

Kapitel 3.9: Eigenschaften von DFS und BFS

*Algorithmen und Datenstrukturen
WS 2019/20*

Prof. Dr. Sándor Fekete

3.9 DFS vs. BFS

Einfach gesagt:

- *DFS ist eine bestmögliche individuelle Suchstrategie mit lokaler Information.*
- *BFS ist eine bestmögliche kooperative Suchstrategie mit globaler Information.*



3.9 DFS vs. BFS

Einfach gesagt:

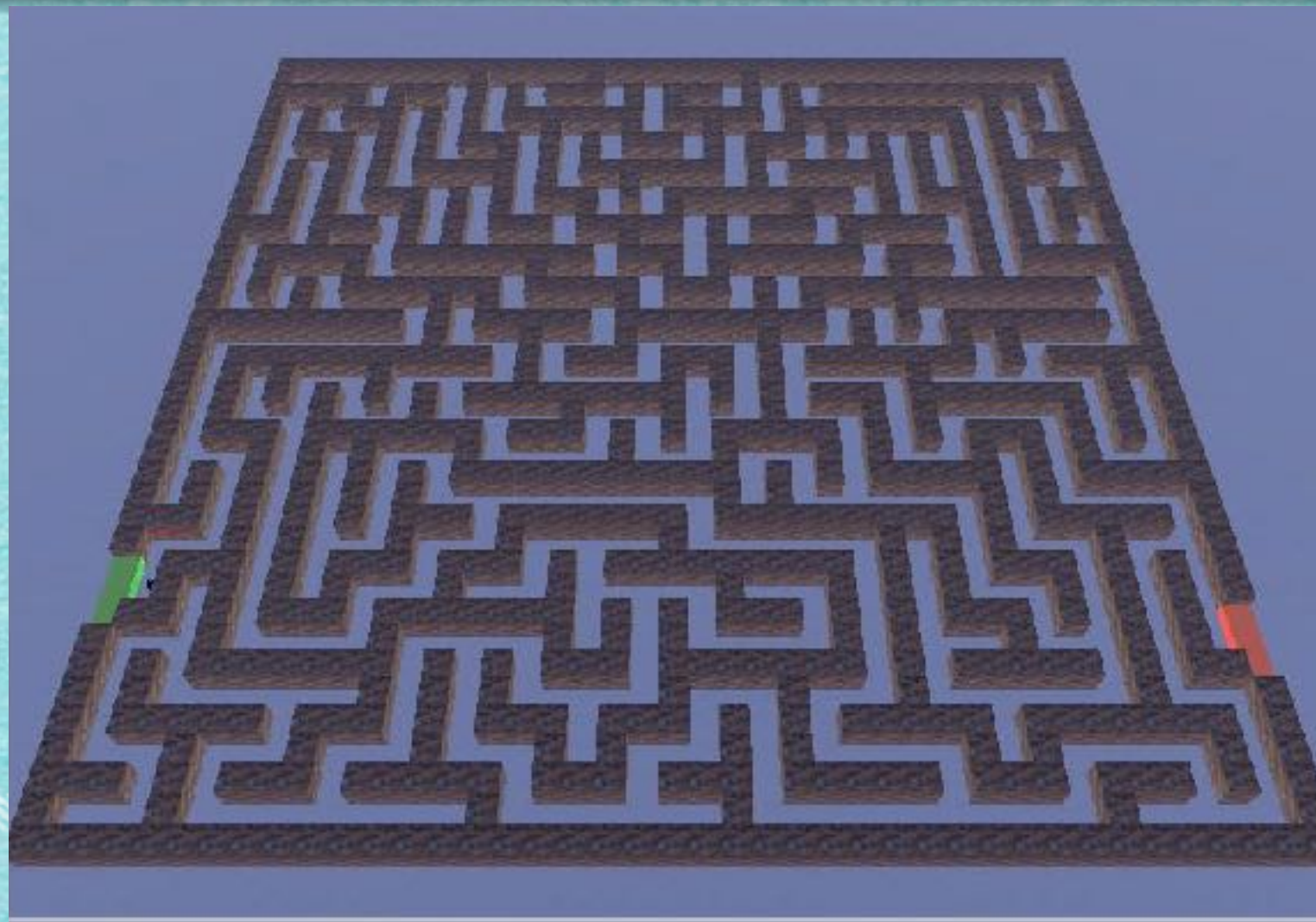
- *DFS ist eine bestmögliche individuelle Suchstrategie mit lokaler Information.*
- *BFS ist eine bestmögliche kooperative Suchstrategie mit globaler Information.*

Konkret:

- *DFS ist gut geeignet für die Suche nach einem Ausweg aus einem Labyrinth.*
- *BFS ist gut geeignet für die Suche nach kürzesten Wegen in einem Graphen.*

Satz 3.16 (Lokale Suche mit DFS)

- (1) DFS findet in jedem zusammenhängenden Graphen mit n Knoten einen Weg der Länge höchstens $2n-1$, der alle Knoten besucht.*
- (2) Für jede lokale Suchstrategie gibt es einen Graphen mit n Knoten, so dass der letzte Knoten erst nach einer Weglänge von $2n-1$ besucht wird.*



Satz 3.16 (Lokale Suche mit DFS)

- (1) DFS findet in jedem zusammenhängenden Graphen mit n Knoten einen Weg der Länge höchstens $2n-1$, der alle Knoten besucht.*
- (2) Für jede lokale Suchstrategie gibt es einen Graphen mit n Knoten, so dass der letzte Knoten erst nach einer Weglänge von $2n-1$ besucht wird.*

Beweis: Übung!

Algorithmus 3.7

3.9 BFS?

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 $R := \{s\}$, $Y := \{s\}$, $T := \emptyset$
2. WHILE ($R \neq \emptyset$) DO {
 - 2.1. wähle Element $v \in R$
 - 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 $w \in V \setminus R$ mit $e = \{v, w\} \in E$;
 - 2.3.2. setze $R := R \cup \{w\}$, $Y := Y \cup \{w\}$, $T := T \cup \{e\}$;}

Auf die Schnelle mit der Welle

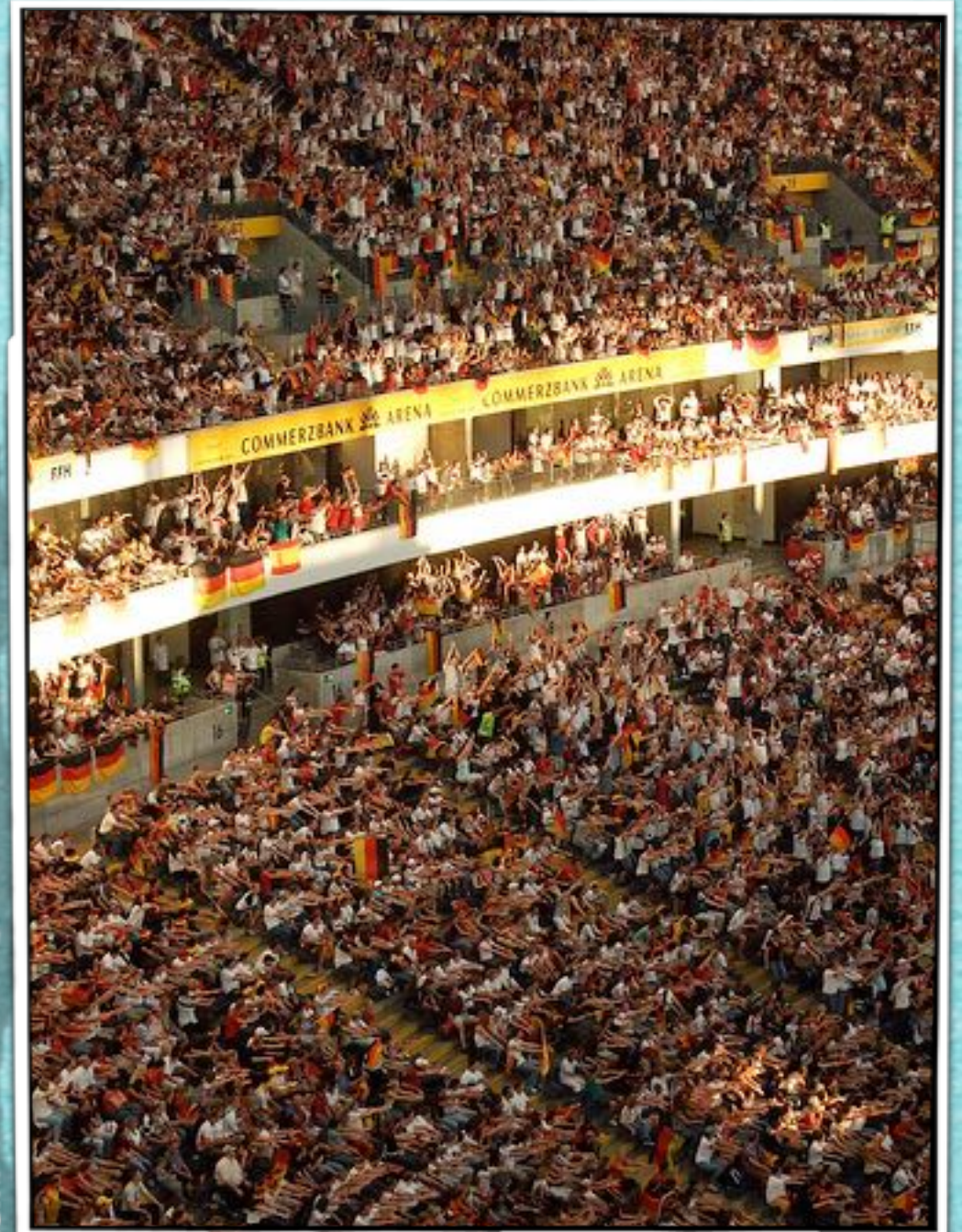
A. LOS bei „NULL“

B. Bis „ANGEKOMMEN!“:

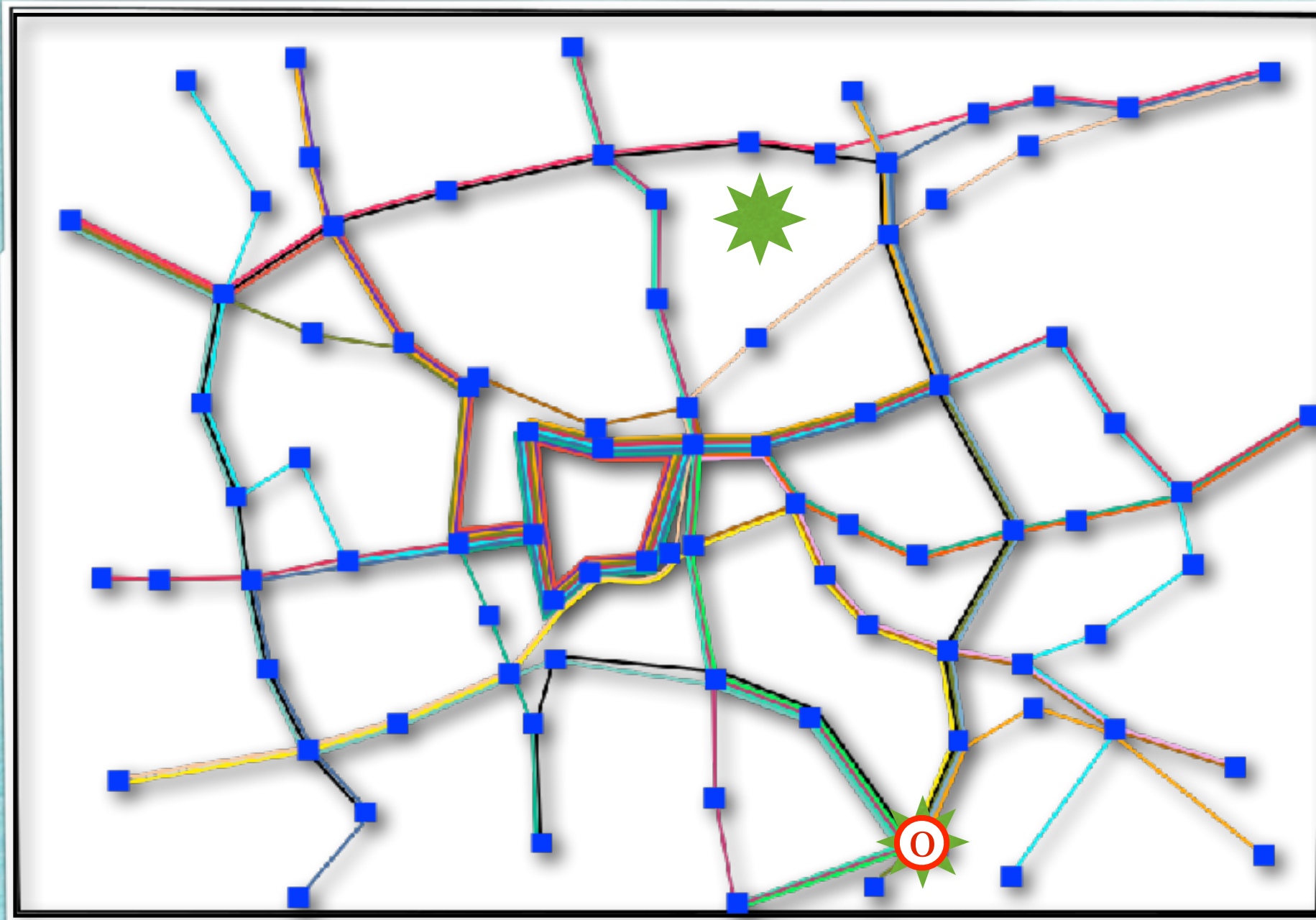
- Solange du noch nicht aufgestanden warst:
 - ▶ Wenn ein oder mehrere direkte Nachbarn aufstehen:
 1. Einen dieser Nachbarn merken
 2. In der nächsten Runde:
 - 2.1. aufstehen
 - 2.2. Zahl merken
 3. In der übernächsten Runde hinsetzen

