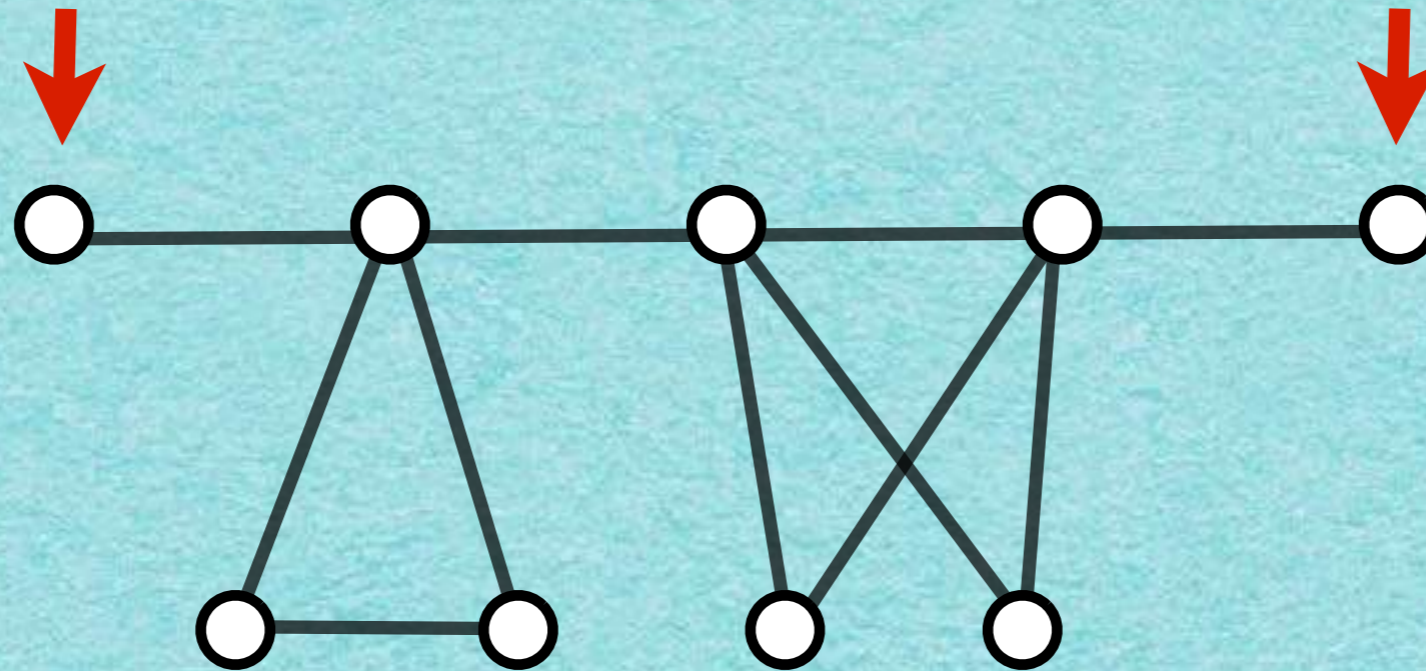
# “Vorlesung”



