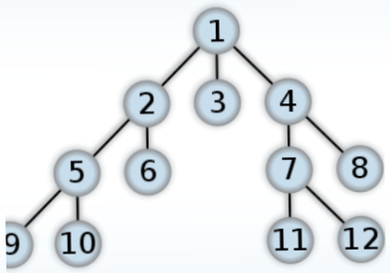



Kapitel 3.8: Laufzeit von DFS und BFS

Algorithmen und Datenstrukturen
WS 2022/23

Prof. Dr. Sándor Fekete

SATZ 3.13

Der Graphen-Scan-Algorithmus 3.7 lässt sich so implementieren, dass die Laufzeit $O(n+m)$ ist.

Satz 3.13

Der Graphen-Scan-Algorithmus 3.7 lässt sich so implementieren, dass die Laufzeit $O(n+m)$ ist.

Algorithmus 3.7

INPUT: Graph G = (V,E), Knoten s

OUTPUT: Knotenmenge $Y \subseteq V$, die von s aus erreichbar ist,

Kantenmenge $T \subseteq E$, die die Erreichbarkeit sicherstellt

1. Sei R:={s}, Y:={s}, T:=∅
2. WHILE (R≠∅) DO {
   - 2.1. Wähle $v \in R$
   - 2.2. IF (es gibt kein $w \in V \setminus Y$ mit $e=\{v,w\} \in E$) THEN
     - 2.2.1. $R := R \setminus \{v\}$
   - 2.3. ELSE {
     - 2.3.1. Wähle ein $w \in V \setminus Y$ mit $e=\{v,w\} \in E$
     - 2.3.2. Setze $R := R \cup \{w\}$, $Y := Y \cup \{w\}$, $T := T \cup \{e\}$
     - }

   }
3. STOP

Satz 3.13

Der Graphen-Scan-Algorithmus 3.7 lässt sich so implementieren, dass die Laufzeit $O(n+m)$ ist.

Algorithmus 3.7

INPUT: Graph G = (V,E), Knoten s

OUTPUT: Knotenmenge $Y \subseteq V$, die von s aus erreichbar ist,

Kantenmenge $T \subseteq E$, die die Erreichbarkeit sicherstellt

1. Sei R:={s}, Y:={s}, T:=∅
2. WHILE (R≠∅) DO {
   - 2.1. Wähle $v \in R$
   - 2.2. IF (es gibt kein $w \in V \setminus Y$ mit $e=\{v,w\} \in E$) THEN
     - 2.2.1. $R := R \setminus \{v\}$
   - 2.3. ELSE {
     - 2.3.1. Wähle ein $w \in V \setminus Y$ mit $e=\{v,w\} \in E$
     - 2.3.2. Setze $R := R \cup \{w\}$, $Y := Y \cup \{w\}$, $T := T \cup \{e\}$

   }
3. STOP

Satz 3.13

Der Graphen-Scan-Algorithmus 3.7 lässt sich so implementieren, dass die Laufzeit $O(n+m)$ ist.

Algorithmus 3.7

INPUT: Graph G = (V,E), Knoten s

OUTPUT: Knotenmenge $Y \subseteq V$, die von s aus erreichbar ist,

Kantenmenge $T \subseteq E$, die die Erreichbarkeit sicherstellt

1. Sei R:={s}, Y:={s}, T:=∅
2. WHILE (R≠∅) DO {
   - 2.1. Wähle $v \in R$
   - 2.2. IF (es gibt kein $w \in V \setminus Y$ mit $e=\{v,w\} \in E$) THEN
     - 2.2.1. $R := R \setminus \{v\}$
   - 2.3. ELSE {
     - 2.3.1. Wähle ein $w \in V \setminus Y$ mit $e=\{v,w\} \in E$
     - 2.3.2. Setze $R := R \cup \{w\}$, $Y := Y \cup \{w\}$, $T := T \cup \{e\}$

   }

3. STOP

Adjazenzliste!

Satz 3.13

Der Graphen-Scan-Algorithmus 3.7 lässt sich so implementieren, dass die Laufzeit $O(n+m)$ ist.

Algorithmus 3.7

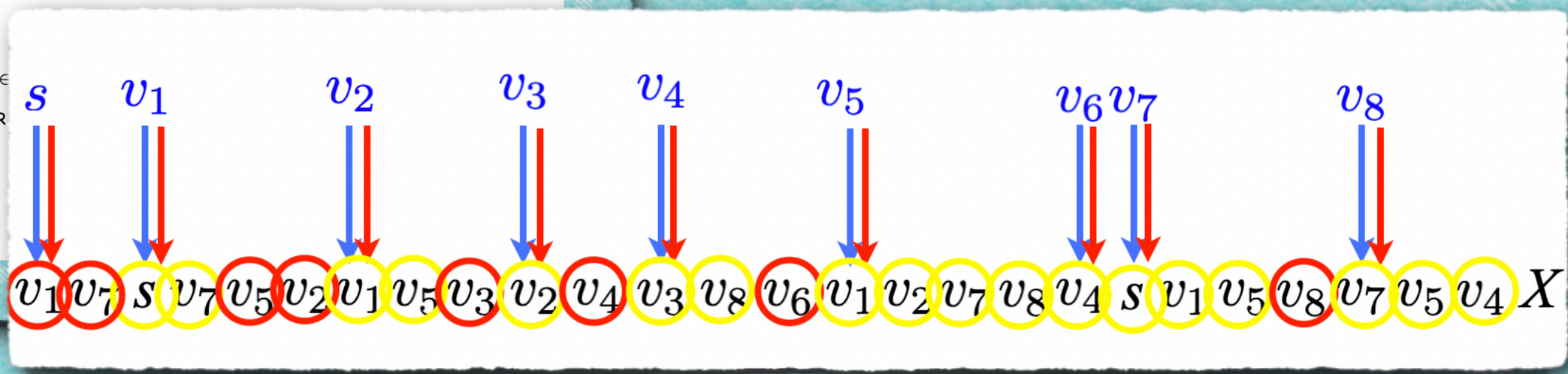
INPUT: Graph G = (V,E), Knoten s

OUTPUT: Knotenmenge $Y \subseteq V$, die von s aus erreichbar ist,

Kantenmenge $T \subseteq E$, die die Erreichbarkeit sicherstellt

1. Sei R:={s}, Y:={s}, T:=∅
2. WHILE (R≠∅) DO {
   - 2.1. Wähle v∈ R
   - 2.2. IF (es gibt kein w∈ V\Y mit e={v,w}∈E) THEN
     - 2.2.1. R:=R\{v}
   - 2.3. ELSE {
     - 2.3.1.Wähle ein w∈
     - 2.3.2. Setze R := R
     - }
   - }
3. STOP

Adjazenzliste!



Satz 3.13

Der Graphen-Scan-Algorithmus 3.7 lässt sich so implementieren, dass die Laufzeit $O(n+m)$ ist.

Algorithmus 3.7

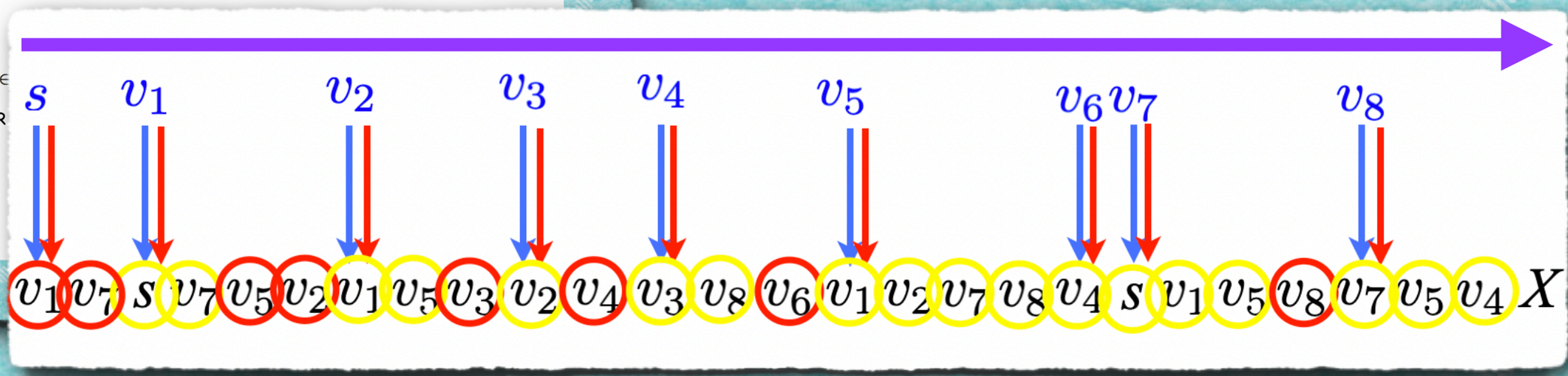
INPUT: Graph G = (V,E), Knoten s

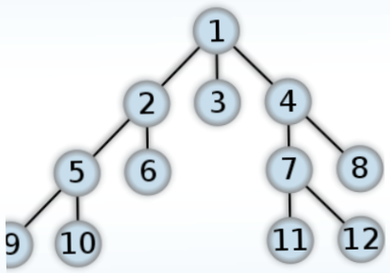
OUTPUT: Knotenmenge $Y \subseteq V$, die von s aus erreichbar ist,

Kantenmenge $T \subseteq E$, die die Erreichbarkeit sicherstellt

1. Sei R:={s}, Y:={s}, T:=∅
2. WHILE (R≠∅) DO {
   - 2.1. Wähle v∈ R
   - 2.2. IF (es gibt kein w∈ V\Y mit e={v,w}∈E) THEN
     - 2.2.1. R:=R\{v}
   - 2.3. ELSE {
     - 2.3.1.Wähle ein w∈
     - 2.3.2. Setze R := R
     - }
   - }
3. STOP

Adjazenzliste!





Kapitel 3.9: Eigenschaften von DFS und BFS

Algorithmen und Datenstrukturen
WS 2022/23

Prof. Dr. Sándor Fekete

Algorithmus 3.17

INPUT: Graph G = (V,E), Knoten s

OUTPUT: Knotenmenge $Y \subseteq V$, die von s aus erreichbar ist,

Kantenmenge $T \subseteq E$, die die Erreichbarkeit sicherstellt

1. Sei R:={s}, Y:={s}, T:=∅

2. WHILE (R≠∅) DO {

   2.1. wähle Element $v \in R$

   2.2. IF (es gibt kein w V\Y mit $e=\{v,w\} \in E$) THEN

      2.2.1. R:=R\{v}

   2.3. ELSE {

      2.3.1. wähle ein $w \in V\backslash R$ mit $e=\{v,w\} \in E$;

      2.3.2. setze R := R u {w}, Y := Y u {w}, T := T u {e};

      }

}

Algorithmus 3.17

INPUT: Graph G = (V,E), Knoten s

OUTPUT: Knotenmenge $Y \subseteq V$, die von s aus erreichbar ist,
für jeden Knoten $v \in Y$ die Länge l(v) eines kürzesten s-v-Weges,
Kantenmenge $T \subseteq E$, die die Erreichbarkeit sicherstellt

1. Sei R:={s}, Y:={s}, T:=∅
2. WHILE (R≠∅) DO {
   - 2.1. wähle Element $v \in R$
   - 2.2. IF (es gibt kein w V\Y mit $e=\{v,w\} \in E$) THEN
     - 2.2.1. R:=R\{v}
   - 2.3. ELSE {
     - 2.3.1. wähle ein $w \in V\backslash R$ mit $e=\{v,w\} \in E$;
     - 2.3.2. setze R := R u {w}, Y := Y u {w}, T := T u {e};
     - }

}

Algorithmus 3.17

INPUT: Graph G = (V,E), Knoten s

OUTPUT: Knotenmenge $Y \subseteq V$, die von s aus erreichbar ist,

für jeden Knoten $v \in Y$ die Länge l(v) eines kürzesten s-v-Weges,

Kantenmenge $T \subseteq E$, die die Erreichbarkeit sicherstellt

1. Sei R:={s}, Y:={s}, T:=∅

2. WHILE (R≠∅) DO {

    2.1. wähle Element $v \in R$

    2.2. IF (es gibt kein w V\Y mit $e=\{v,w\} \in E$) THEN

        2.2.1. R:=R\{v}

    2.3. ELSE {

        2.3.1. wähle ein $w \in V\backslash R$ mit $e=\{v,w\} \in E$;

        2.3.2. setze R := R u {w}, Y := Y u {w}, T := T u {e};

        }

}

Algorithmus 3.17

INPUT: Graph G = (V,E), Knoten s

OUTPUT: Knotenmenge $Y \subseteq V$, die von s aus erreichbar ist,

für jeden Knoten $v \in Y$ die Länge l(v) eines kürzesten s-v-Weges,

Kantenmenge $T \subseteq E$, die die Erreichbarkeit sicherstellt

1. Sei R:={s}, Y:={s}, T:=∅, l(s):=0
2. WHILE (R≠∅) DO {
   - 2.1. wähle Element $v \in R$
   - 2.2. IF (es gibt kein w V\Y mit $e=\{v,w\} \in E$) THEN
     - 2.2.1. R:=R\{v}
   - 2.3. ELSE {
     - 2.3.1. wähle ein $w \in V\backslash R$ mit $e=\{v,w\} \in E$;
     - 2.3.2. setze R := R u {w}, Y := Y u {w}, T := T u {e};
     - 2.3.3. setze l(w):=l(v)+1
     - }

}

Algorithmus 3.17

INPUT: Graph G = (V,E), Knoten s

OUTPUT: Knotenmenge $Y \subseteq V$, die von s aus erreichbar ist,

für jeden Knoten $v \in Y$ die Länge l(v) eines kürzesten s-v-Weges,

Kantenmenge $T \subseteq E$, die die Erreichbarkeit sicherstellt

1. Sei R:={s}, Y:={s}, T:=∅, l(s):=0
2. WHILE (R≠∅) DO {
   - 2.1. wähle Element $v \in R$
   - 2.2. IF (es gibt kein w V\Y mit $e=\{v,w\} \in E$) THEN
     - 2.2.1. R:=R\{v}
   - 2.3. ELSE {
     - 2.3.1. wähle ein $w \in V\backslash R$ mit $e=\{v,w\} \in E$;
     - 2.3.2. setze R := R u {w}, Y := Y u {w}, T := T u {e};
     - 2.3.3. setze l(w):=l(v)+1
     - }

}

3.9 BFS

Satz 3.18

Satz 3.18

(1) Das Verfahren 3.17 ist endlich.

Satz 3.18

(1) Das Verfahren 3.17 ist endlich.

(2) Die Laufzeit ist O(n+m).

Satz 3.18

(1) Das Verfahren 3.17 ist endlich.

(2) Die Laufzeit ist O(n+m).

(3) Am Ende ist für jeden erreichbaren Knoten $v \in Y$ die Länge eines kürzesten Weges von s nach v im Baum (Y,T) durch l(v) gegeben.

Satz 3.18

(1) Das Verfahren 3.17 ist endlich.

(2) Die Laufzeit ist O(n+m).

(3) Am Ende ist für jeden erreichbaren Knoten $v \in Y$ die Länge eines kürzesten Weges von s nach v im Baum (Y,T) durch l(v) gegeben.

(4) Am Ende ist für jeden erreichbaren Knoten $v \in Y$ die Länge eines kürzesten Weges von s nach v im Graphen (V,E) durch l(v) gegeben.

Satz 3.18

(1) Das Verfahren 3.17 ist endlich.

(2) Die Laufzeit ist O(n+m).

(3) Am Ende ist für jeden erreichbaren Knoten $v \in Y$ die Länge eines kürzesten Weges von s nach v im Baum (Y,T) durch l(v) gegeben.

(4) Am Ende ist für jeden erreichbaren Knoten $v \in Y$ die Länge eines kürzesten Weges von s nach v im Graphen (V,E) durch l(v) gegeben.

Beweis:

Satz 3.18

(1) ***Das Verfahren 3.17 ist endlich.***

(2) ***Die Laufzeit ist O(n+m).***

(3) ***Am Ende ist für jeden erreichbaren Knoten $v \in Y$ die Länge eines kürzesten Weges von s nach v im Baum (Y,T) durch l(v) gegeben.***

(4) ***Am Ende ist für jeden erreichbaren Knoten $v \in Y$ die Länge eines kürzesten Weges von s nach v im Graphen (V,E) durch l(v) gegeben.***

Beweis:

(1) **Wie für Algorithmus 3.7 gelten alle Eigenschaften. zusätzlich ist für jeden Knoten $v \in Y$ per Induktion, der Wert l(v) tatsächlich definiert.**

Satz 3.18

(1) Das Verfahren 3.17 ist endlich.

(2) Die Laufzeit ist O(n+m).

(3) Am Ende ist für jeden erreichbaren Knoten $v \in Y$ die Länge eines kürzesten Weges von s nach v im Baum (Y,T) durch l(v) gegeben.

(4) Am Ende ist für jeden erreichbaren Knoten $v \in Y$ die Länge eines kürzesten Weges von s nach v im Graphen (V,E) durch l(v) gegeben.

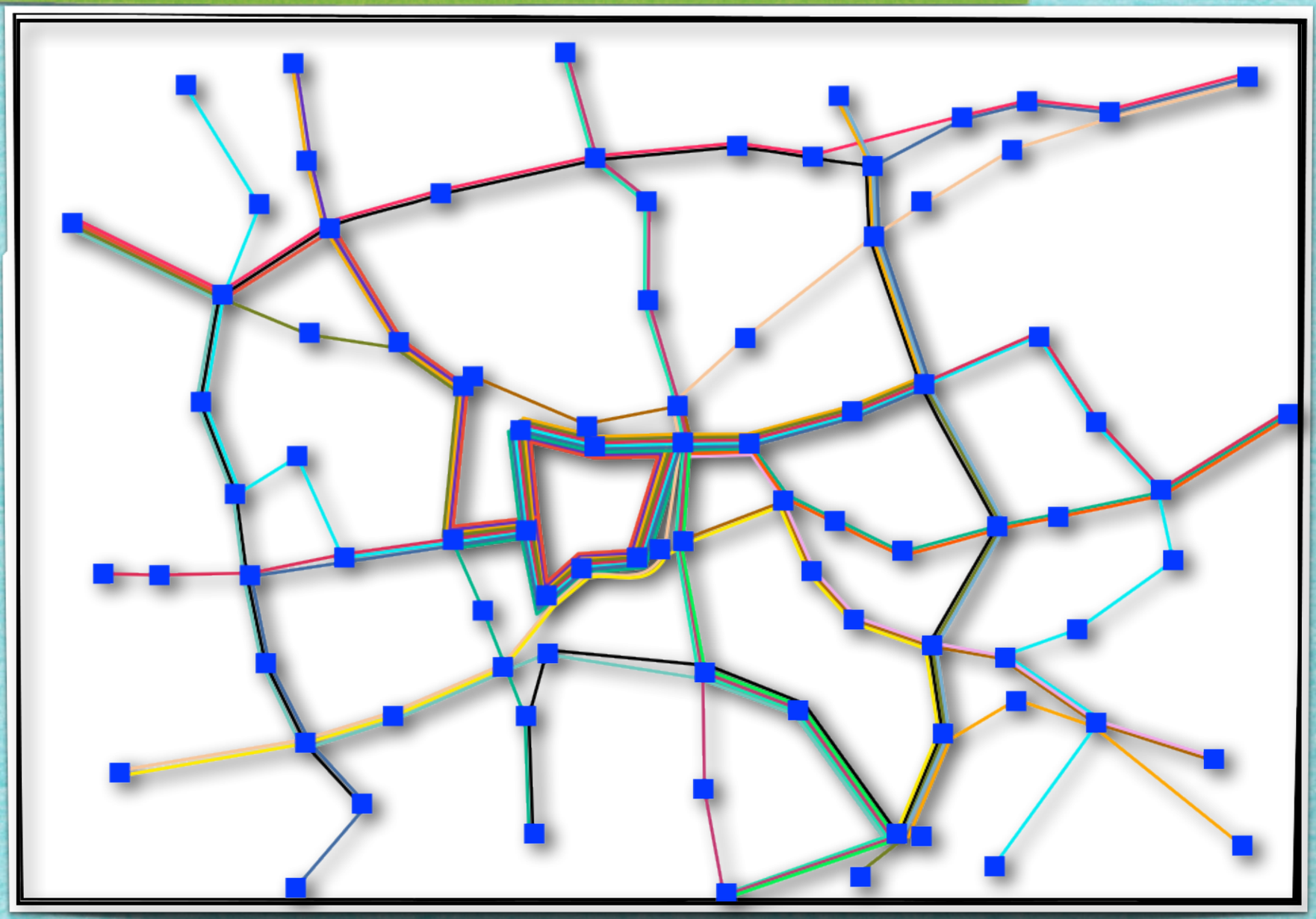
Beweis:

(1) **Wie für Algorithmus 3.7 gelten alle Eigenschaften. zusätzlich ist für jeden Knoten $v \in Y$ per Induktion, der Wert l(v) tatsächlich definiert.**

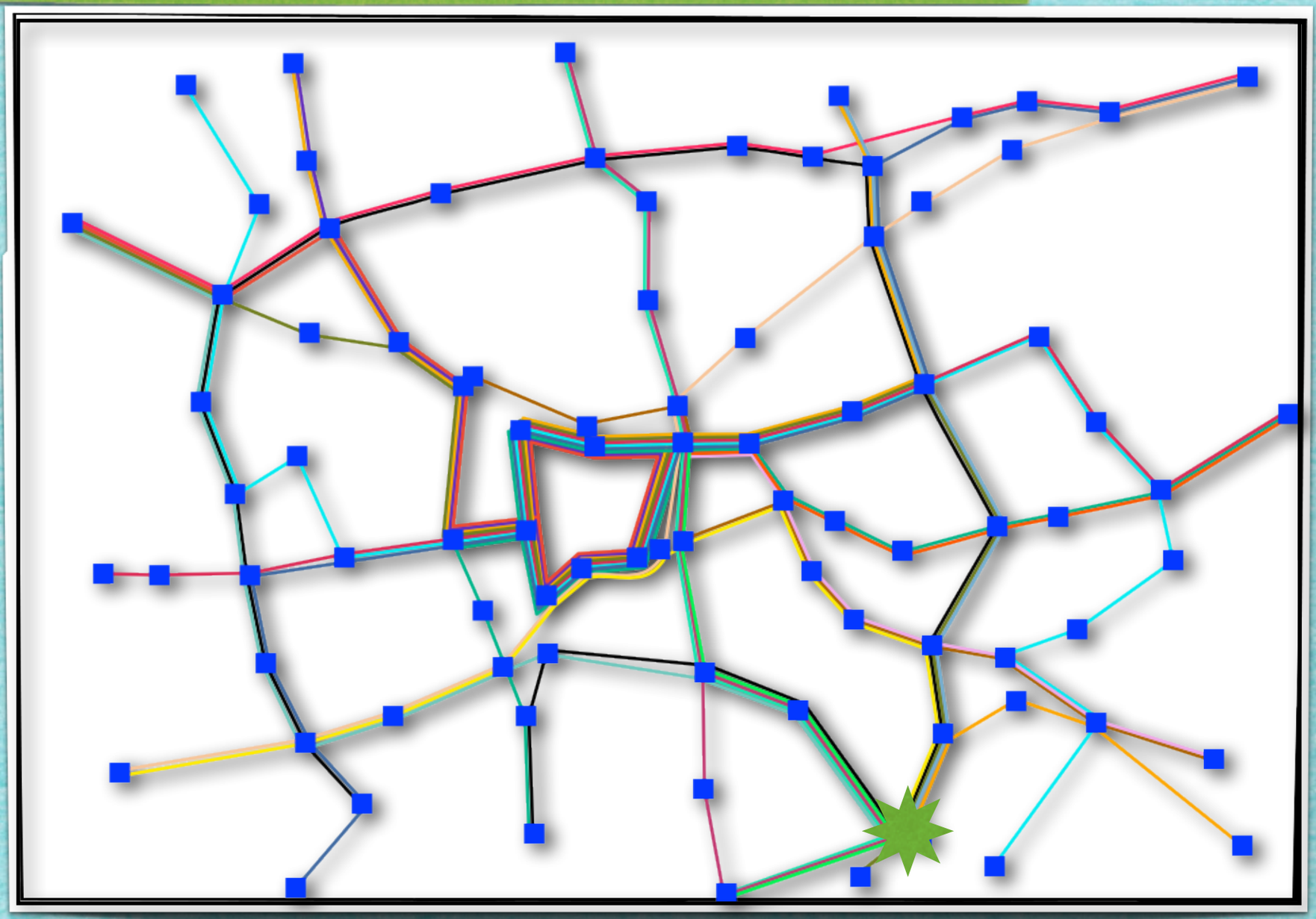
(2) **Die Laufzeit bleibt von Algorithmus 3.7 erhalten.**