C. Nach „ANGEKOMMEN!“:

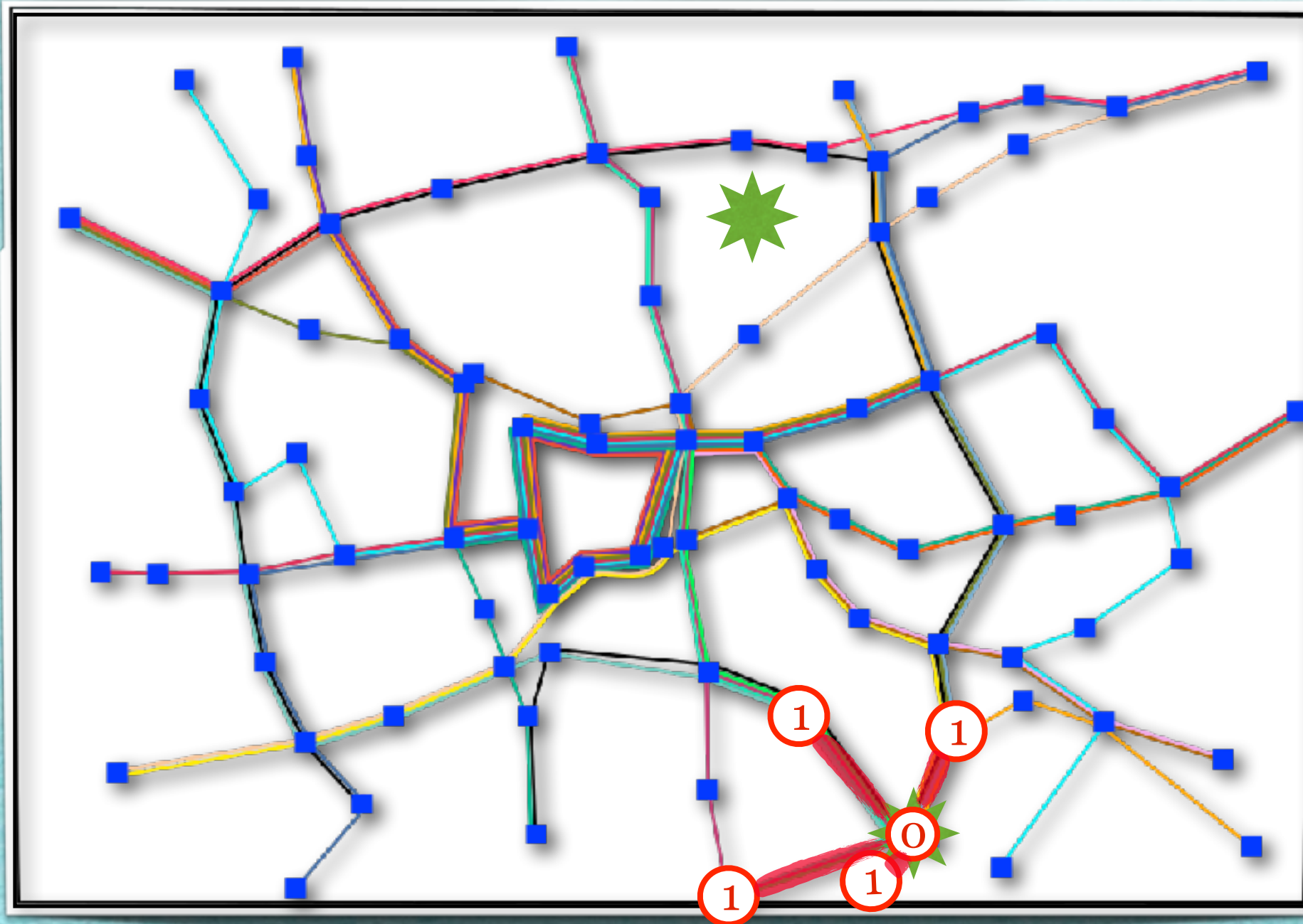
- Auf gemerkten Nachbarn zeigen



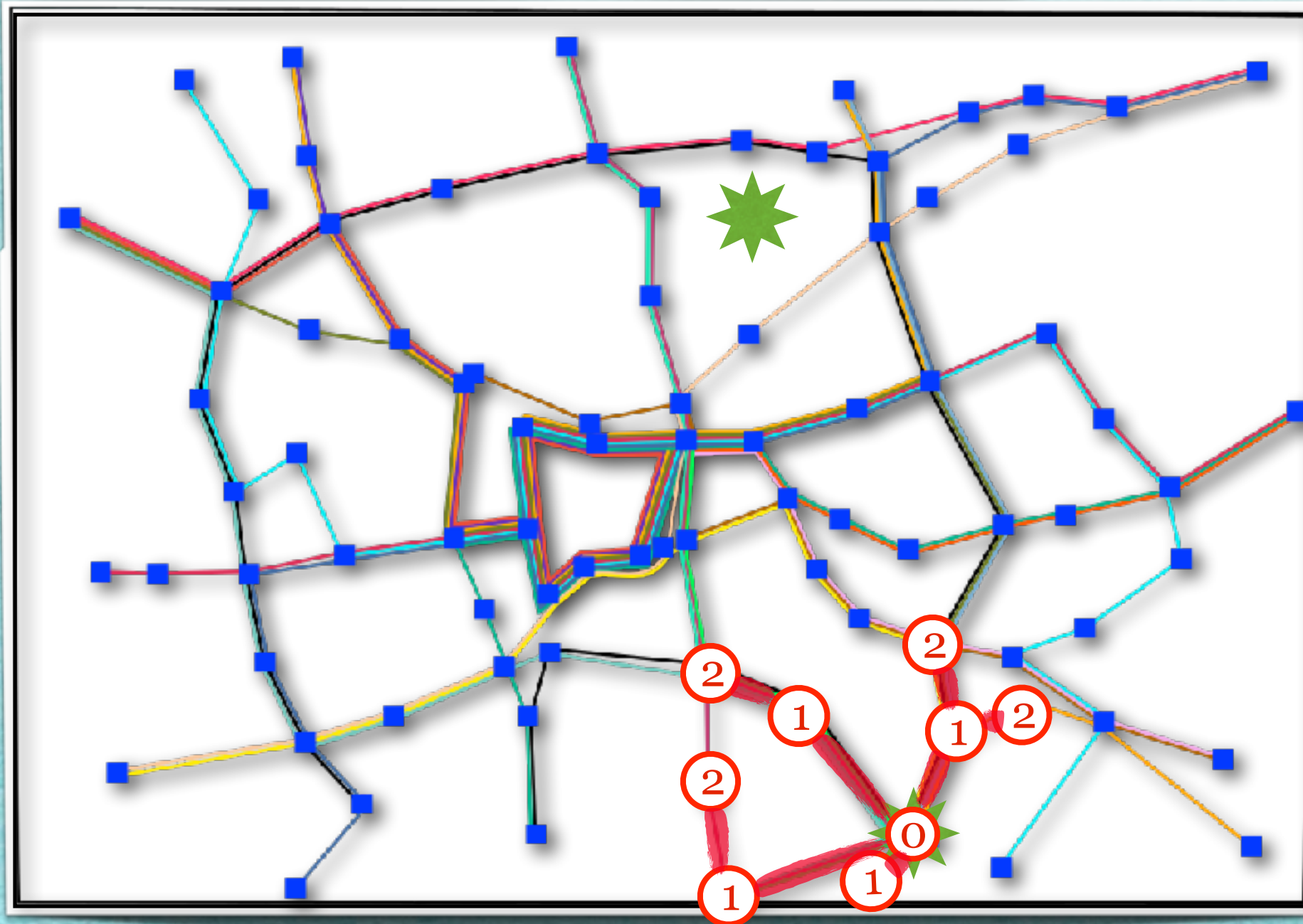
Wellenreiten in Graphen



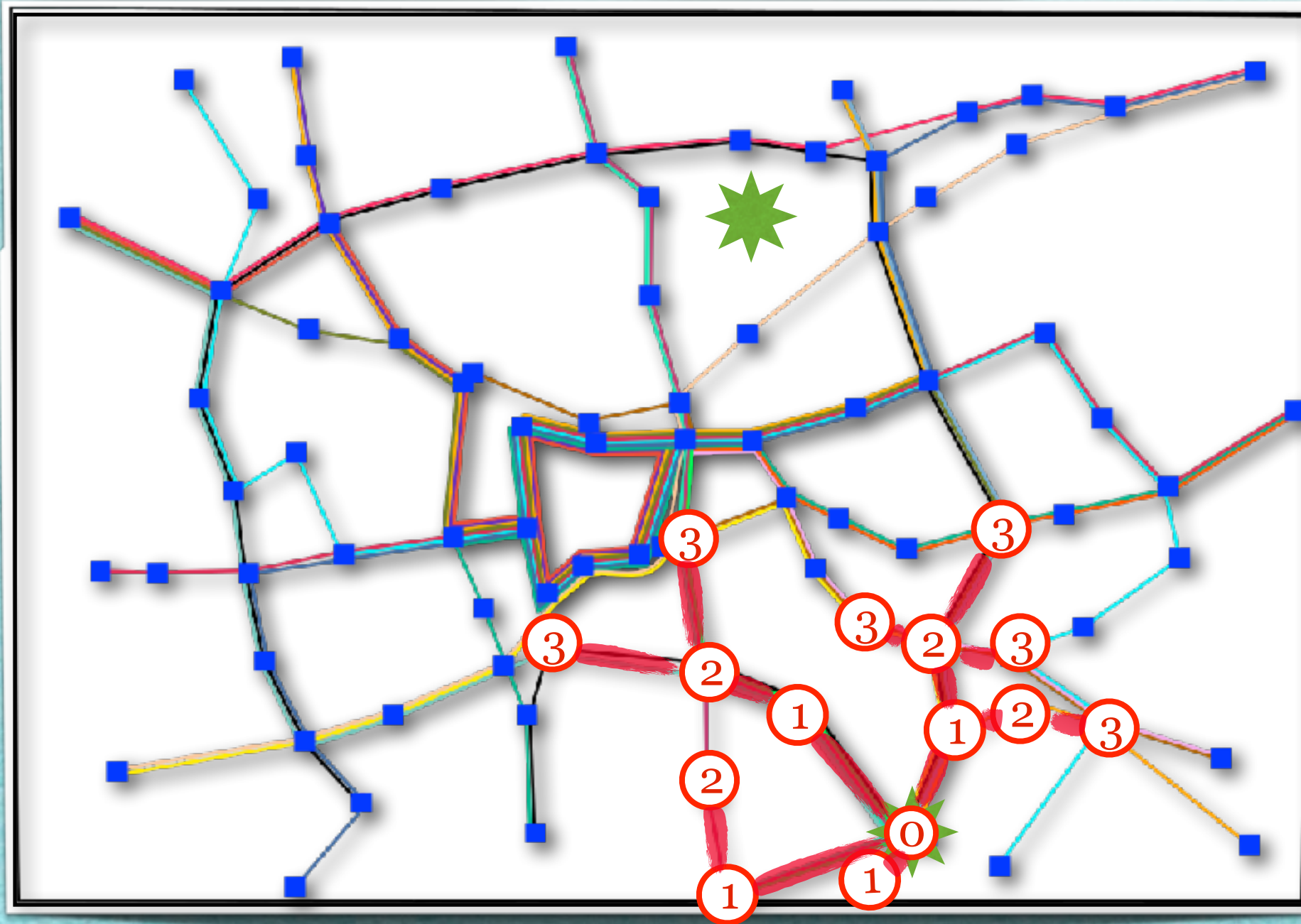
Wellenreiten in Graphen



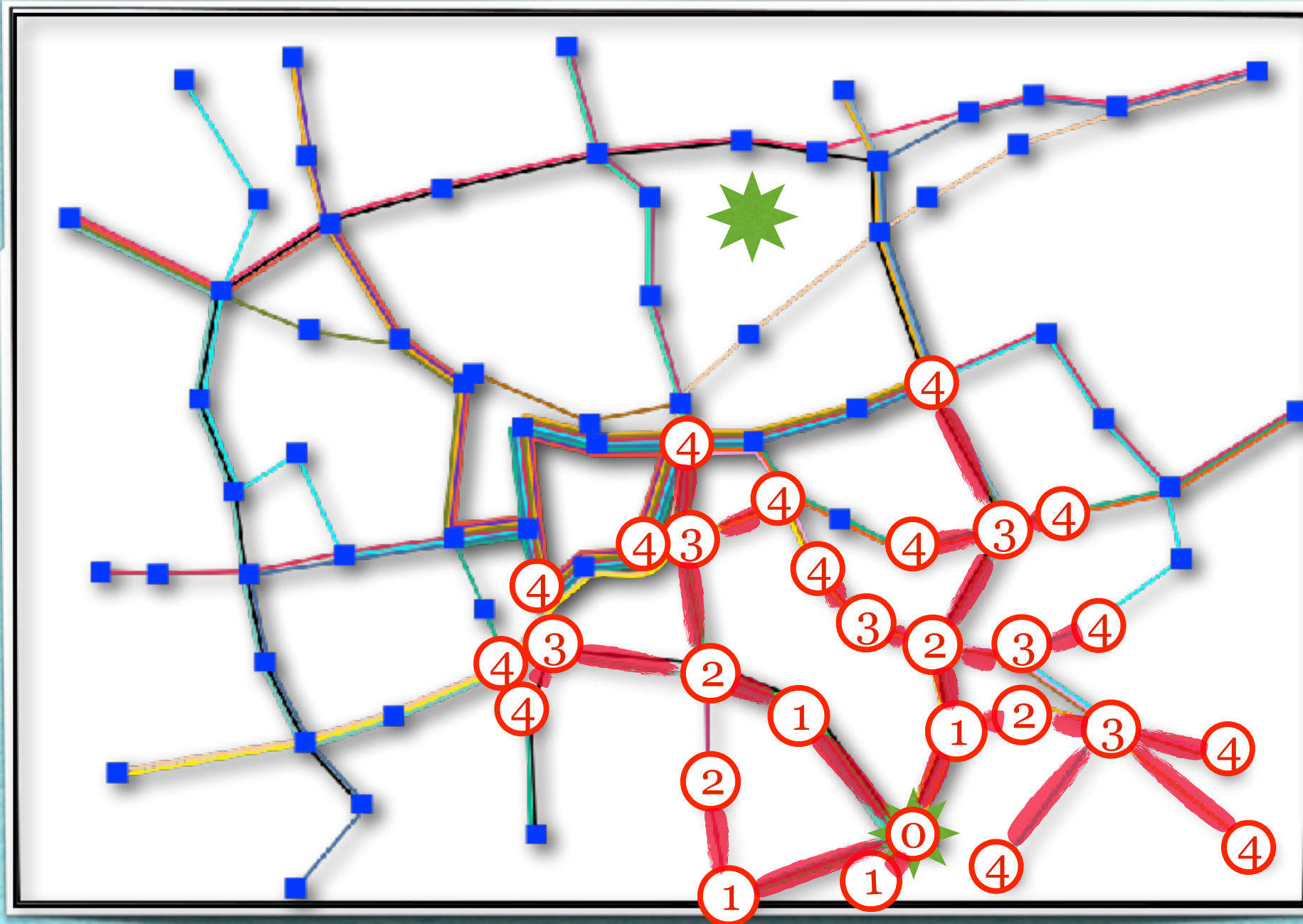
Wellenreiten in Graphen



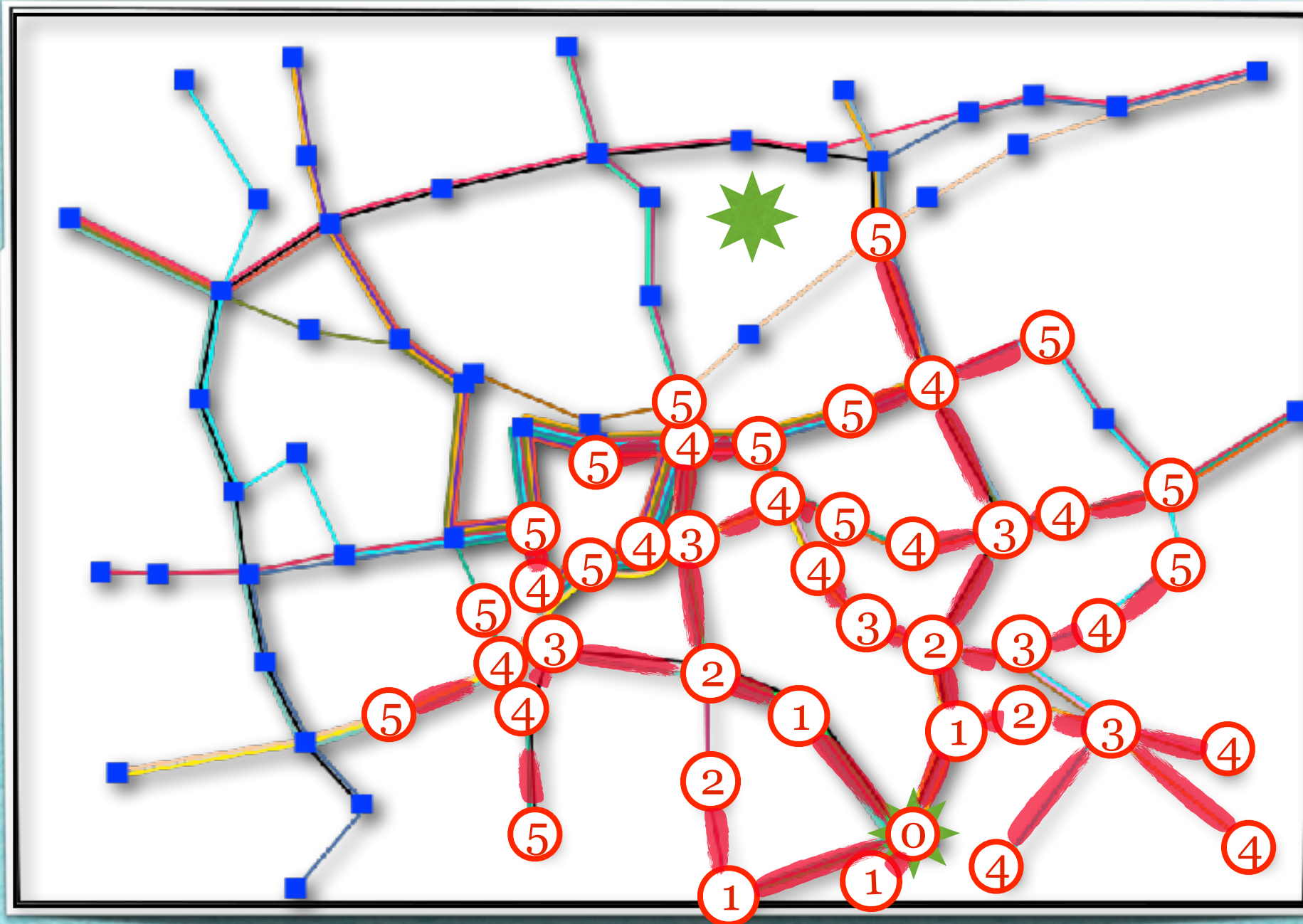
Wellenreiten in Graphen