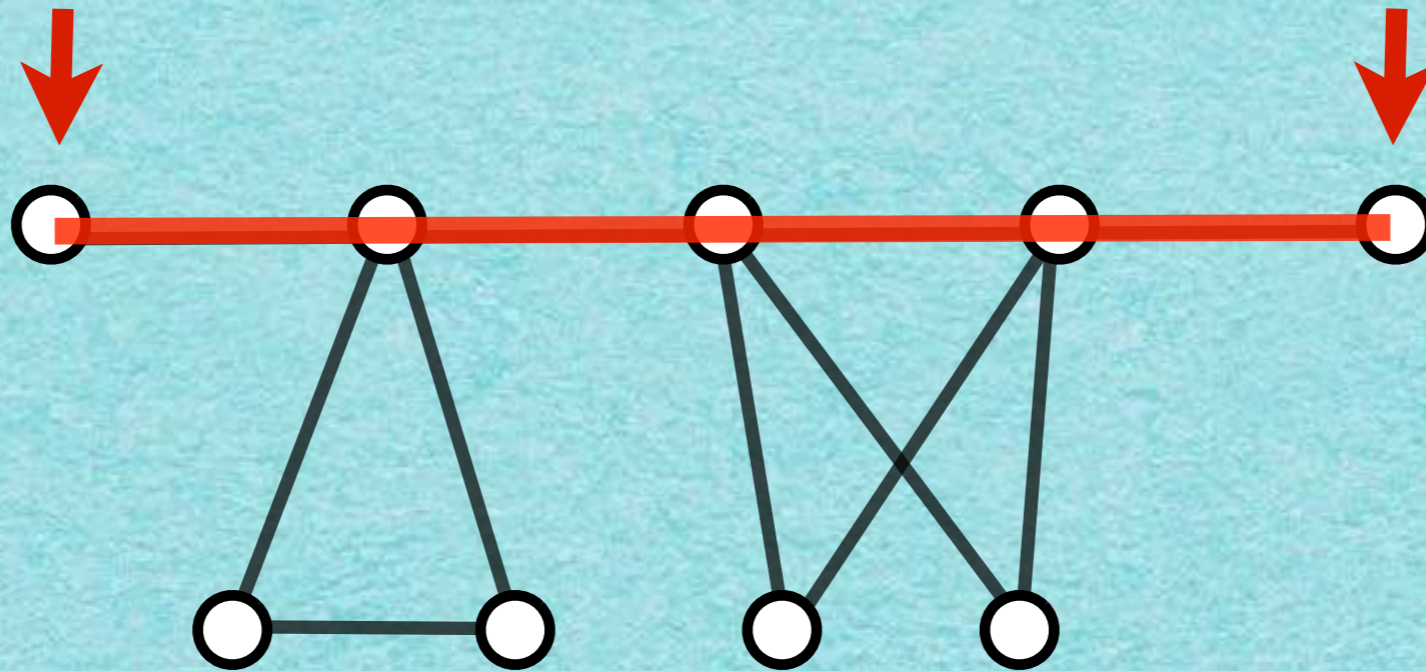
# Wegebau



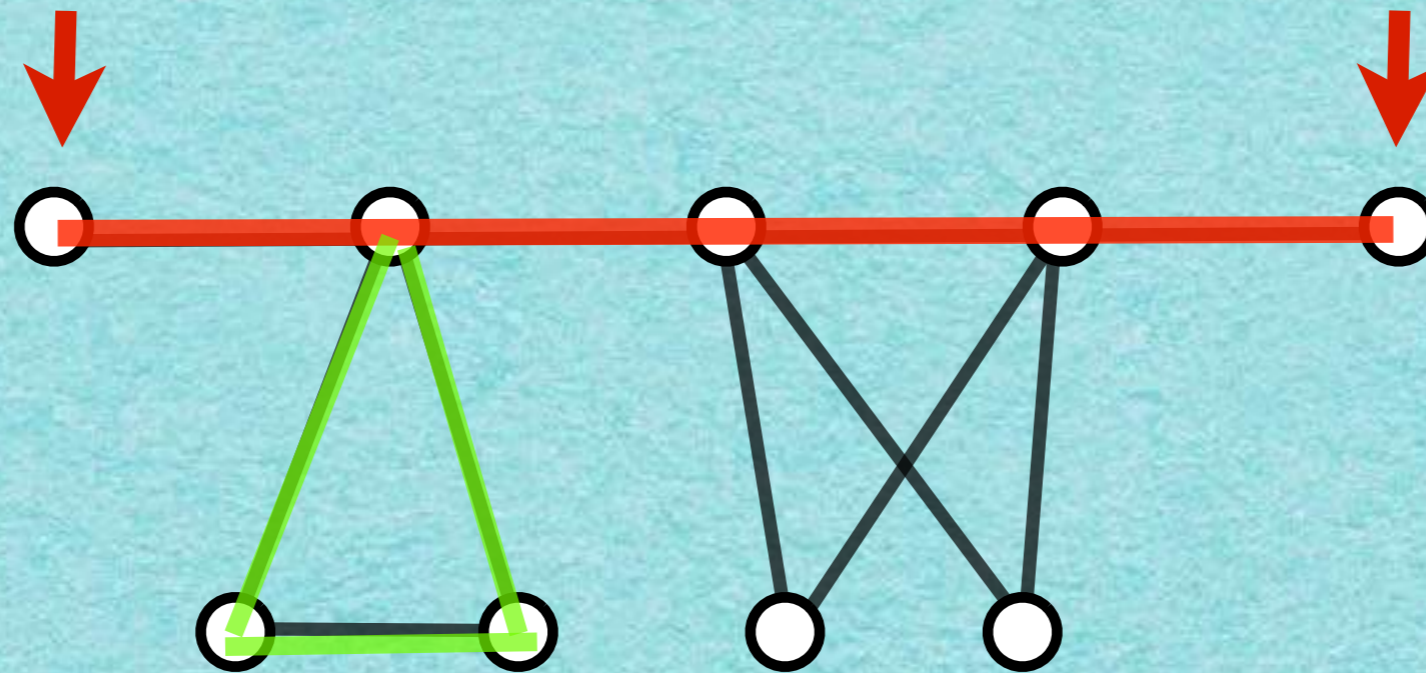
# Wegebau



# Wegebau

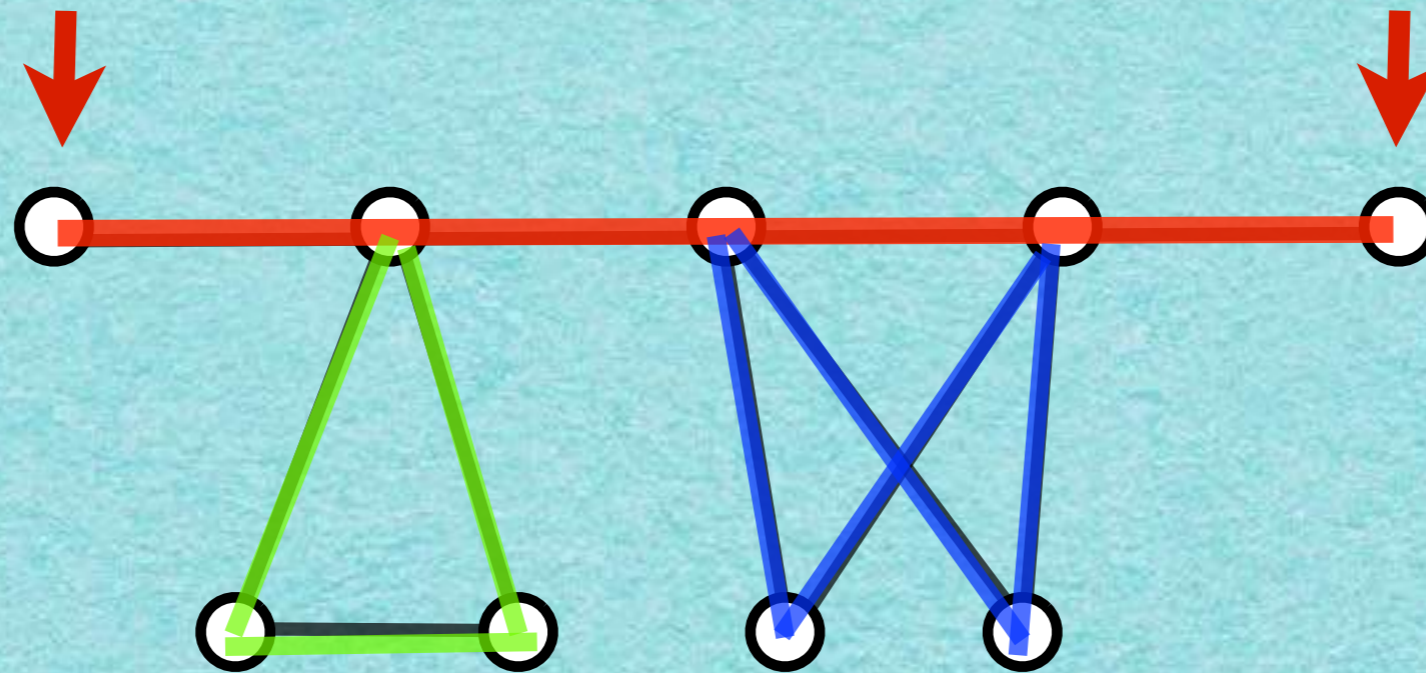


# Wegebau

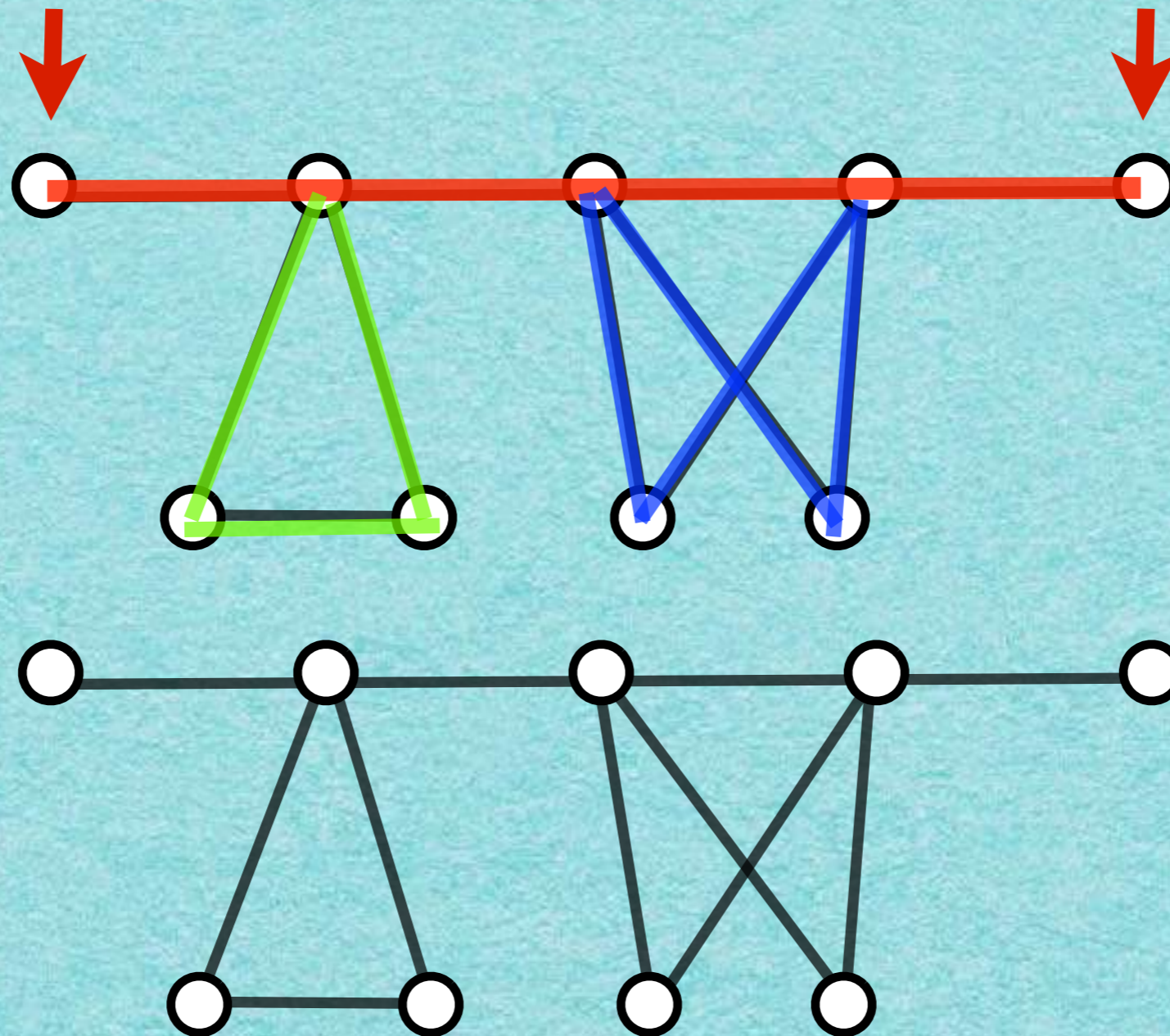




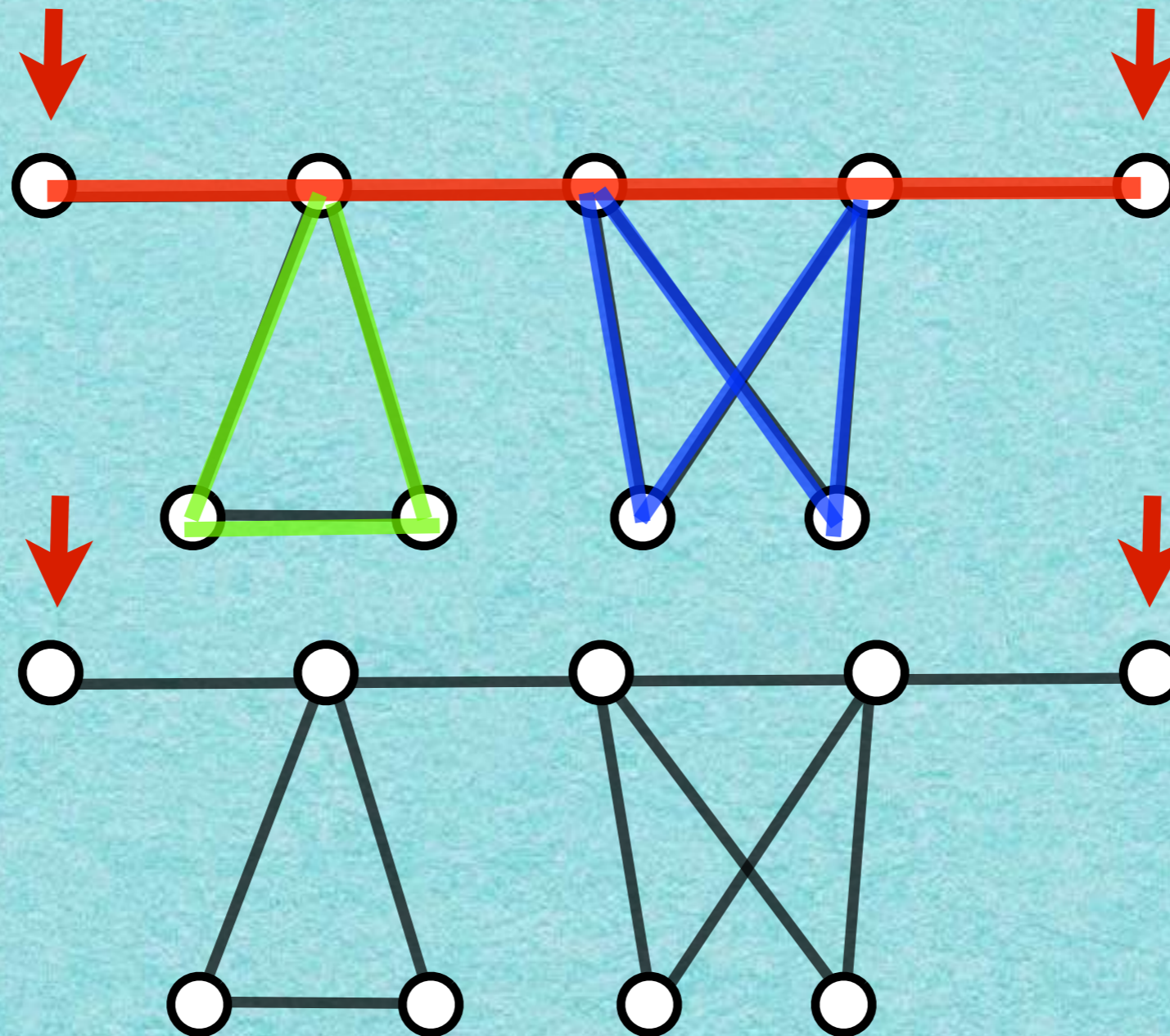
# Wegebau



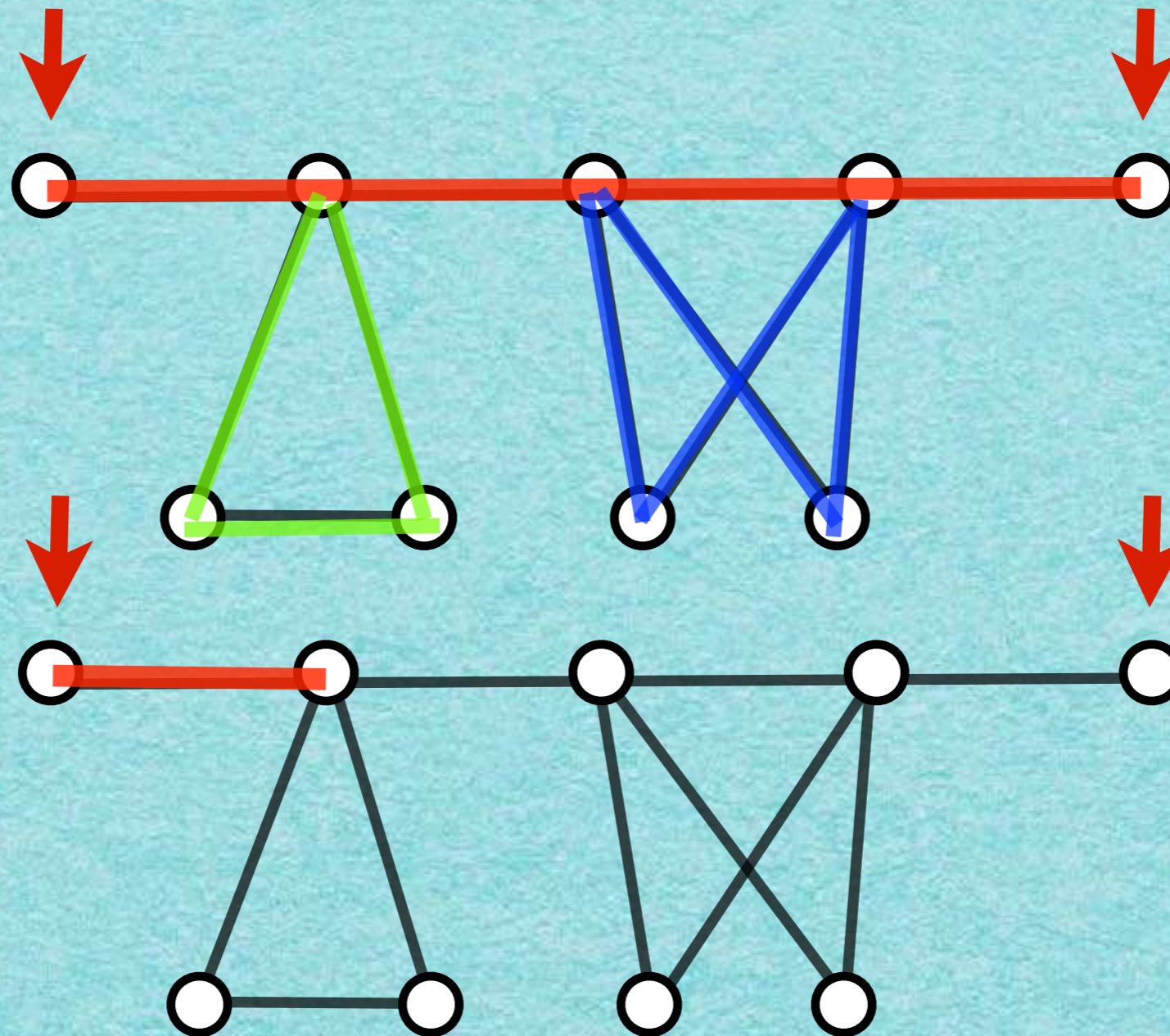
# Wegebau



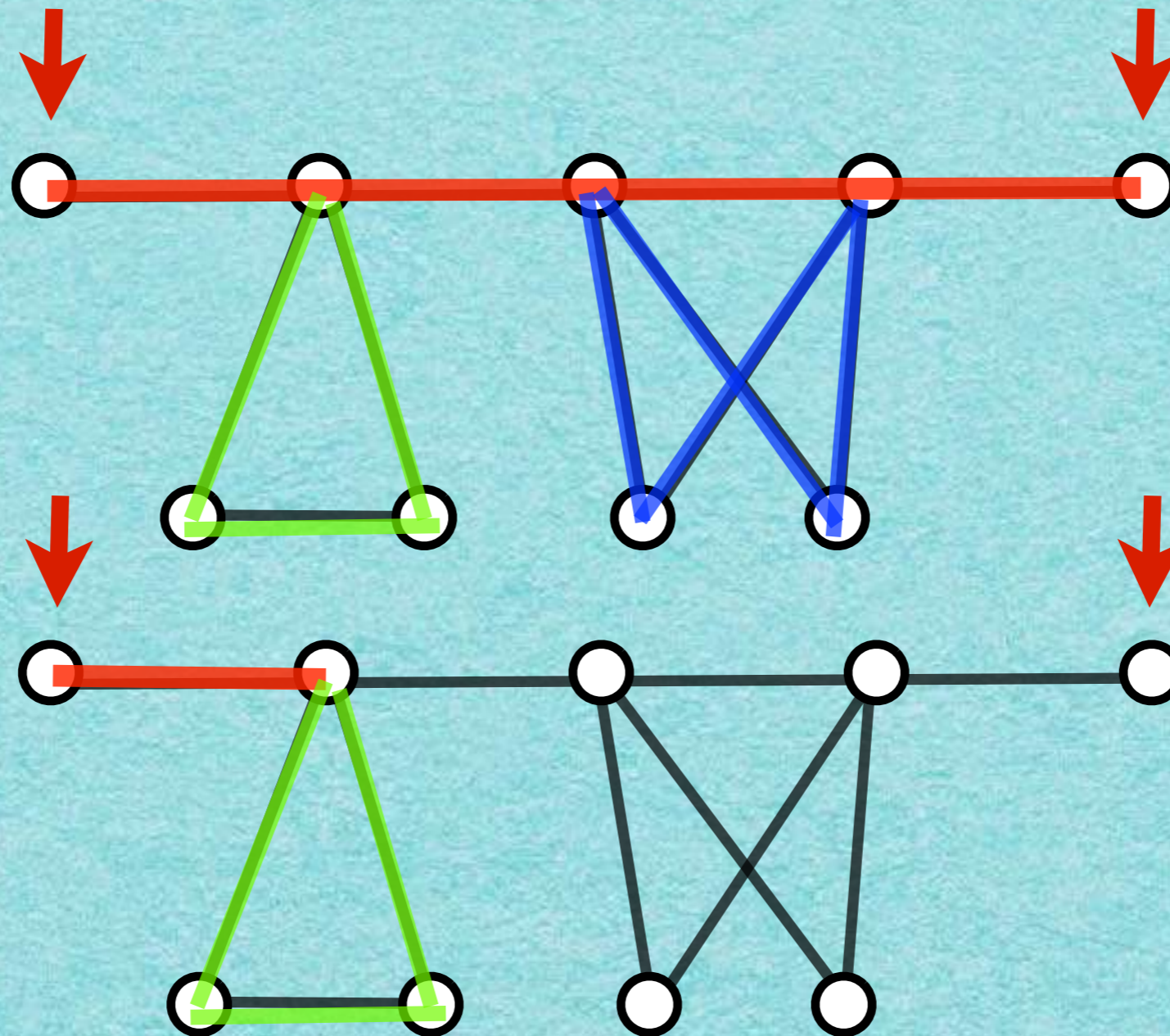
# Wegebau



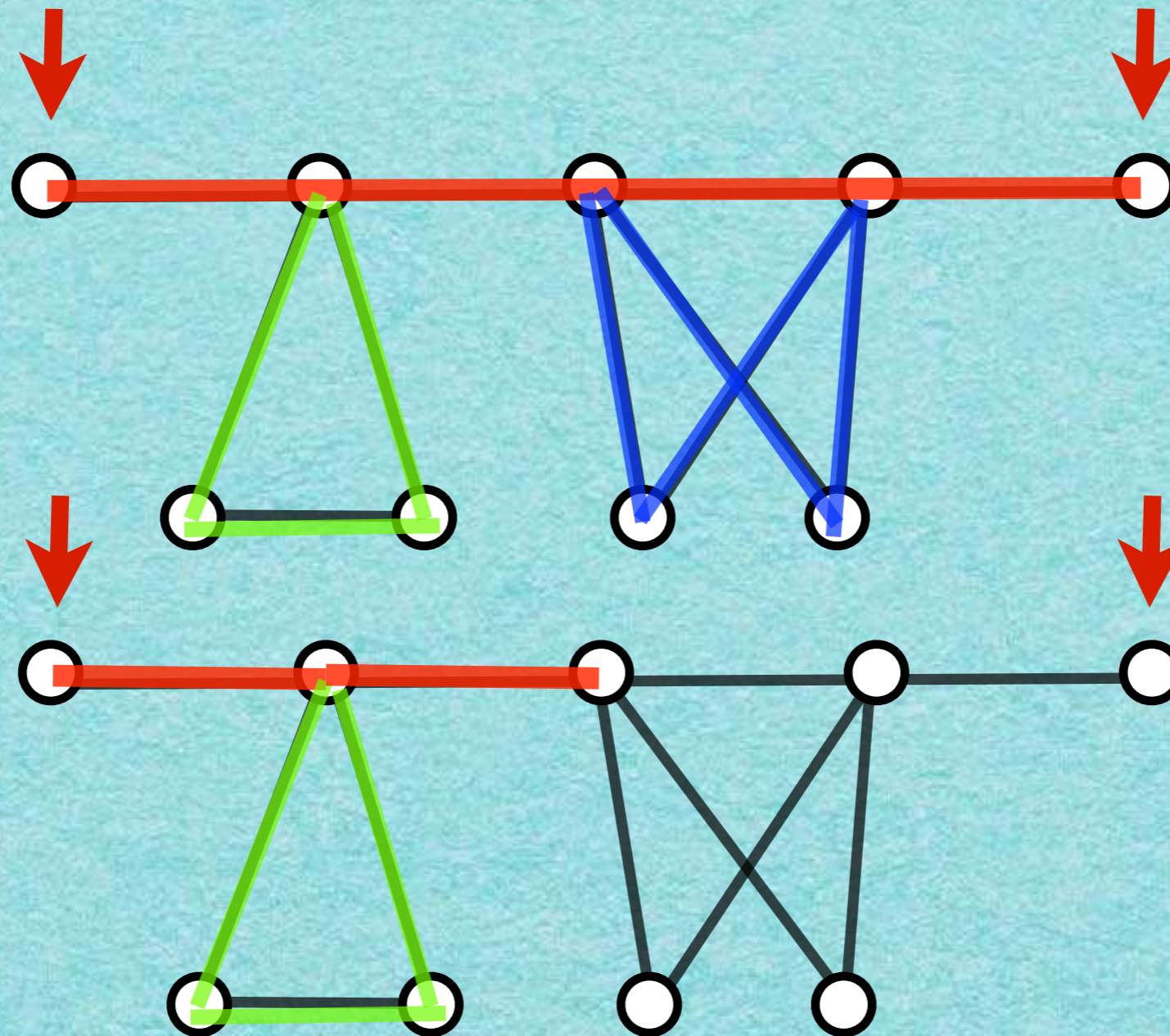
# Wegebau



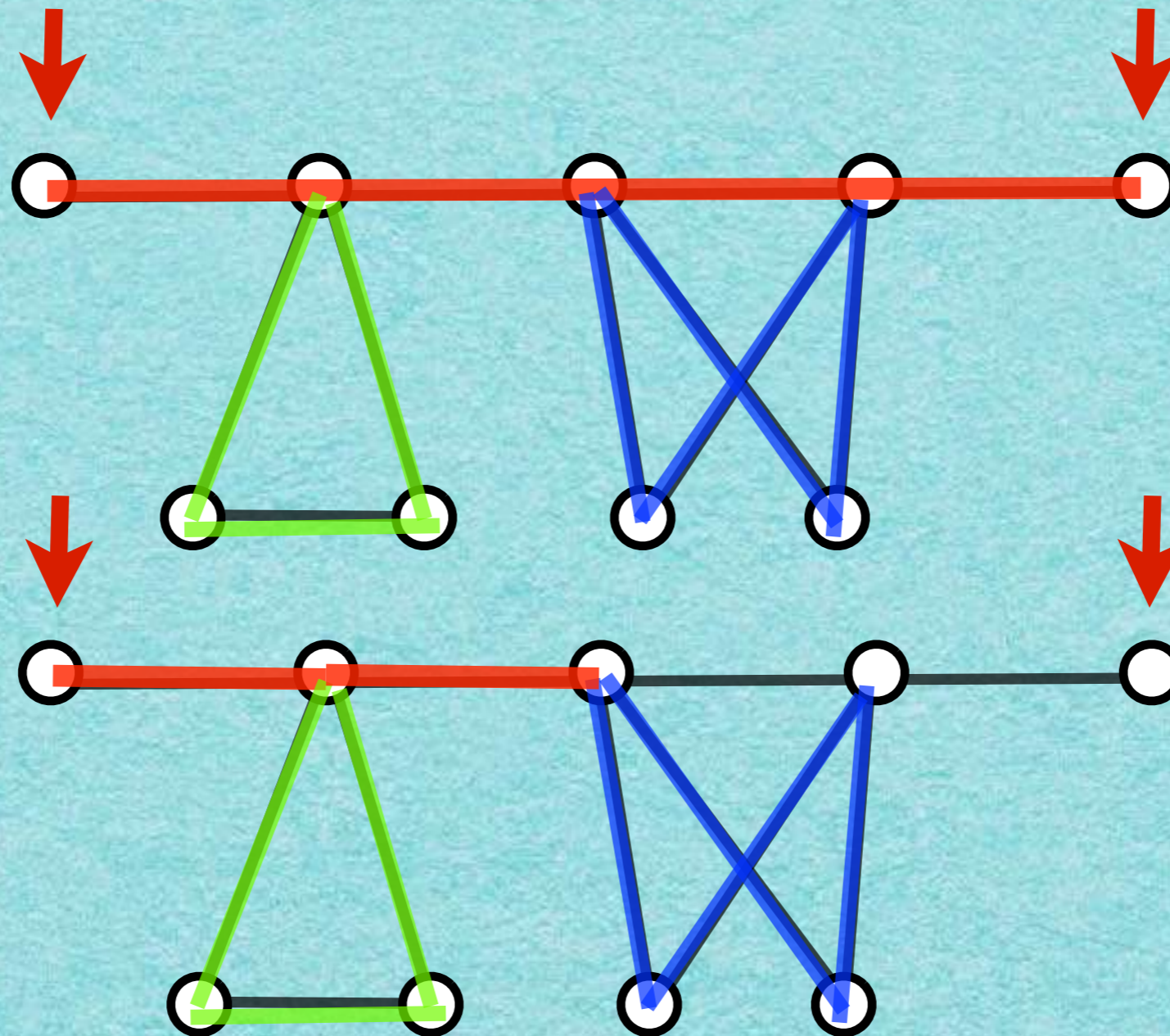
# Wegebau



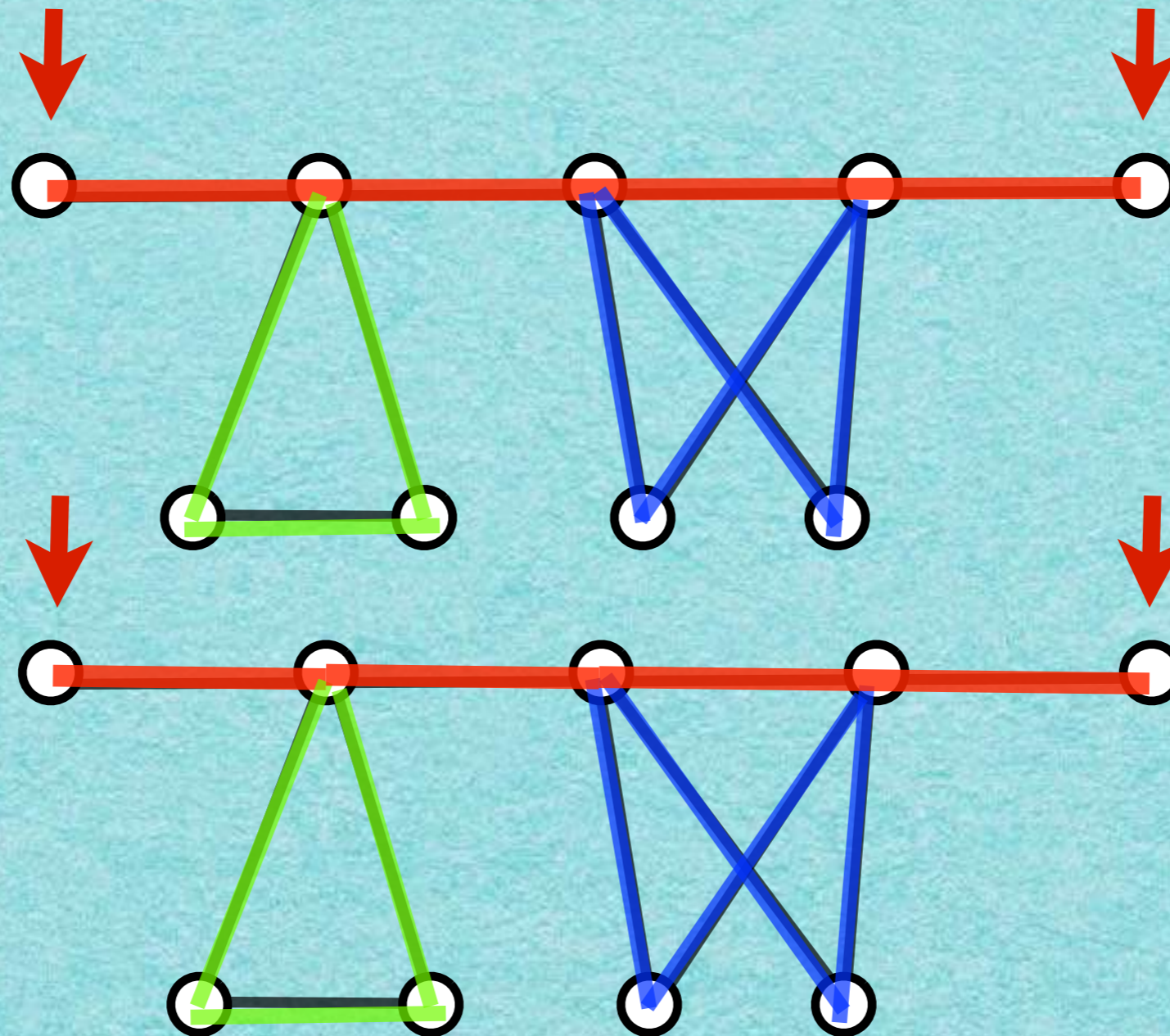
# Wegebau



# Wegebau



# Wegebau





SOLVTIO PROBLEMATIS  
AD  
GEOMETRIAM SITVS  
PERTINENTIS.  
AVCTORE  
*Leonb. Eulero.*

§. 1.

Tab. VIII. **P**ariter illam Geometriae partem, quae circa quæ-  
ritates versatur, et omni tempore firmo studio  
est excolta, alterius partis etiamnum admodum  
ignotae primis mentionem fecit *Leibnitzius*, quam Geo-  
metriam sine vocauit. Illa pars ab ipso in solo sine  
determinando, sineque propriis creendis occupata  
esse statuitur; in quo negotio neque ad quantitates re-  
spiciendum, neque calculo quantitarum vtrum sit.  
Cuiusmodi autem problemata ad hanc partem Geometriam  
pertinent, et quali methodo in eis resolendis uti oportet,  
non satis est definitum. Quomobrem, cum nuper  
problematis eiusdem mentio esset facta, quod quidem  
ad geometriam pertinere videbatur, at ita erat con-  
paratum, ut neque determinationem quantitarum requi-  
reret, neque solutionem calculi quantitarum ope admi-  
ttere, sed ad geometriam sine referre haud dubitanti-  
præsertim quod in eius solutione sine sine in conside-  
rationem veniat, calculi vero nullius prorsus sit usus.  
Methodam ergo meam quam ad hanc generis proble-  
mata

Euler: (1) Das gilt für jede beliebige Instanz: Mit mehr als zwei ungeraden Knoten gibt es keinen solchen Weg.

(2) Man kann auch charakterisieren, unter welchen Bedingungen es einen Weg tatsächlich gibt.

Hierholzer proved that a [graph](#) has an [Eulerian cycle](#) if and only if it is connected and every vertex has an even degree (excluding the starting and terminal vertices). This result had been given, without proof, by [Leonhard Euler](#) in 1736. Hierholzer apparently explained his proof, just before his premature death in 1871, to a colleague who then arranged for its posthumous publication which appeared in 1873.<sup>[1]</sup>

### [Mathematische Annalen](#)

..... March 1873, Volume 6, [Issue 1](#), pp 30–32

## Ueber die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren.

VON CARL HIERHOLZER.

Mitgetheilt von CHR. WISNER\*).