Wellenreiten in Graphen

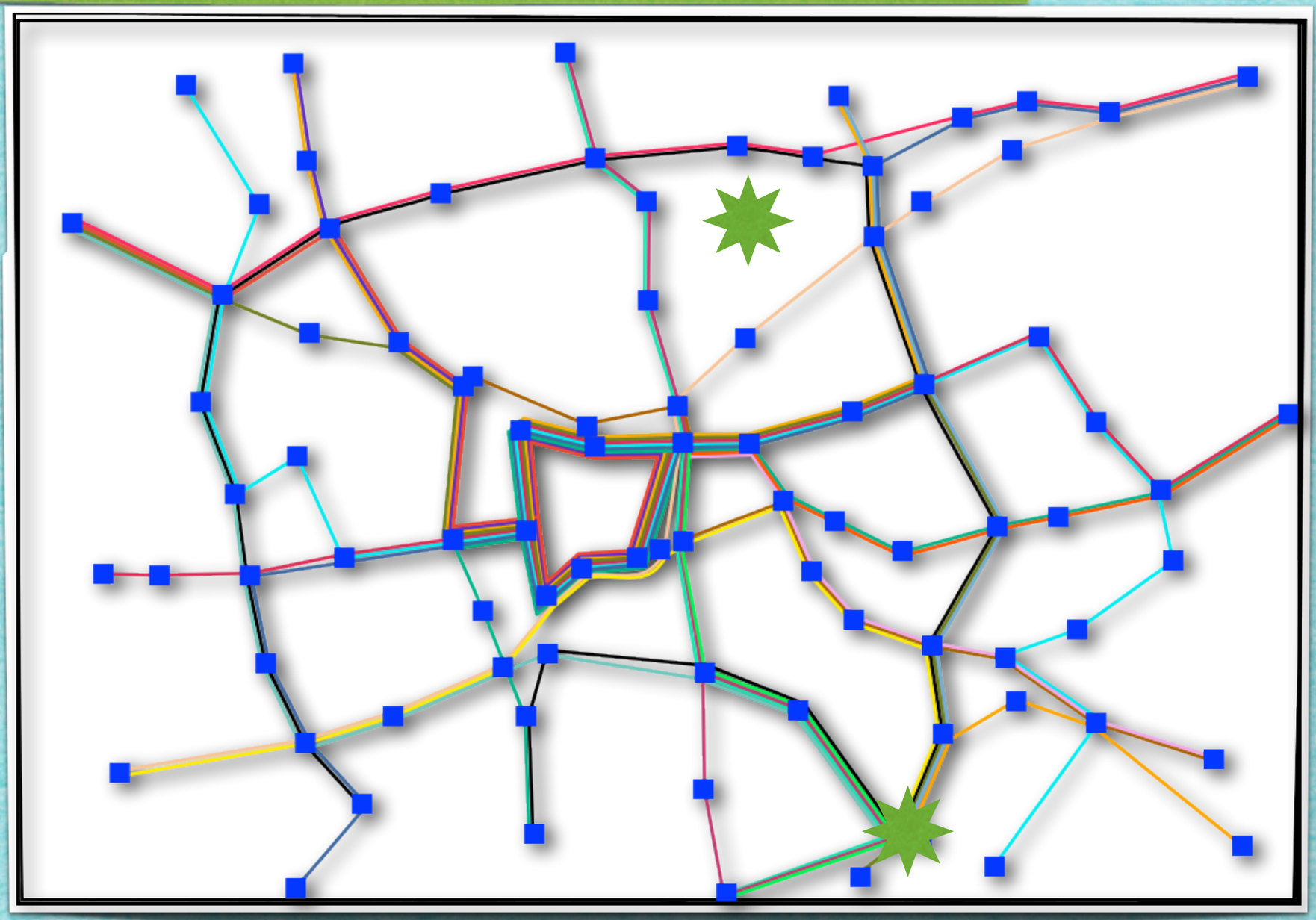
Wellenreiten in Graphen



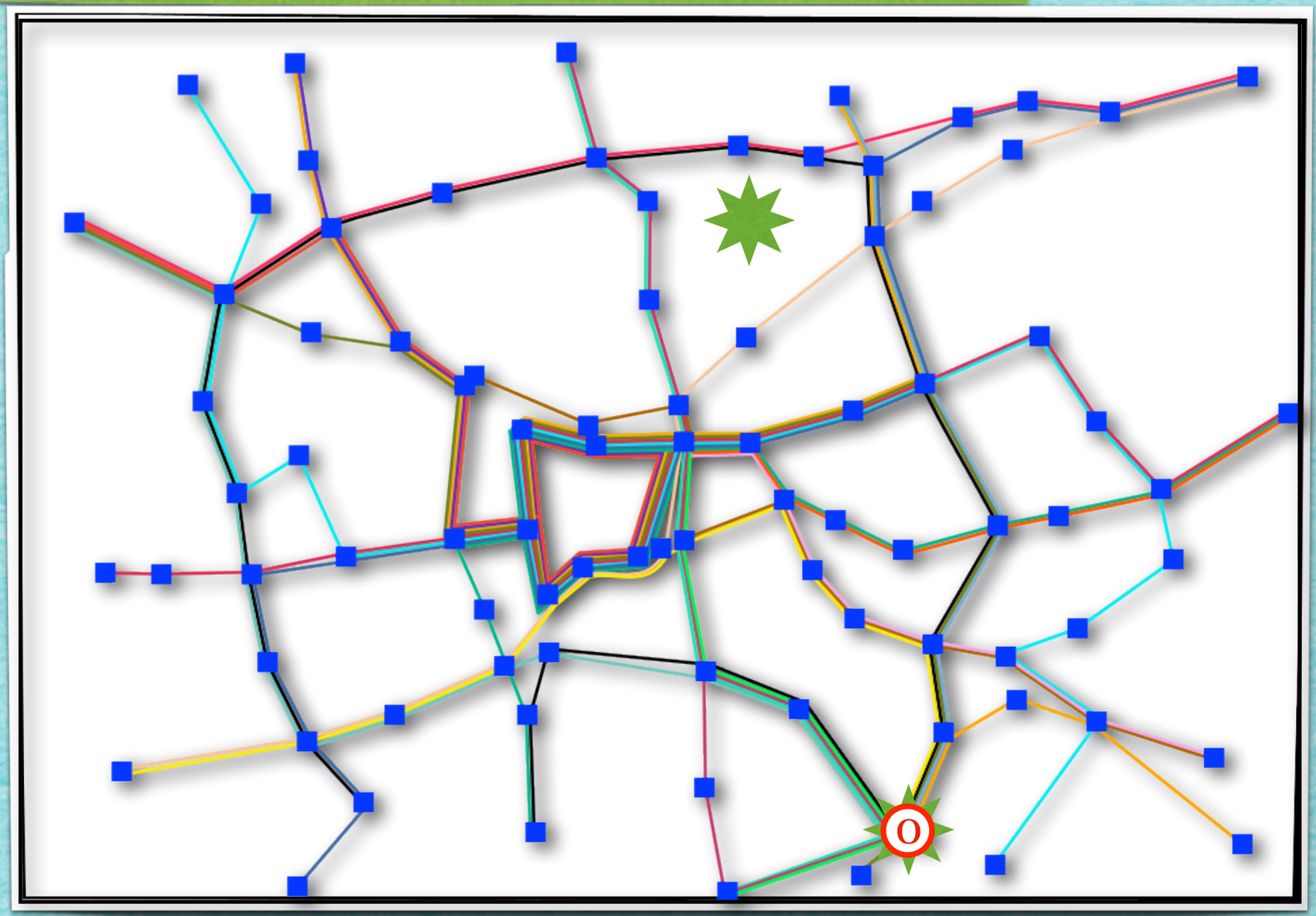
Wellenreiten in Graphen



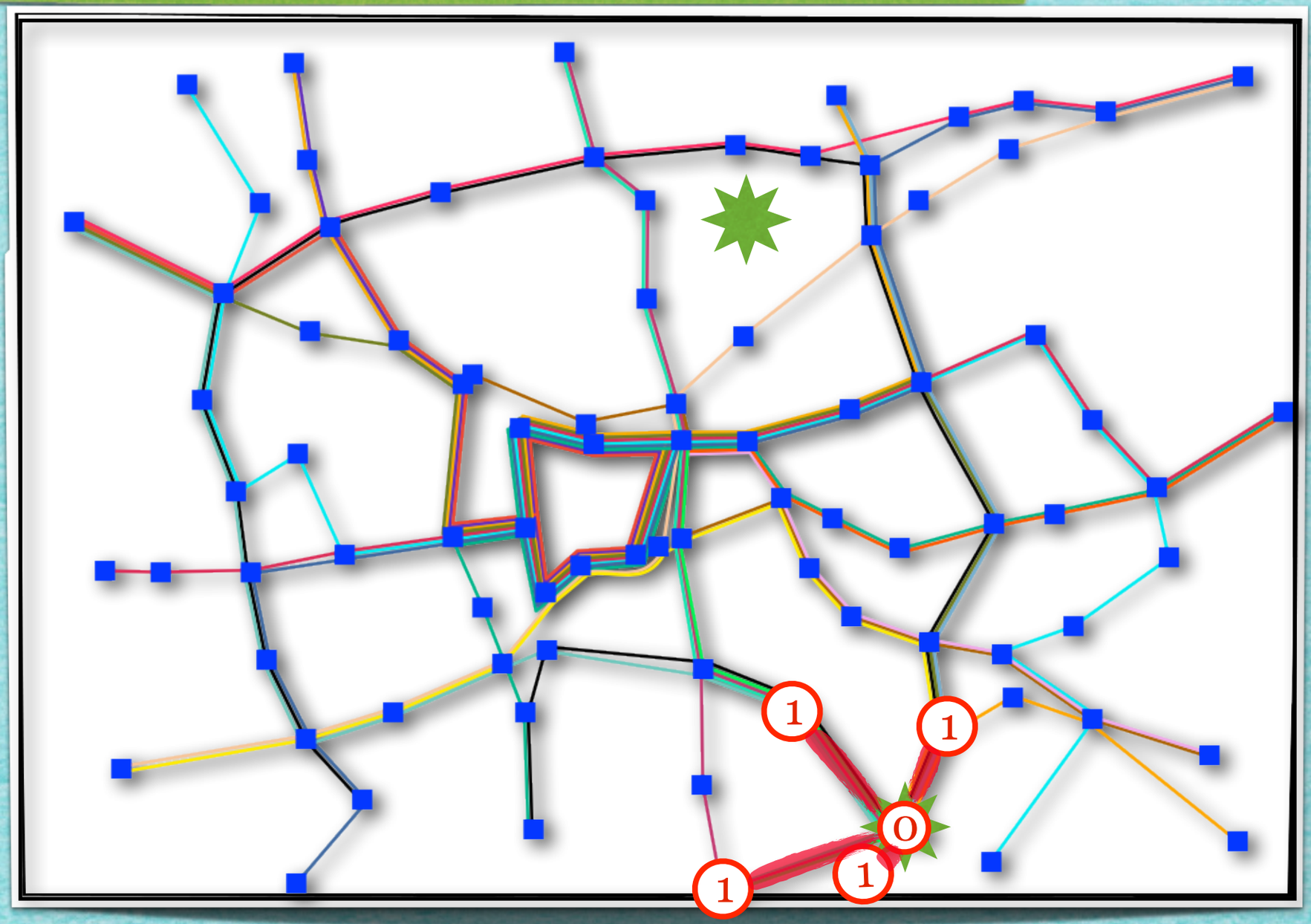
Wellenreiten in Graphen



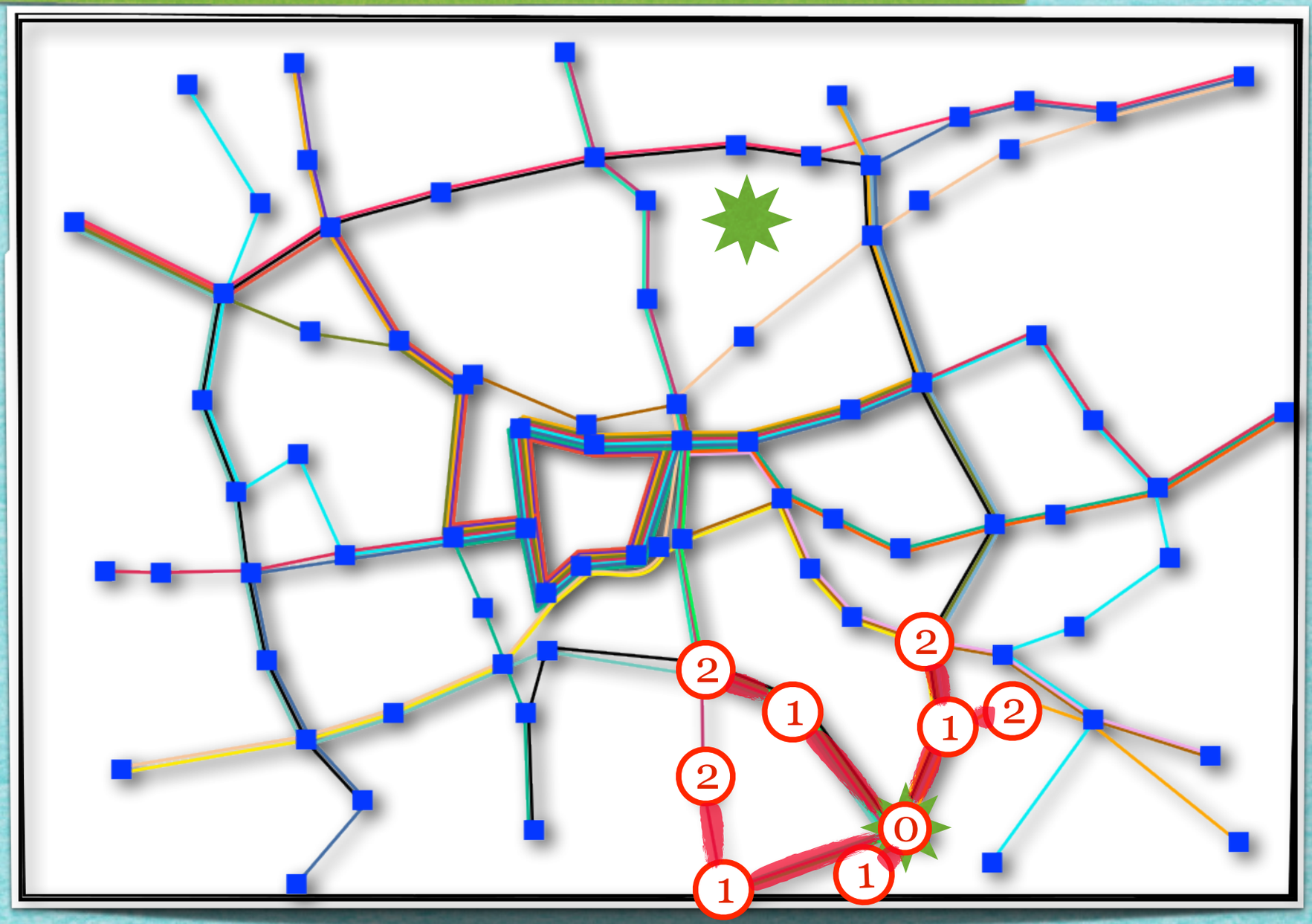
Wellenreiten in Graphen



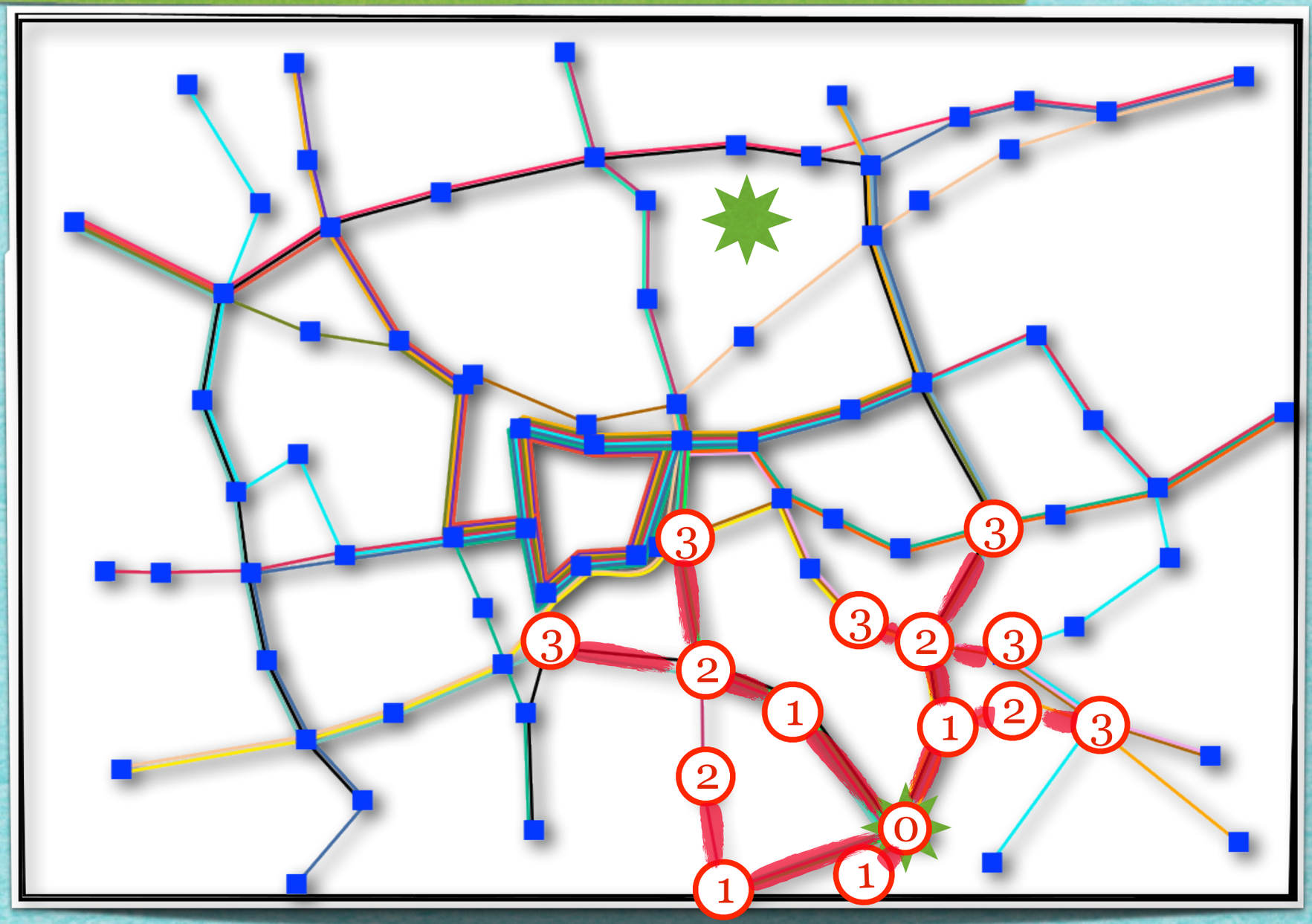
Wellenreiten in Graphen



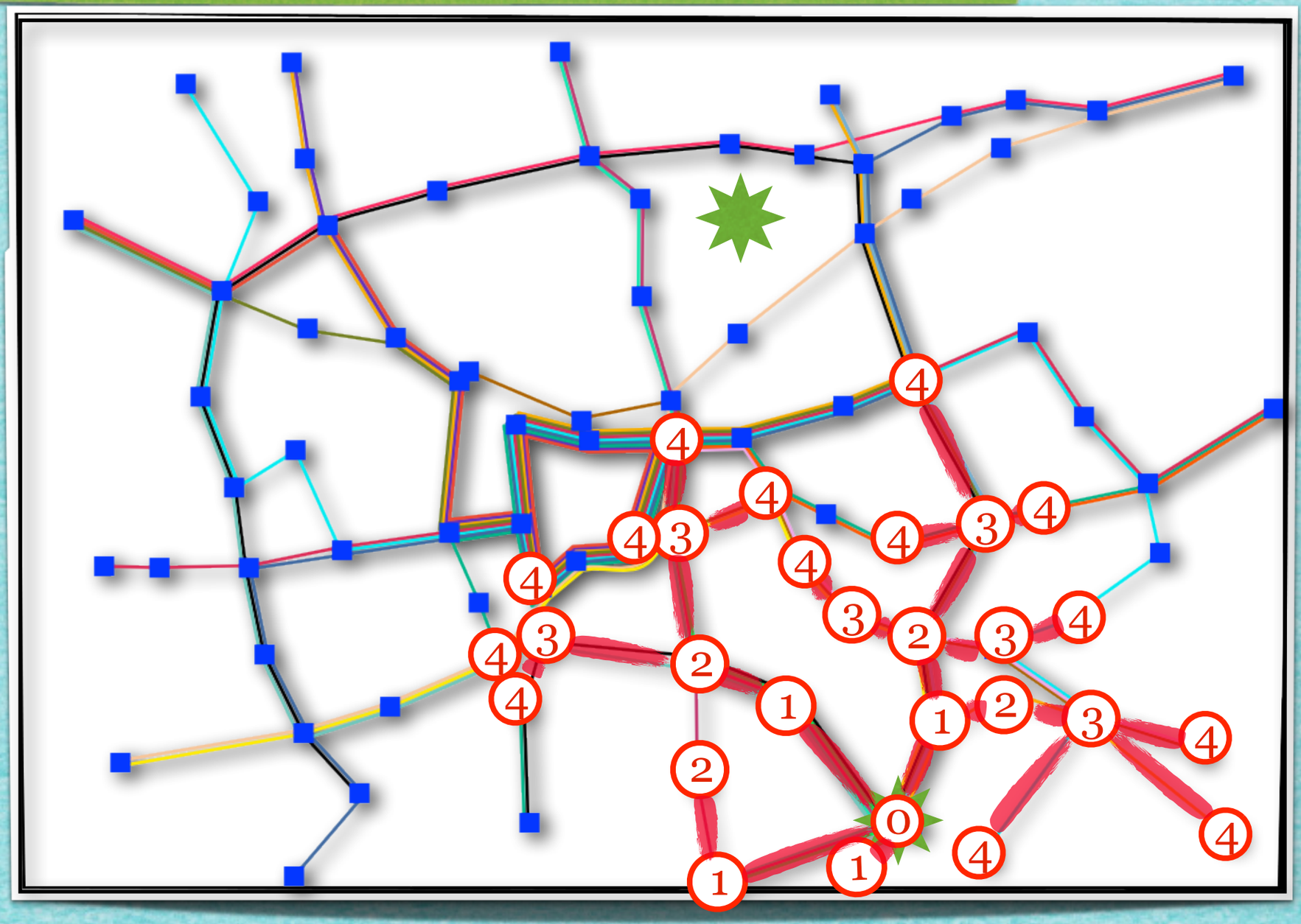
Wellenreiten in Graphen



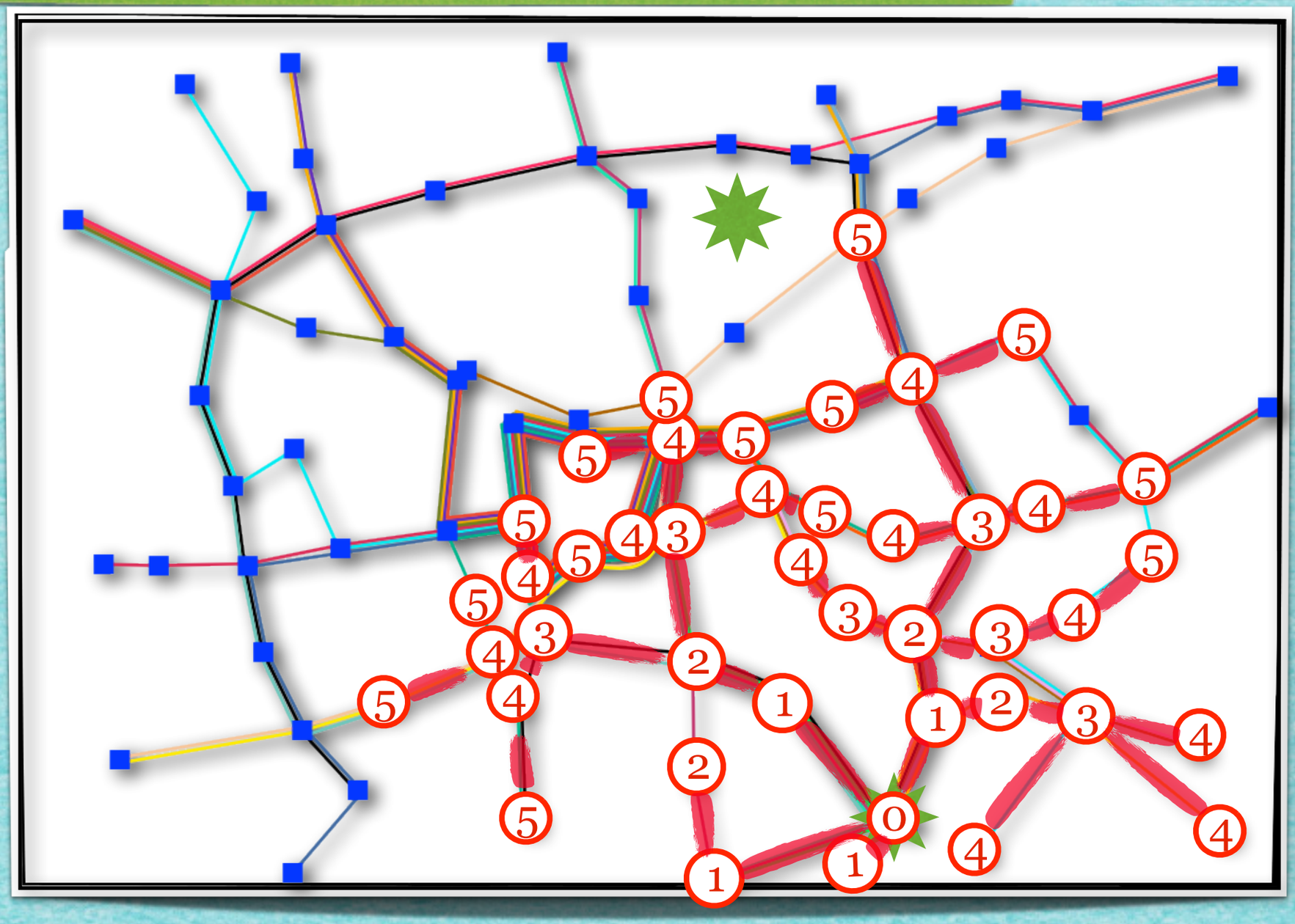
Wellenreiten in Graphen



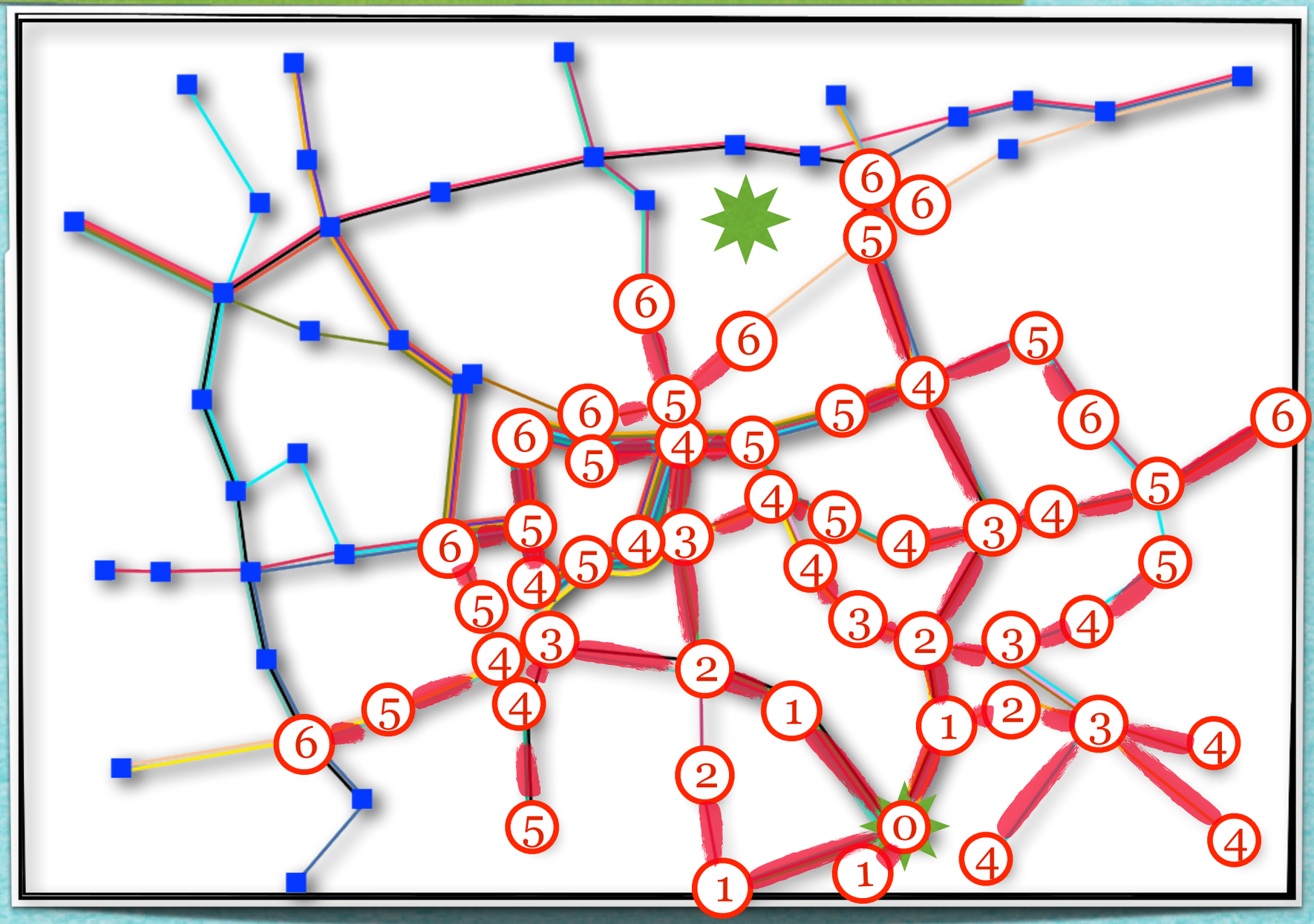
Wellenreiten in Graphen



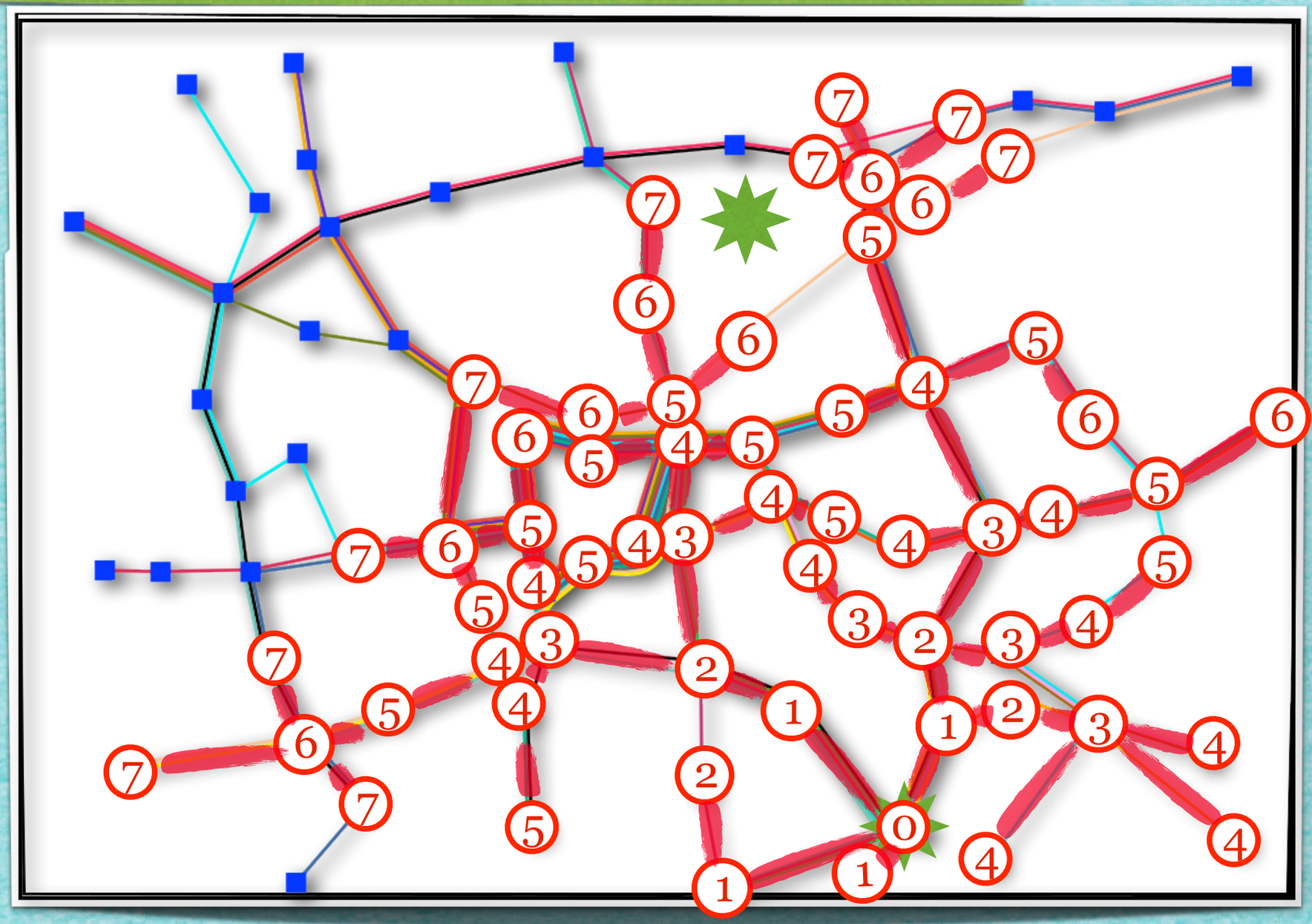