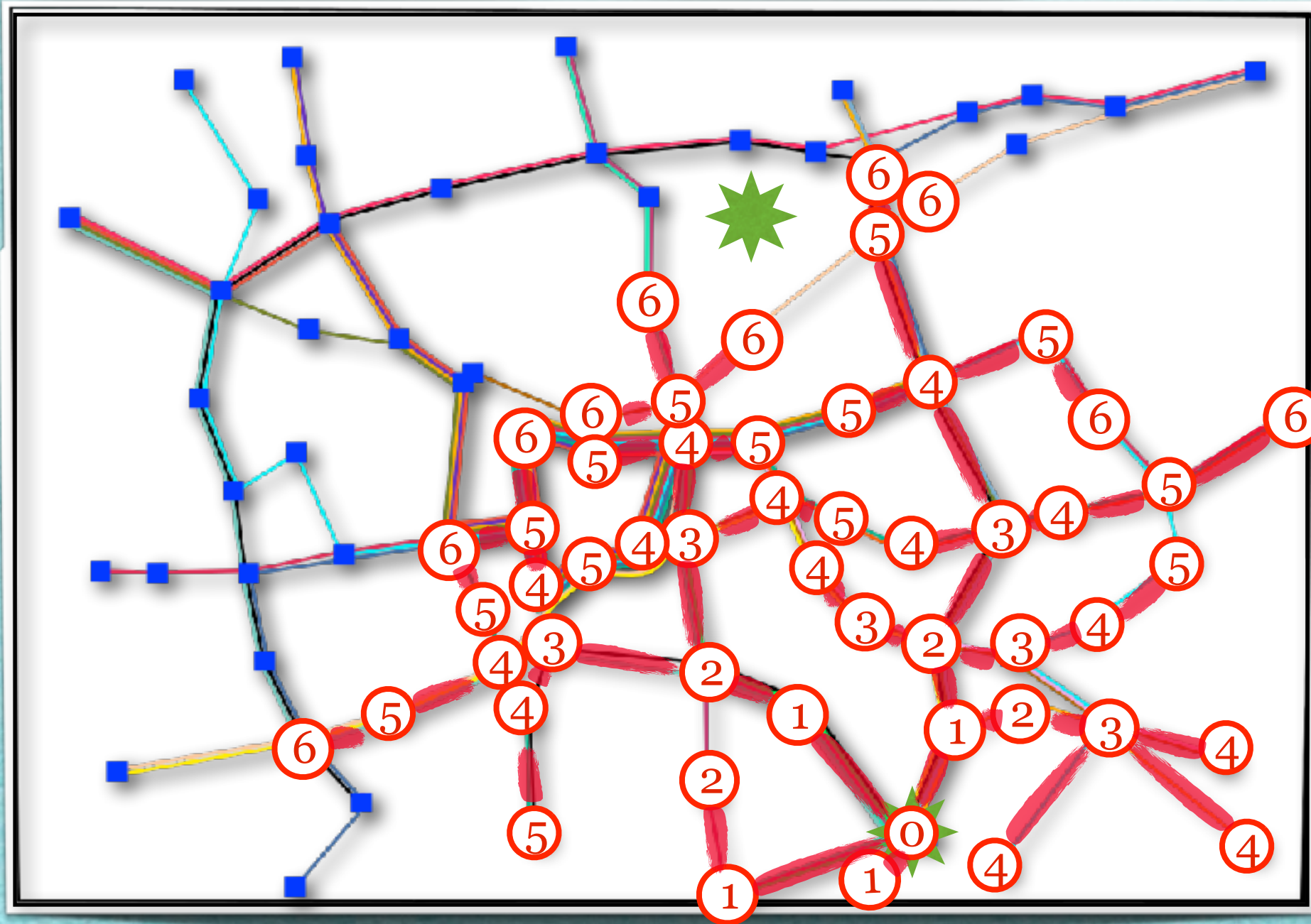
Wellenreiten in Graphen



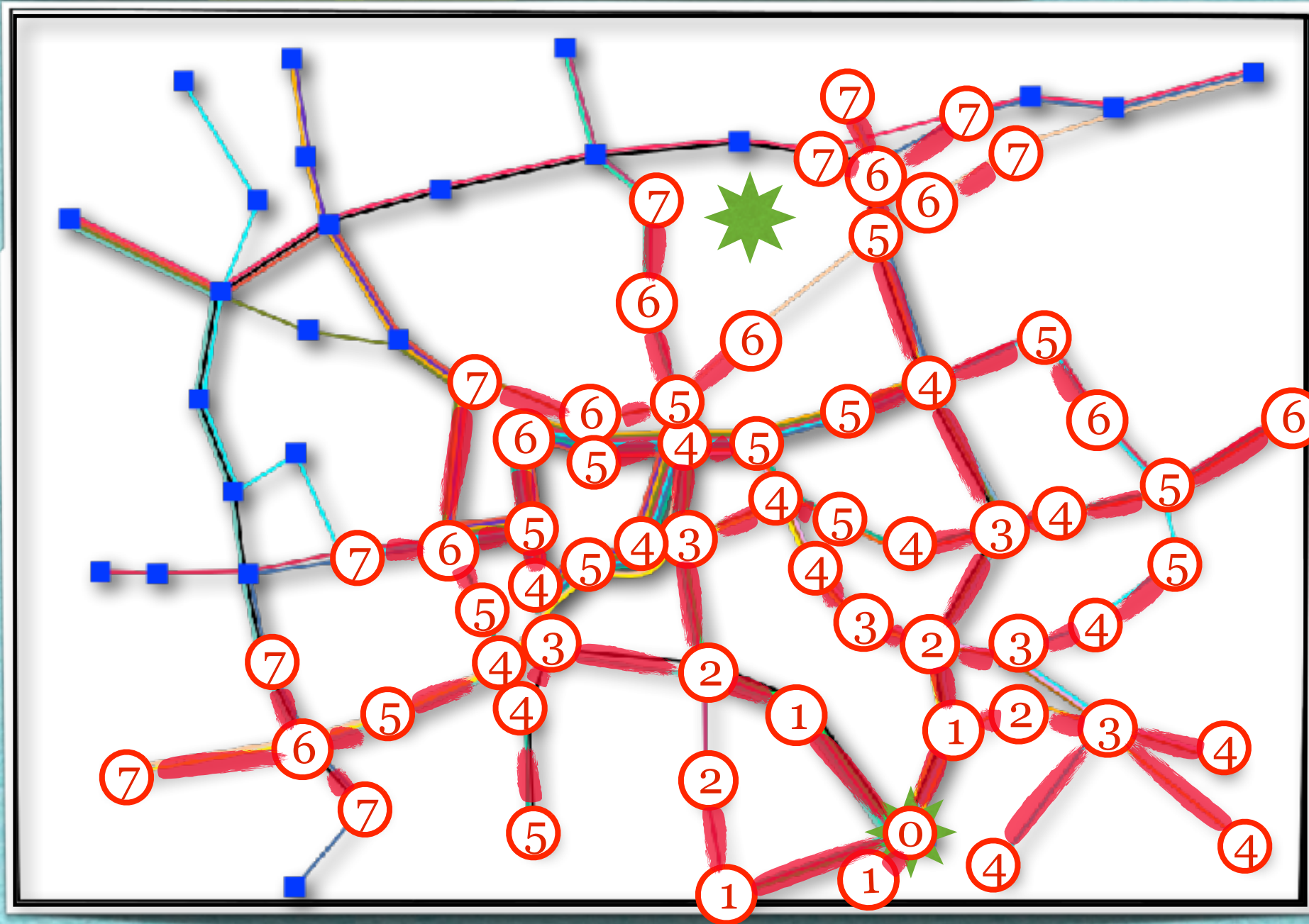
Wellenreiten in Graphen



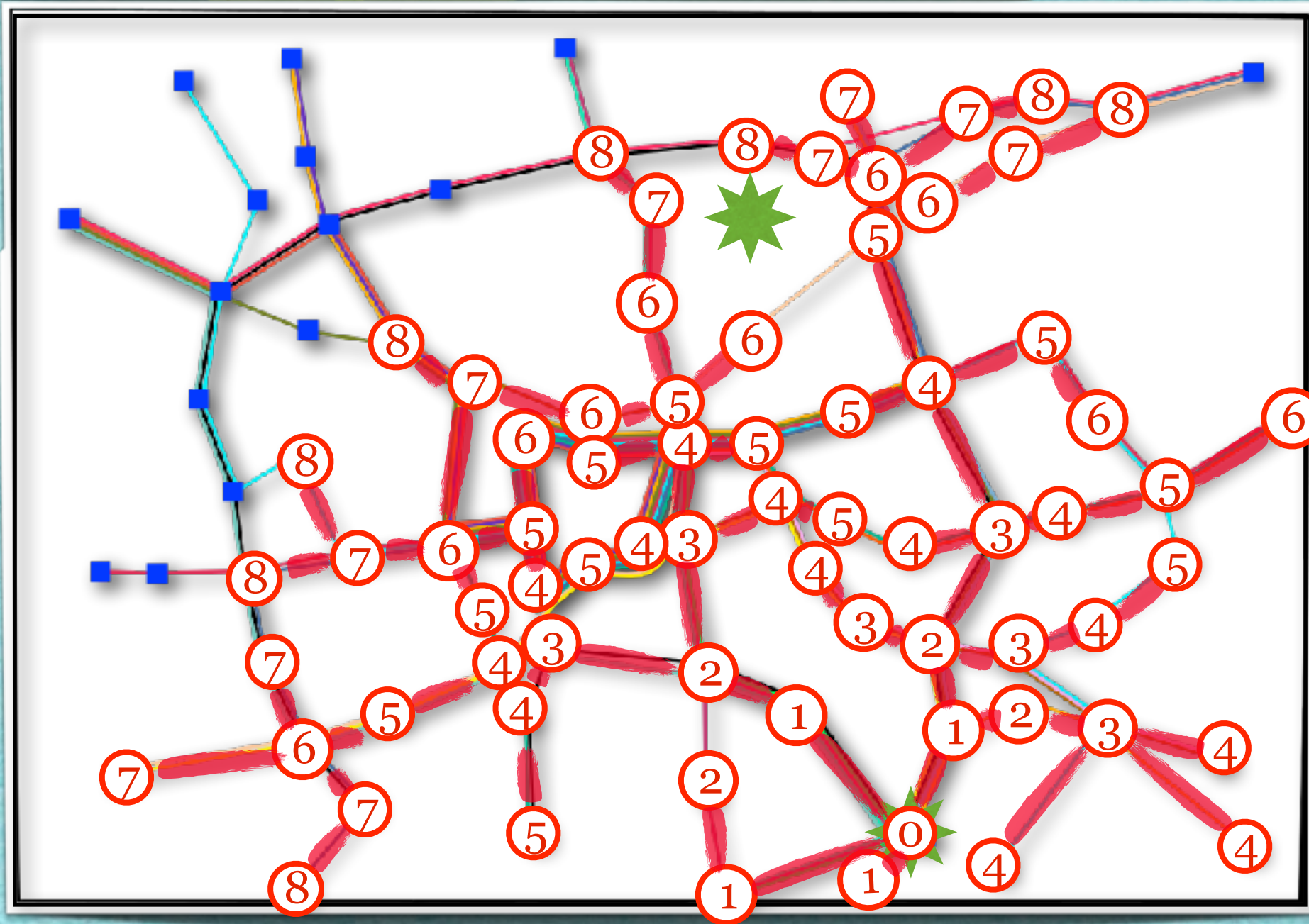
Wellenreiten in Graphen



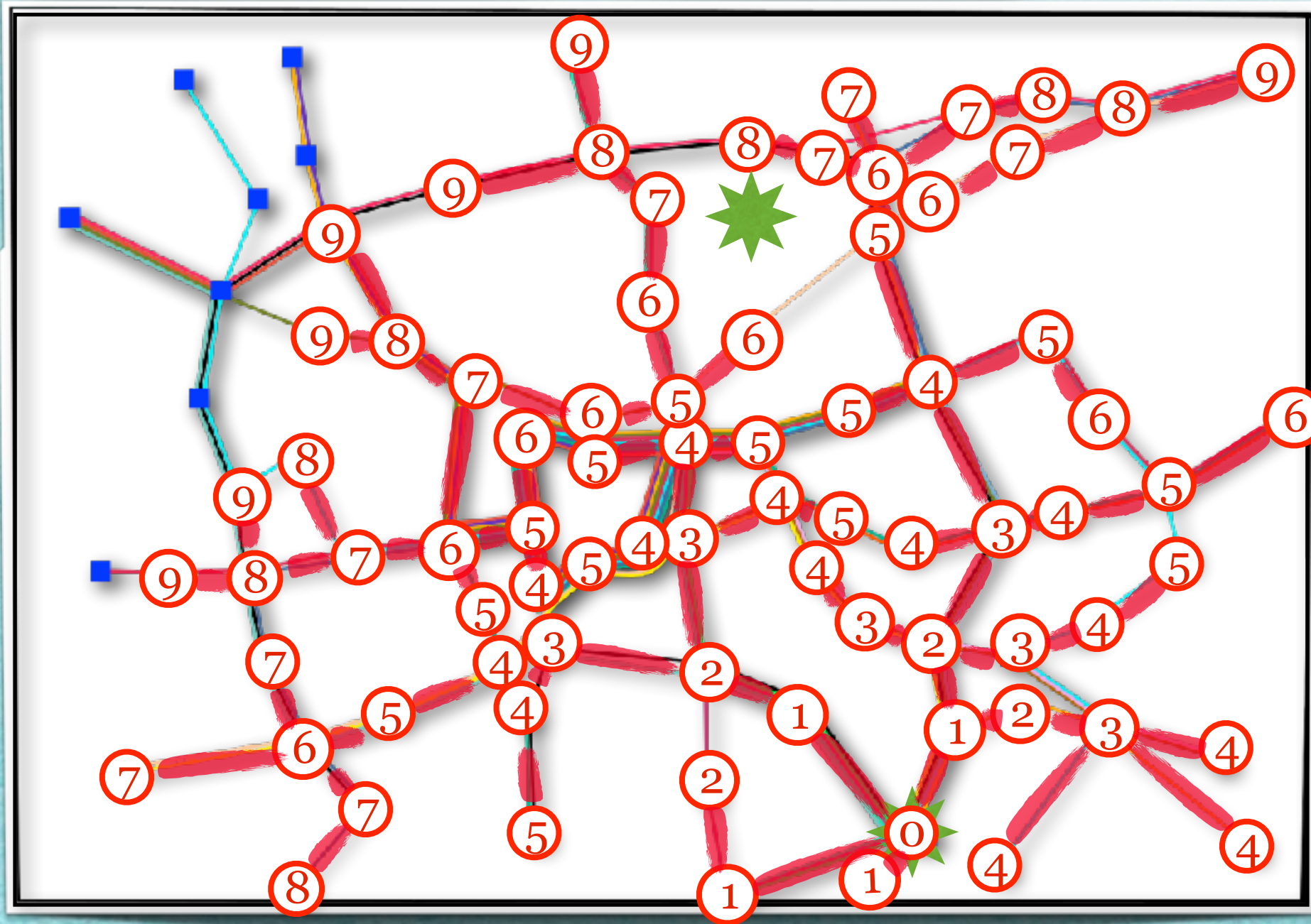
Wellenreiten in Graphen



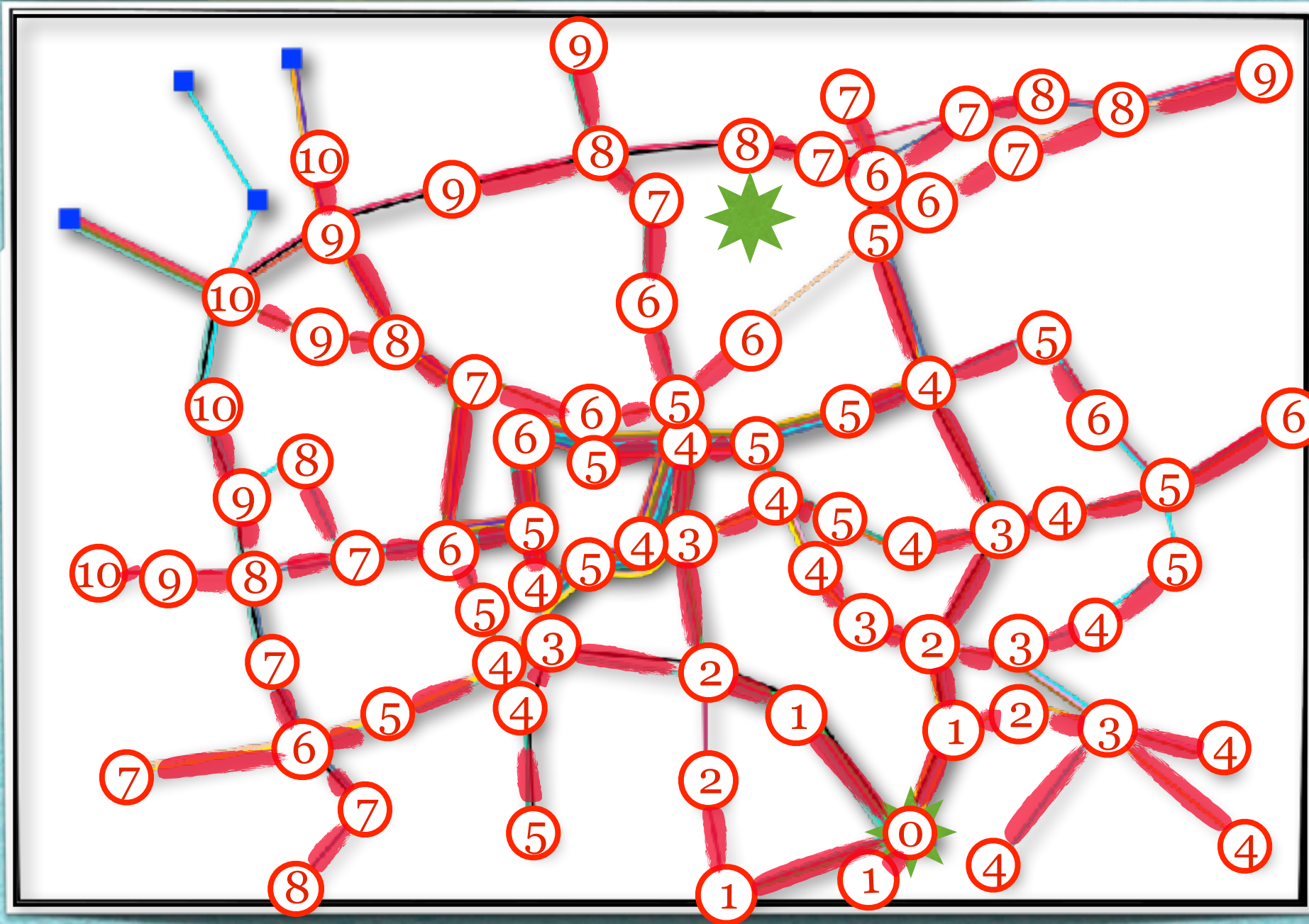
Wellenreiten in Graphen



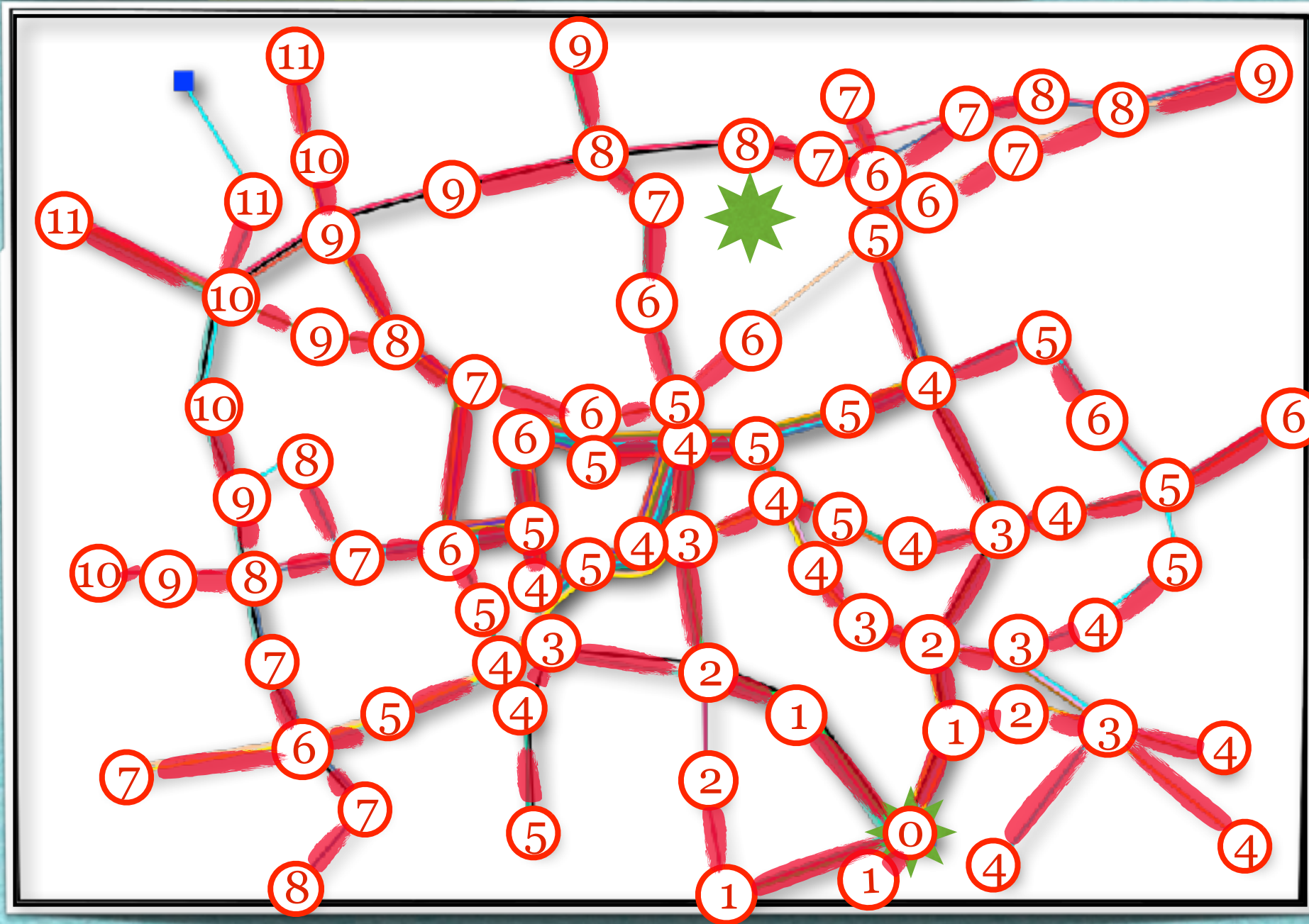
Wellenreiten in Graphen