In einem beliebig verschlungenen Linienzuge mögen *Zweige* eines Punktes diejenigen verschiedenen Theile des Zuges heissen, auf welchen man den fraglichen Punkt verlassen kann. Ein Punkt mit mehreren *Zweigen* heisse ein *Knotenpunkt*, der so vielfach genannt werde, als

---

\*) Die folgende Untersuchung trug der leider so früh dem Dienste der Wissenschaft durch den Tod entrissene Privatdocent Dr. Hierholzer dahier (gest. 13. Sept. 1871) einem Kreise befreundeter Mathematiker vor. Um sie vor Vergessenheit zu bewahren, musste sie bei dem Mangel jeder schriftlichen Aufzeichnung aus dem Gedächtniss wieder hergestellt werden, was ich unter Beihilfe meines verehrten Collegen Lüröth durch das Folgende möglichst getren auszuführen suchte.

# Algorithmus 2.7

INPUT: Graph  $G$  mit höchstens zwei ungeraden Knoten

OUTPUT: Ein Weg in  $G$ .

1. Starte in einem Knoten  $v_0$  (ungerade, sonst beliebig);
2. Solange es eine zum gegenwärtigen Knoten  $v_i$  inzidente unbenutzte Kante  $\{v_i, v_j\}$  gibt:
  - 2.1. Wähle eine dieser Kanten aus,  $e_i = \{v_i, v_j\}$
  - 2.2. Laufe zum Nachbarknoten  $v_j$
  - 2.3. Lösche die Kante aus der Liste der unbenutzten Kanten.
  - 2.4. Setze  $v_{i+1} := v_j$
  - 2.5. Setze  $i := i+1$
3. STOP

# Algorithmus 2.8

## Algorithmus von Hierholzer

INPUT: Ein zusammenhängender Graph  $G$  mit höchstens zwei ungeraden Knoten

OUTPUT: Ein Eulerweg bzw. eine Eulertour in  $G$

- A. Wähle einen Startknoten  $v$  (ungerade falls vorhanden);
- B. Verwende Algorithmus 2.7, um einen Weg  $W$  von  $v$  aus zu bestimmen;
- C. Solange es noch unbenutzte Kanten gibt:
  - C.1. Wähle einen von  $W$  besuchten Knoten  $w$  mit positivem Grad im Restgraphen;
  - C.2. Verwende Algorithmus 2.7, um einen Weg  $W'$  von  $w$  aus zu bestimmen;
  - C.3. Verschmelze  $W$  und  $W'$
- D. STOP

## 2.3 Eulerwege

### Satz 2.9

- (i) Das Verfahren 2.7 stoppt immer in endlicher Zeit, ist also ein Algorithmus.*
- (ii) Der Algorithmus liefert einen Weg.*
- (iii) Ist  $v_0$  ungerade, stoppt der Algorithmus im zweiten ungeraden Knoten.*
- (iv) Ist  $G$  eulersch (d.h. haben alle Knoten gerade Grad), stoppt der Algorithmus in  $v_0$ , liefert also einen geschlossenen Weg.*

### Beweis:

*(i)*

## 2.3 Eulerwege

### Satz 2.9

- (i) Das Verfahren 2.7 stoppt immer in endlicher Zeit, ist also ein Algorithmus.*
- (ii) Der Algorithmus liefert einen Weg.*
- (iii) Ist  $v_0$  ungerade, stoppt der Algorithmus im zweiten ungeraden Knoten.*
- (iv) Ist  $G$  eulersch (d.h. haben alle Knoten gerade Grad), stoppt der Algorithmus in  $v_0$ , liefert also einen geschlossenen Weg.*

### Beweis:

*(i)*

# Algorithmus 2.7

INPUT: Graph  $G$  mit höchstens zwei ungeraden Knoten

OUTPUT: Ein Weg in  $G$ .