Wellenreiten in Graphen



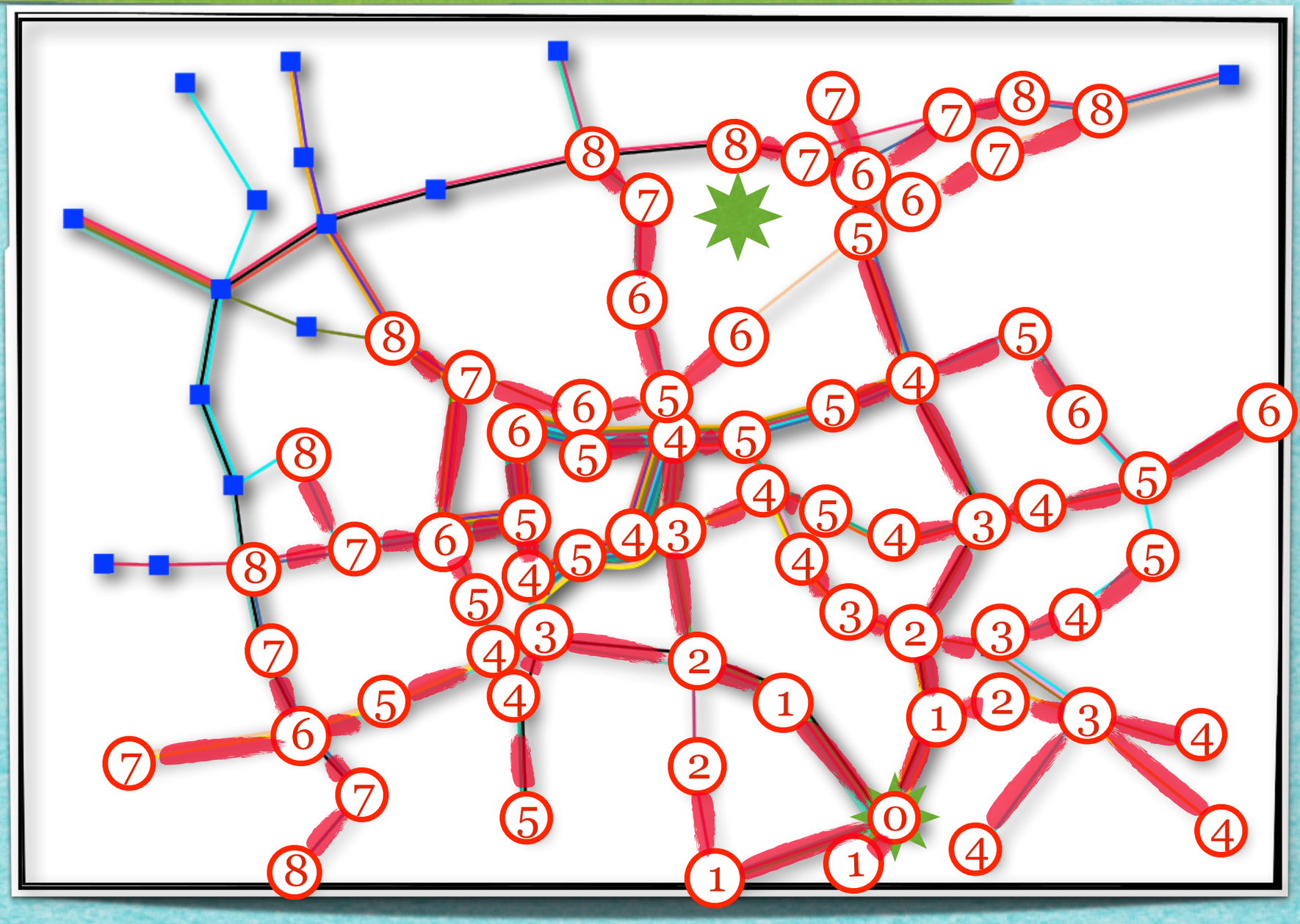
Wellenreiten in Graphen



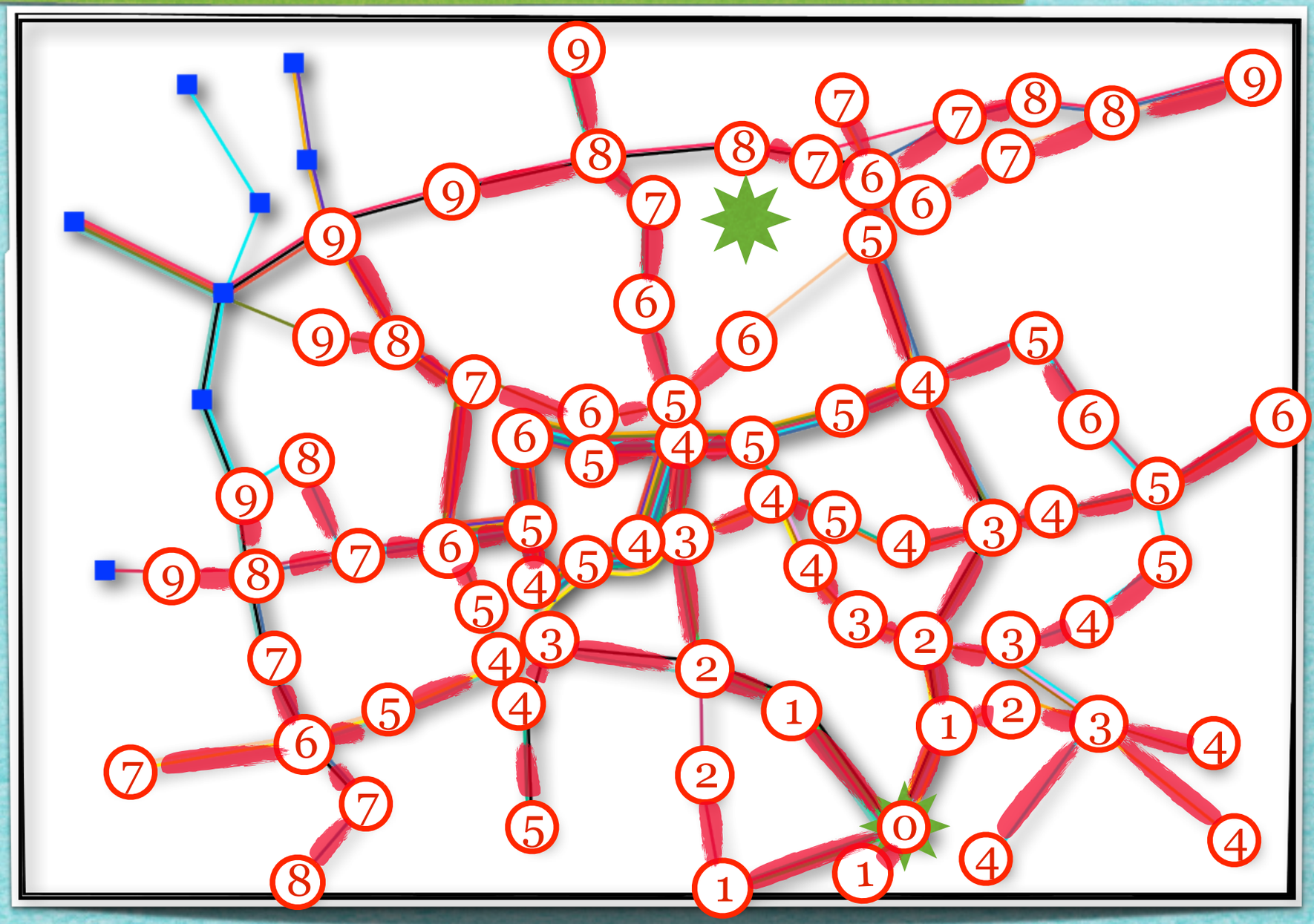
Wellenreiten in Graphen



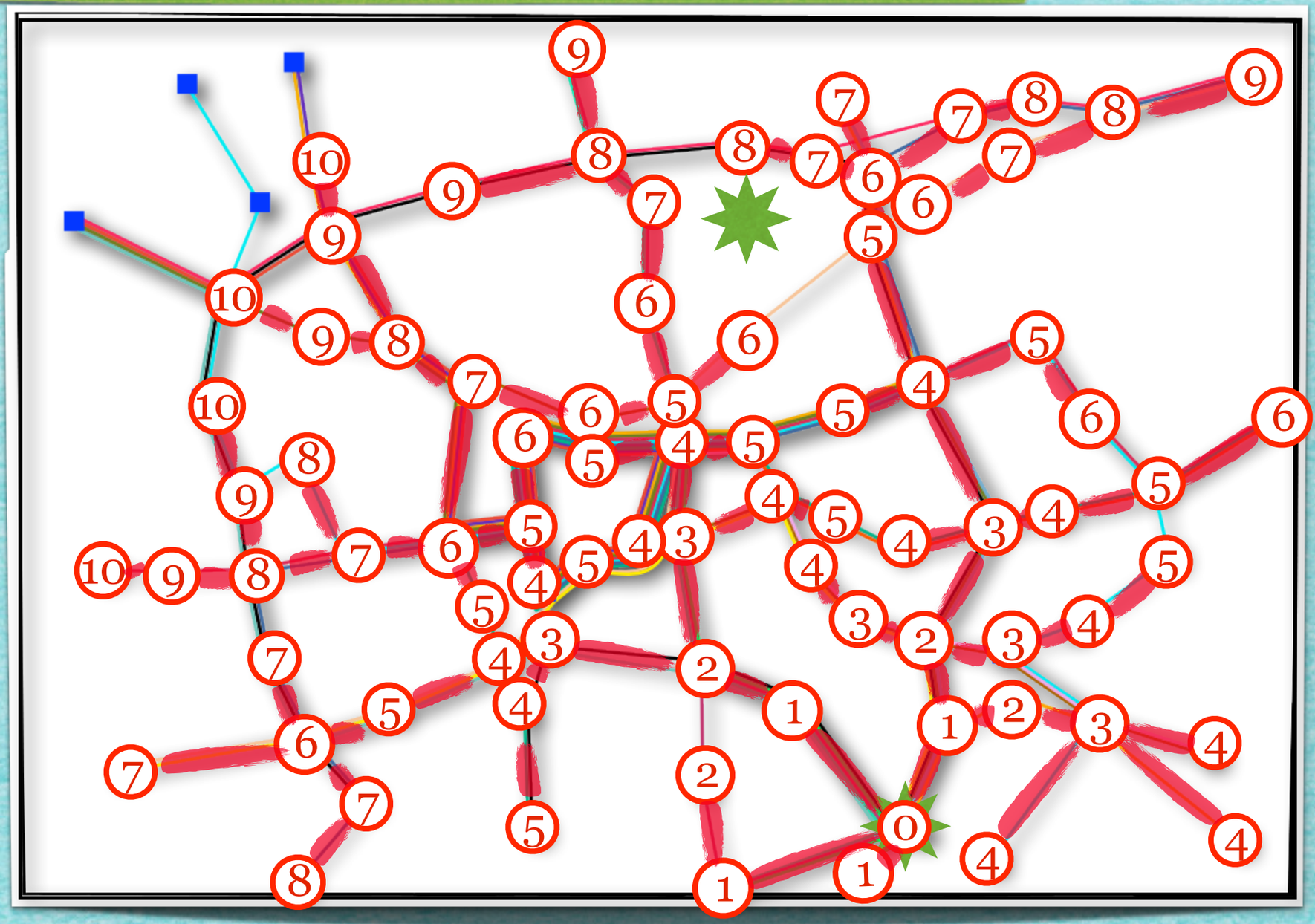
Wellenreiten in Graphen



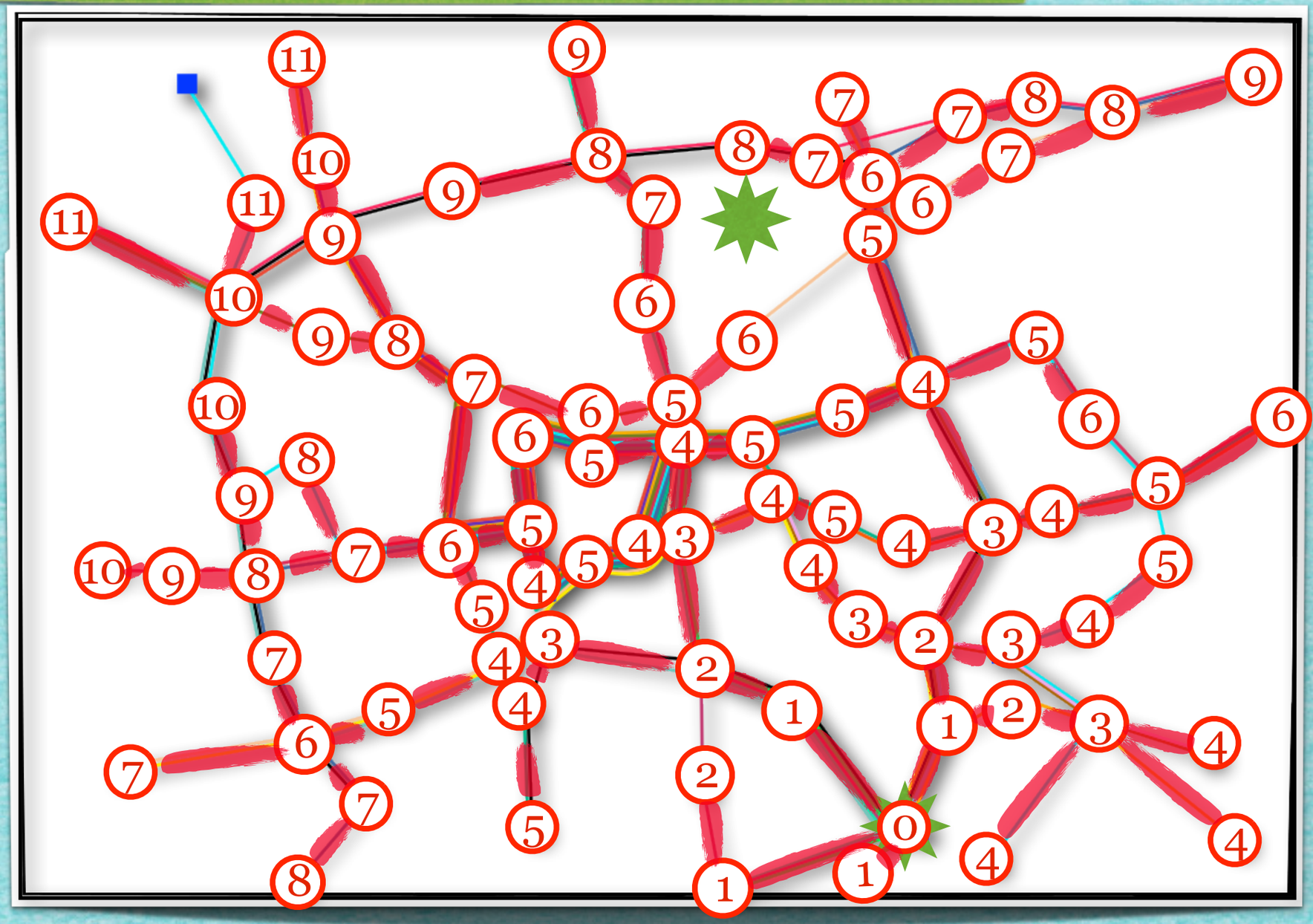
Wellenreiten in Graphen



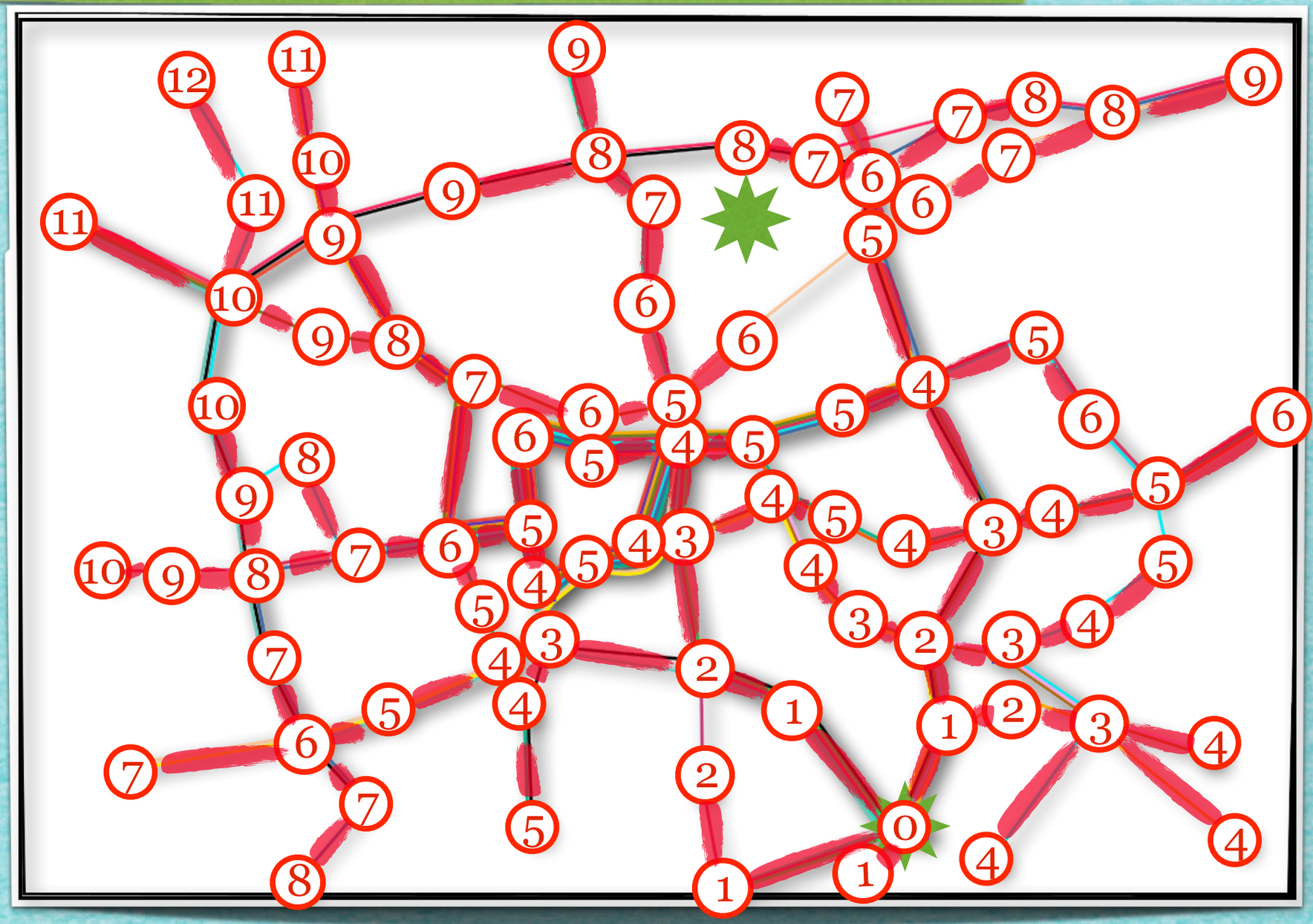
Wellenreiten in Graphen



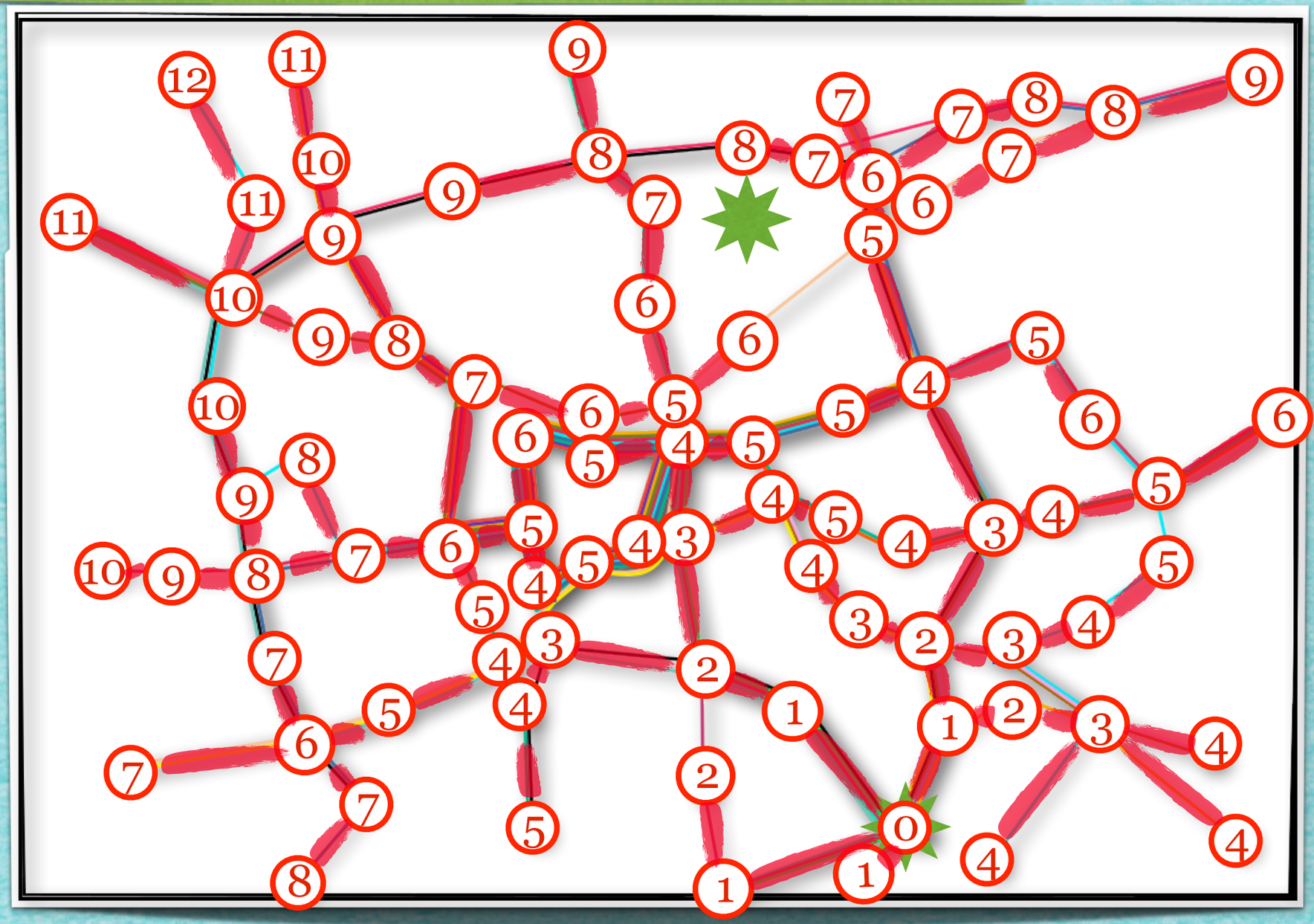
Wellenreiten in Graphen



Wellenreiten in Graphen

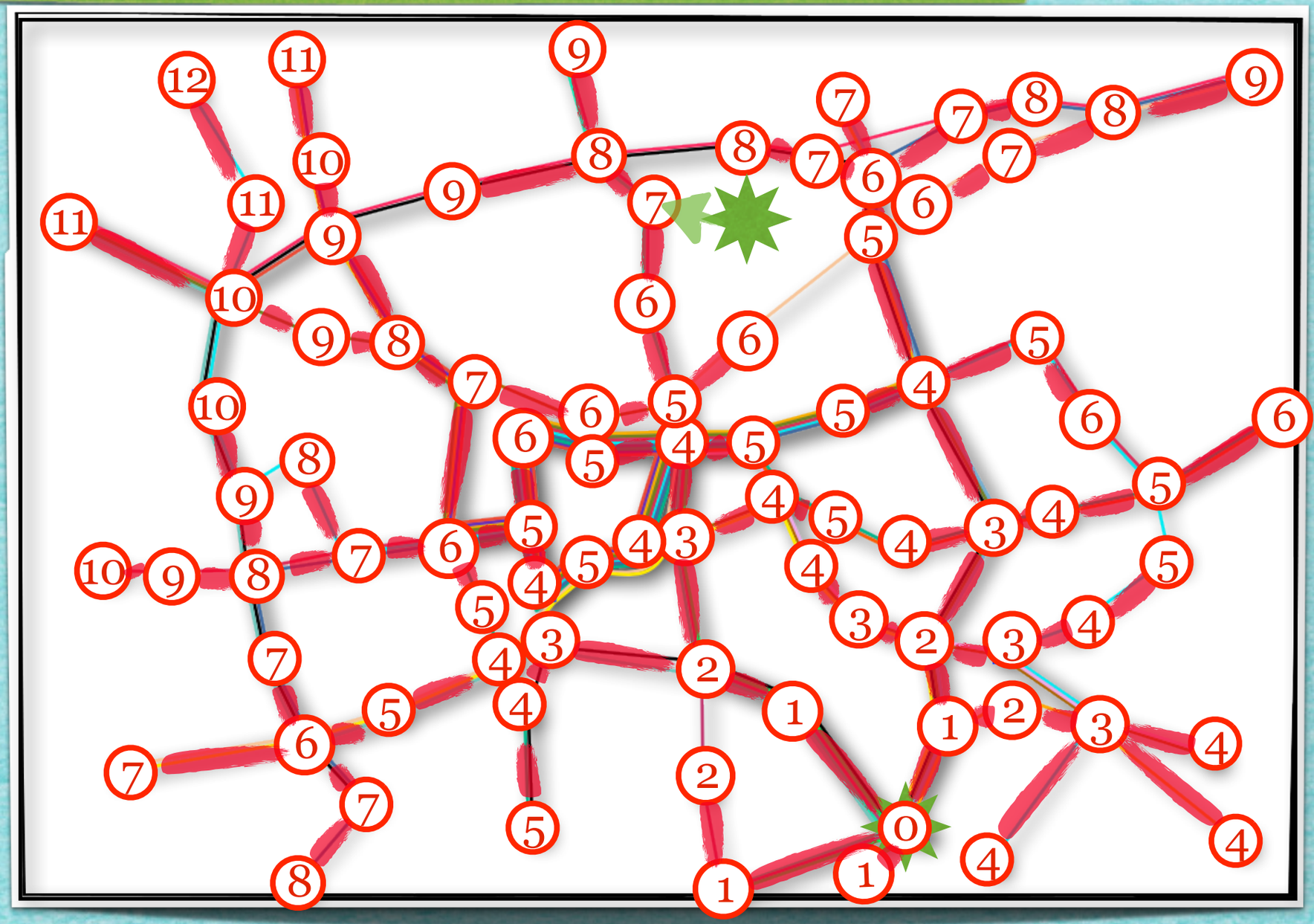


Wellenreiten in Graphen



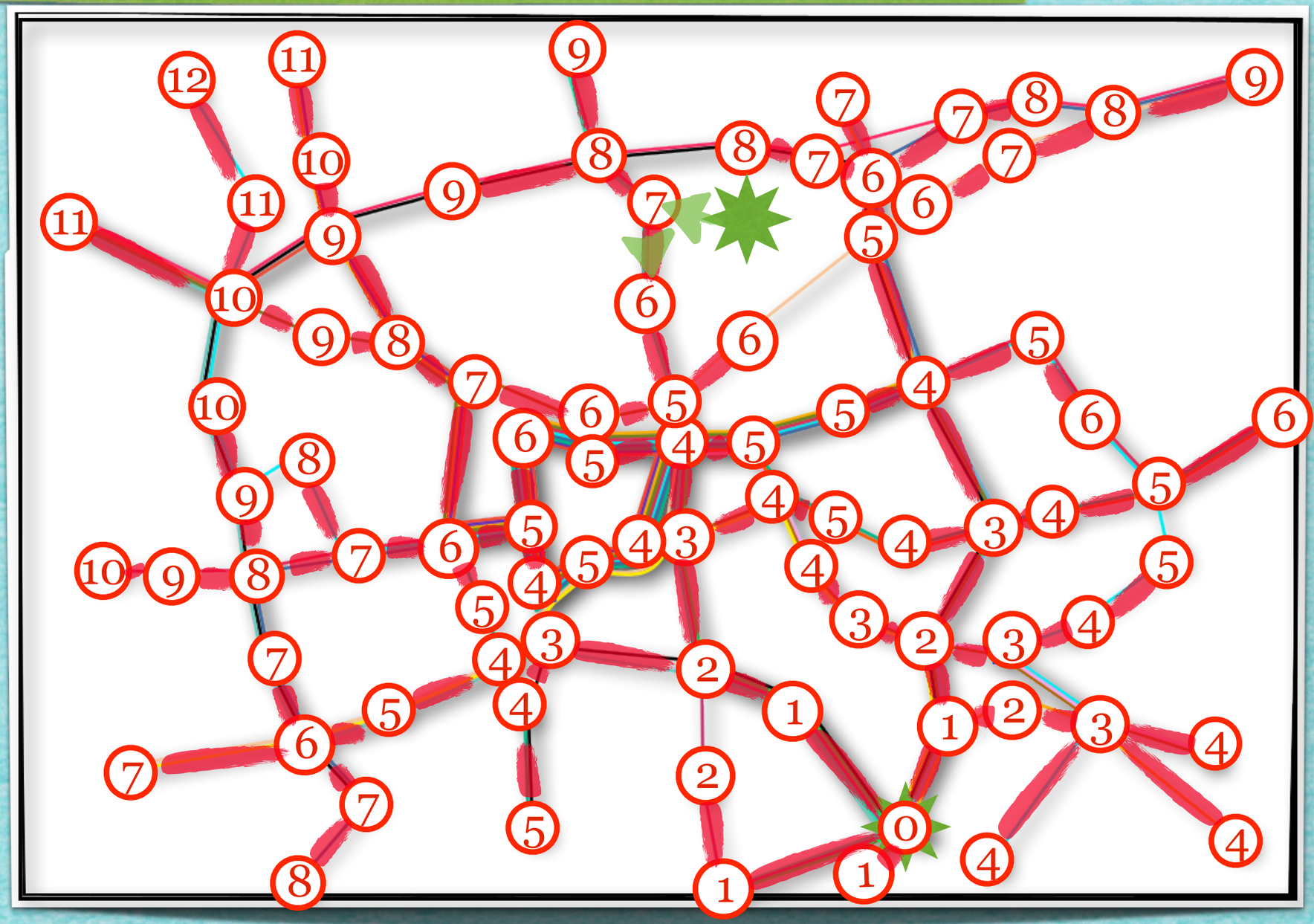
Breitensuche