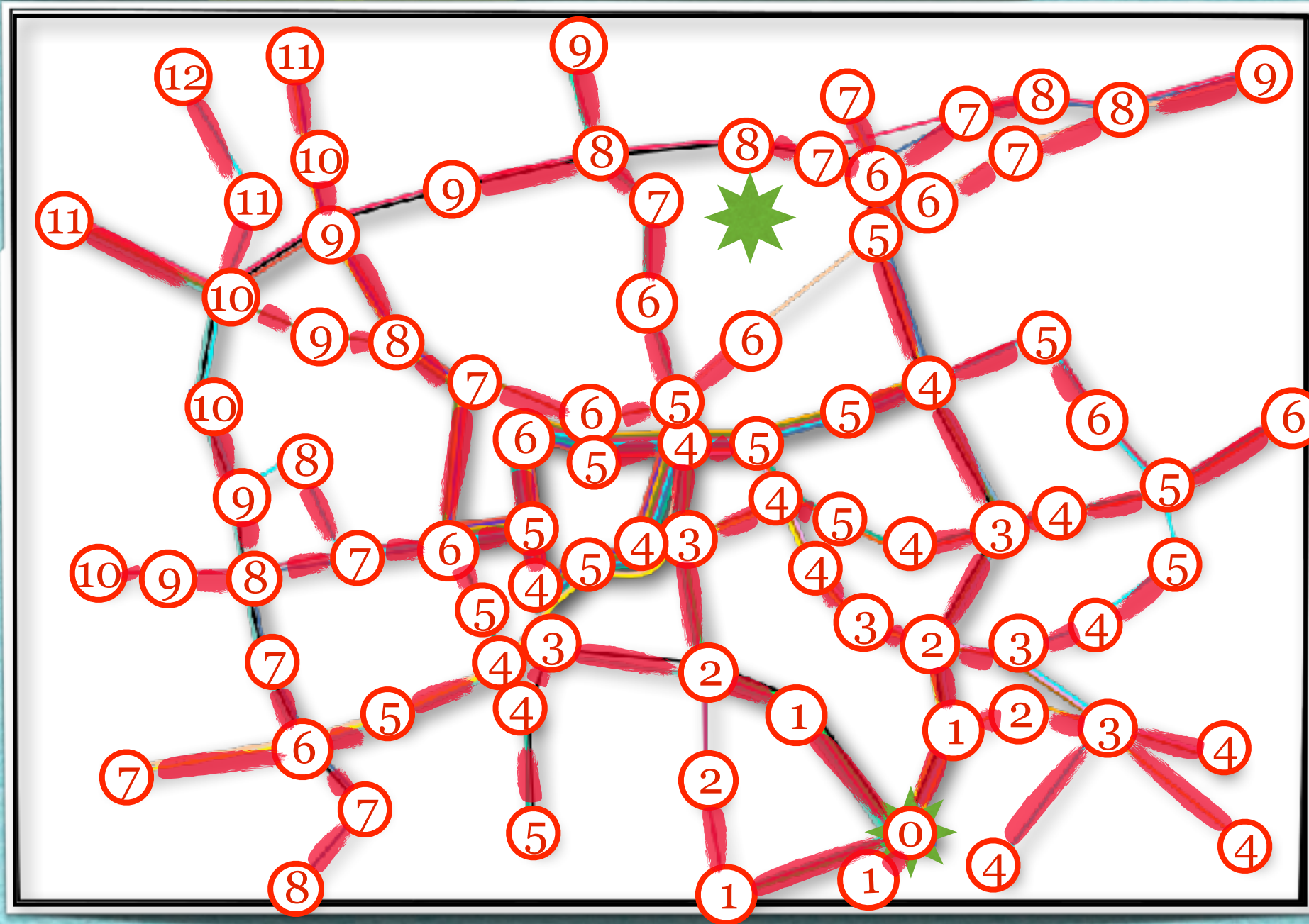
Wellenreiten in Graphen



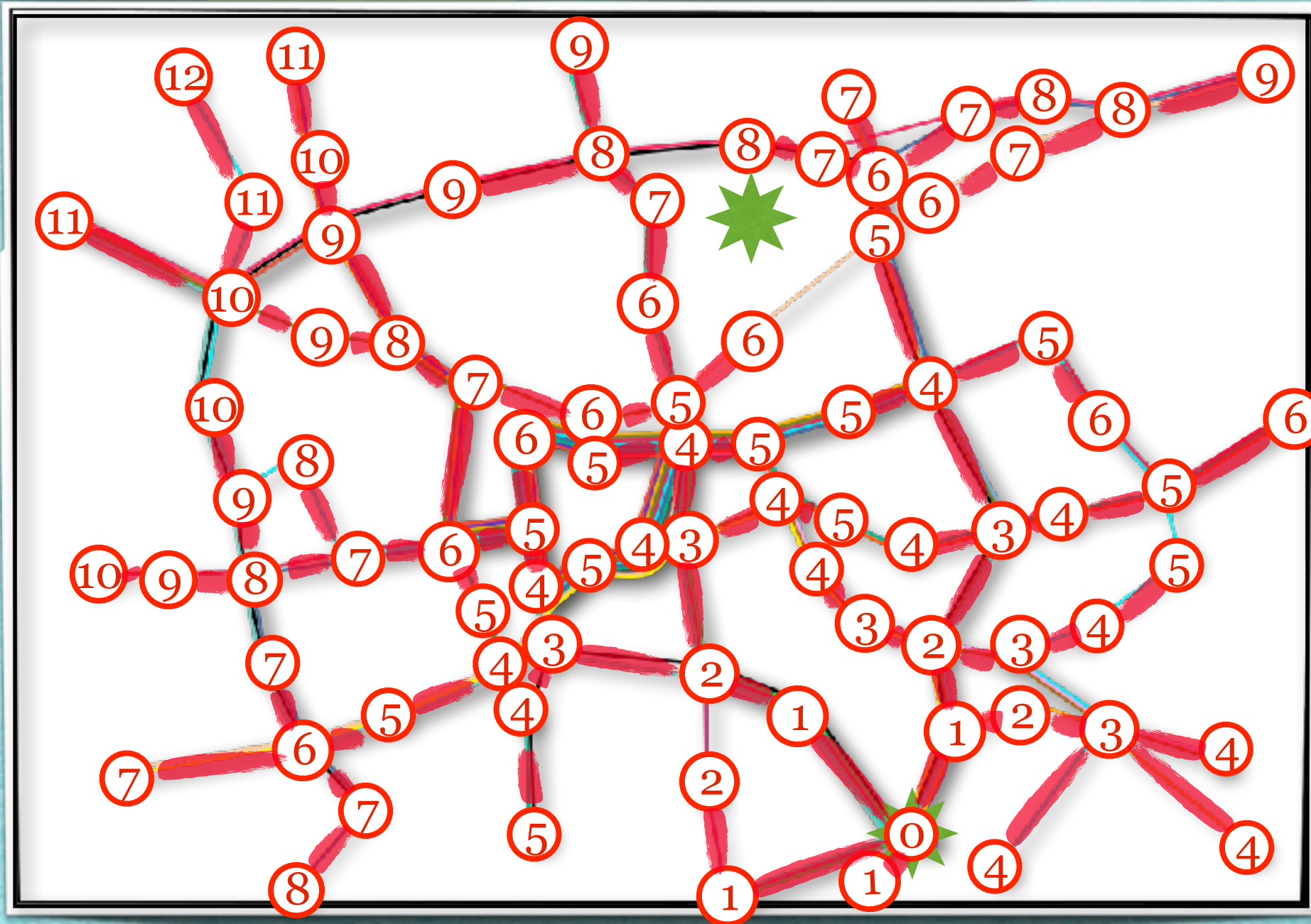
Wellenreiten in Graphen



Wellenreiten in Graphen

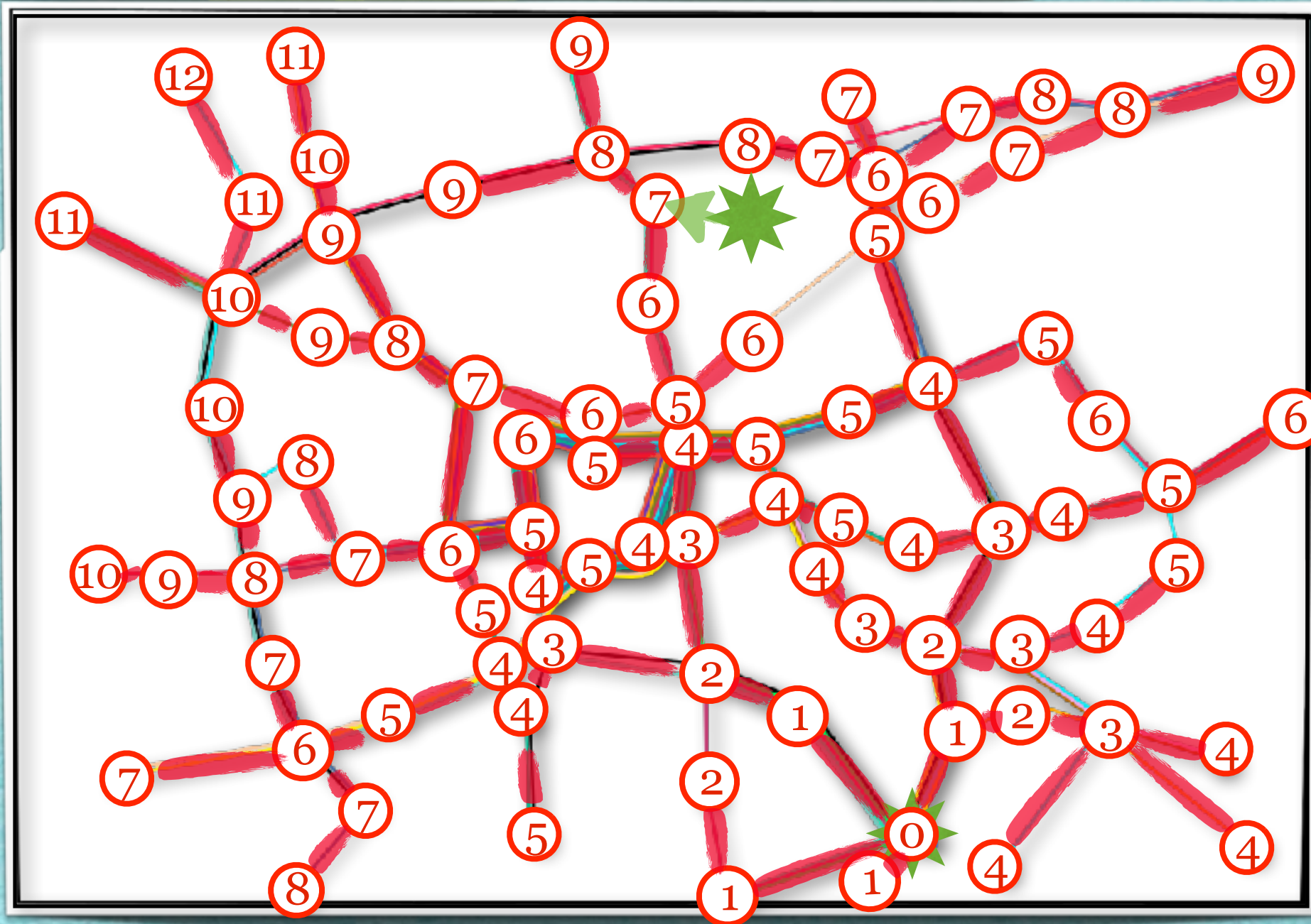


Wellenreiten in Graphen



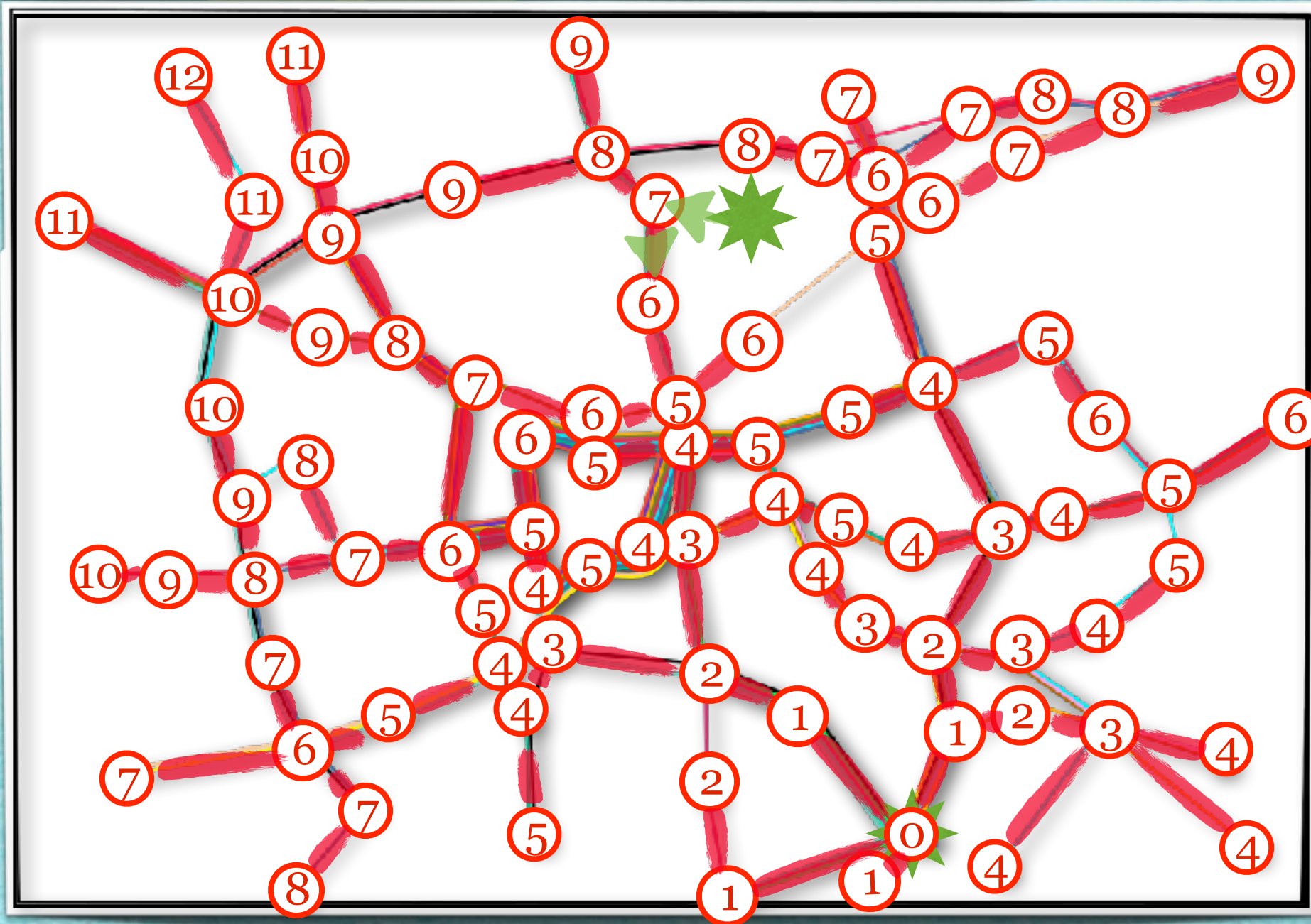
Breitensuche

Wellenreiten in Graphen



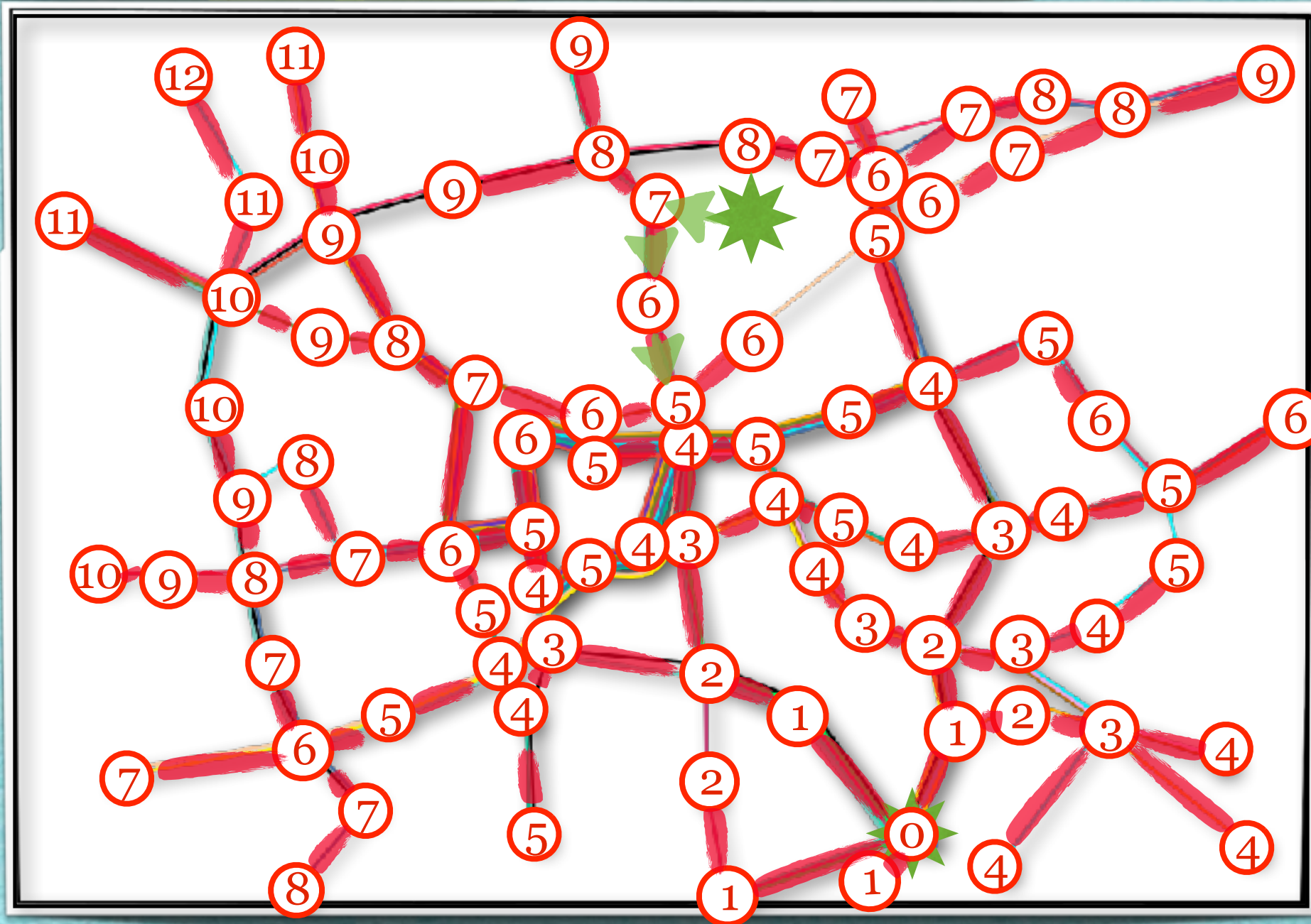
Breitensuche

Wellenreiten in Graphen