1. Starte in einem Knoten  $v_0$  (ungerade, sonst beliebig);
2. Solange es eine zum gegenwärtigen Knoten  $v_i$  inzidente unbenutzte Kante  $\{v_i, v_j\}$  gibt:
  - 2.1. Wähle eine dieser Kanten aus,  $e_i = \{v_i, v_j\}$
  - 2.2. Laufe zum Nachbarknoten  $v_j$
  - 2.3. Lösche die Kante aus der Liste der unbenutzten Kanten.
  - 2.4. Setze  $v_{i+1} := v_j$
  - 2.5. Setze  $i := i+1$
3. STOP

## 2.3 Eulerwege

### Satz 2.9

- (i) Das Verfahren 2.7 stoppt immer in endlicher Zeit, ist also ein Algorithmus.*
- (ii) Der Algorithmus liefert einen Weg.*
- (iii) Ist  $v_0$  ungerade, stoppt der Algorithmus im zweiten ungeraden Knoten.*
- (iv) Ist  $G$  eulersch (d.h. haben alle Knoten gerade Grad), stoppt der Algorithmus in  $v_0$ , liefert also einen geschlossenen Weg.*

### Beweis:

- (i) Bei jedem Durchlauf der Schleifen 2.1-2.5 wird in 2.3 eine Kante entfernt. Das kann nur endlich oft passieren. Also muss das Verfahren irgendwann stoppen.*



## 2.3 Eulerwege

### **Satz 2.9**

- (i) Das Verfahren 2.7 stoppt immer in endlicher Zeit, ist also ein Algorithmus.***
- (ii) Der Algorithmus liefert einen Weg.***
- (iii) Ist  $v_0$  ungerade, stoppt der Algorithmus im zweiten ungeraden Knoten.***
- (iv) Ist  $G$  eulersch (d.h. haben alle Knoten gerade Grad), stoppt der Algorithmus in  $v_0$ , liefert also einen geschlossenen Weg.***

## 2.3 Eulerwege

### Satz 2.9

- (i) Das Verfahren 2.7 stoppt immer in endlicher Zeit, ist also ein Algorithmus.*
- (ii) Der Algorithmus liefert einen Weg.*
- (iii) Ist  $v_0$  ungerade, stoppt der Algorithmus im zweiten ungeraden Knoten.*
- (iv) Ist  $G$  eulersch (d.h. haben alle Knoten gerade Grad), stoppt der Algorithmus in  $v_0$ , liefert also einen geschlossenen Weg.*

### Beweis:

*(ii)*

# Algorithmus 2.7

INPUT: Graph  $G$  mit höchstens zwei ungeraden Knoten

OUTPUT: Ein Weg in  $G$ .

1. Starte in einem Knoten  $v_0$  (ungerade, sonst beliebig);
2. Solange es eine zum gegenwärtigen Knoten  $v_i$  inzidente unbenutzte Kante  $\{v_i, v_j\}$  gibt:
  - 2.1. Wähle eine dieser Kanten aus,  $e_i = \{v_i, v_j\}$
  - 2.2. Laufe zum Nachbarknoten  $v_j$
  - 2.3. Lösche die Kante aus der Liste der unbenutzten Kanten.
  - 2.4. Setze  $v_{i+1} := v_j$
  - 2.5. Setze  $i := i+1$
3. STOP

## 2.3 Eulerwege

### Satz 2.9

- (i) Das Verfahren 2.7 stoppt immer in endlicher Zeit, ist also ein Algorithmus.*
- (ii) Der Algorithmus liefert einen Weg.*
- (iii) Ist  $v_0$  ungerade, stoppt der Algorithmus im zweiten ungeraden Knoten.*
- (iv) Ist  $G$  eulersch (d.h. haben alle Knoten gerade Grad), stoppt der Algorithmus in  $v_0$ , liefert also einen geschlossenen Weg.*

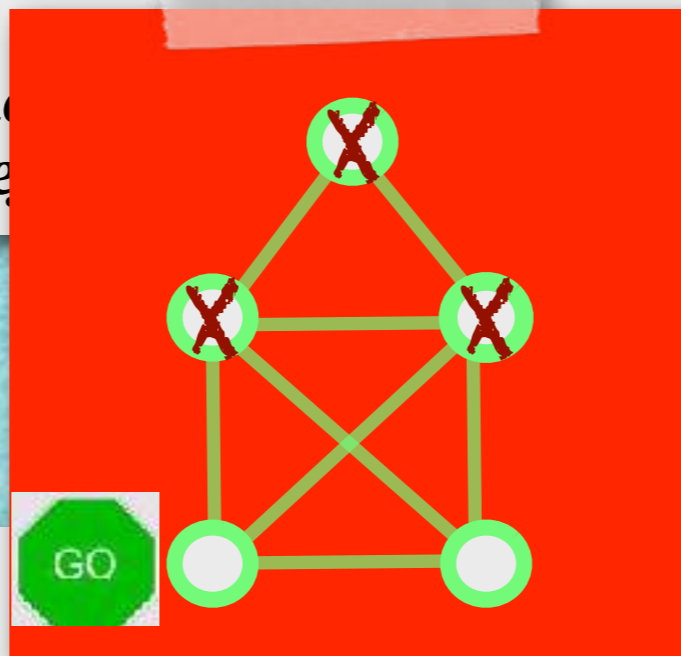
### Beweis:

- (ii) Nach Konstruktion erhalten wir eine Kantenfolge; keine Kante wird doppelt verwendet.*

## 2.3 Eulerwege

### Satz 2.9

- (i) *Das Verfahren 2.7 stoppt immer in endlicher Zeit, ist also ein Algorithmus.*
- (ii) *Der Algorithmus liefert einen Weg.*
- (iii) *Ist  $v_0$  ungerade, stoppt der Algorithmus im zweiten ungeraden Knoten.*
- (iv) *Ist  $G$  eulersch (d.h. hat alle Knoten ungeraden Grad), stoppt der Algorithmus in  $v_0$ , liefert also einen geschlossenen Weg.*



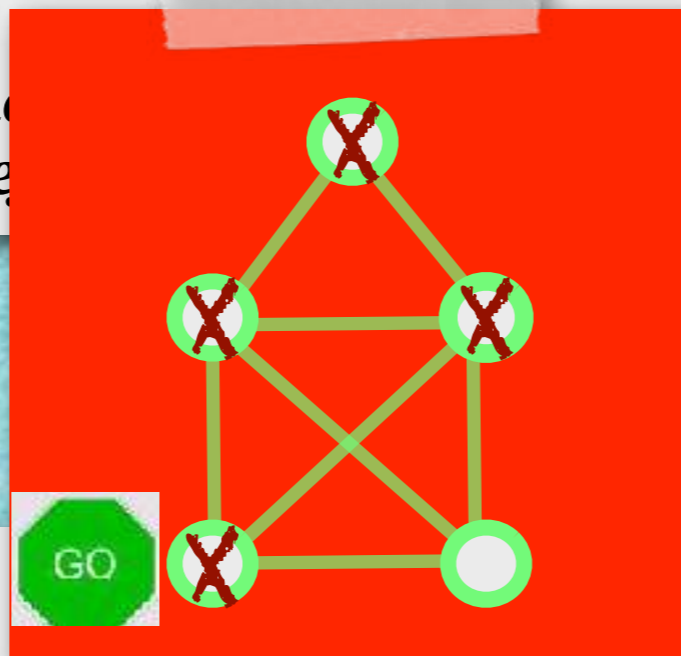
### Beweis:

- (iii) *Ein gerader Knoten wird genauso oft verlassen wie betreten; also kann der Algorithmus in keinem davon stoppen,*

## 2.3 Eulerwege

### Satz 2.9

- (i) *Das Verfahren 2.7 stoppt immer in endlicher Zeit, ist also ein Algorithmus.*
- (ii) *Der Algorithmus liefert einen Weg.*
- (iii) *Ist  $v_0$  ungerade, stoppt der Algorithmus im zweiten ungeraden Knoten.*
- (iv) *Ist  $G$  eulersch (d.h. hat alle Knoten ungeraden Grad), stoppt der Algorithmus in  $v_0$ , liefert aber keinen geschlossenen Weg.*



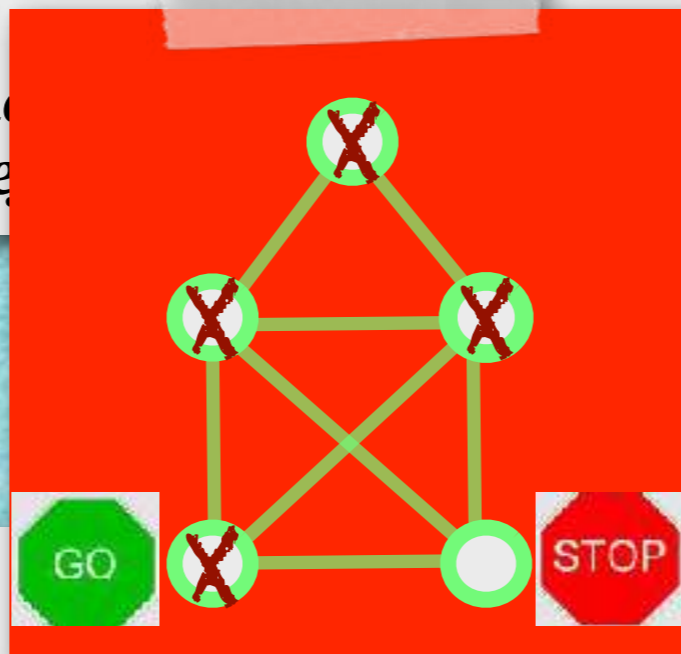
### Beweis:

- (iii) **Ein gerader Knoten wird genauso oft verlassen wie betreten; also kann der Algorithmus in keinem davon stoppen, aber auch nicht im Startknoten, wenn dieser ungerade ist.**

## 2.3 Eulerwege

### Satz 2.9

- (i) *Das Verfahren 2.7 stoppt immer in endlicher Zeit, ist also ein Algorithmus.*
- (ii) *Der Algorithmus liefert einen Weg.*
- (iii) *Ist  $v_0$  ungerade, stoppt der Algorithmus im zweiten ungeraden Knoten.*
- (iv) *Ist  $G$  eulersch (d.h. hat alle Knoten ungeraden Grad), stoppt der Algorithmus in  $v_0$ , liefert also einen geschlossenen Weg.*



### Beweis:

- (iii) **Ein gerader Knoten wird genauso oft verlassen wie betreten; also kann der Algorithmus in keinem davon stoppen, aber auch nicht im Startknoten, wenn dieser ungerade ist. Es bleibt nur der andere ungerade Knoten.**

## 2.3 Eulerwege

### Satz 2.9

- (i) Das Verfahren 2.7 stoppt immer in endlicher Zeit, ist also ein Algorithmus.*
- (ii) Der Algorithmus liefert einen Weg.*
- (iii) Ist  $v_0$  ungerade, stoppt der Algorithmus im zweiten ungeraden Knoten.*
- (iv) Ist  $G$  eulersch (d.h. haben alle Knoten gerade Grad), stoppt der Algorithmus in  $v_0$ , liefert also einen geschlossenen Weg.*

### Beweis:

- (iv) Wie in (iii) kann der Algorithmus in keinem geraden Knoten stoppen, der nicht der Startknoten ist.  
Es bleibt nur der Startknoten.*



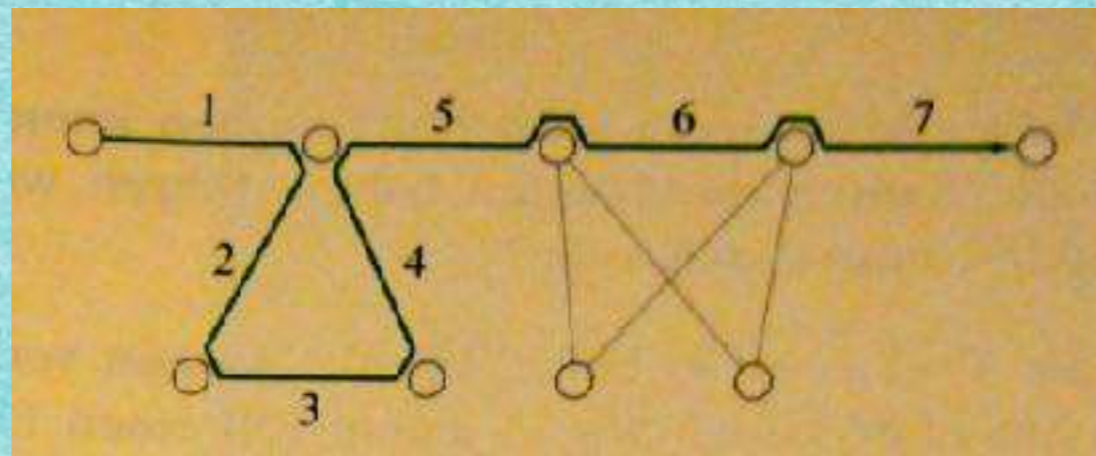
## 2.3 Eulerwege

### Satz 2.10

Wenn Algorithmus 2.7 stoppt, bleibt ein eulerscher Graph zurück, also ein Graph mit lauter geraden Knoten.