Wellenreiten in Graphen



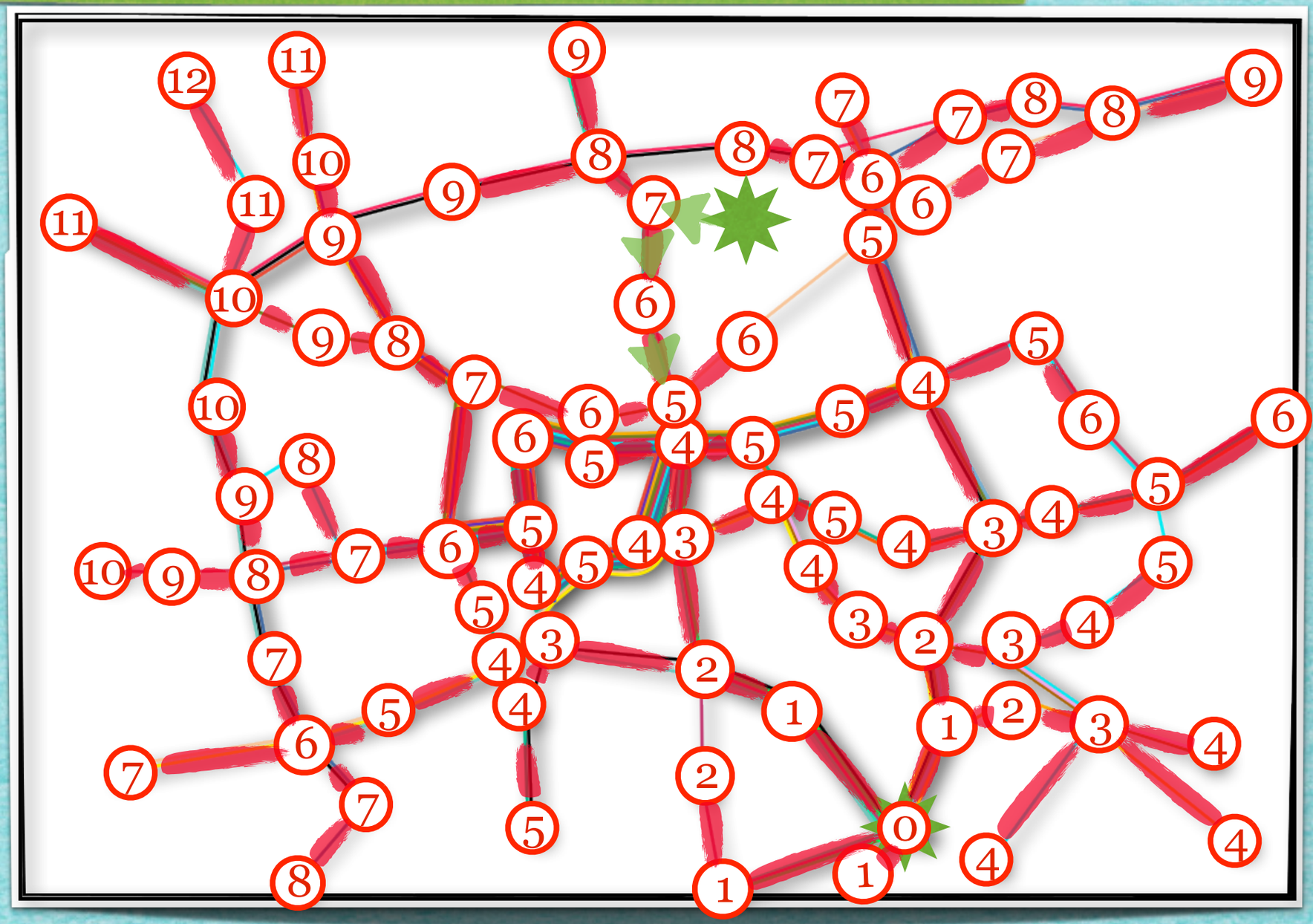
Breitensuche

Wellenreiten in Graphen



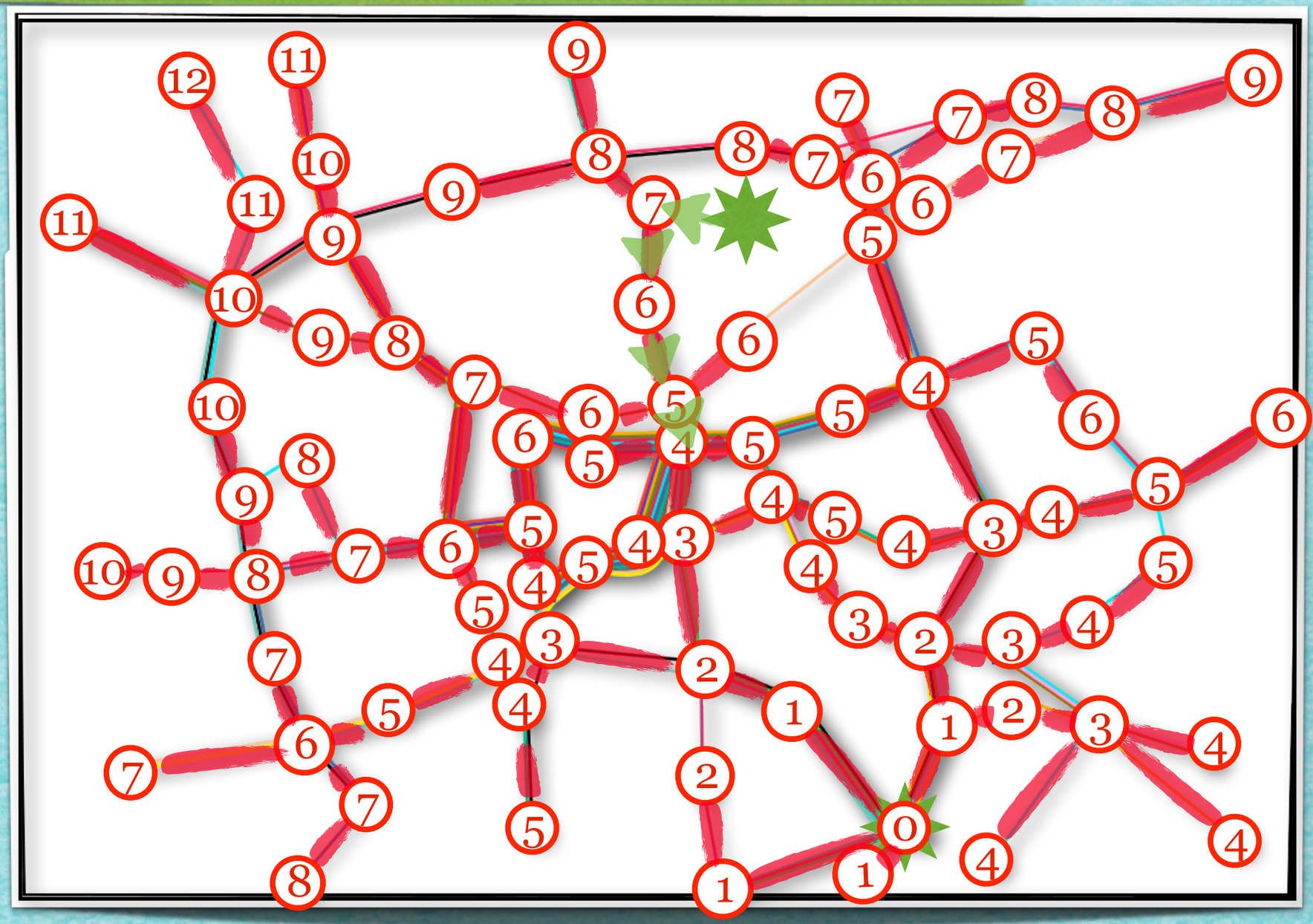
Breitensuche

Wellenreiten in Graphen



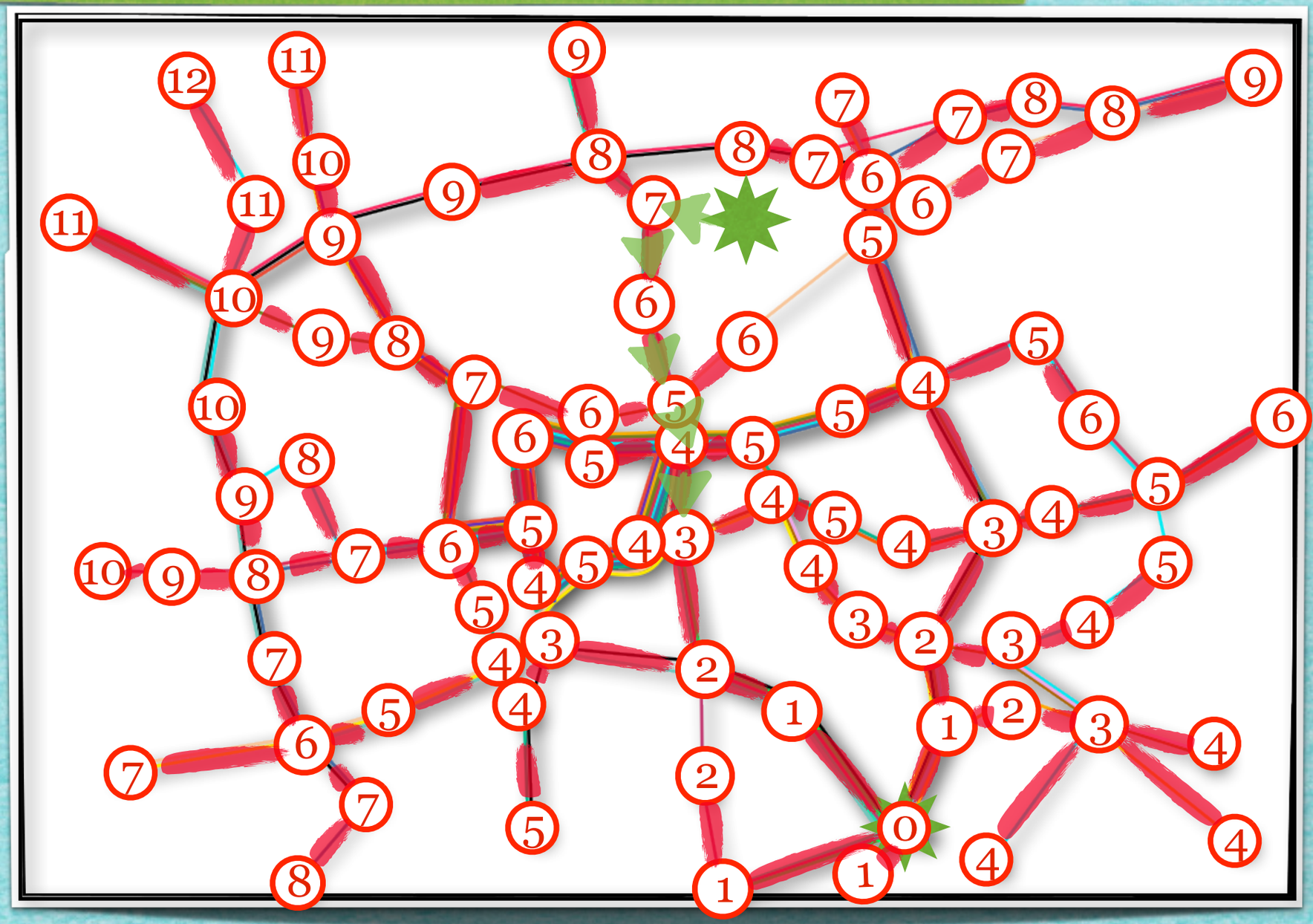
Breitensuche

Wellenreiten in Graphen



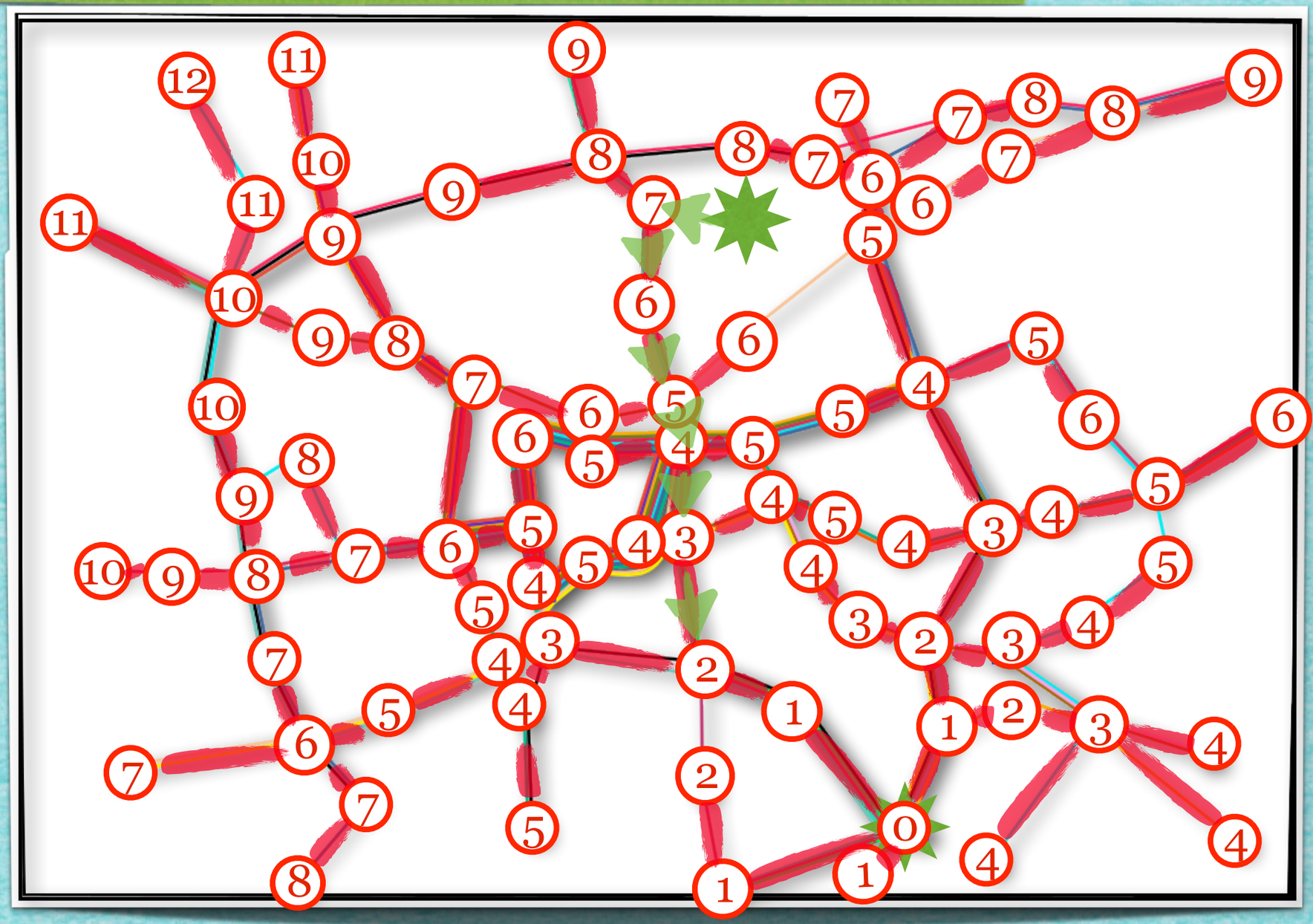
Breitensuche

Wellenreiten in Graphen



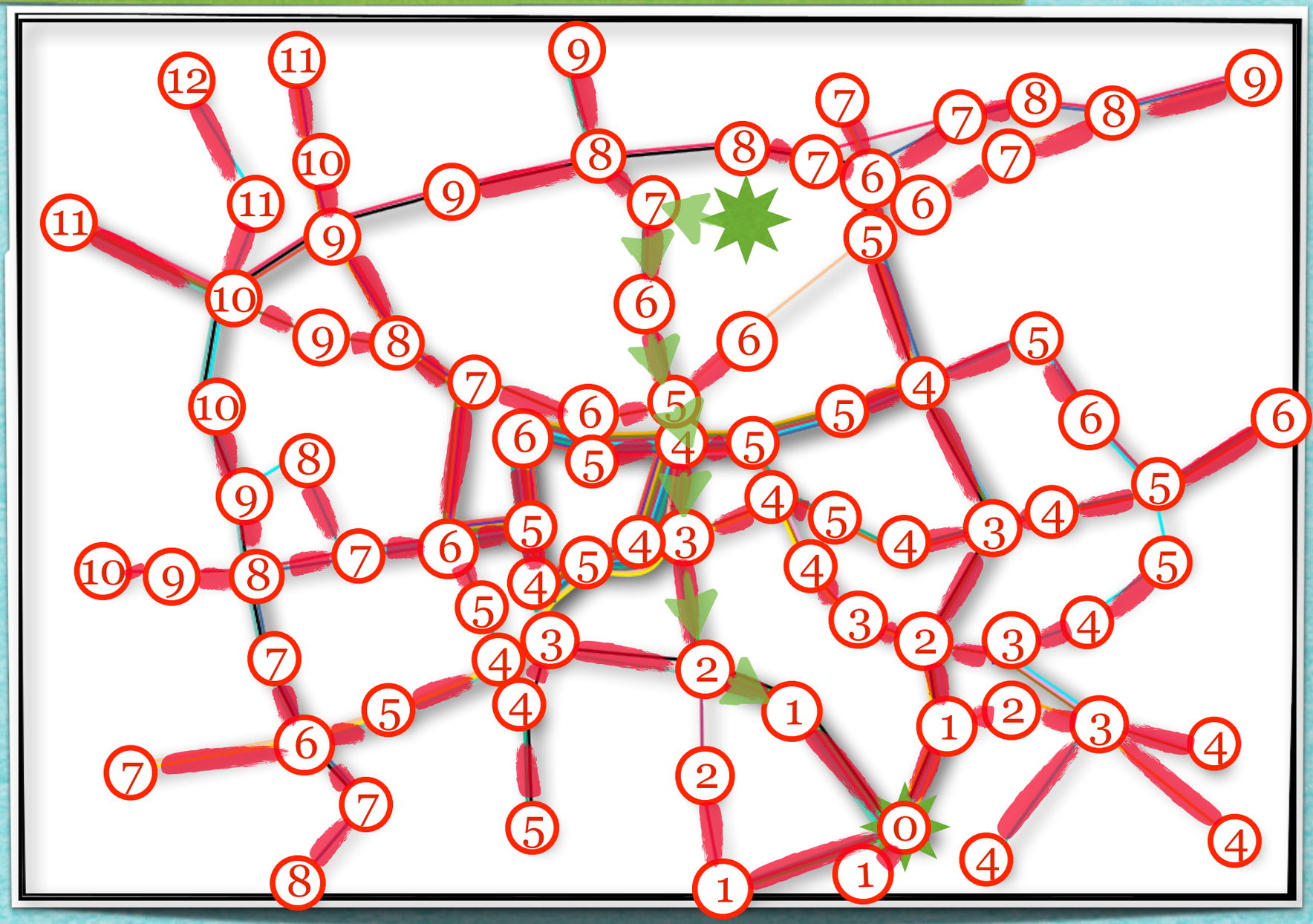
Breitensuche

Wellenreiten in Graphen



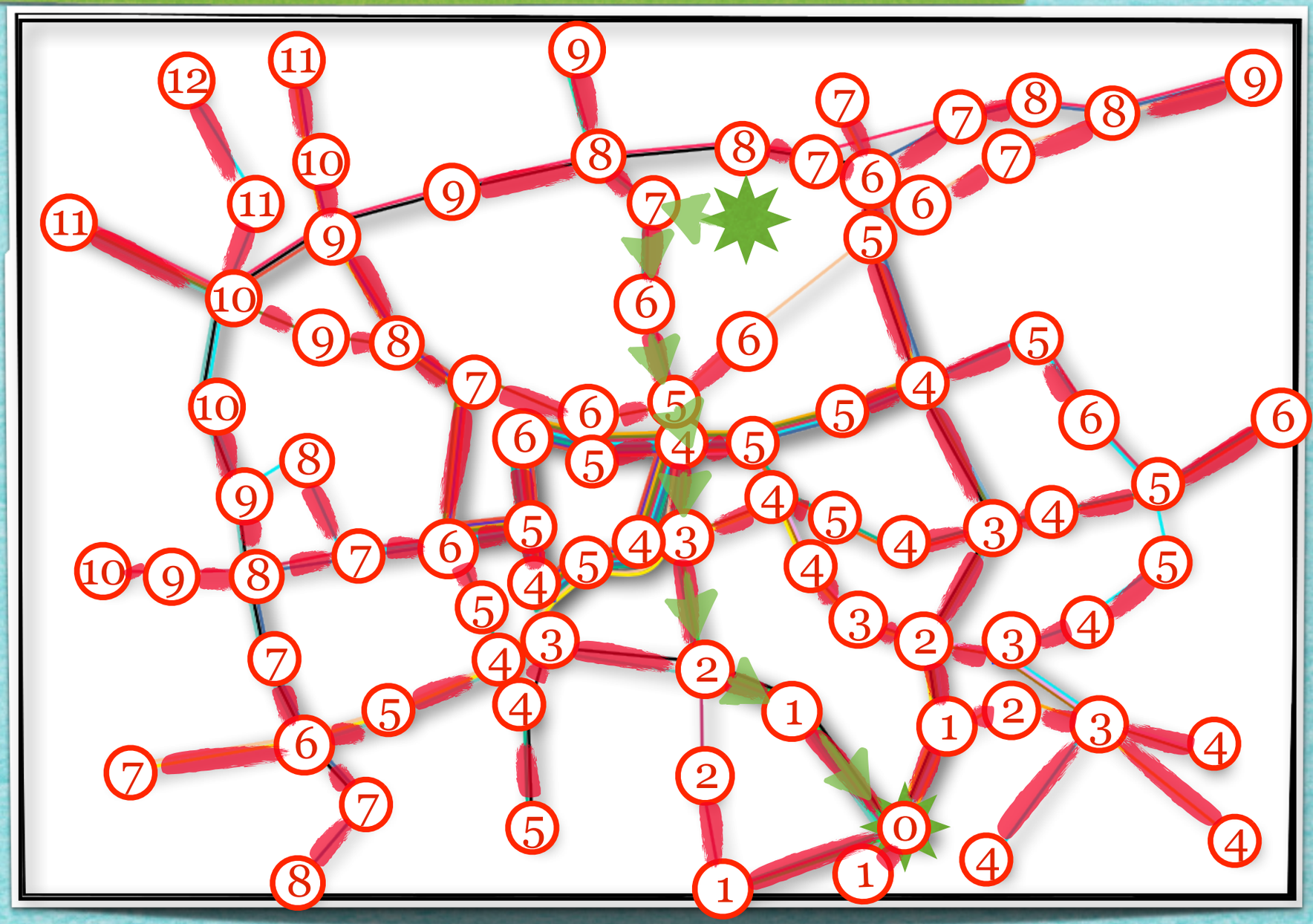
Breitensuche

Wellenreiten in Graphen



Breitensuche

Wellenreiten in Graphen



Breitensuche

Mehr Details!