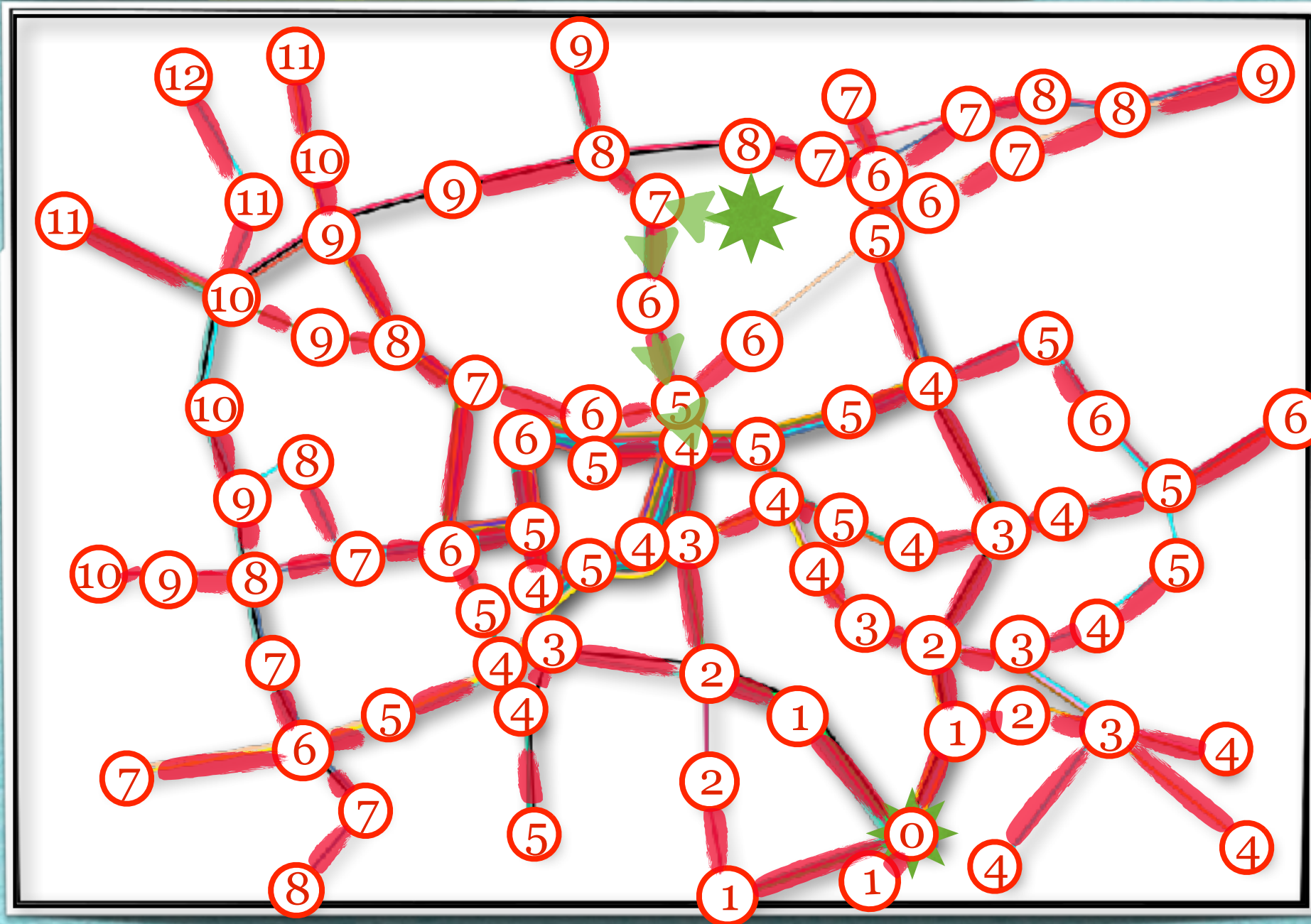
Breitensuche

Wellenreiten in Graphen



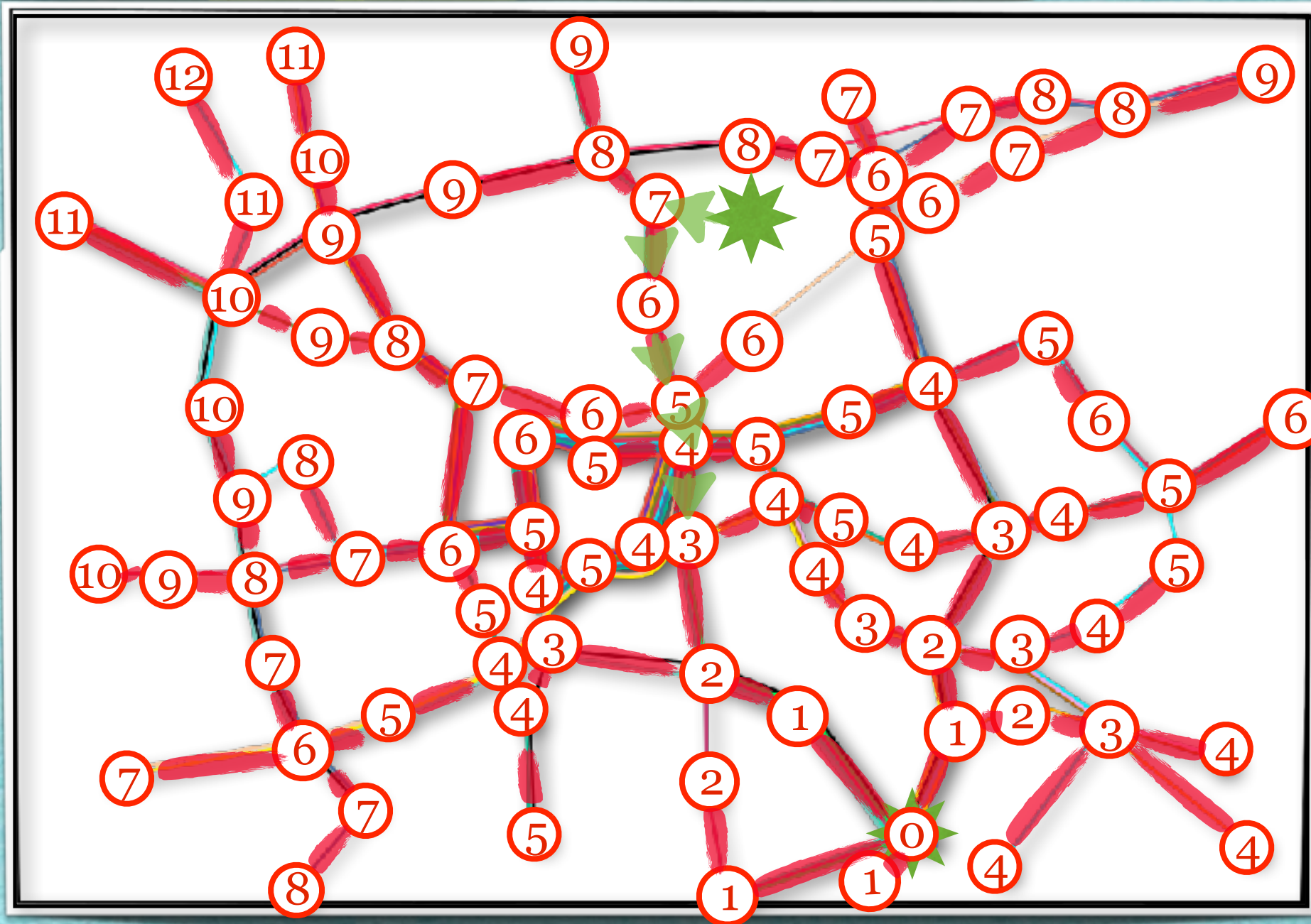
Breitensuche

Wellenreiten in Graphen



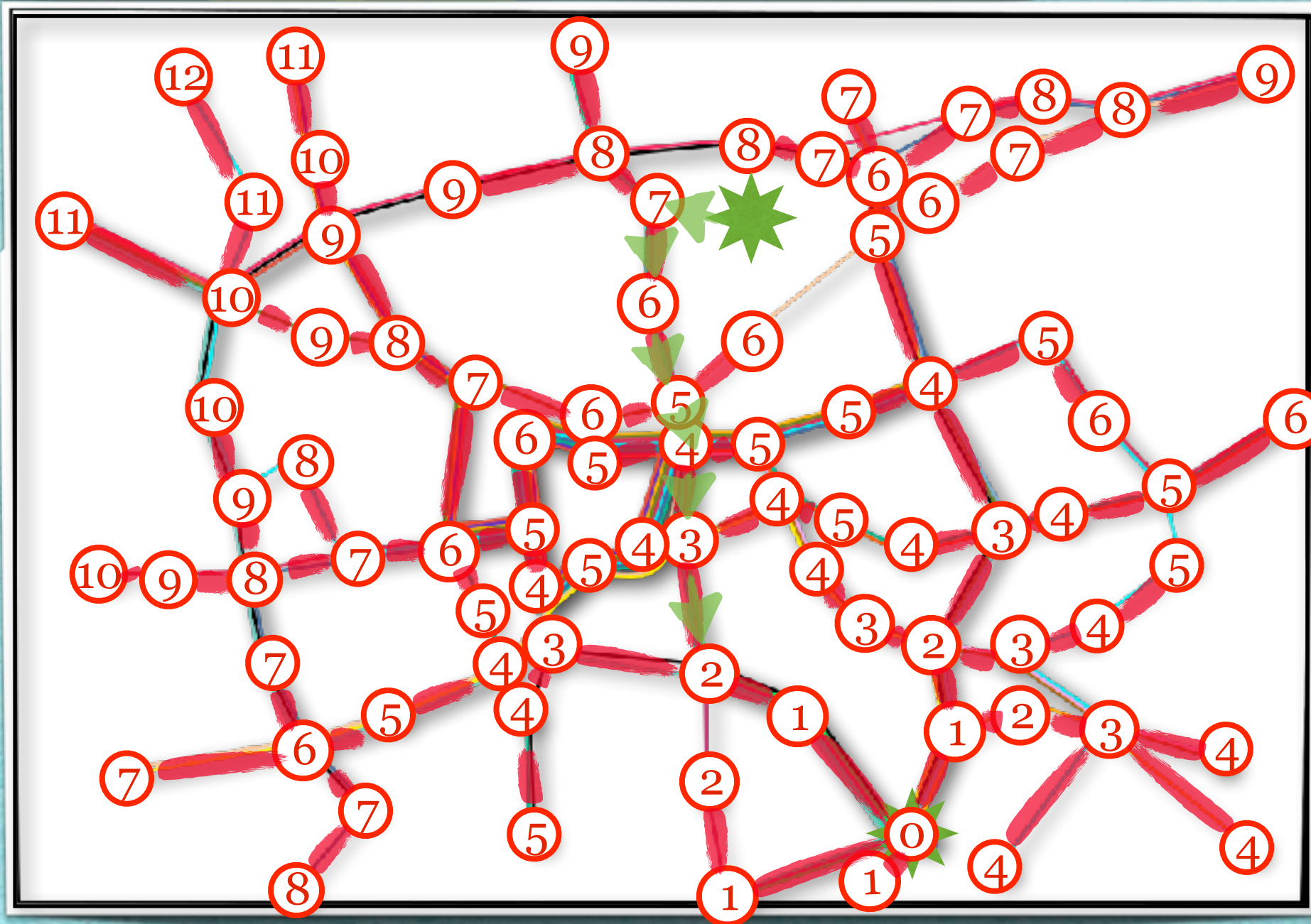
Breitensuche

Wellenreiten in Graphen



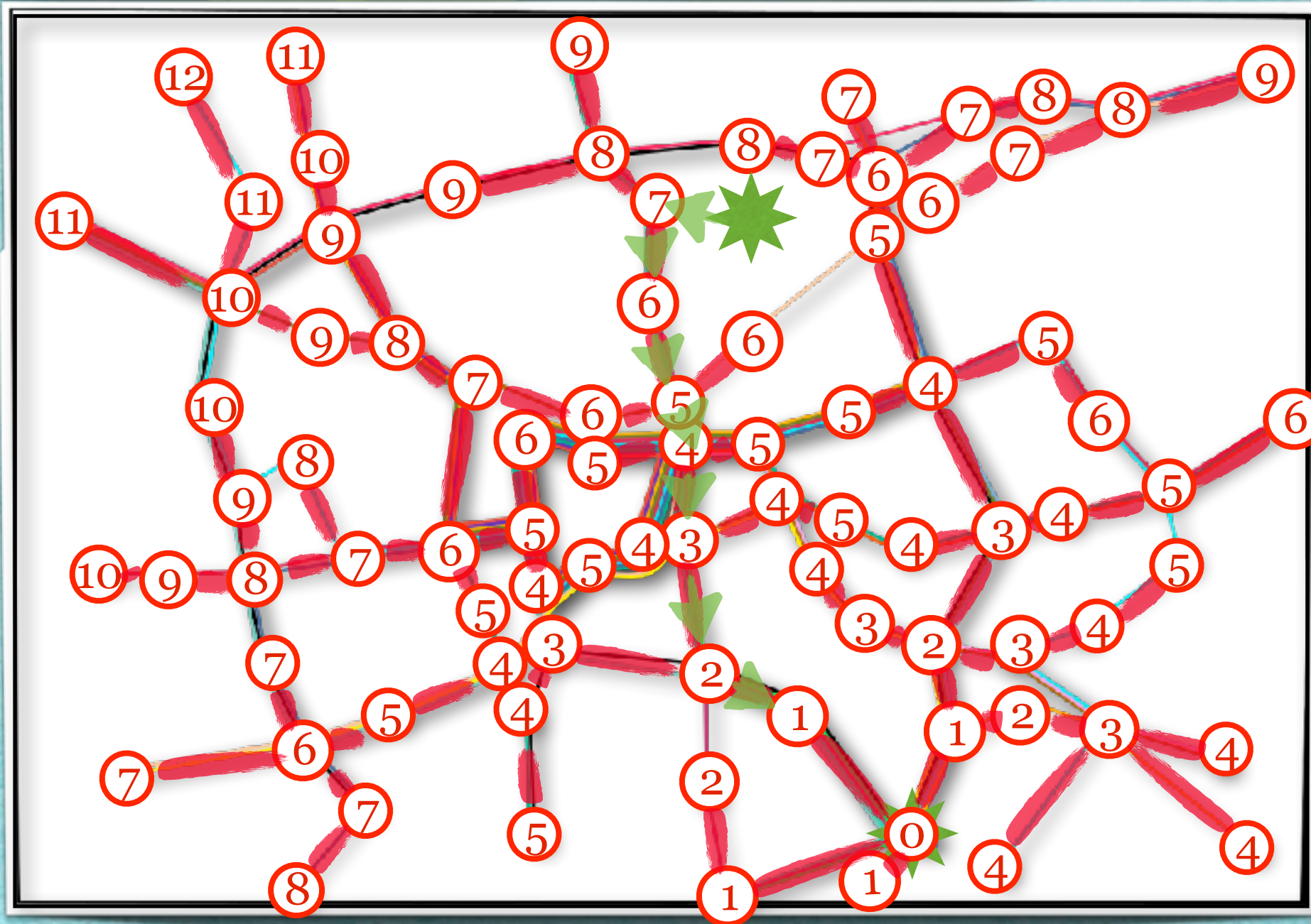
Breitensuche

Wellenreiten in Graphen



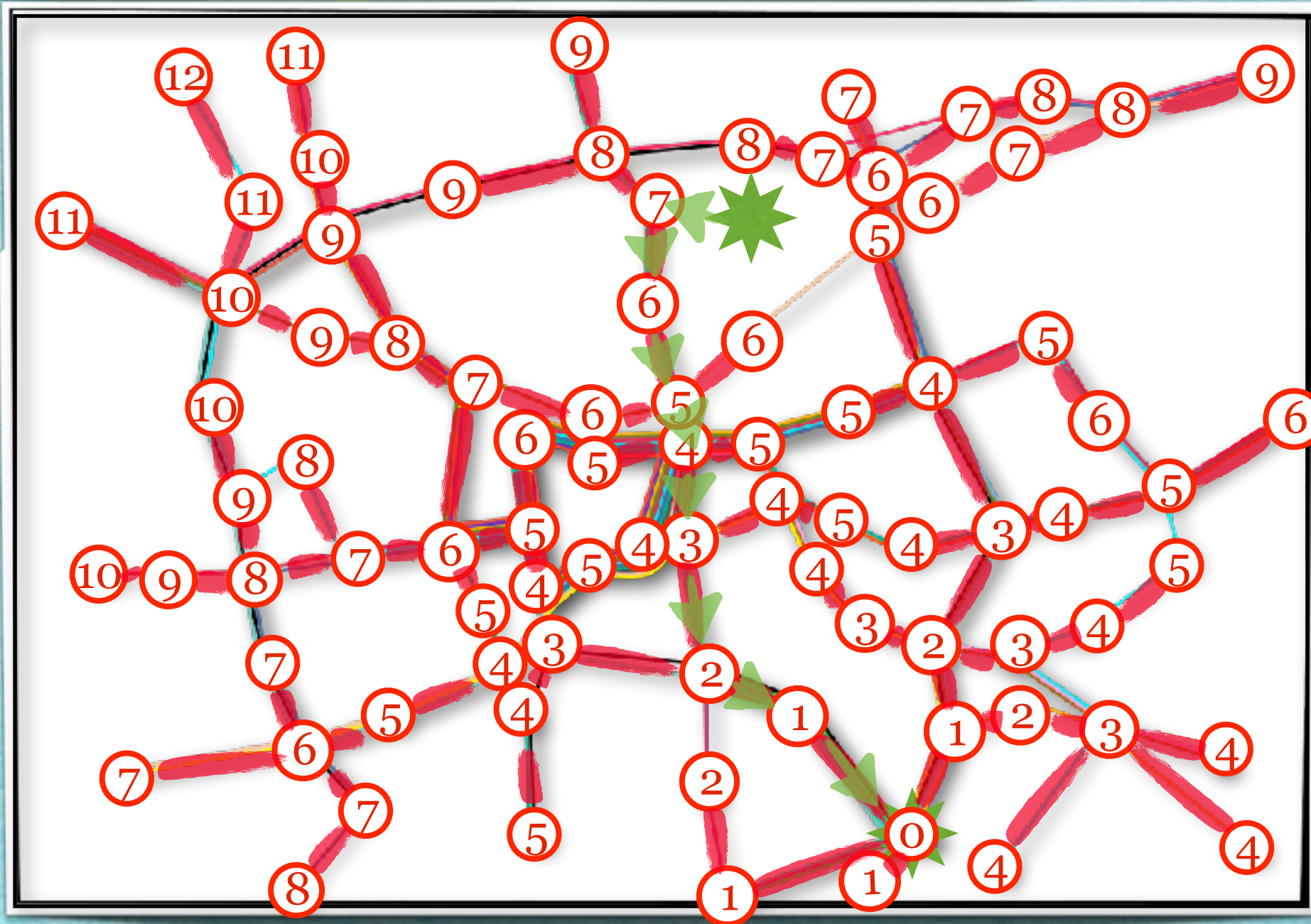
Breitensuche