### Beweis

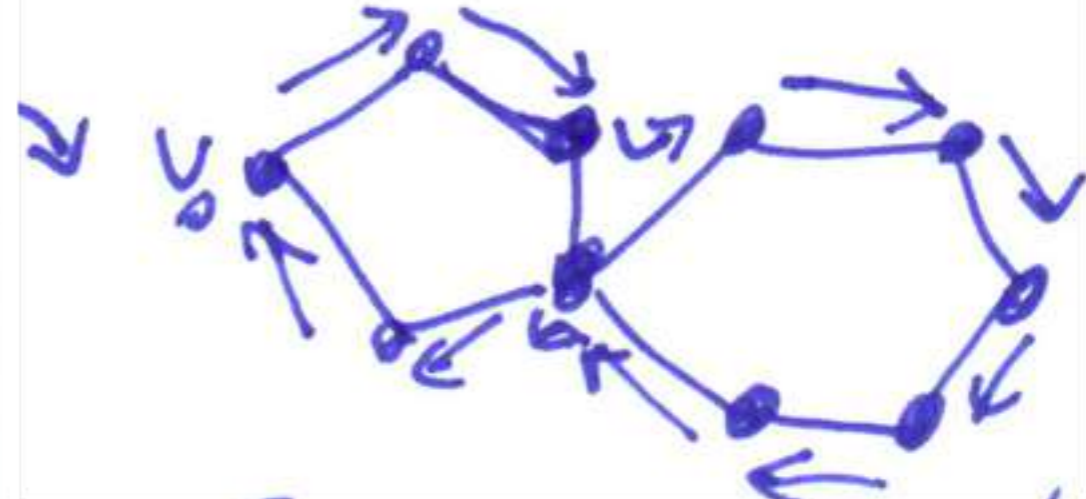
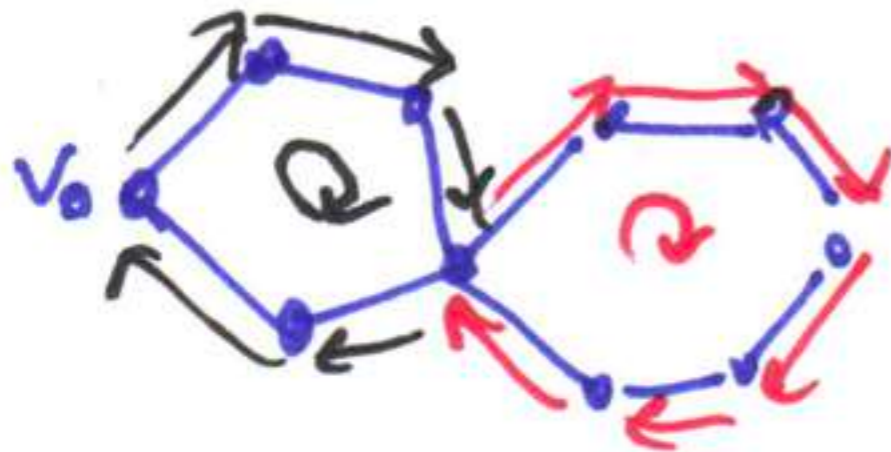
Durch Entfernen des konstruierten Weges wird für jeden geraden Knoten der Grad um einen gerade Zahl geändert, bleibt also gerade. Für einen der ggf. vorhandenen ungeraden Knoten ändert sich der Grad um eine ungerade Zahl, wird also auch gerade.



## 2.3 Eulerwege

### Satz 2.10

Wenn Algorithmus 2.7 stoppt, bleibt ein eulerscher Graph zurück, also ein Graph mit lauter geraden Knoten.



### Beobachtung 2.11

- (i) Zwei geschlossene Wege mit einem gemeinsamen Knoten kann man in einen geschlossenen Weg verwandeln.
- (ii) Man kann aus allen Wegen einen Weg machen, wenn der Graph zusammenhängend ist.

# Algorithmus 2.8

INPUT: Ein zusammenhängender Graph  $G$  mit höchstens zwei ungeraden Knoten

OUTPUT: Ein Eulerweg bzw. eine Eulertour in  $G$

- A. Wähle einen Startknoten  $v$  (ungerade falls vorhanden);
- B. Verwende Algorithmus 2.7, um einen Weg  $W$  von  $v$  aus zu bestimmen;
- C. Solange es noch unbenutzte Kanten gibt:
  - C.1. Wähle einen von  $W$  besuchten Knoten  $w$  mit positivem Grad im Restgraphen;
  - C.2. Verwende Algorithmus 2.7, um einen Weg  $W'$  von  $w$  aus zu bestimmen;
  - C.3. Verschmelze  $W$  und  $W'$
- D. STOP

## 2.3 Eulerwege

### Satz 2.12

- (i) *Das Verfahren 2.8 ist endlich.*
- (ii) *Alle Schritte lassen sich korrekt ausführen.*
- (iii) *Algorithmus 2.8 liefert einen Eulerweg bzw. eine Eulertour.*

### Beweis:

- (i) **Wie gehabt: Es gibt nur endlich viele Kanten, also muss das Verfahren irgendwann stoppen.**

## 2.3 Eulerwege

### **Satz 2.12**

- (i) Das Verfahren 2.8 ist endlich.*
- (ii) Alle Schritte lassen sich korrekt ausführen.*
- (iii) Algorithmus 2.8 liefert einen Eulerweg bzw. eine Eulertour.*

### **Beweis:**

- (i) Wie gehabt: Es gibt nur endlich viele Kanten, also muss das Verfahren irgendwann stoppen.*

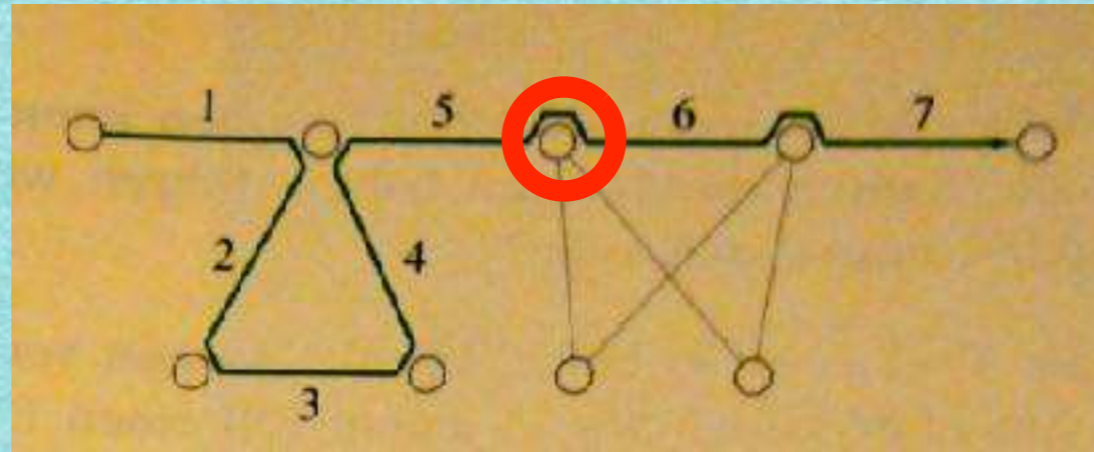
# Algorithmus 2.8

INPUT: Ein zusammenhängender Graph  $G$  mit höchstens zwei ungeraden Knoten

OUTPUT: Ein Eulerweg bzw. eine Eulertour in  $G$

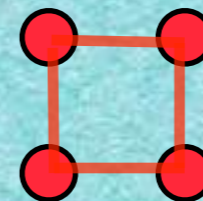
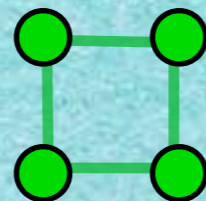
- A. Wähle einen Startknoten  $v$  (ungerade falls vorhanden);
- B. Verwende Algorithmus 2.7, um einen Weg  $W$  von  $v$  aus zu bestimmen;
- C. Solange es noch unbenutzte Kanten gibt:
  - C.1. Wähle einen von  $W$  besuchten Knoten  $w$  mit positivem Grad im Restgraphen;
  - C.2. Verwende Algorithmus 2.7, um einen Weg  $W'$  von  $w$  aus zu bestimmen;
  - C.3. Verschmelze  $W$  und  $W'$
- D. STOP

## (Beweis Satz 2.12, (ii) )



**(ii)**

**Solange es noch unbenutzte Kanten gibt, kann man diese auch von den benutzten Kanten aus besuchen:  
Weil  $G$  zusammenhängend ist, muss es einen Knoten geben, der sowohl zu einer benutzten als auch zu einer unbenutzten Kante inzident ist.**

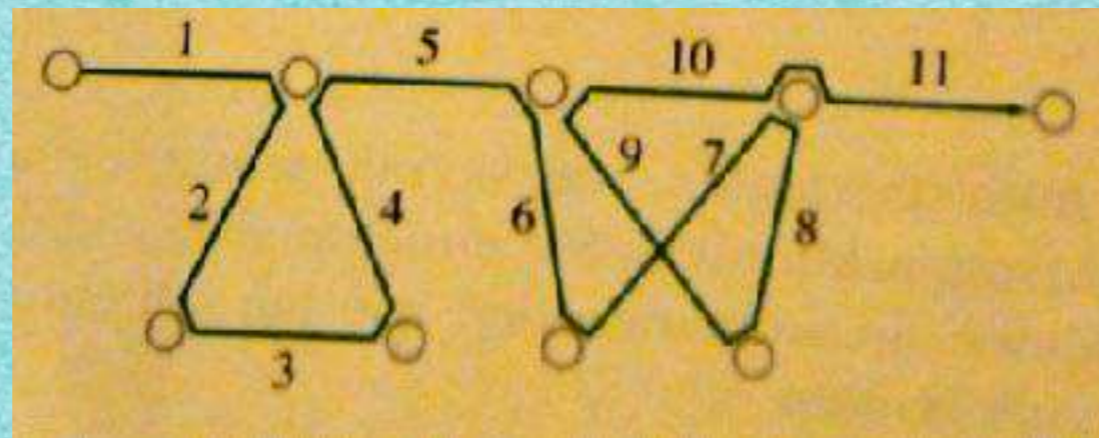


## (Beweis Satz 2.12, (iii) )

### Satz 2.12

- (i) *Das Verfahren 2.8 ist endlich.*
- (ii) *Alle Schritte lassen sich korrekt ausführen.*
- (iii) *Algorithmus 2.8 liefert einen Eulerweg bzw. eine Eulertour.*

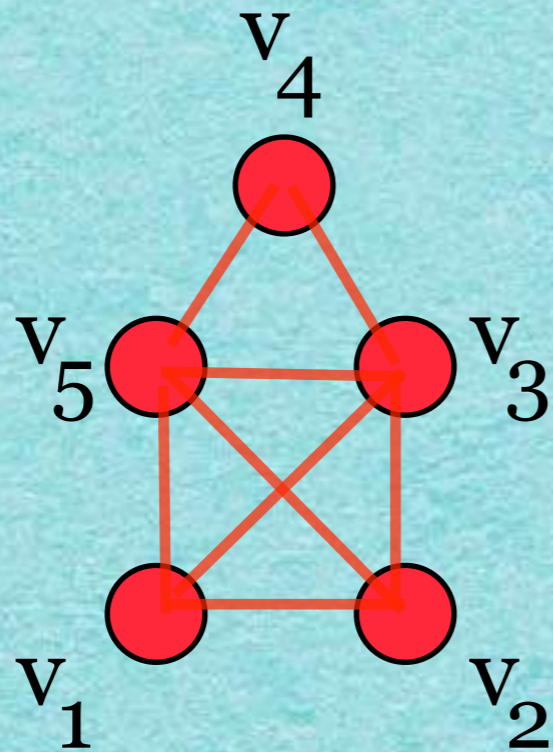
- (iii) **Am Ende haben wir einen Weg und es gibt keine unbenutzten Kanten mehr!**





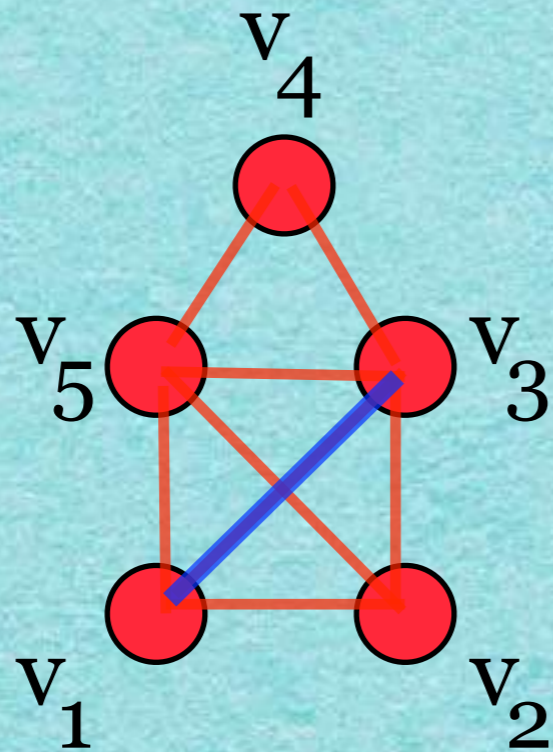
## 2.3 Eulerwege

Es geht auch anders!



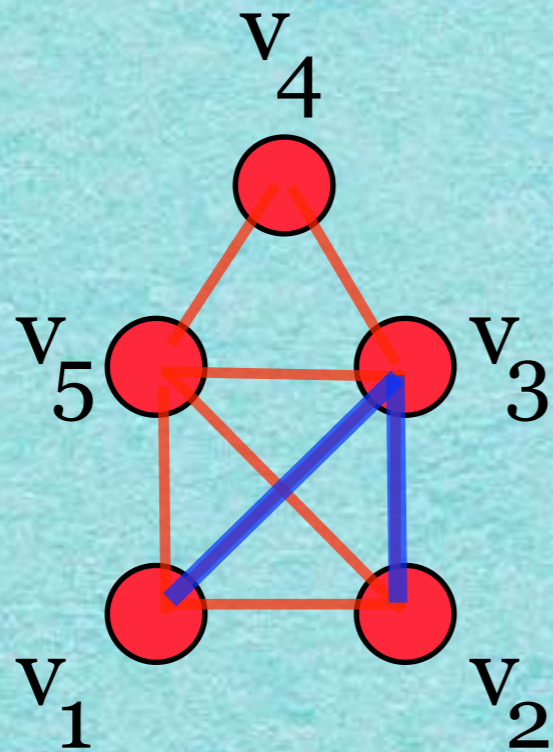
## 2.3 Eulerwege

Es geht auch anders!



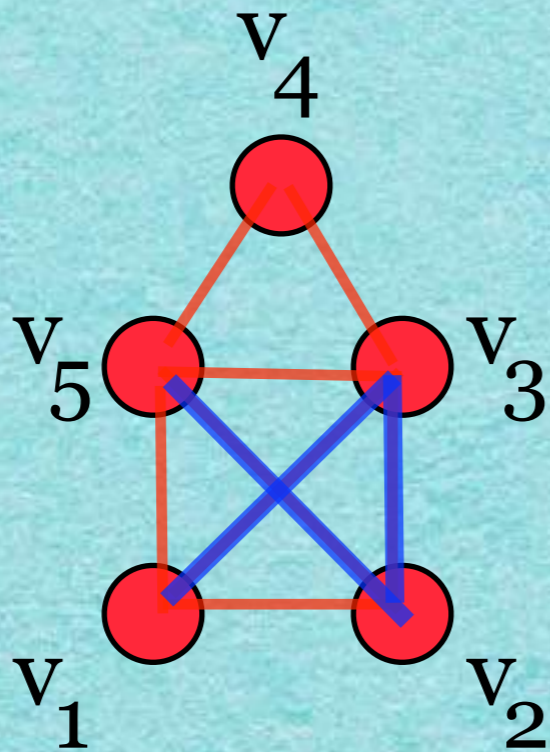
## 2.3 Eulerwege

Es geht auch anders!