s.fekete@tu-bs.de

GRÅPH SCÄN

GRÅPH SCÄN

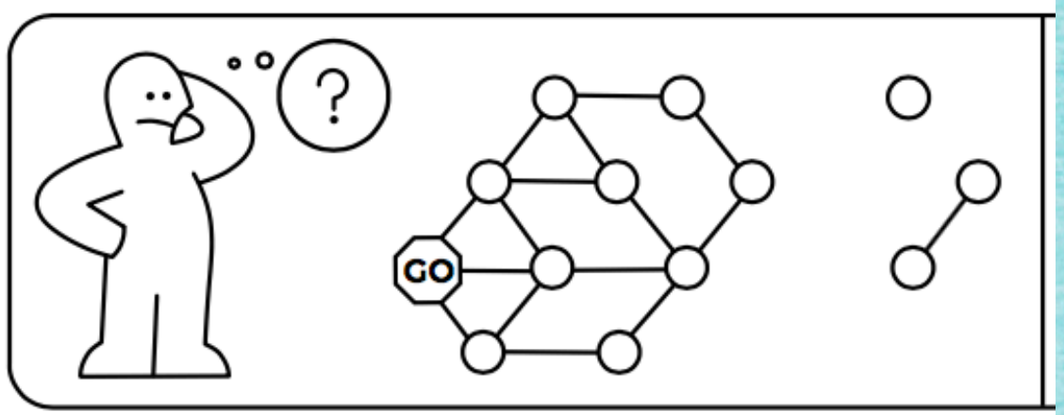
idea-instructions.com/graph-scan/

IDEA

GRÅPH SCÄN

idea-instructions.com/graph-scan/
v1.0, CC by-nc-sa 4.0

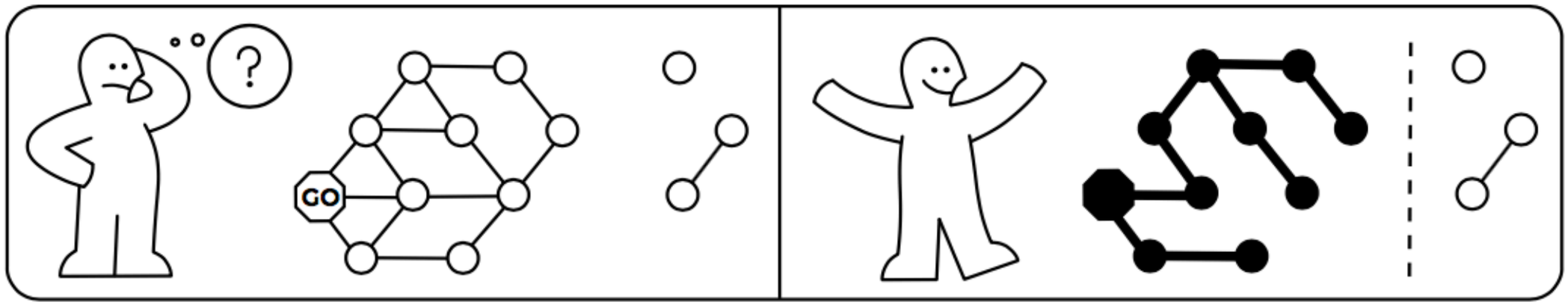
IDEA



GRÅPH SCÄN

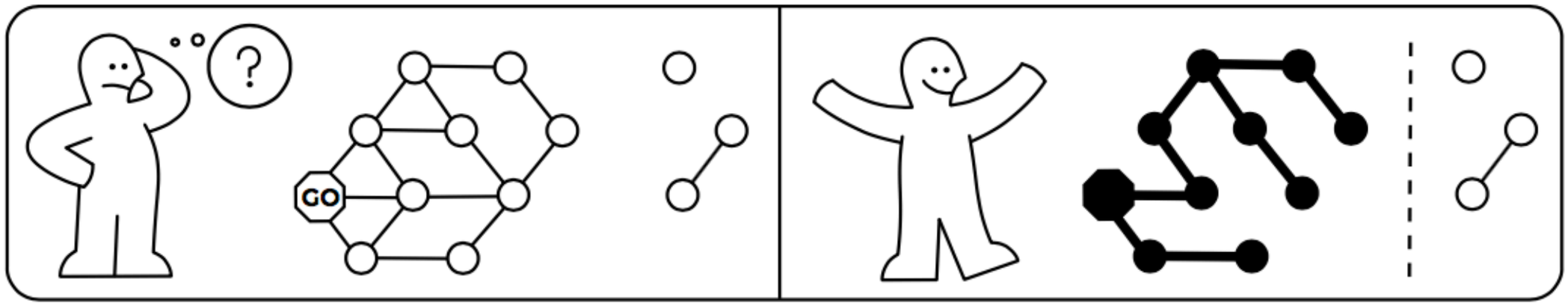
idea-instructions.com/graph-scan/
v1.0, CC by-nc-sa 4.0

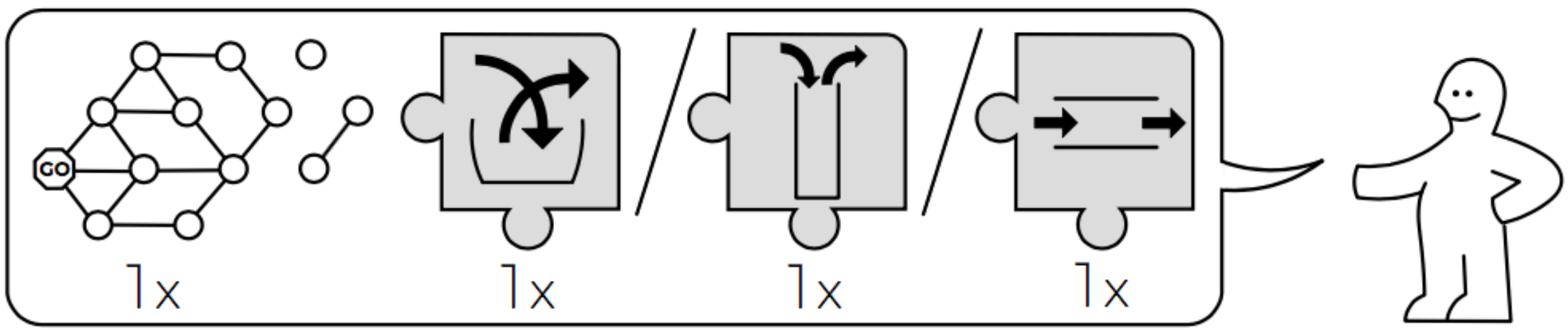
IDEA

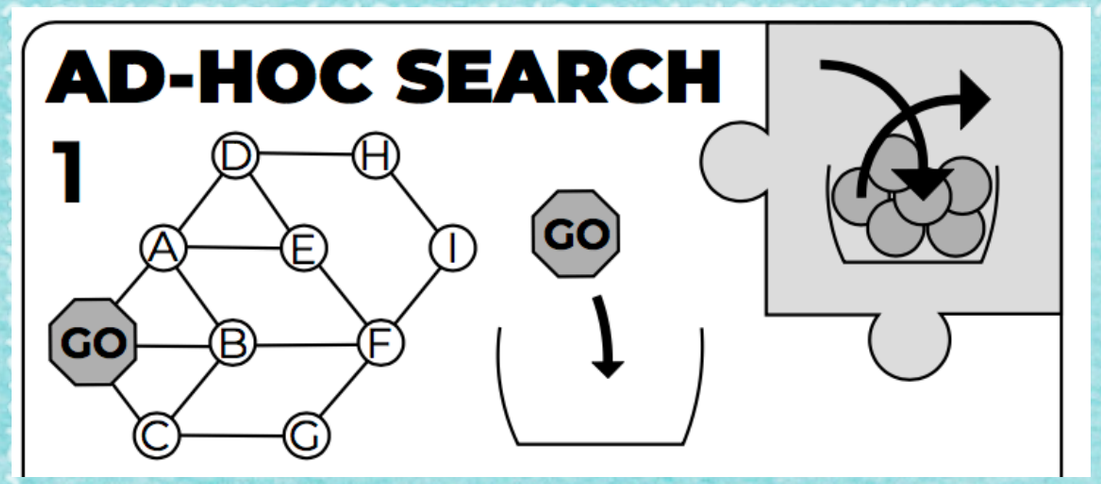


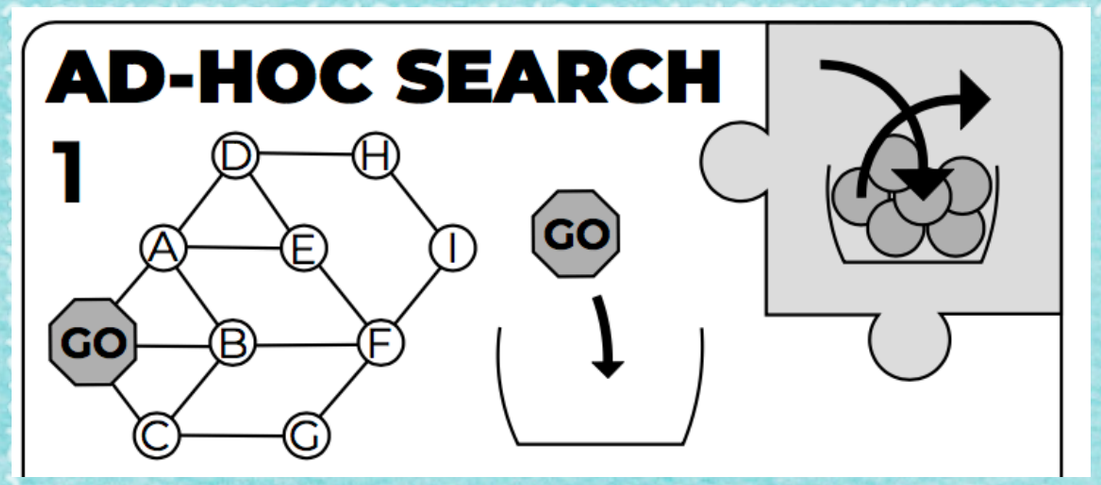
GRÅPH SCÄN

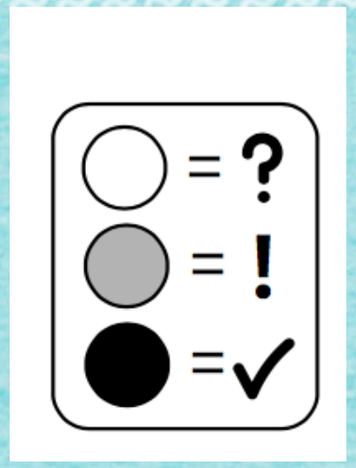
idea-instructions.com/graph-scan/
v1.0, CC by-nc-sa 4.0