Wellenreiten in Graphen



Breitensuche

Wellenreiten in Graphen



Breitensuche

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 $R := \{s\}$, $Y := \{s\}$, $T := \emptyset$
2. WHILE ($R \neq \emptyset$) DO {
 - 2.1. wähle Element $v \in R$
 - 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 $w \in V \setminus R$ mit $e = \{v, w\} \in E$;
 - 2.3.2. setze $R := R \cup \{w\}$, $Y := Y \cup \{w\}$, $T := T \cup \{e\}$;
 - 2.3.3. setze $l(w) := l(v) + 1$}

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.**

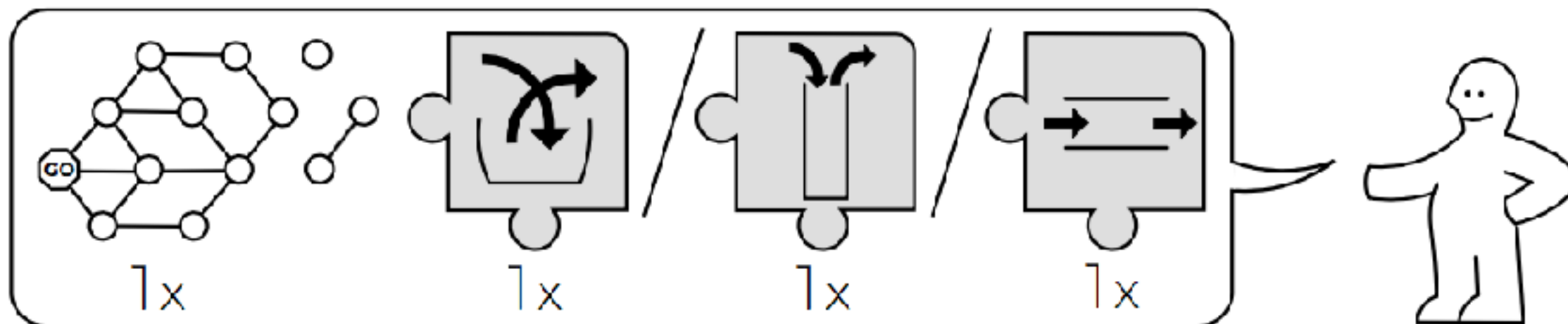
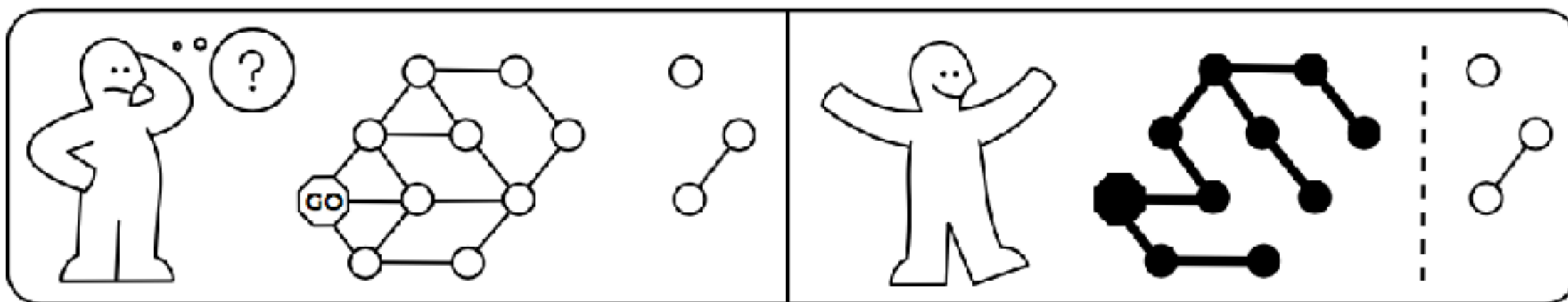
Mehr Details!

s.fekete@tu-bs.de

GRÅPH SCÄN

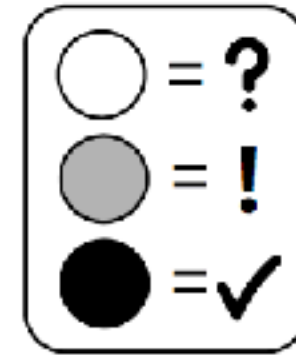
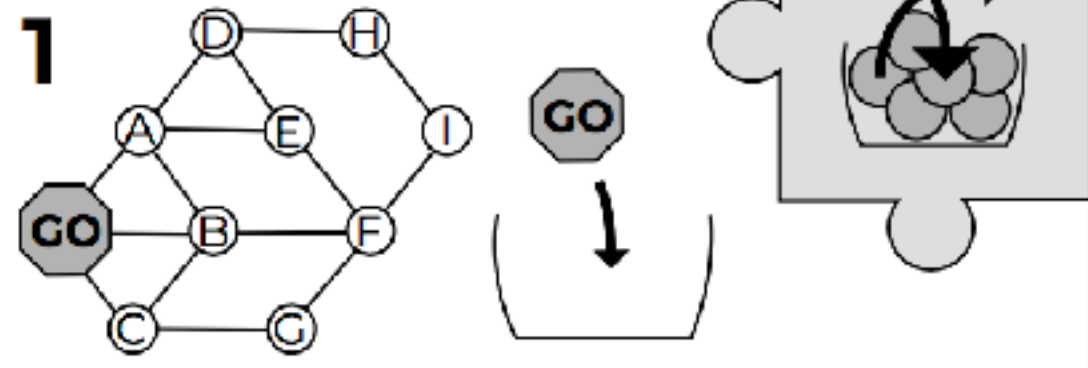
idea-instructions.com/graph-scan/
v1.0, CC by-nc-sa 4.0