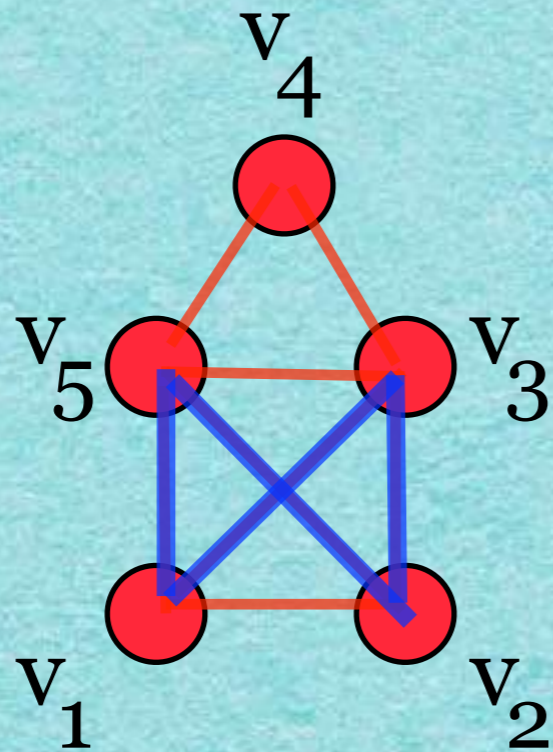
## 2.3 Eulerwege

Es geht auch anders!



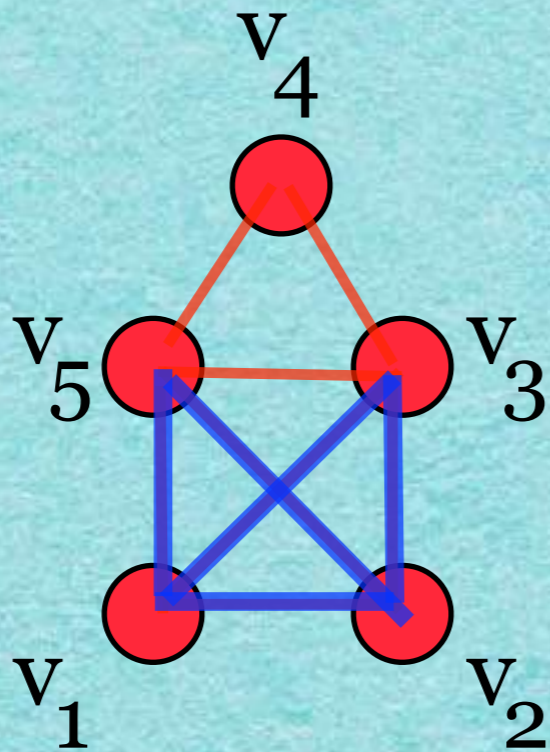
## 2.3 Eulerwege

Es geht auch anders!



## 2.3 Eulerwege

Es geht auch anders!

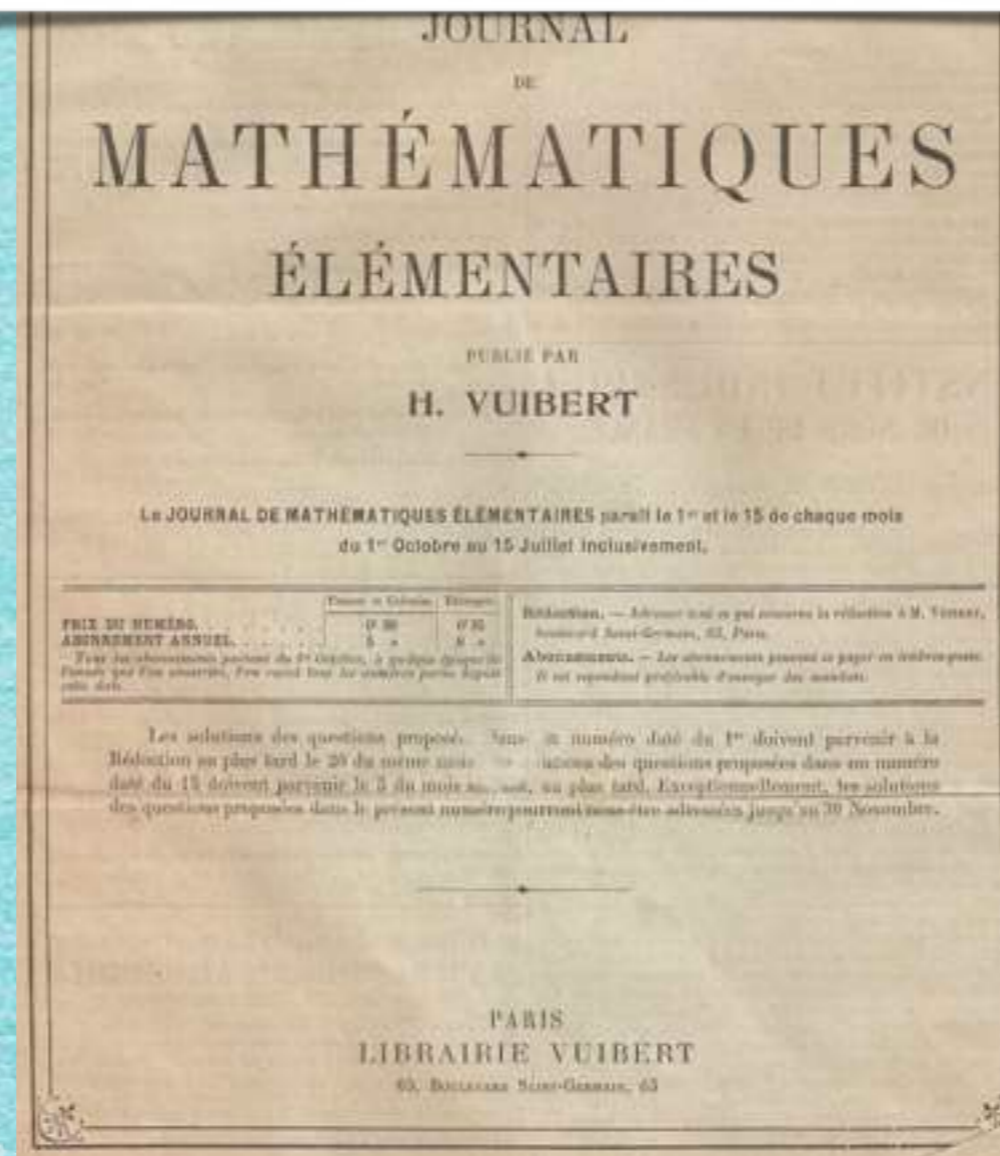


Wir hinterlassen Kanten?!

# Algorithmus 2.13

## Algorithmus von Fleury

<sup>^</sup> Fleury, M. (1883), "Deux problèmes de Géométrie de situation" [↗](#), *Journal de mathématiques élémentaires*, 2nd ser. (in French), 2: 257–261.



# Algorithmus 2.13

## Algorithmus von Fleury

INPUT: Graph  $G$  mit höchstens zwei ungeraden Knoten

OUTPUT: Ein Weg in  $G$ .

1. Starte in einem Knoten  $v_0$  (ungerade, sonst beliebig);
2. Solange es eine zum gegenwärtigen Knoten  $v_i$  inzidente unbenutzte Kante  $\{v_i, v_j\}$  gibt:
  - 2.1. Wähle eine dieser Kanten aus,  $e_i = \{v_i, v_j\}$  **der den Restgraphen zshgd. lässt**
  - 2.2. Laufe zum Nachbarknoten  $v_j$
  - 2.3. Lösche die Kante aus der Liste der unbenutzten Kanten.
  - 2.4. Setze  $v_{i+1} := v_j$
  - 2.5. Setze  $i := i+1$
3. STOP



## 2.3 Eulerwege

### **Satz 2.14**

- (i)** *Das Verfahren 2.13 ist endlich.*
- (ii)** *Alle Schritte lassen sich korrekt ausführen.*
- (iii)** *Algorithmus 2.13 liefert einen Eulerweg bzw. eine Eulertour.*

### **Beweis:**

- (i)** **Wie gehabt! Es gibt nur endlich viele Kanten, also muss das Verfahren irgendwann stoppen.**

# Algorithmus 2.13

## Algorithmus von Fleury

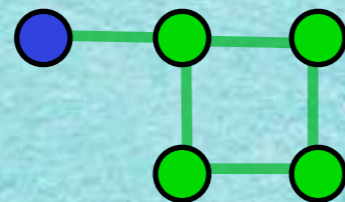
INPUT: Graph  $G$  mit höchstens zwei ungeraden Knoten

OUTPUT: Ein Weg in  $G$ .

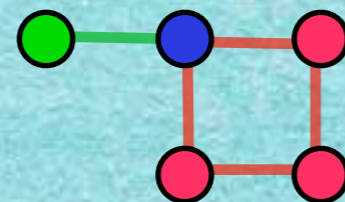
1. Starte in einem Knoten  $v_0$  (ungerade, sonst beliebig);
2. Solange es eine zum gegenwärtigen Knoten  $v_i$  inzidente unbenutzte Kante  $\{v_i, v_j\}$  gibt:
  - 2.1. Wähle eine dieser Kanten aus,  $e_i = \{v_i, v_j\}$ , der den Restgraphen zshgd. lässt
  - 2.2. Laufe zum Nachbarknoten  $v_j$
  - 2.3. Lösche die Kante aus der Liste der unbenutzten Kanten.
  - 2.4. Setze  $v_{i+1} := v_j$
  - 2.5. Setze  $i := i+1$
3. STOP

## (Beweis Satz 2.14, (ii) )

**(ii)** Kritisch ist 2.1. Wenn es nur eine Kante gibt, um den aktuellen Knoten zu verlassen, bleibt der Restgraph nach deren Benutzung zusammenhängend.



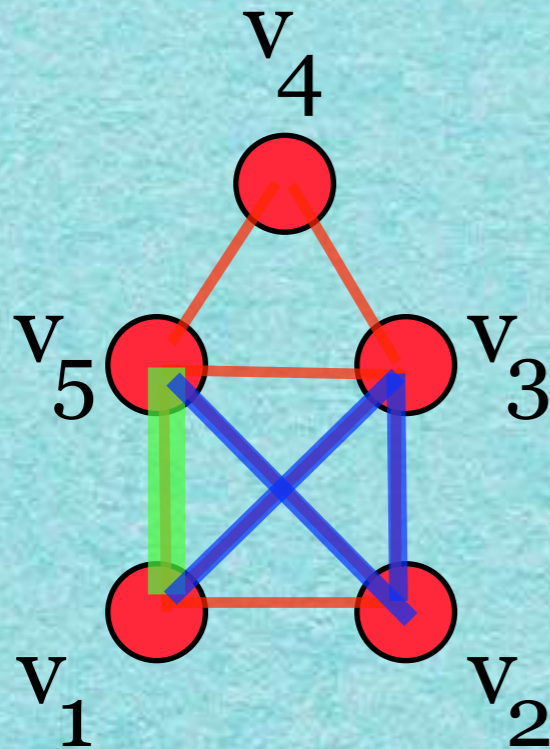
Wenn es mehr als eine Kante gibt, um den aktuellen Knoten zu verlassen, von denen eine bei Entfernen den Graphen unzusammenhängend macht, sind alle anderen Kanten in geschlossenenen Wegen enthalten.



Also wählt man eine der anderen.

## (Beweis Satz 2.14, (iii) )

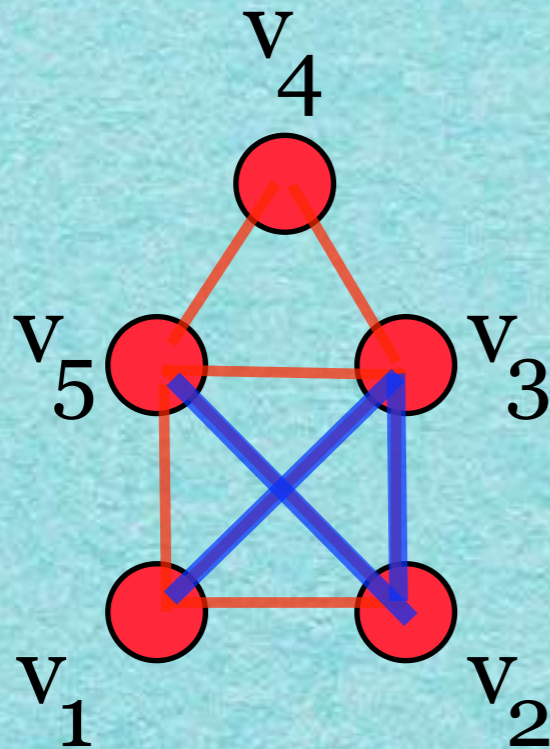
**(iii)** Nach Konstruktion bekommen wir einen Weg. Man hat immer einen zusammenhängenden Graphen, kann also keine Kanten zurücklassen.



**Wir hinterlassen keine Kanten!**

## (Beweis Satz 2.14, (iii) )

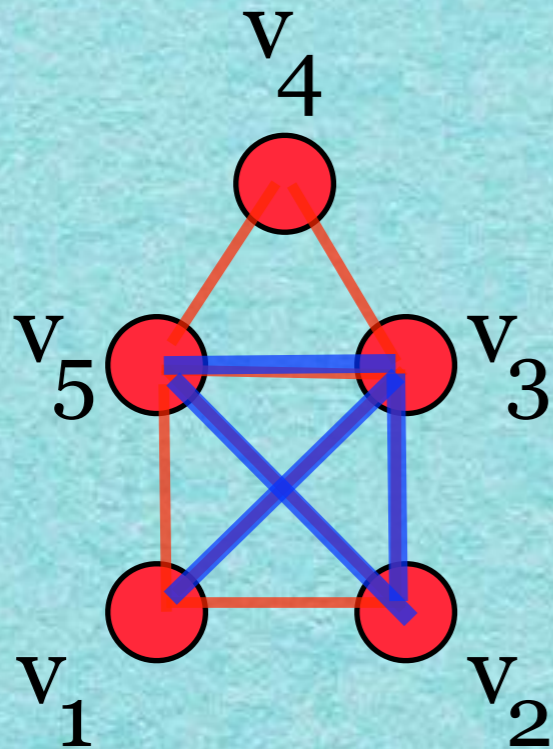
**(iii)** Nach Konstruktion bekommen wir einen Weg. Man hat immer einen zusammenhängenden Graphen, kann also keine Kanten zurücklassen.