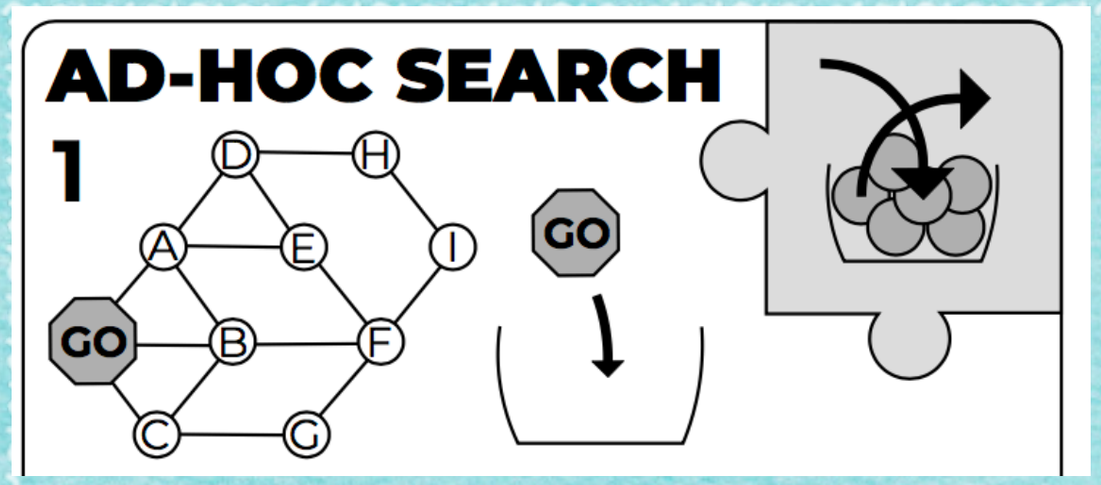


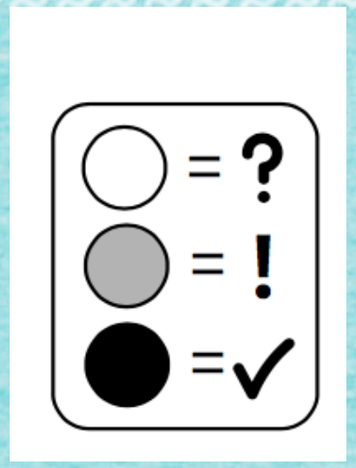


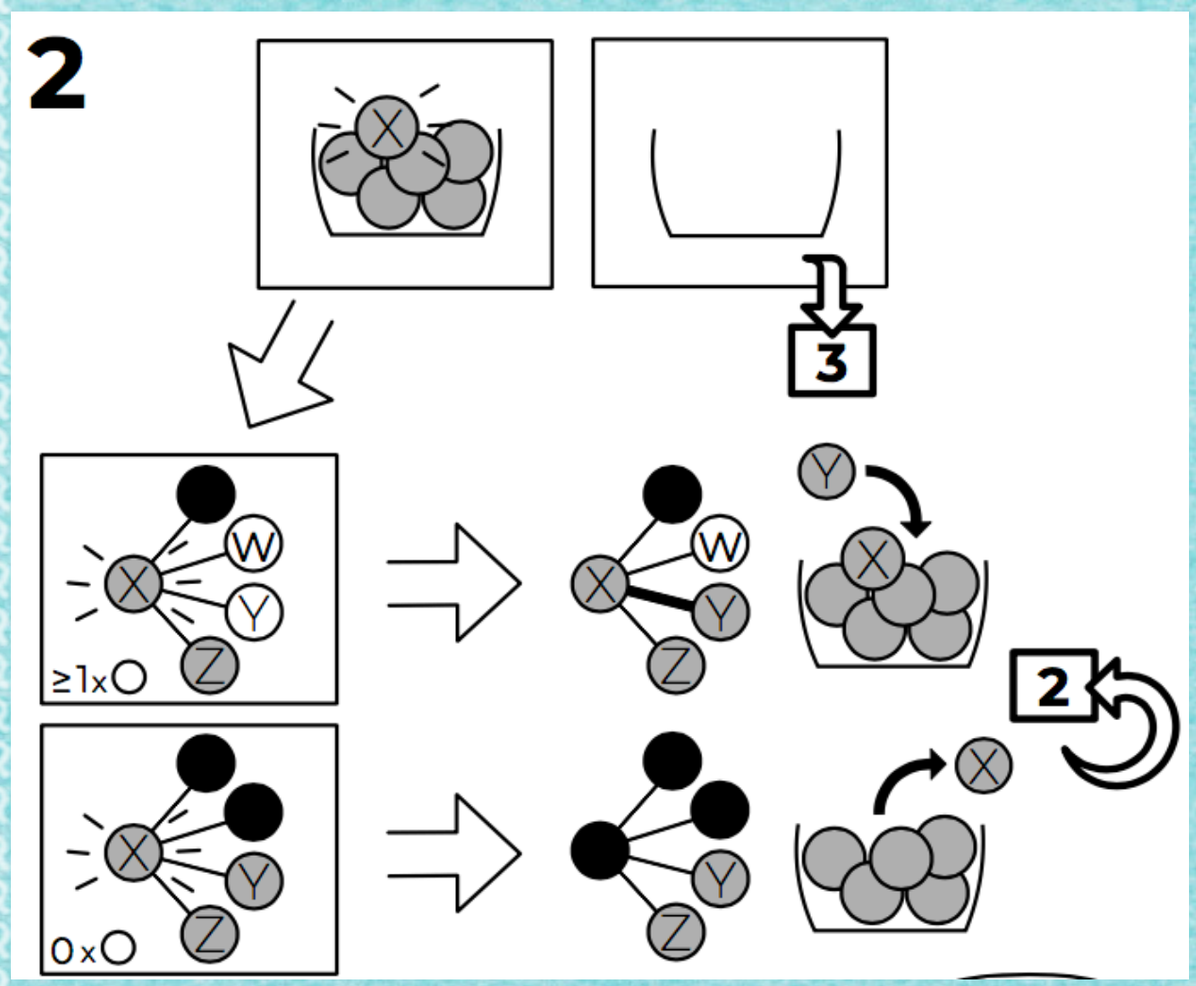

AD-HOC SEARCH
1
D
H
A
E
I
GO
B
F
C
G
GO

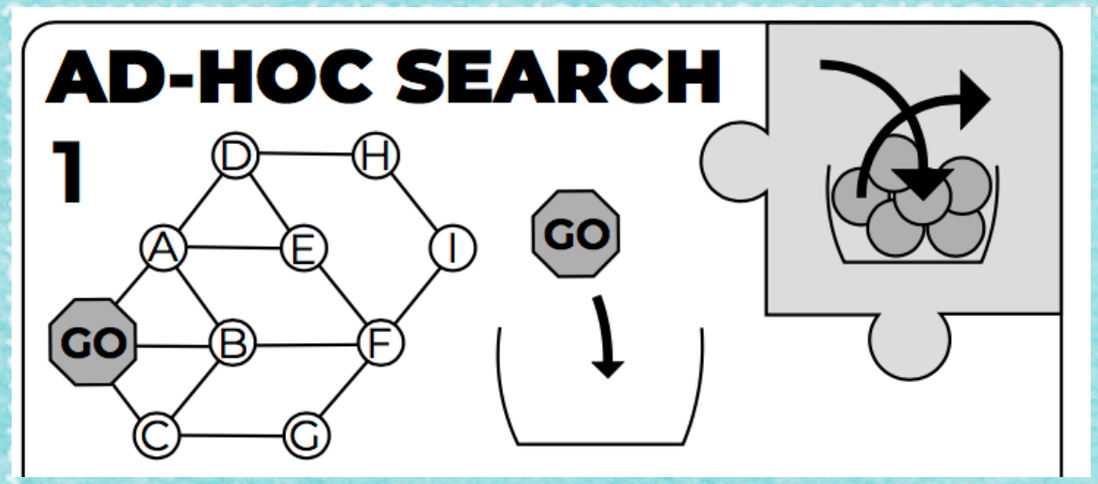

AD-HOC SEARCH
1
D
H
A
E
I
GO
B
F
C
G
GO

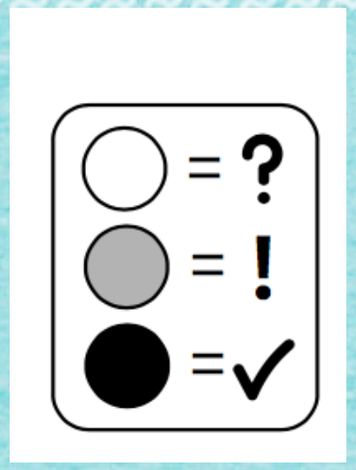

= ?
= !
=

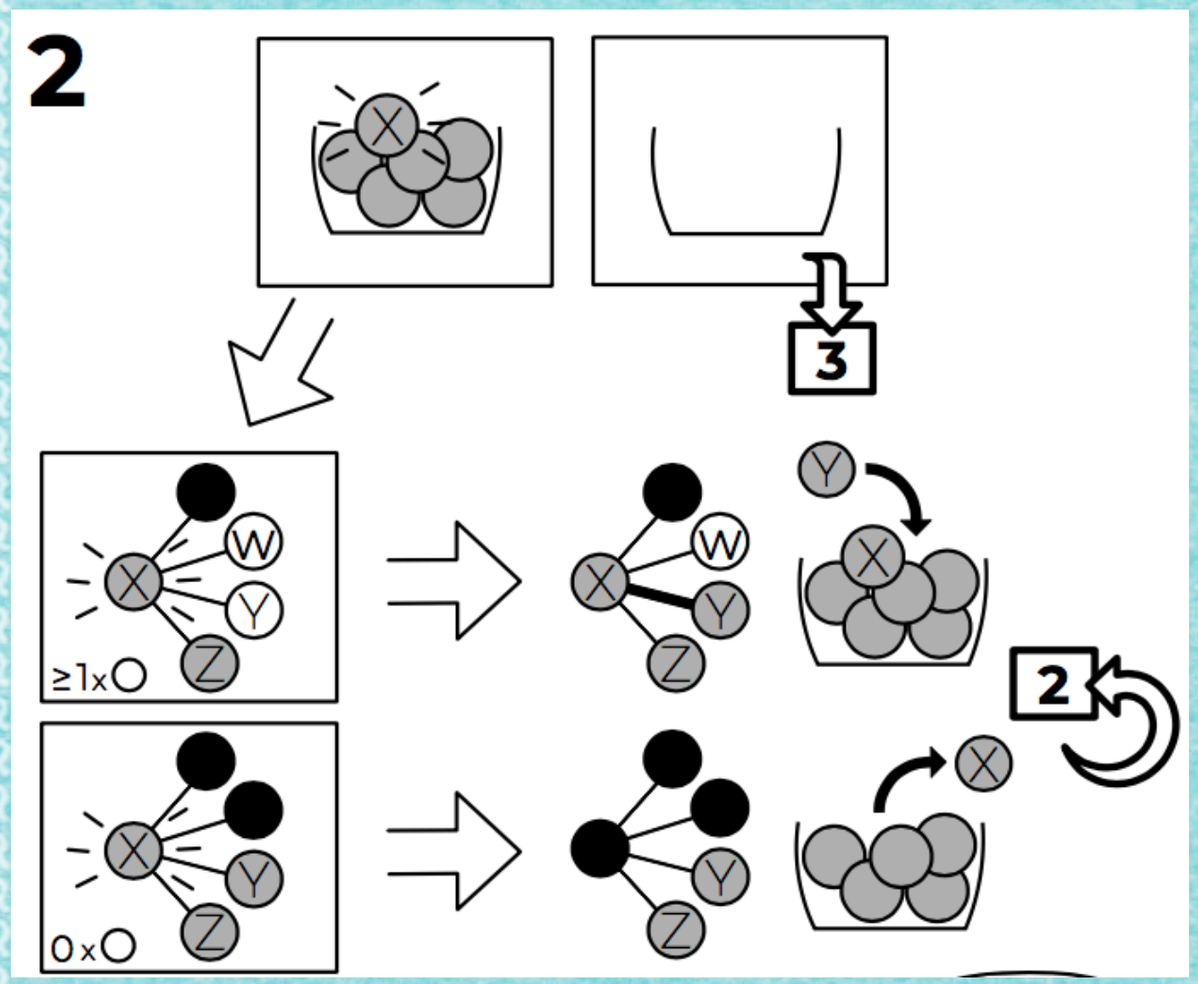

AD-HOC SEARCH
1
D
H
A
E
I
GO
B
F
C
G
GO

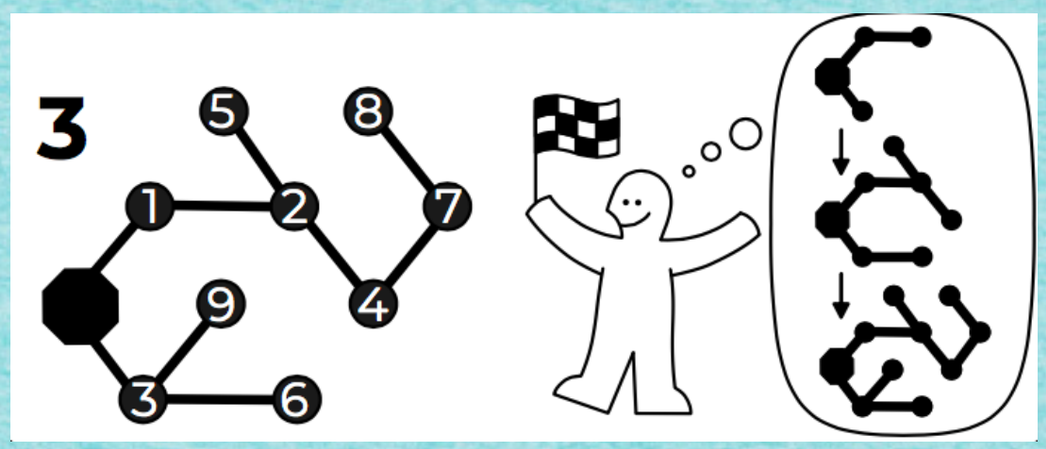

= ?
= !
=

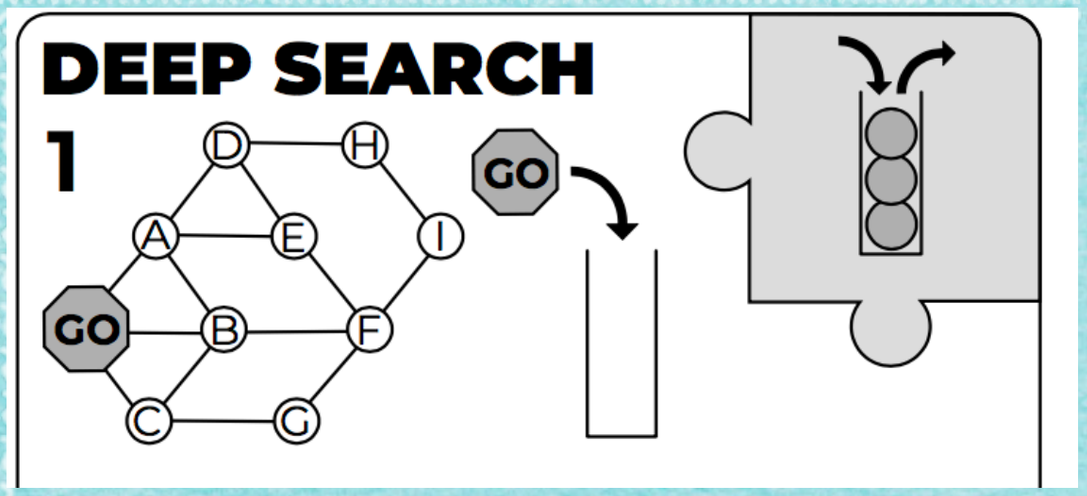

2
X
3
W
X
Y
Z
≥1xO
Y
X
W
X
Y
Z
2
0xO
X
Y
X
Z

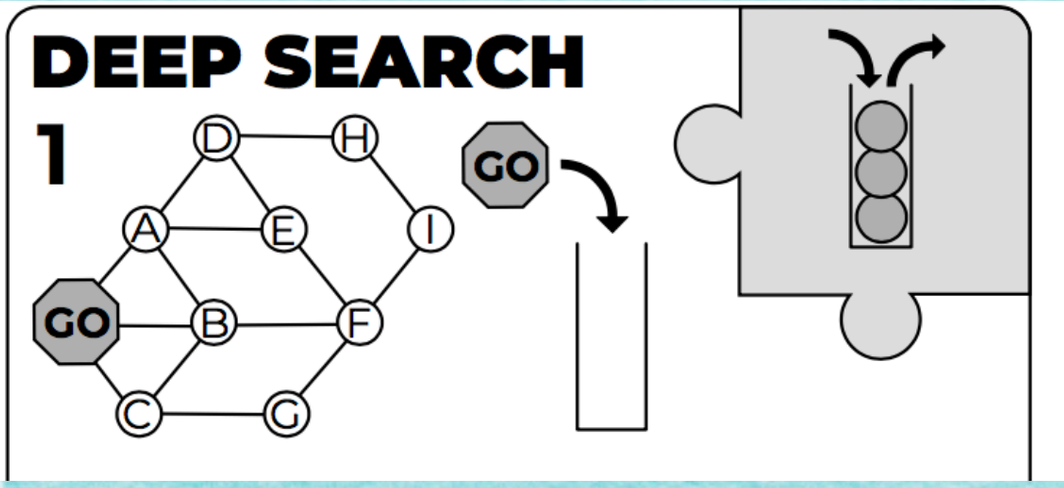

AD-HOC SEARCH
1
D
H
A
E
I
GO
B
F
C
G
GO

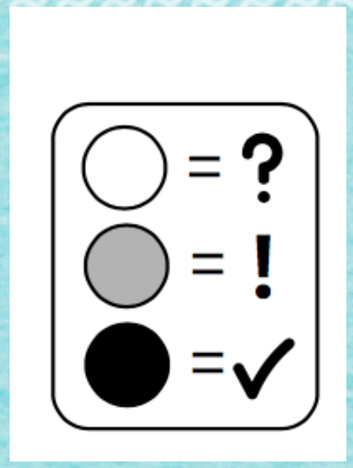

= ?
= !
=

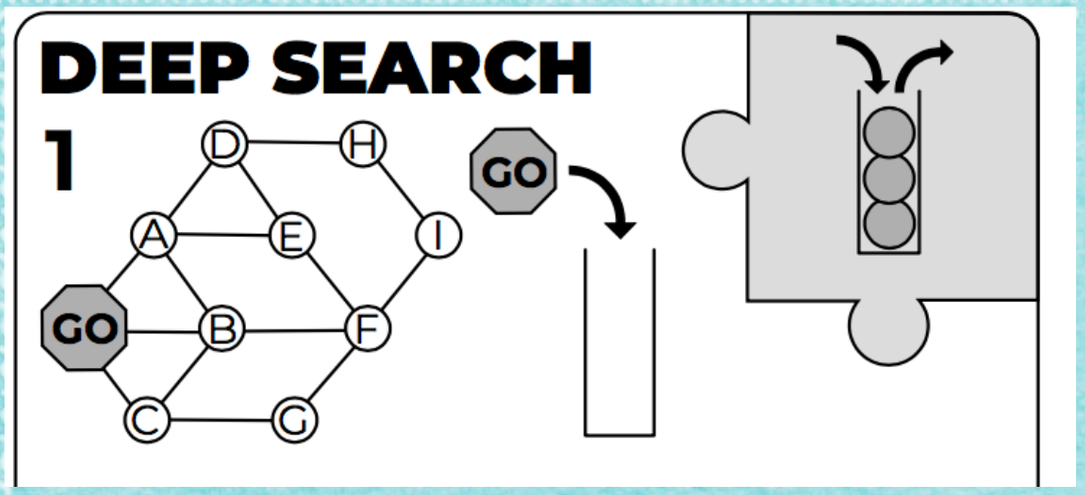

2
X
3
W
X
Y
Z
≥1xO
0xO
2

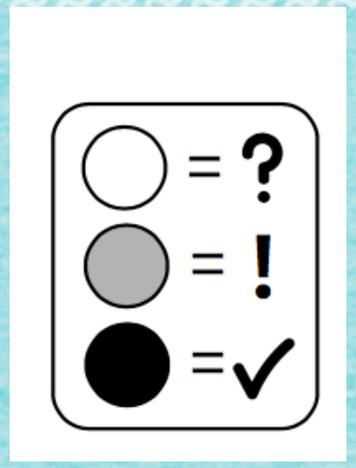

3
5
8
1
2
7
9
4
3
6

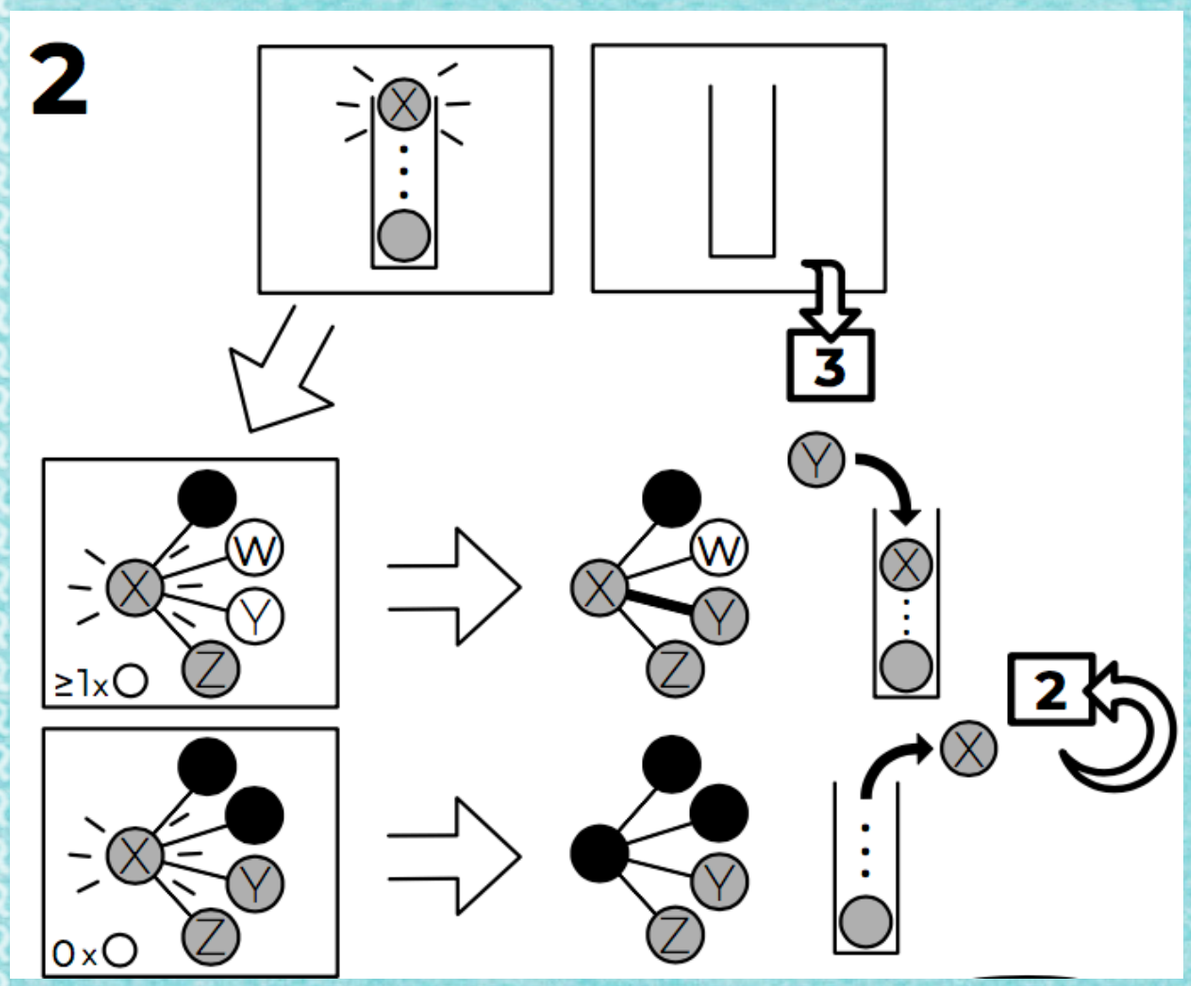

DEEP SEARCH
1
D
H
GO
A
E
I
GO
B
F
C
G


DEEP SEARCH
1
D
H
GO
A
E
I
GO
B
F
C
G

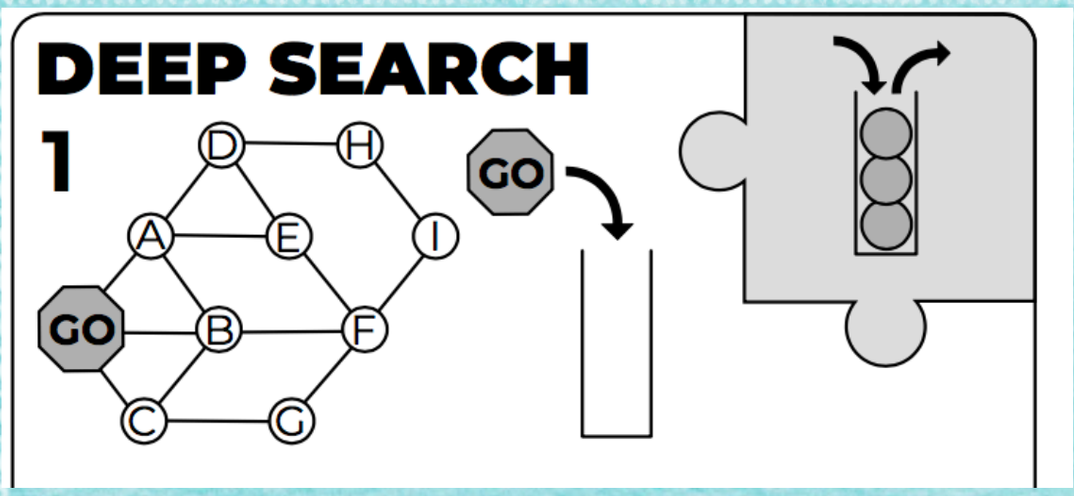

= ?
= !
=