**Wir hinterlassen keine Kanten!**

## (Beweis Satz 2.14, (iii) )

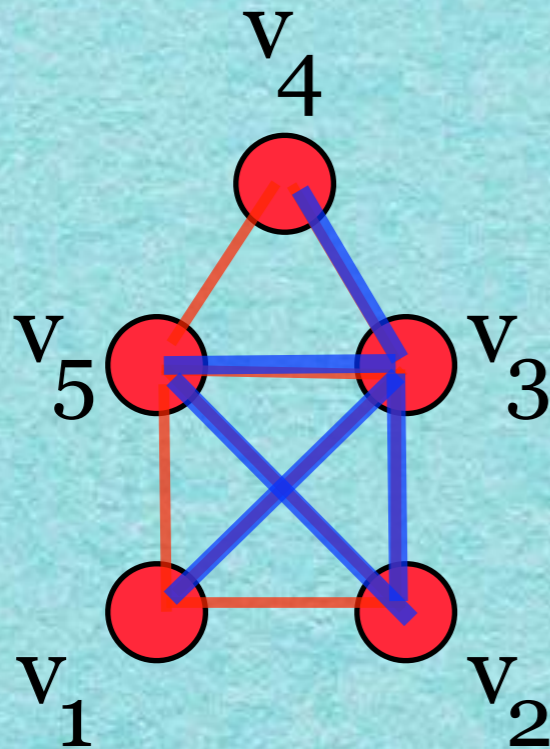
**(iii)** Nach Konstruktion bekommen wir einen Weg. Man hat immer einen zusammenhängenden Graphen, kann also keine Kanten zurücklassen.



**Wir hinterlassen keine Kanten!**

## (Beweis Satz 2.14, (iii) )

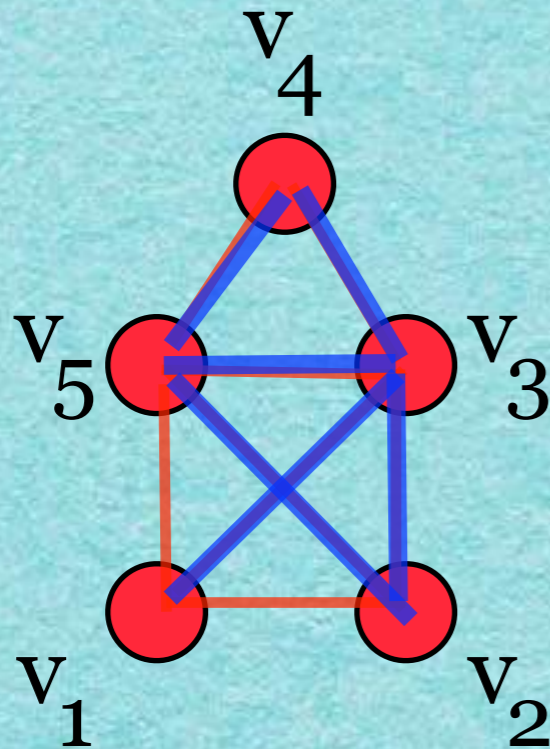
**(iii)** Nach Konstruktion bekommen wir einen Weg. Man hat immer einen zusammenhängenden Graphen, kann also keine Kanten zurücklassen.



**Wir hinterlassen keine Kanten!**

## (Beweis Satz 2.14, (iii) )

**(iii)** Nach Konstruktion bekommen wir einen Weg. Man hat immer einen zusammenhängenden Graphen, kann also keine Kanten zurücklassen.

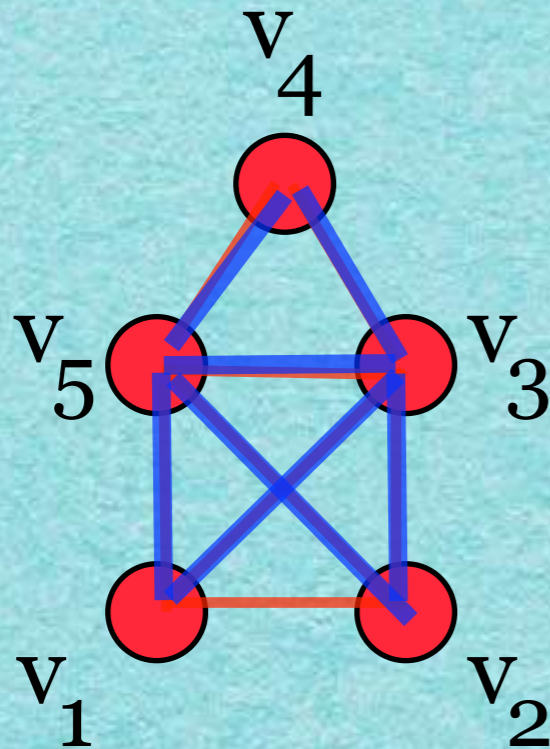


**Wir hinterlassen keine Kanten!**



## (Beweis Satz 2.14, (iii) )

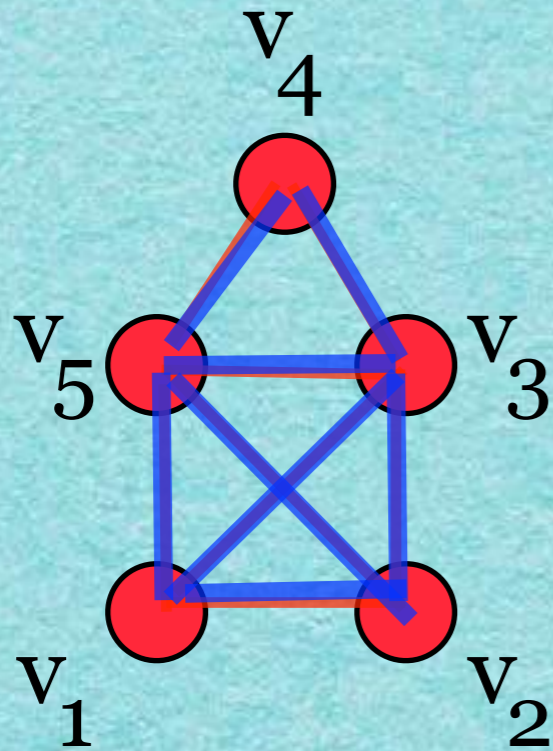
**(iii)** Nach Konstruktion bekommen wir einen Weg. Man hat immer einen zusammenhängenden Graphen, kann also keine Kanten zurücklassen.



**Wir hinterlassen keine Kanten!**

## (Beweis Satz 2.14, (iii) )

**(iii)** Nach Konstruktion bekommen wir einen Weg. Man hat immer einen zusammenhängenden Graphen, kann also keine Kanten zurücklassen.



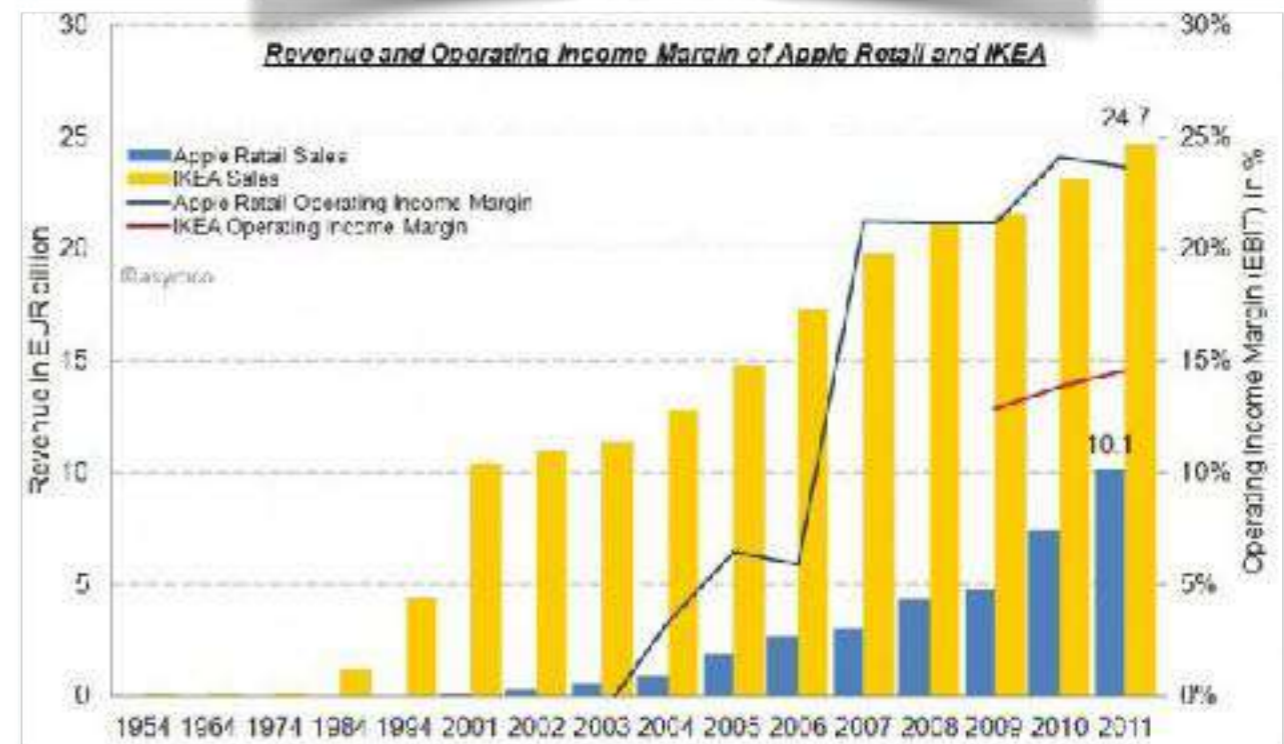
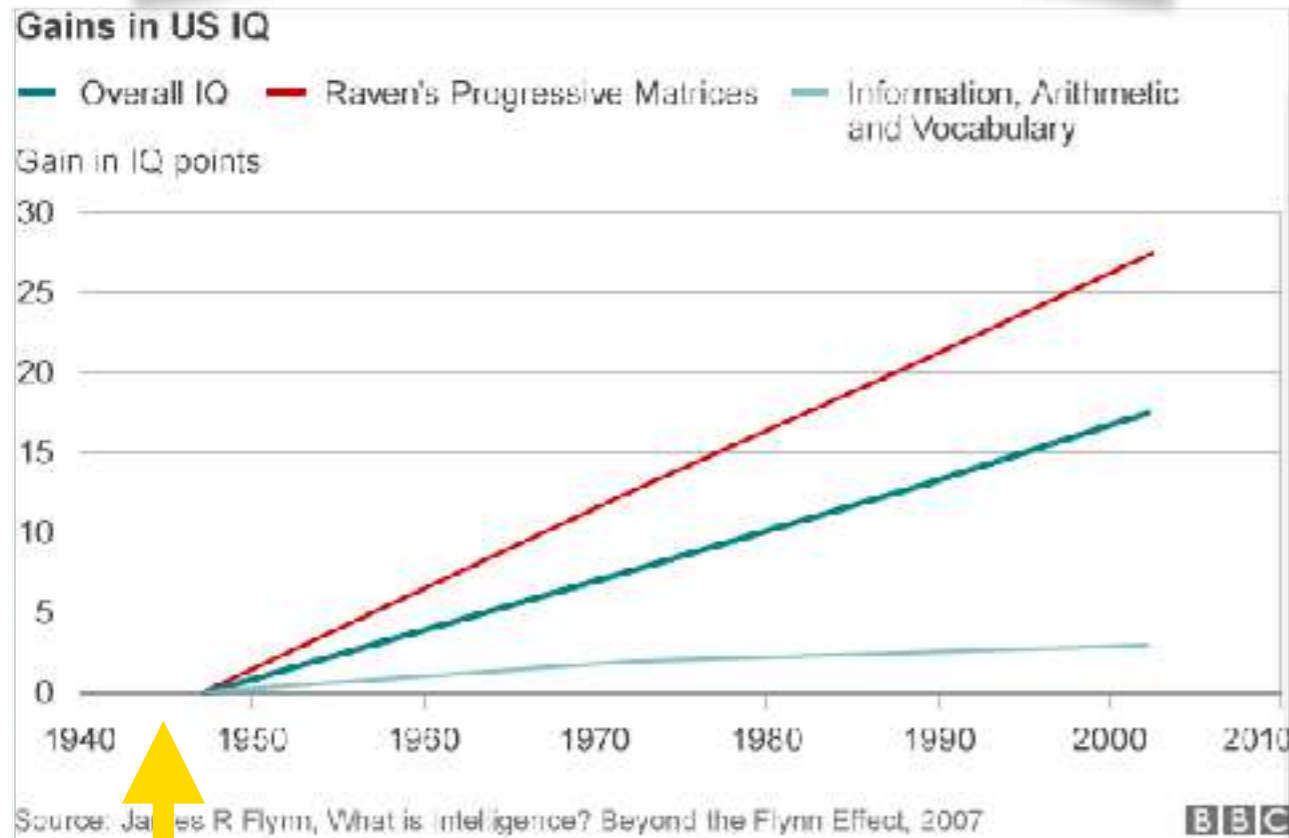
**Wir hinterlassen keine Kanten!**

# Feeling smarter already?!

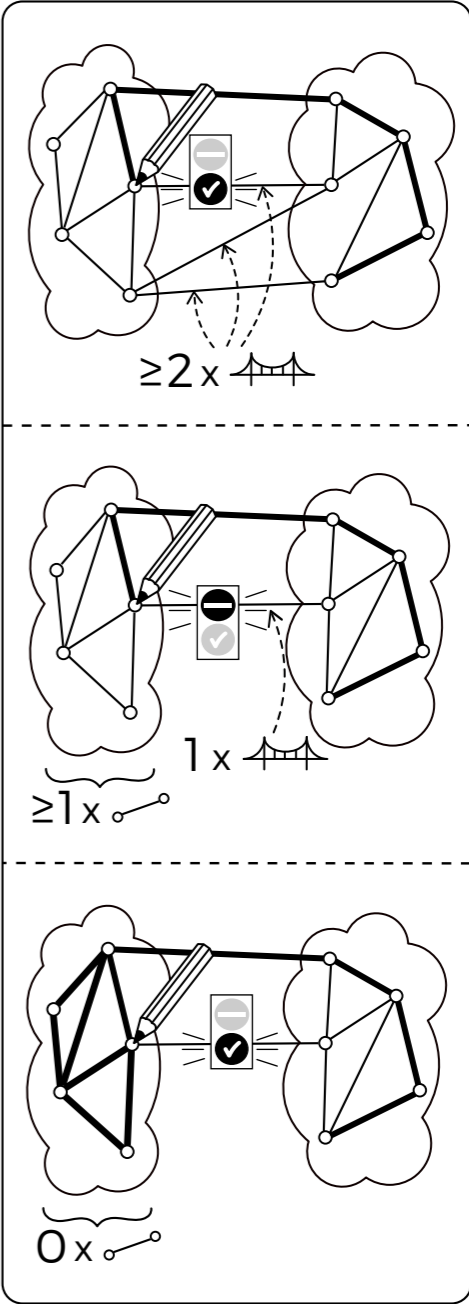
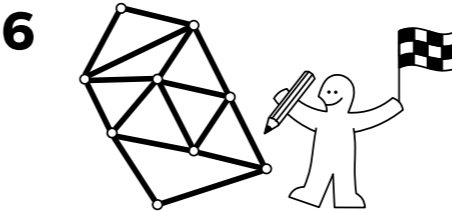
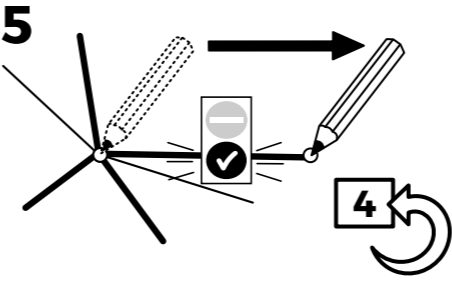
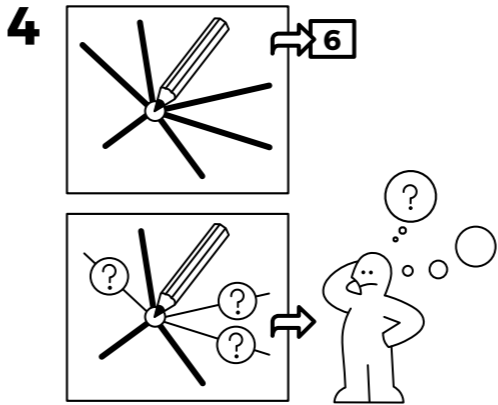
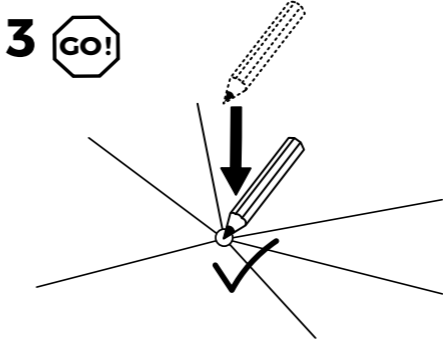
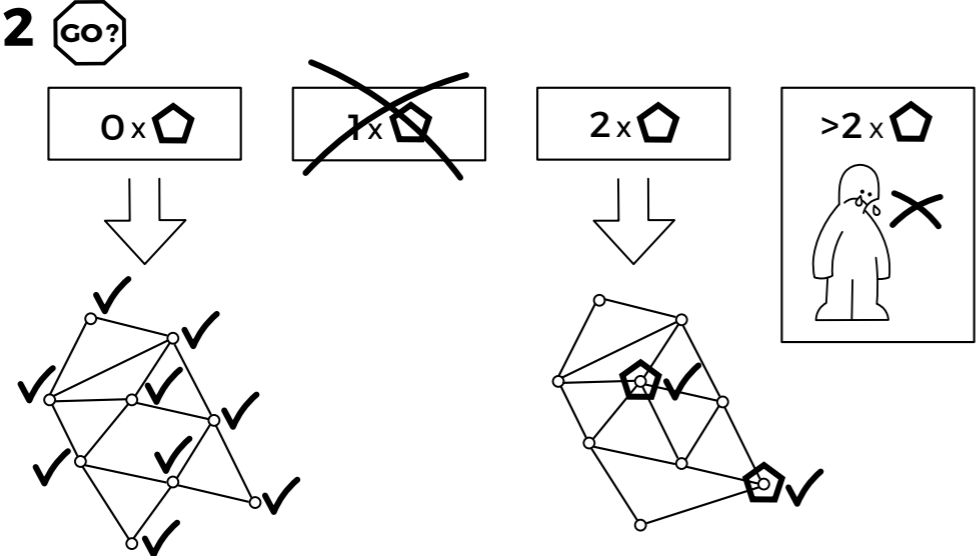
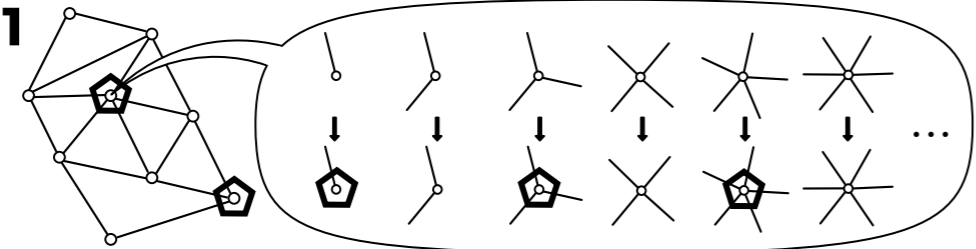
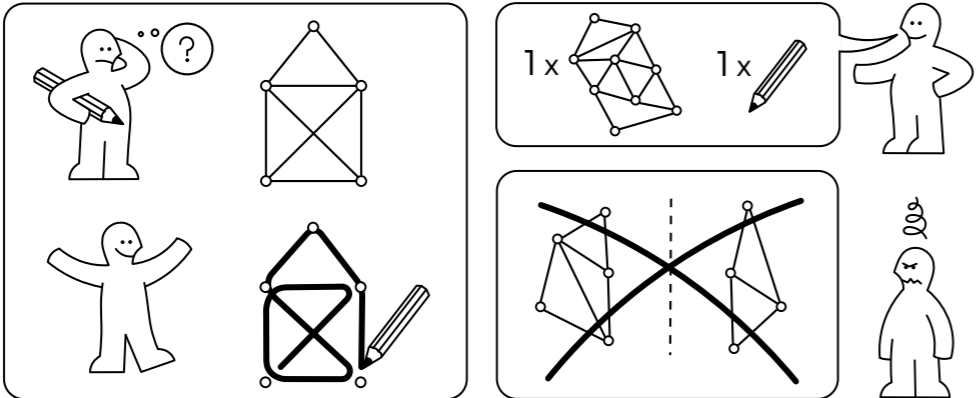
## Intelligenz nimmt zu: „Flynn-Effekt“



Type	Private
Industry	Retail
Founded	1943; 74 years ago Ålmhult, Sweden <sup>[1][2]</sup>

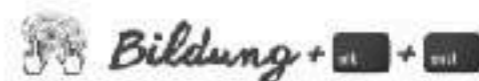
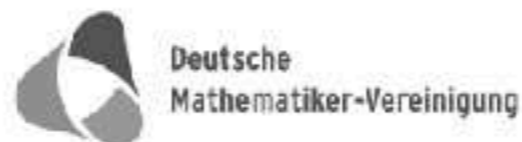


# ONE STRÖKE DRAW



**IDEA** is a series of nonverbal algorithm assembly instructions by [Sándor P. Fekete](#), [Sebastian Morr](#), and [Sebastian Stiller](#). They were originally created for Sándor's [algorithms and datastructures lecture](#) at TU Braunschweig, but we hope they will be useful in all sorts of context. We publish them here so that they can be used by teachers, students, and curious people alike. Visit the [about page](#) to learn more.

ZEIT  ONLINE



life hacker

BOINGBOING

microsiervos



KOTTKE.ORG

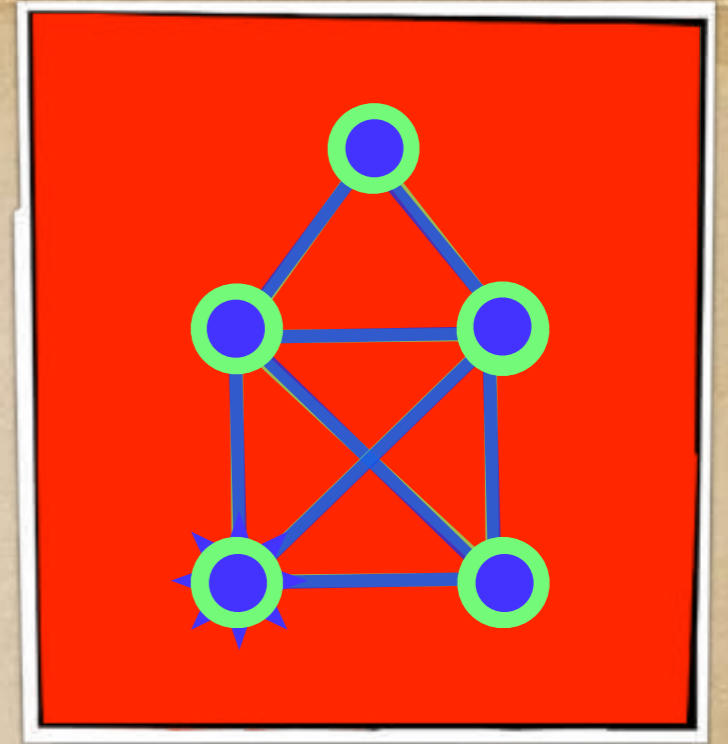
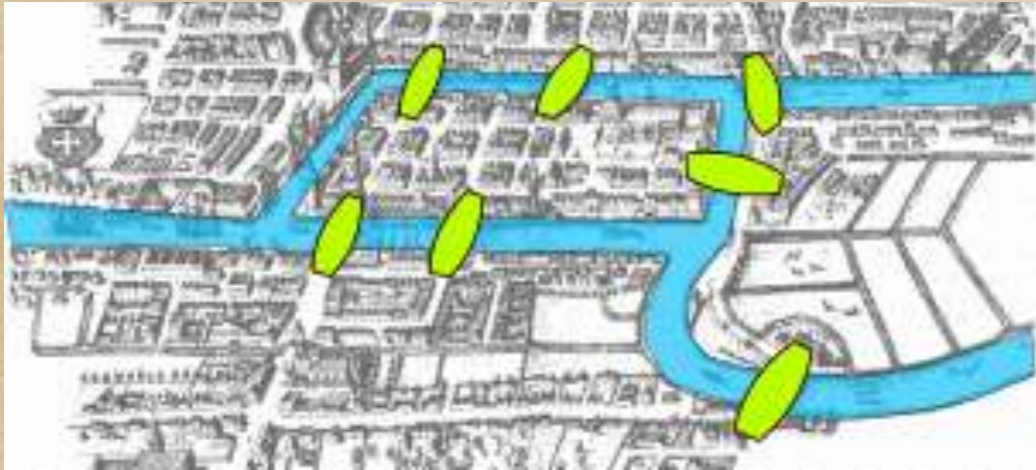


# Euler Time!



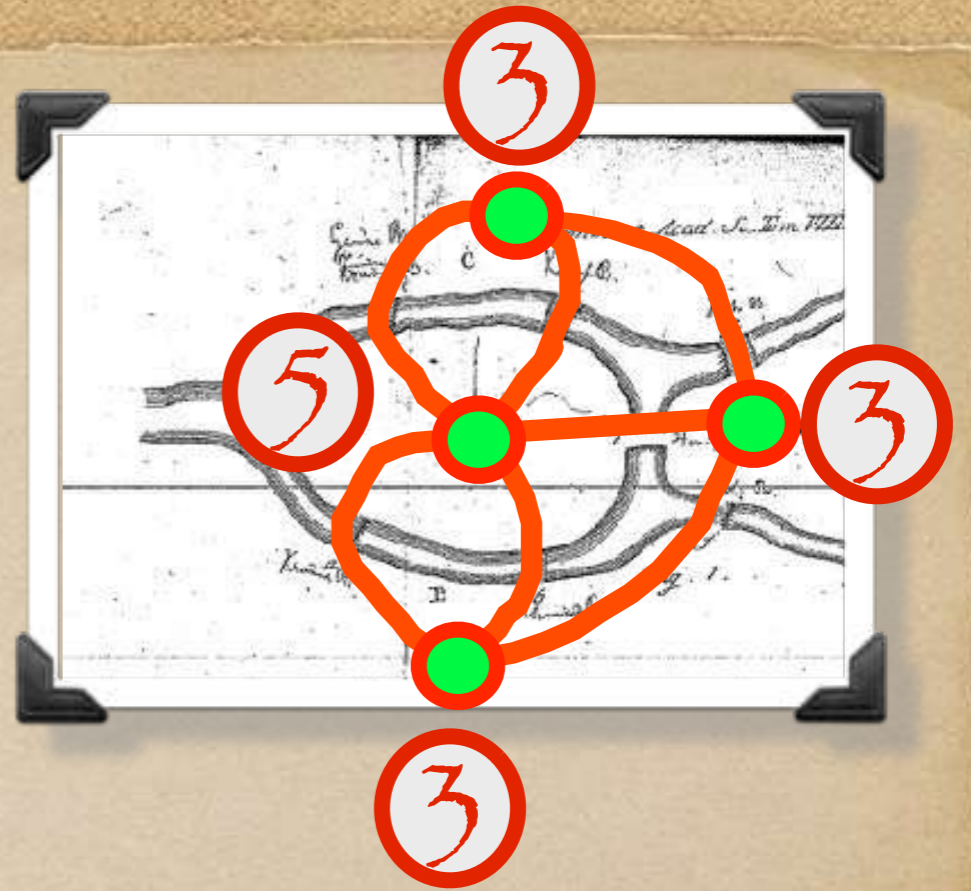
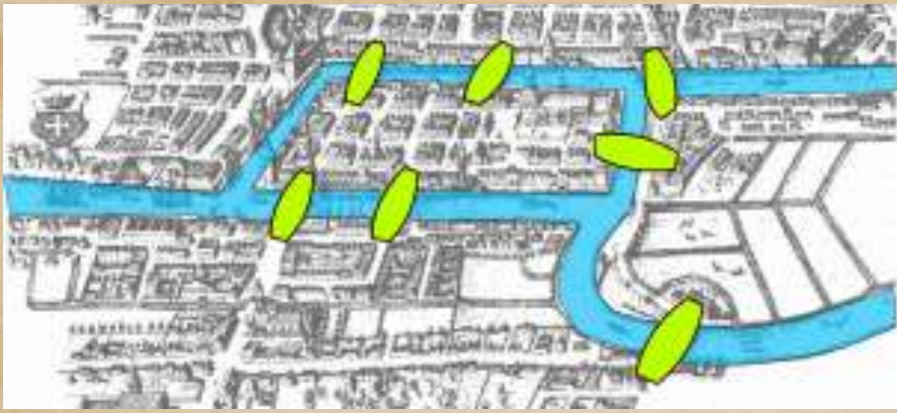


Some citizens of Königsberg  
Were walking on the strand  
Beside the river Pregel  
With its seven bridges spanned



“Oh Euler, come and walk with us!”  
Those burghers did beseech  
“We’ll walk the seven bridges o’er  
And pass but once by each.”





“It can’t be done” then Euler cried.  
“Here comes the Q. E. D.  
Your islands are but vertices,  
And all of odd degree!”

**You can’t walk this!**



**PROFS**  
**@TUNRTABLES**  
Eure Profs legen auf!

**18.11.2021**



**BONDI** | **THE ENDLESS SUMMER**

"THE CROWN JEWEL"

"THESE WILL REMOVAL AND/OR MAY"

ALSO KNOWN AS: FEELING

18 90