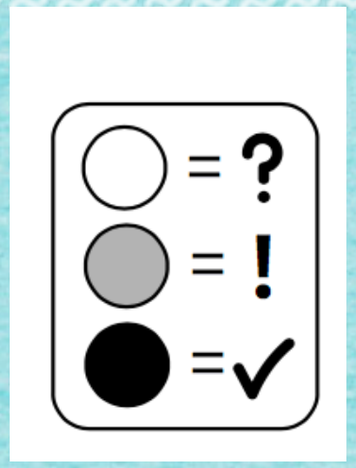

DEEP SEARCH
1
GO
D
H
A
E
I
GO
B
F
C
G

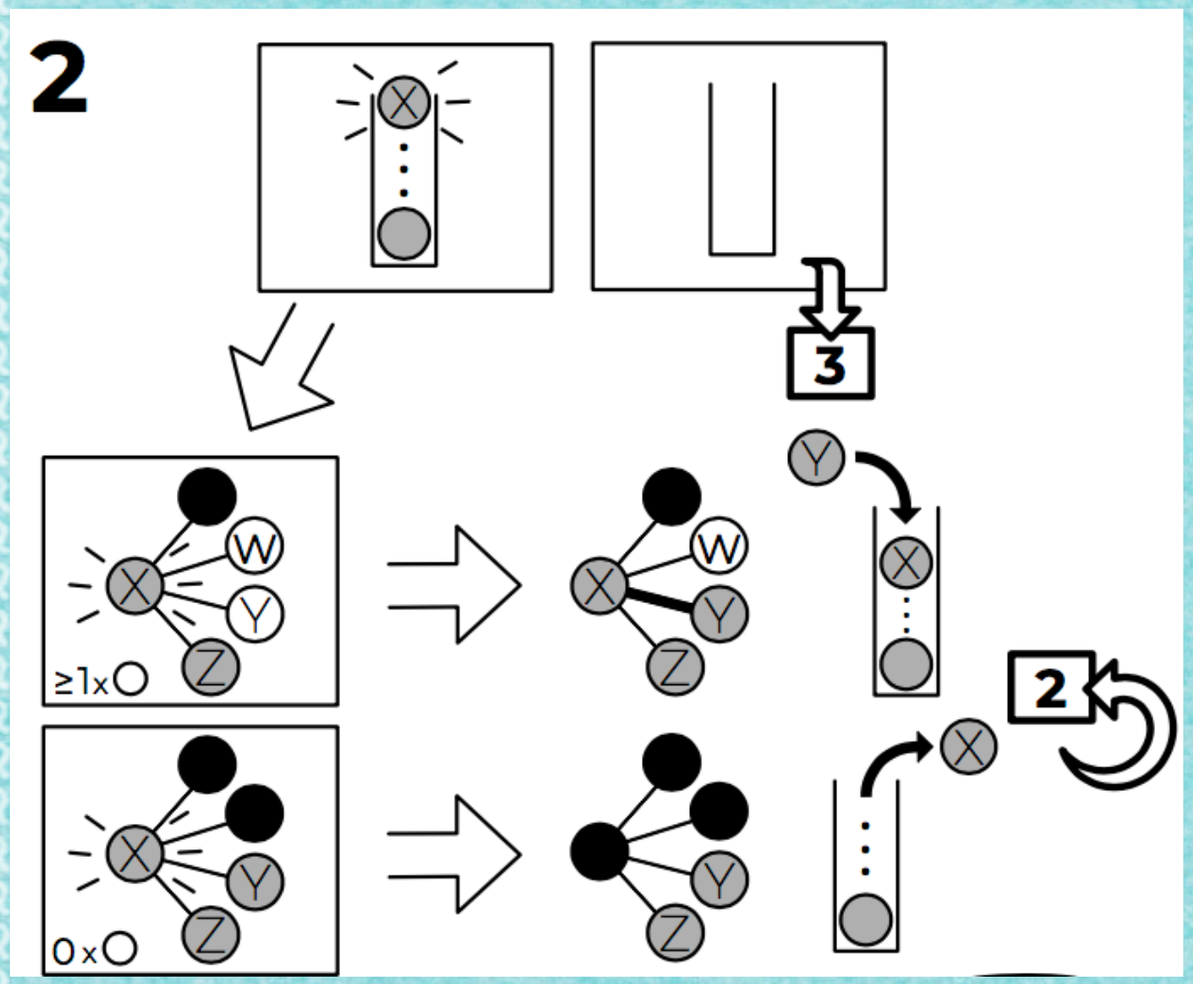

= ?
= !
= ✓

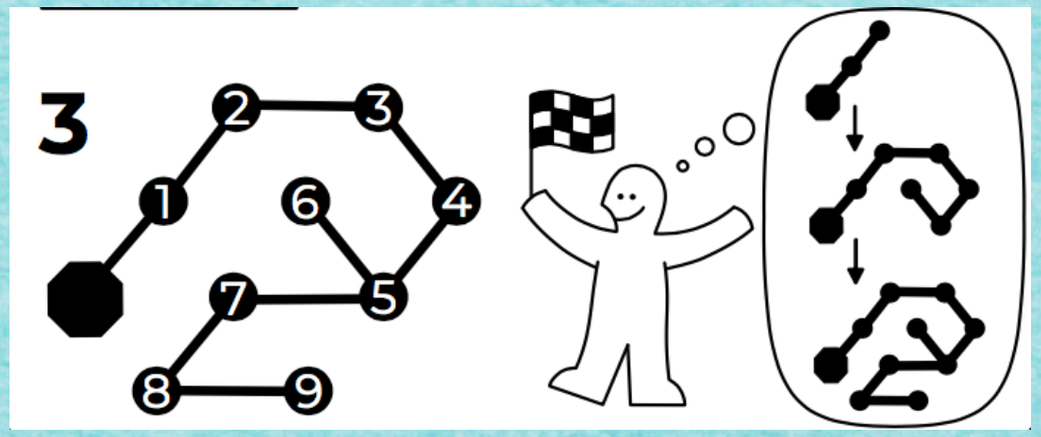

2
X
3
Y
X
W
Y
Z
≥1x○
0x○
2

DEEP SEARCH



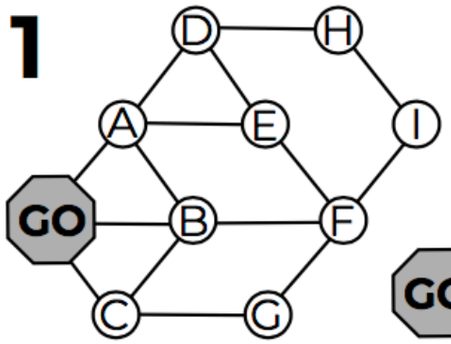






BROAD SEARCH

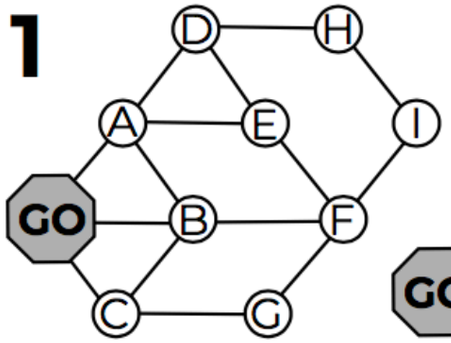
1



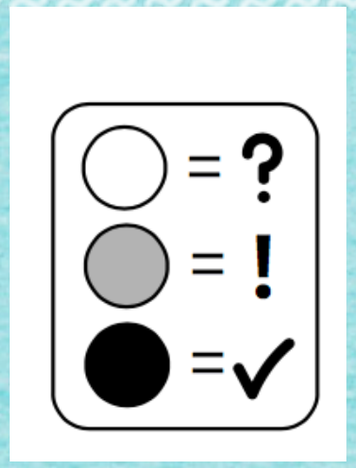
GO

BROAD SEARCH

1

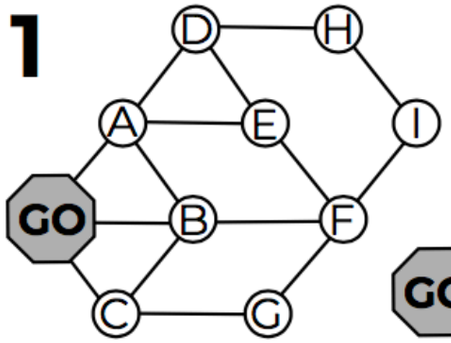


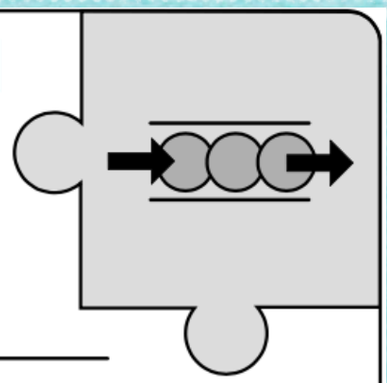
GO

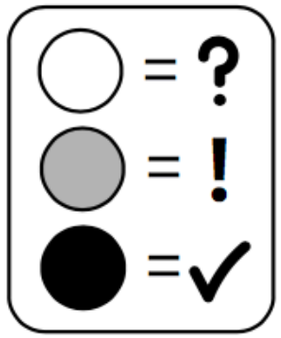


BROAD SEARCH

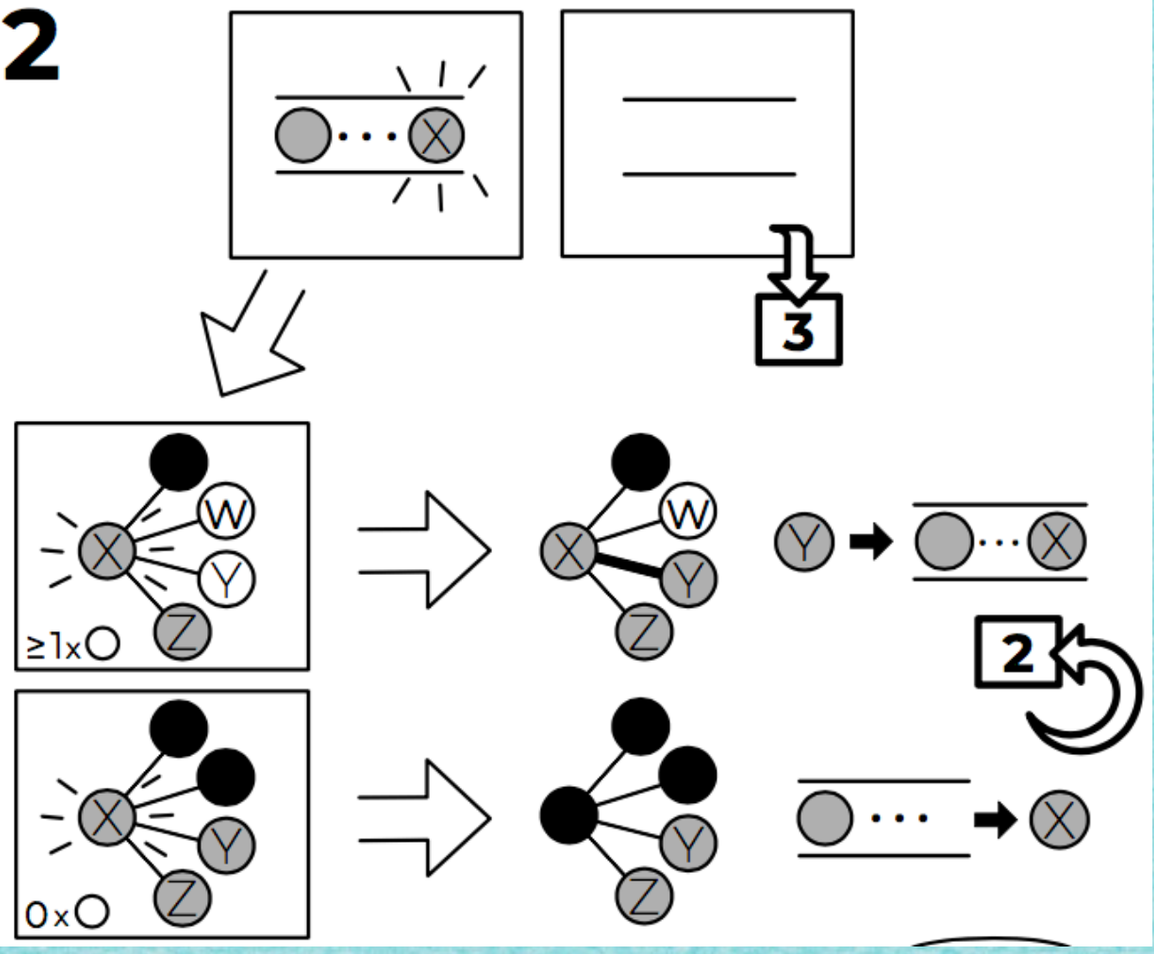
1



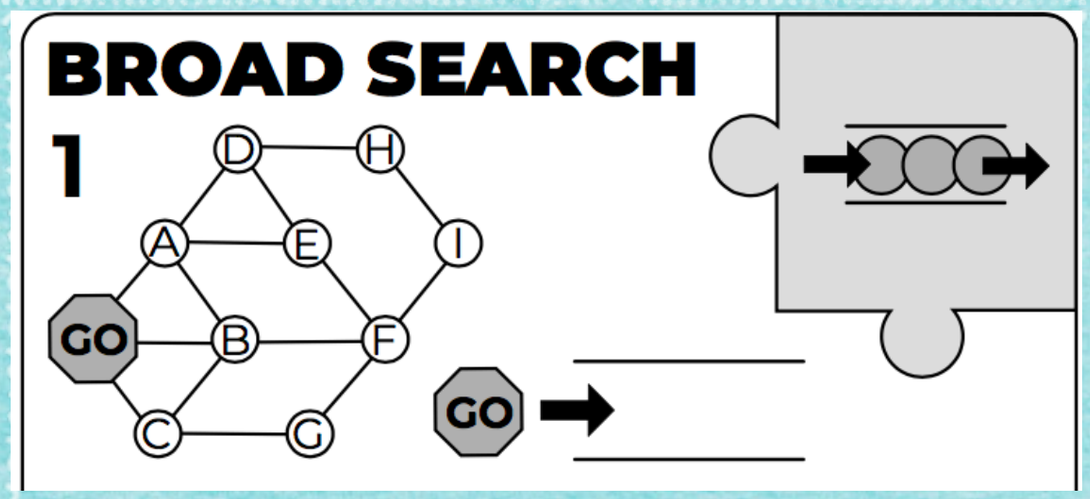


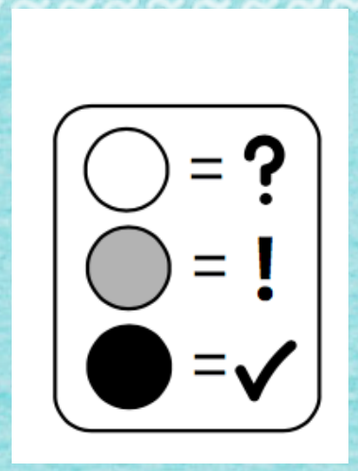


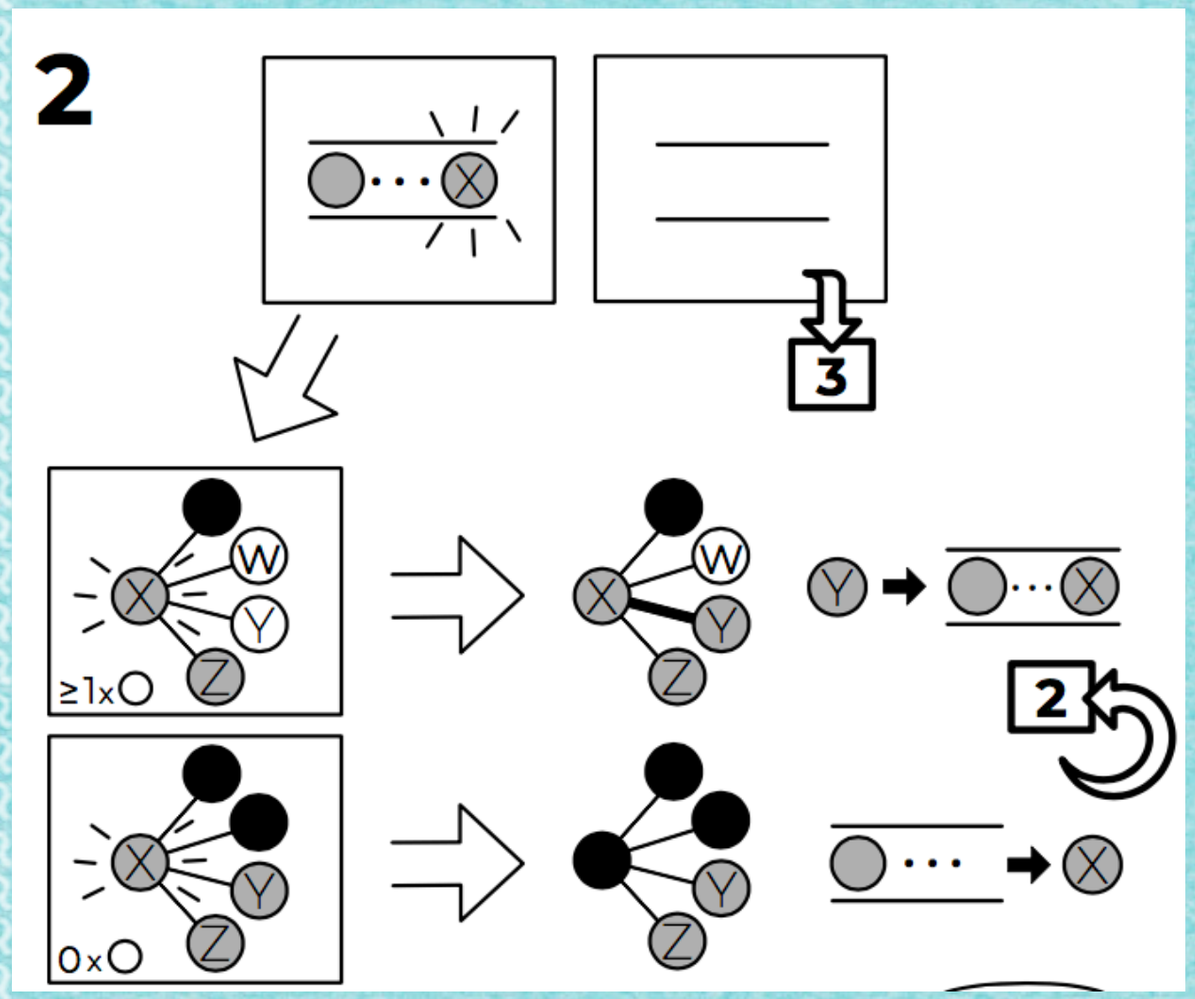
2

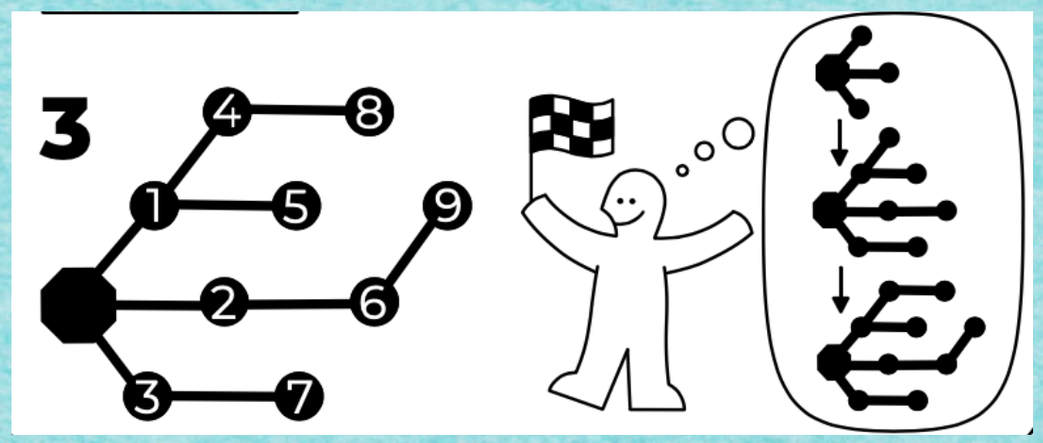


BROAD SEARCH





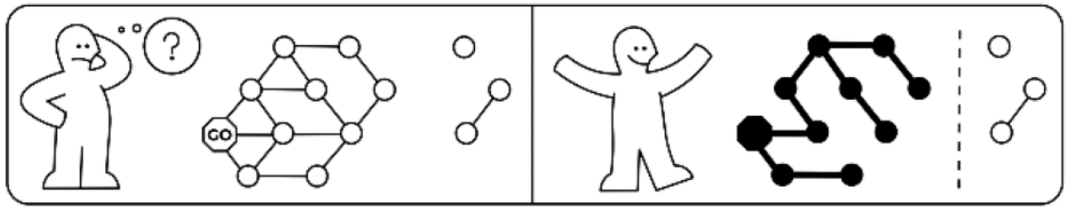


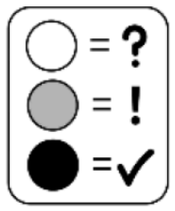


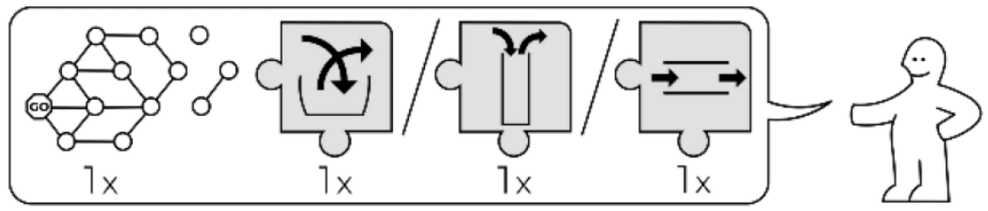
GRÅPH SKÄN

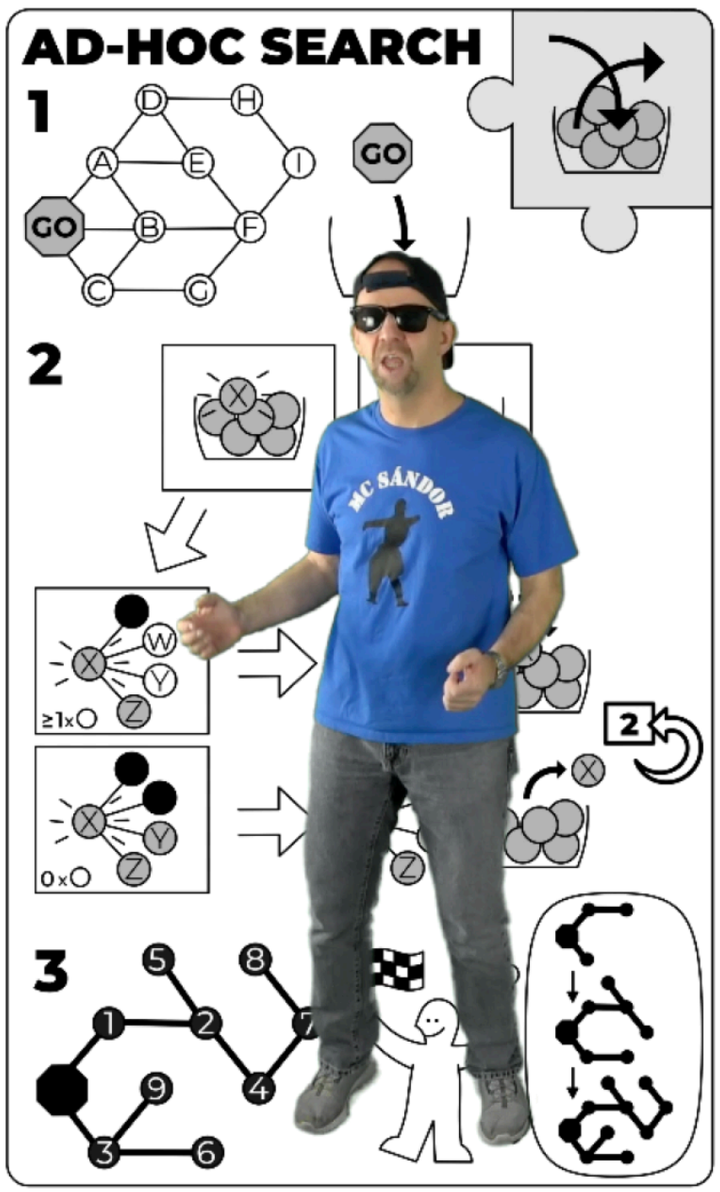
idea-instructions.com/graph-scan/
v1.3, CC by-nc-sa 4.0

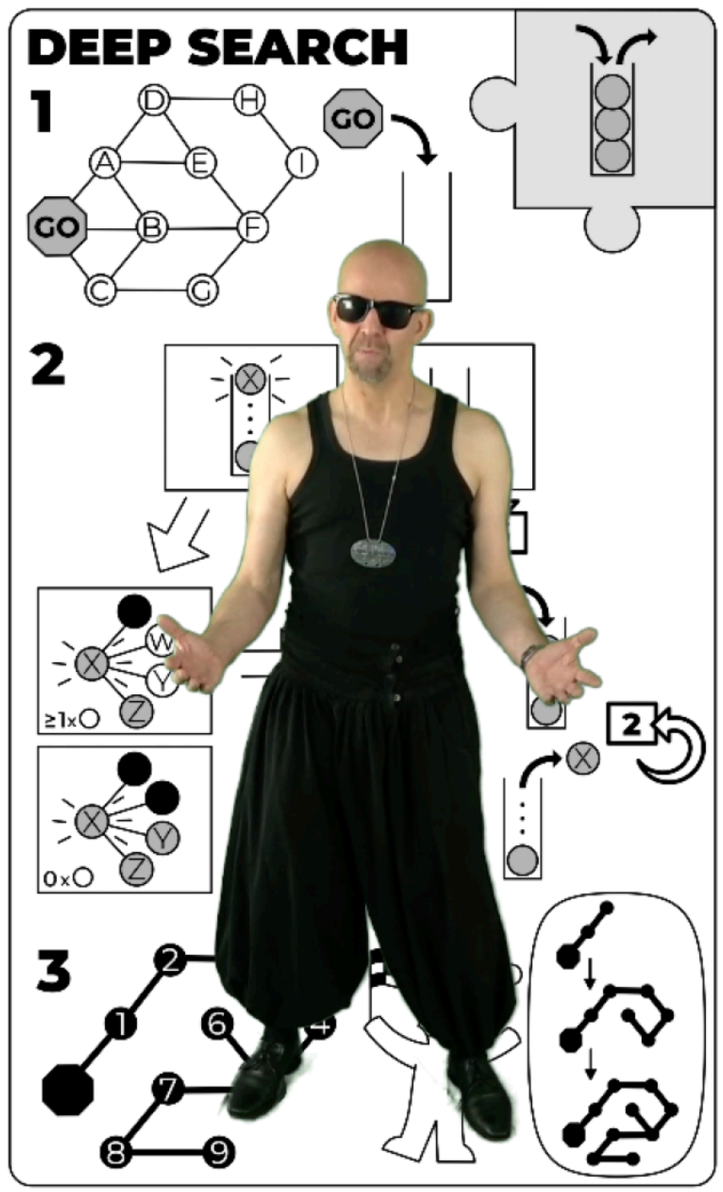
IDEA

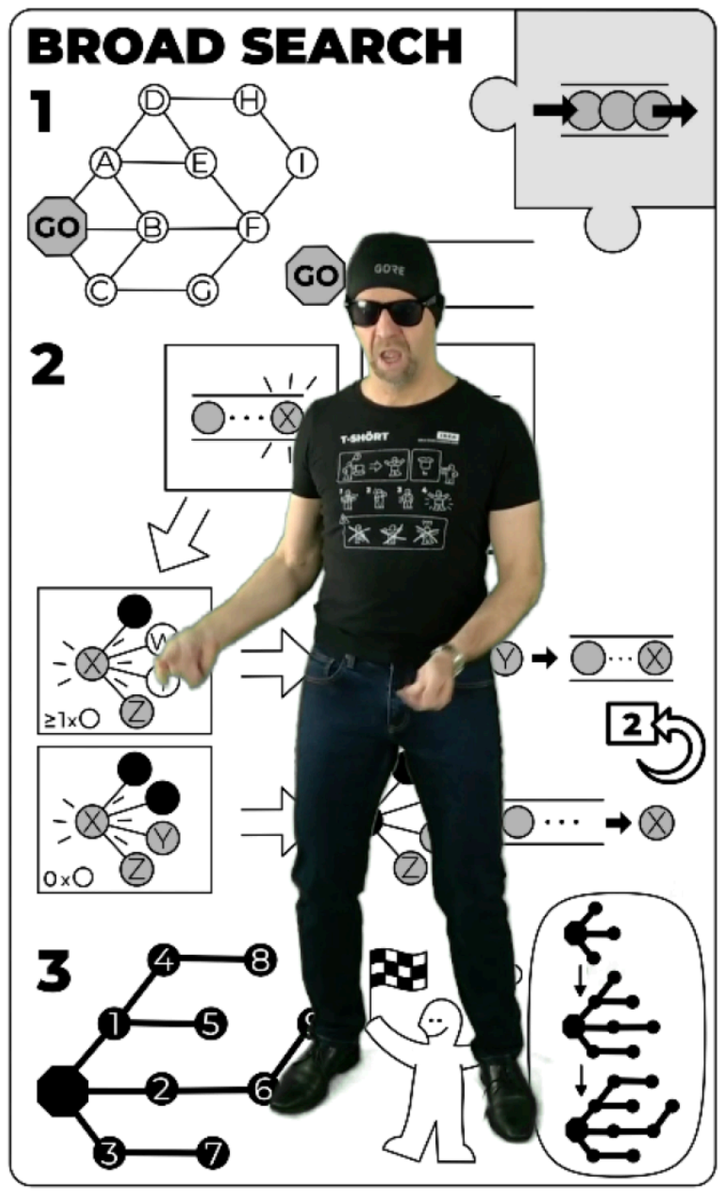








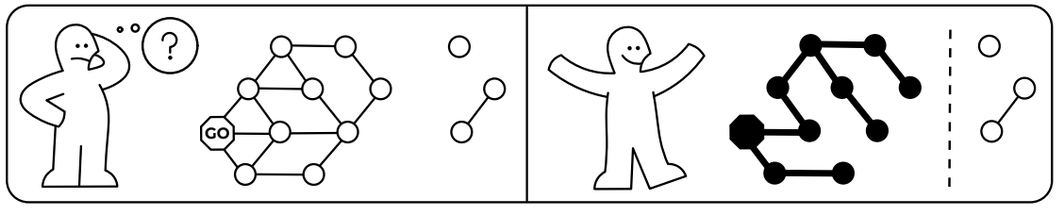


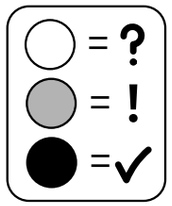


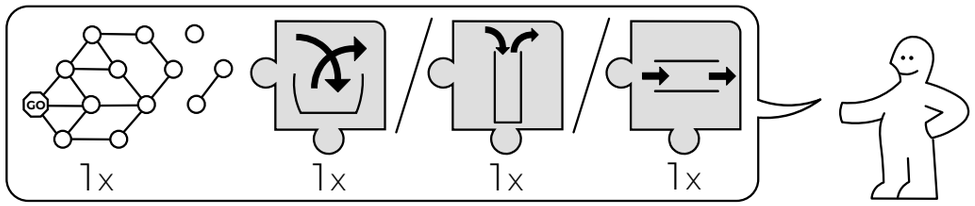
GRÅPH SKÄN

idea-instructions.com/graph-scan/
v1.3, CC by-nc-sa 4.0

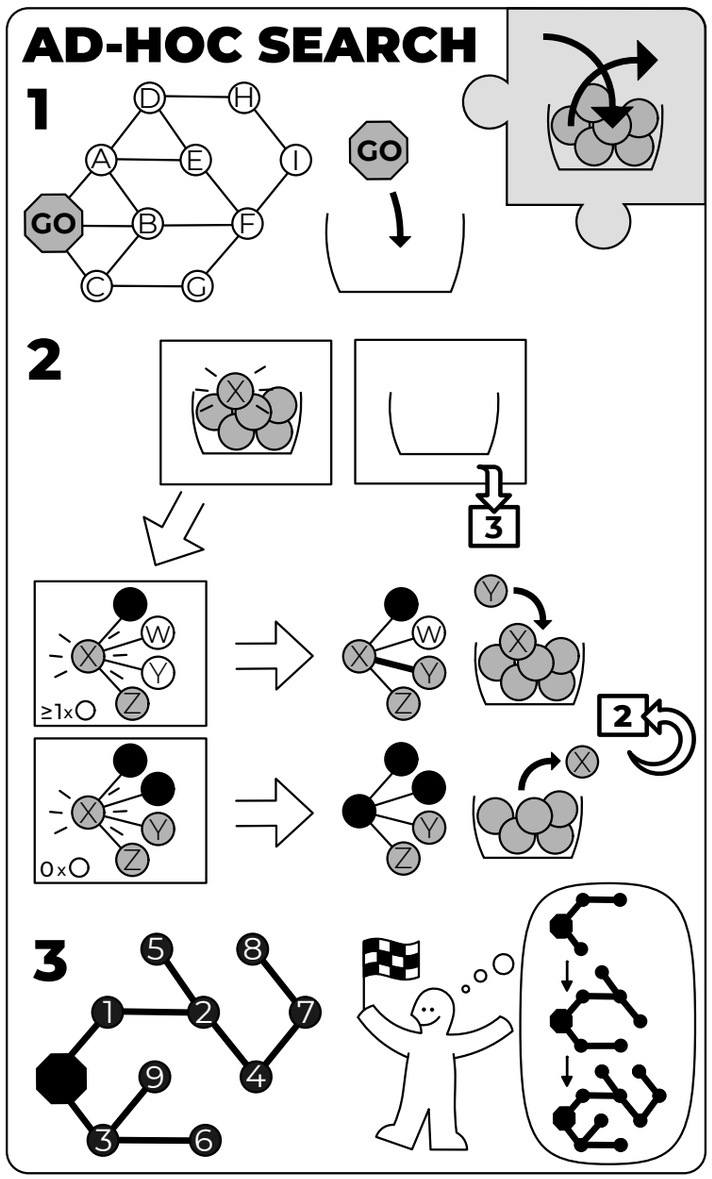
IDEA



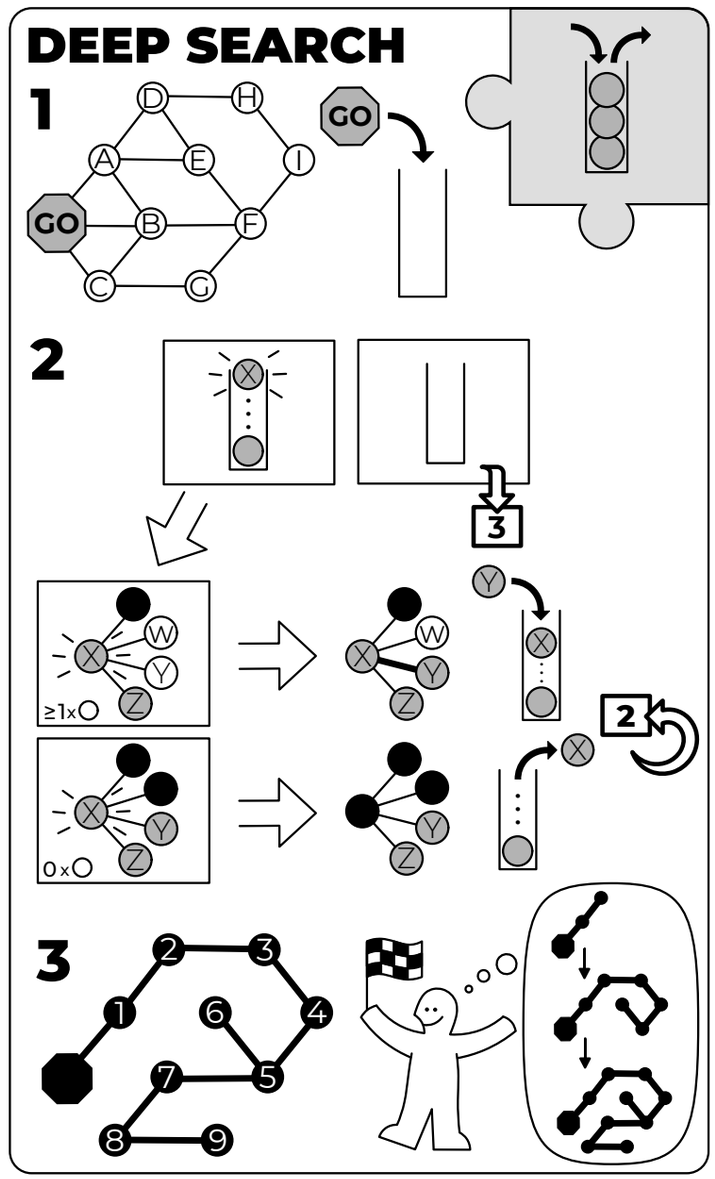




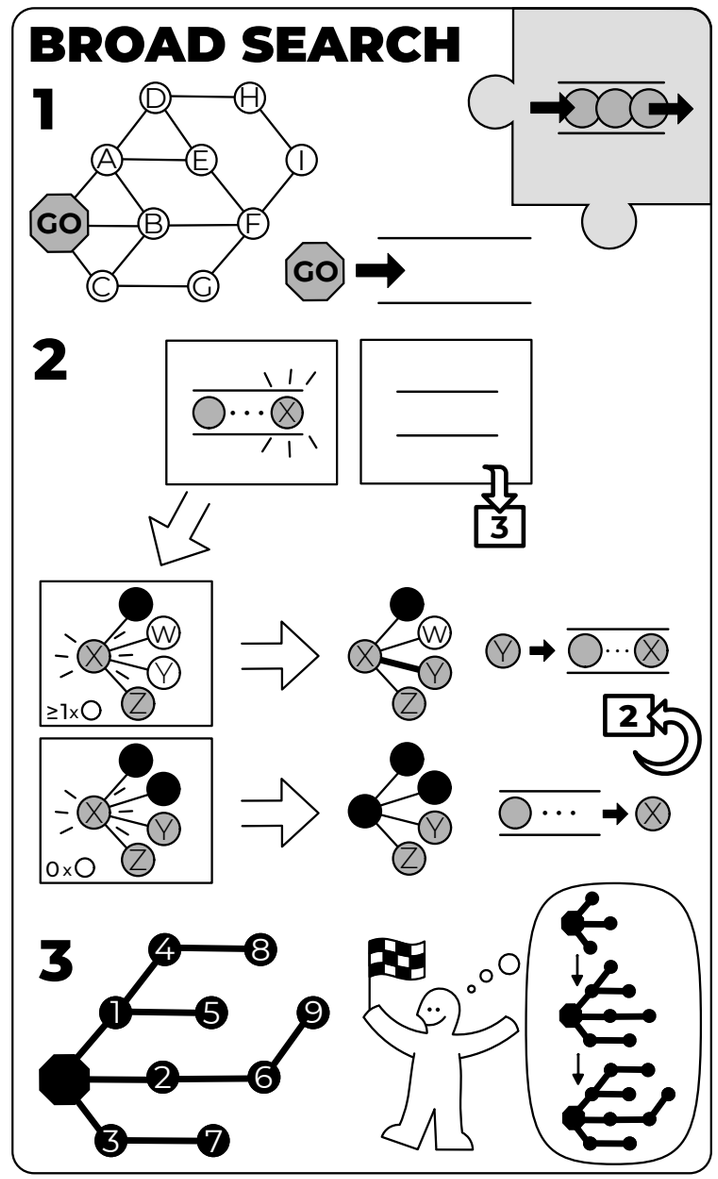
AD-HOC SEARCH



DEEP SEARCH

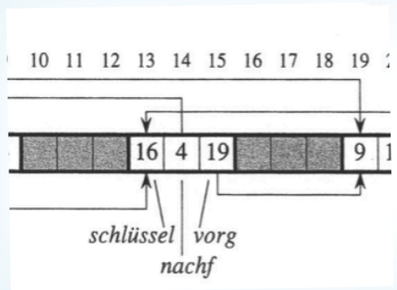


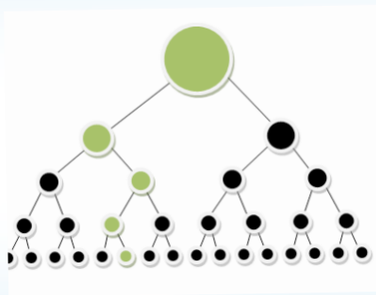
BROAD SEARCH



Kapitelende!

s.fekete@tu-bs.de






Kapitel 4: Dynamische Datenstrukturen

Algorithmen und Datenstrukturen
WS 2022/23

Prof. Dr. Sándor Fekete

Wie verwalten wir dynamische Mengen von Objekten?

Wie verwalten wir dynamische Mengen von Objekten?

Waschkorb

4.1 Grundoperationen

4.1 Grundoperationen

Aufgabenstellung:

4.1 Grundoperationen

Aufgabenstellung:

- ***Verwalten einer Menge S von Objekten***

4.1 Grundoperationen

Aufgabenstellung:

- ***Verwalten einer Menge S von Objekten***
- ***Ausführen von verschiedenen Operationen (s.u.)***

4.1 Grundoperationen

<u>Aufgabenstellung:</u>

- ***Verwalten einer Menge S von Objekten***
- ***Ausführen von verschiedenen Operationen (s.u.)***

<u>Im Folgenden:</u>

4.1 Grundoperationen

Aufgabenstellung:

- ***Verwalten einer Menge S von Objekten***
- ***Ausführen von verschiedenen Operationen (s.u.)***

Im Folgenden:

S Menge von Objekten

4.1 Grundoperationen

Aufgabenstellung:

- ***Verwalten einer Menge S von Objekten***
- ***Ausführen von verschiedenen Operationen (s.u.)***

Im Folgenden:

S Menge von Objekten

k Wert eines Elements (“Schlüssel”)

4.1 Grundoperationen

Aufgabenstellung:

- ***Verwalten einer Menge S von Objekten***
- ***Ausführen von verschiedenen Operationen (s.u.)***

Im Folgenden:

S **Menge von Objekten**

k **Wert eines Elements (“Schlüssel”)**

x **Zeiger auf Element**

4.1 Grundoperationen

Aufgabenstellung:

- ***Verwalten einer Menge S von Objekten***
- ***Ausführen von verschiedenen Operationen (s.u.)***

Im Folgenden:

S Menge von Objekten

k Wert eines Elements (“Schlüssel”)

x Zeiger auf Element

NIL spezieller, “leerer” Zeiger

4.1 Grundoperationen

4.1 Grundoperationen

<u>SEARCH(S,k):</u> **“Suche in S nach k”**

<u>SEARCH(S,k):</u> **“Suche in S nach k”**

Durchsuche die Menge S nach einem Element von Wert k.

<u>SEARCH(S,k):</u> **“Suche in S nach k”**

Durchsuche die Menge S nach einem Element von Wert k.

Ausgabe: Zeiger x, falls x existent

<u>SEARCH(S,k):</u> **“Suche in S nach k”**

Durchsuche die Menge S nach einem Element von Wert k.

Ausgabe: Zeiger x, falls x existent
NIL, falls kein Element Wert k hat.

4.1 Grundoperationen

4.1 Grundoperationen

<u>INSERT(S,x):</u> **“Füge x in S ein”**

4.1 Grundoperationen

INSERT(S,x): **“Füge x in S ein”**

Erweitere S um das Element, das unter der Adresse x steht.

4.1 Grundoperationen

DELETE(S,x): **“Entferne x aus S”**

DELETE(S,x): **“Entferne x aus S”**

Lösche das unter der Adresse x stehende Element aus der Menge S.

4.1 Grundoperationen

MINIMUM(S): **“Suche das Minimum in S”**

<u>MINIMUM(S):</u> **“Suche das Minimum in S”**

Finde in S ein Element von kleinstem Wert.

<u>MINIMUM(S):</u> **“Suche das Minimum in S”**

Finde in S ein Element von kleinstem Wert. (Annahme: Die Werte lassen sich vollständig vergleichen!)

<u>MINIMUM(S):</u> **“Suche das Minimum in S”**

Finde in S ein Element von kleinstem Wert. (Annahme: Die Werte lassen sich vollständig vergleichen!)

Ausgabe: Zeiger x auf solch ein Element

4.1 Grundoperationen

MAXIMUM(S): **“Suche das Maximum in S”**

<u>MAXIMUM(S):</u> **“Suche das Maximum in S”**

Finde in S ein Element von größtem Wert.

MAXIMUM(S): **“Suche das Maximum in S”**

Finde in S ein Element von größtem Wert. (Annahme: Die Werte lassen sich vollständig vergleichen!)

MAXIMUM(S): **"Suche das Maximum in S"**

Finde in S ein Element von größtem Wert. (Annahme: Die Werte lassen sich vollständig vergleichen!)

Ausgabe: Zeiger x auf solch ein Element

4.1 Grundoperationen

PREDECESSOR(S,x):
“Finde das nächstkleinere Element”

<u>PREDECESSOR(S,x):</u>
“Finde das nächstkleinere Element”

Für ein in x stehendes Element in S, bestimme ein Element von nächstkleinerem Wert in S.

PREDECESSOR(S,x):
“Finde das nächstkleinere Element”

Für ein in x stehendes Element in S, bestimme ein Element von nächstkleinerem Wert in S.

Ausgabe: Zeiger y auf Element

<u>PREDECESSOR(S,x):</u>
“Finde das nächstkleinere Element”

Für ein in x stehendes Element in S, bestimme ein Element von nächstkleinerem Wert in S.

Ausgabe: Zeiger y auf Element
NIL, falls x Minimum von S angibt

4.1 Grundoperationen

SUCCESSOR(S,x):

“Finde das nächstgrößere Element”

SUCCESSOR(S,x):
“Finde das nächstgrößere Element”

Für ein in x stehendes Element in S, bestimme ein Element von nächstgrößerem Wert in S.

<u>SUCCESSOR(S,x):</u>
“Finde das nächstgrößere Element”

Für ein in x stehendes Element in S, bestimme ein Element von nächstgrößerem Wert in S.

Ausgabe: Zeiger y auf Element

SUCCESSOR(S,x):
“Finde das nächstgrößere Element”

Für ein in x stehendes Element in S, bestimme ein Element von nächstgrößerem Wert in S.

Ausgabe: Zeiger y auf Element
NIL, falls x Maximum von S angibt

4.1 Grundoperationen

4.1 Grundoperationen

Wie nimmt man das vor?

4.1 Grundoperationen

Wie nimmt man das vor?

Wie lange dauert das,
in Abhängigkeit von der Größe von S?

4.1 Grundoperationen

<u>Wie nimmt man das vor?</u>

**<u>Wie lange dauert das,
in Abhängigkeit von der Größe von S?</u>**

Unsortierte Unterlagen:

4.1 Grundoperationen

<u>Wie nimmt man das vor?</u>

**<u>Wie lange dauert das,
in Abhängigkeit von der Größe von S?</u>**

Unsortierte Unterlagen:
Immer alles durchgehen, also: **O(n)**

4.1 Grundoperationen

<u>Wie nimmt man das vor?</u>

**<u>Wie lange dauert das,
in Abhängigkeit von der Größe von S?</u>**

Unsortierte Unterlagen:
Immer alles durchgehen, also: **O(n)**

Sortierte Unterlagen: Geht schneller!

4.1 Grundoperationen

4.1 Grundoperationen

Langsam:

4.1 Grundoperationen

Langsam:

- ***O(n):*** ***lineare Zeit***

4.1 Grundoperationen

Langsam:

- ***O(n):*** ***lineare Zeit***

  Alle Objekte anschauen

4.1 Grundoperationen

Langsam:

- ***O(n):*** ***lineare Zeit***

  Alle Objekte anschauen

Sehr schnell:

4.1 Grundoperationen

Langsam:

- ***O(n):*** ***lineare Zeit***

  Alle Objekte anschauen

Sehr schnell:

- ***O(1):*** ***konstante Zeit***

4.1 Grundoperationen

Langsam:

- ***O(n):*** ***lineare Zeit***

  Alle Objekte anschauen

Sehr schnell:

- ***O(1):*** ***konstante Zeit***

  Immer gleich schnell, egal wie groß S ist.

4.1 Grundoperationen

Langsam:

- ***O(n):*** ***lineare Zeit***

  Alle Objekte anschauen

Sehr schnell:

- ***O(1):*** ***konstante Zeit***

  Immer gleich schnell, egal wie groß S ist.

Schnell:

4.1 Grundoperationen

Langsam:

- ***O(n):*** ***lineare Zeit***

Alle Objekte anschauen

Sehr schnell:

- ***O(1):*** ***konstante Zeit***

Immer gleich schnell, egal wie groß S ist.

Schnell:

- ***O(log n): logarithmische Zeit***

4.1 Grundoperationen

Langsam:

- ***O(n):*** ***lineare Zeit***

  Alle Objekte anschauen

Sehr schnell:

- ***O(1):*** ***konstante Zeit***

  Immer gleich schnell, egal wie groß S ist.

Schnell:

- ***O(log n):*** ***logarithmische Zeit***

  Wiederholtes Halbieren

Mehr demnächst!

s.fekete@tu-bs.de