IDEA

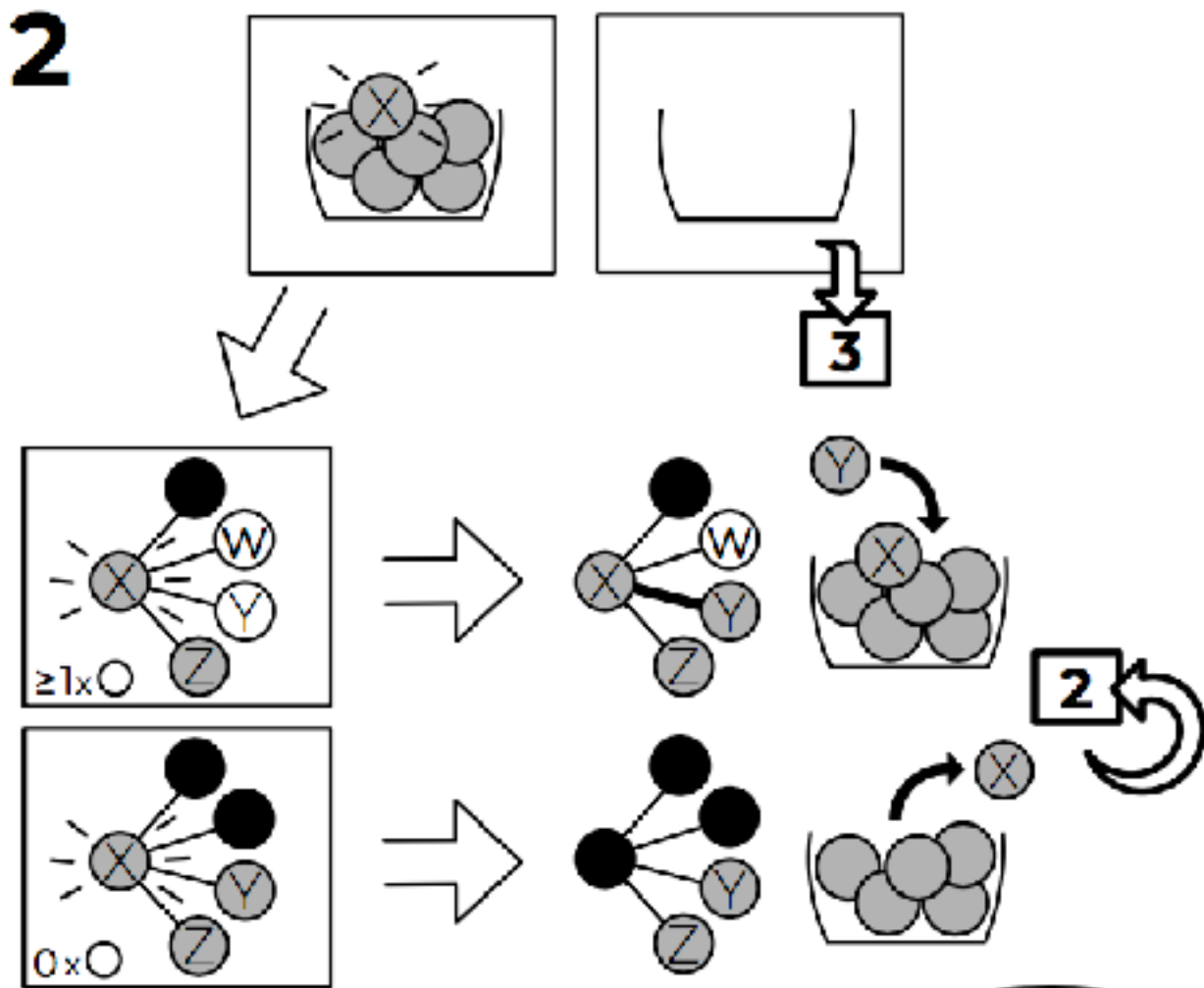


AD-HOC SEARCH

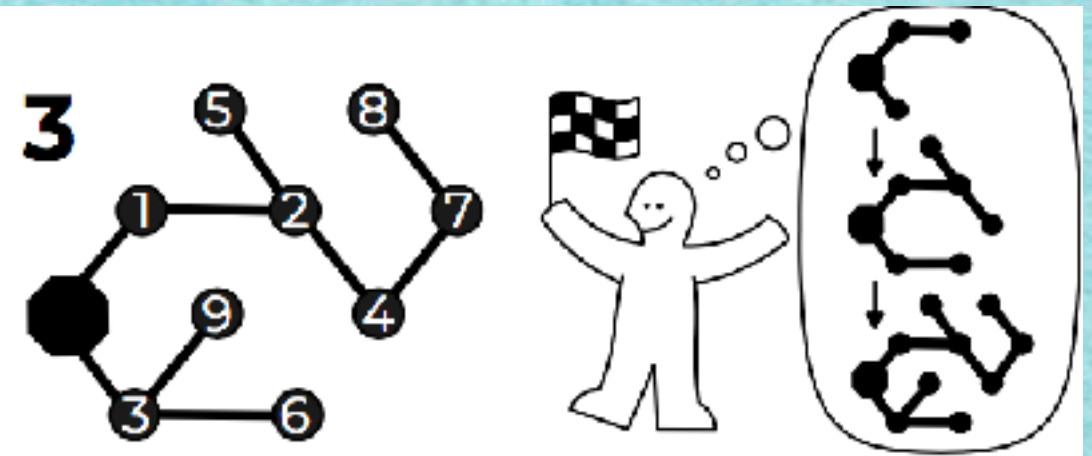
1



2

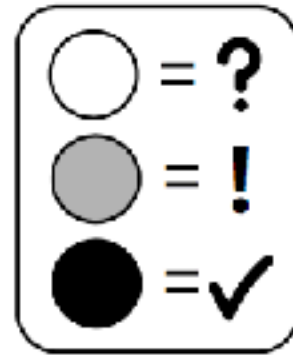
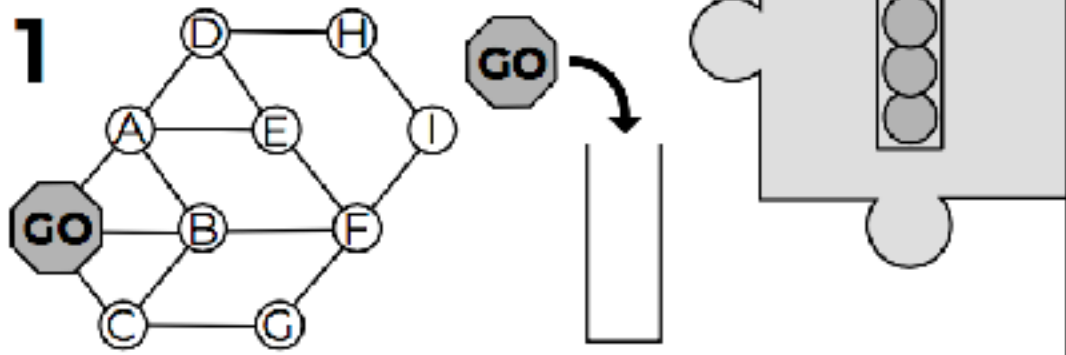


3

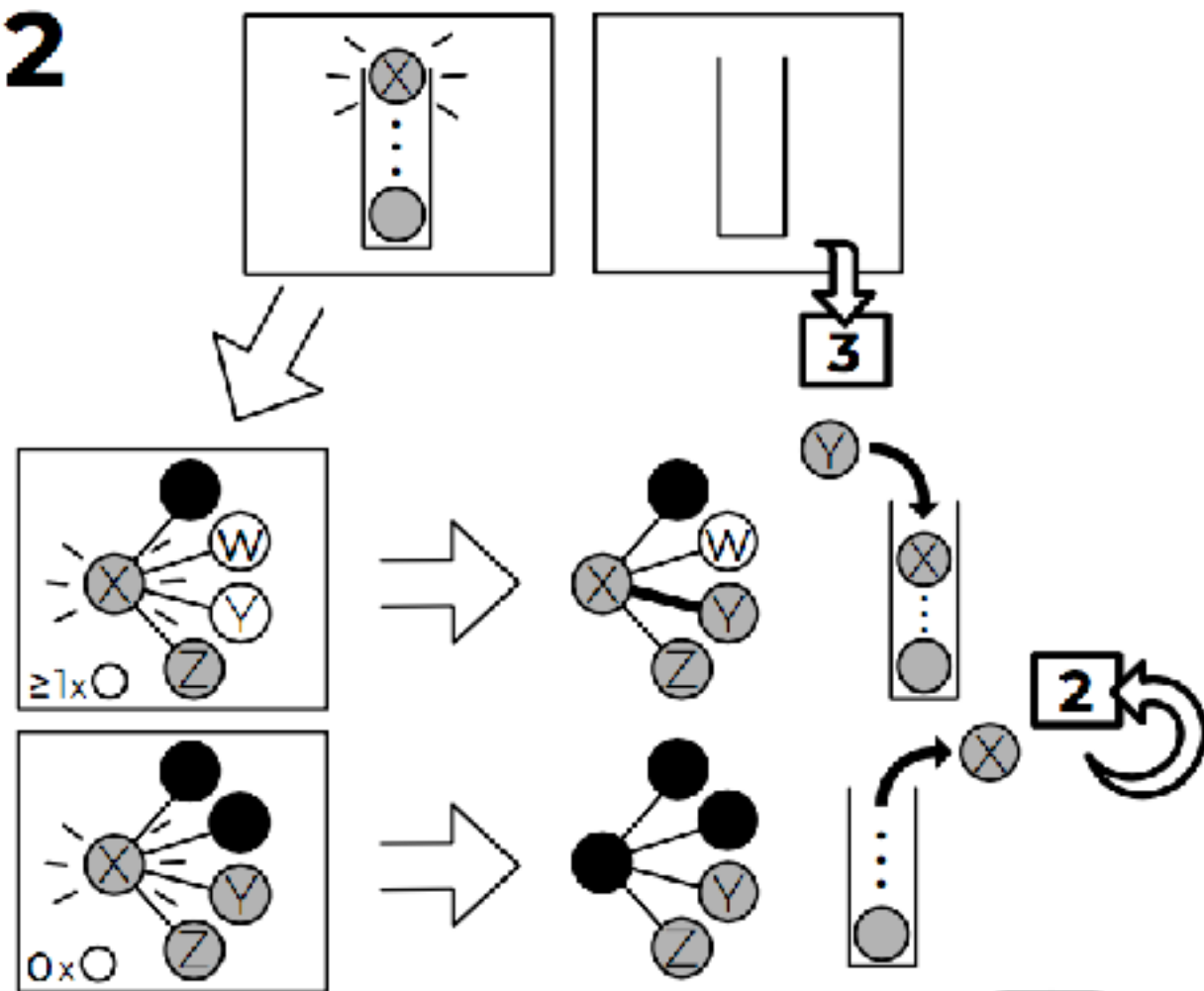


DEEP SEARCH

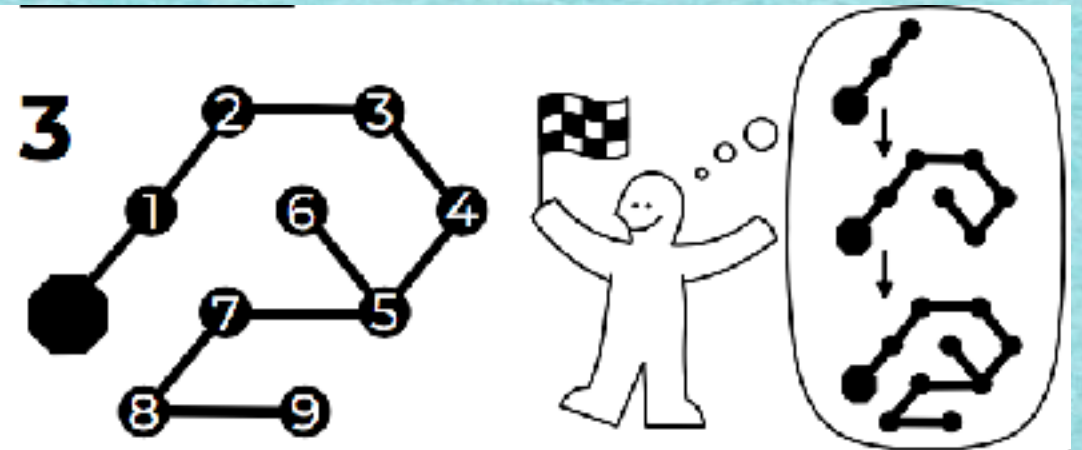
1



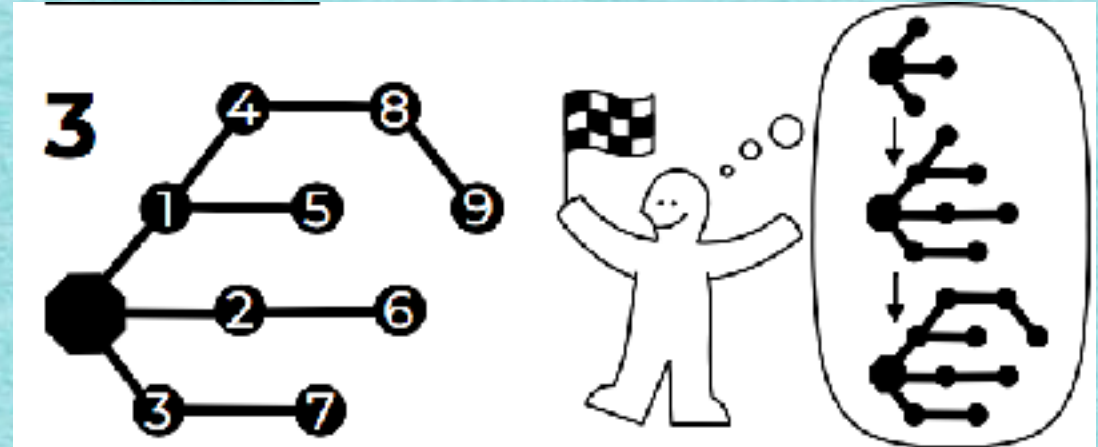
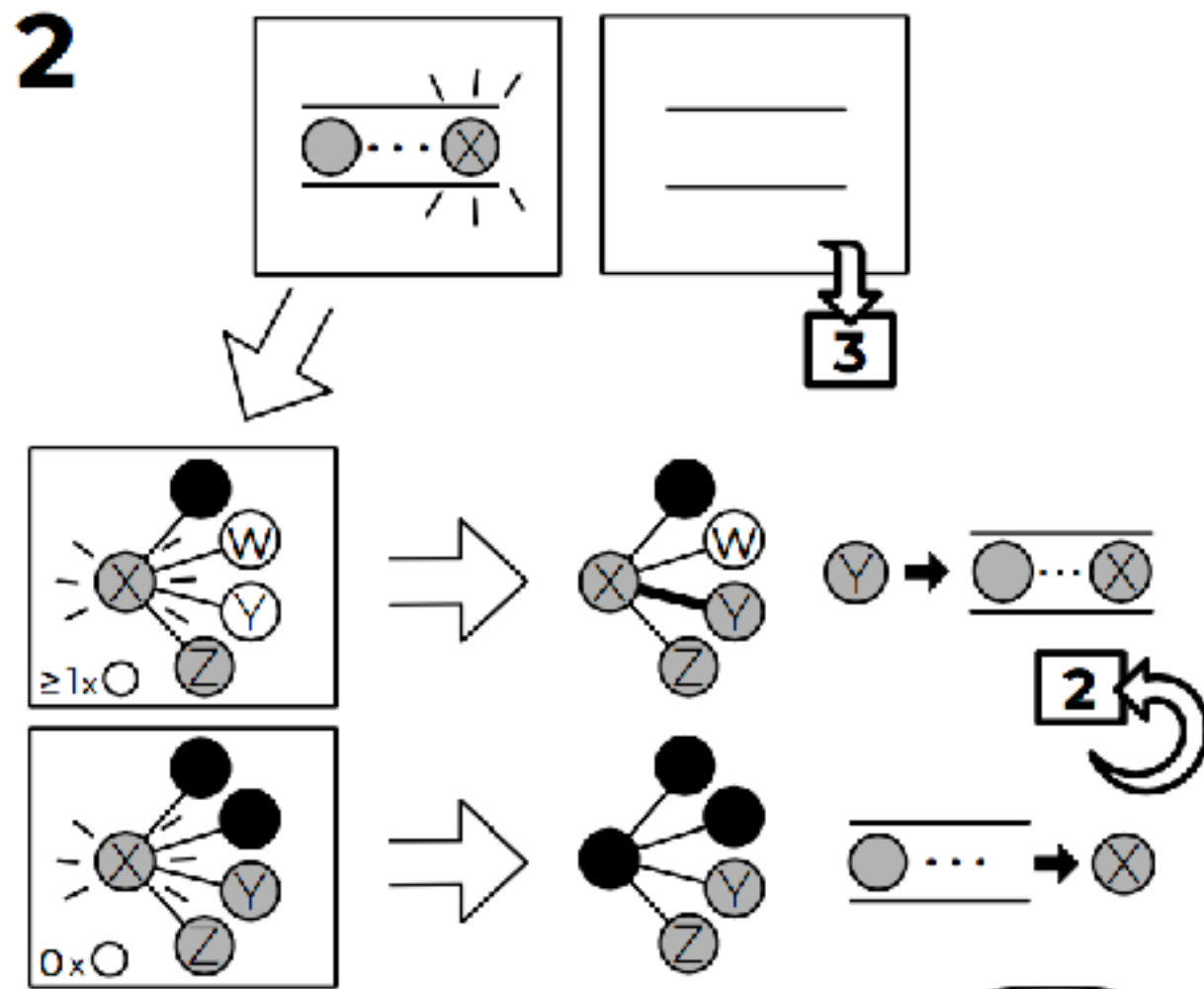
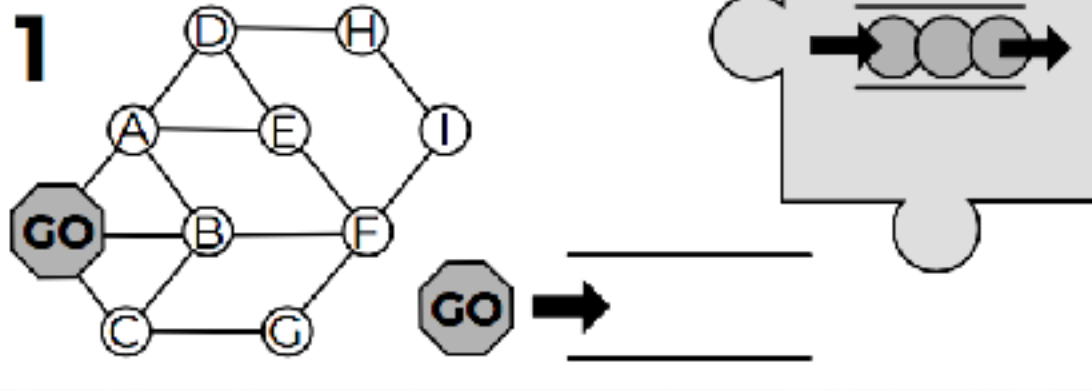
2



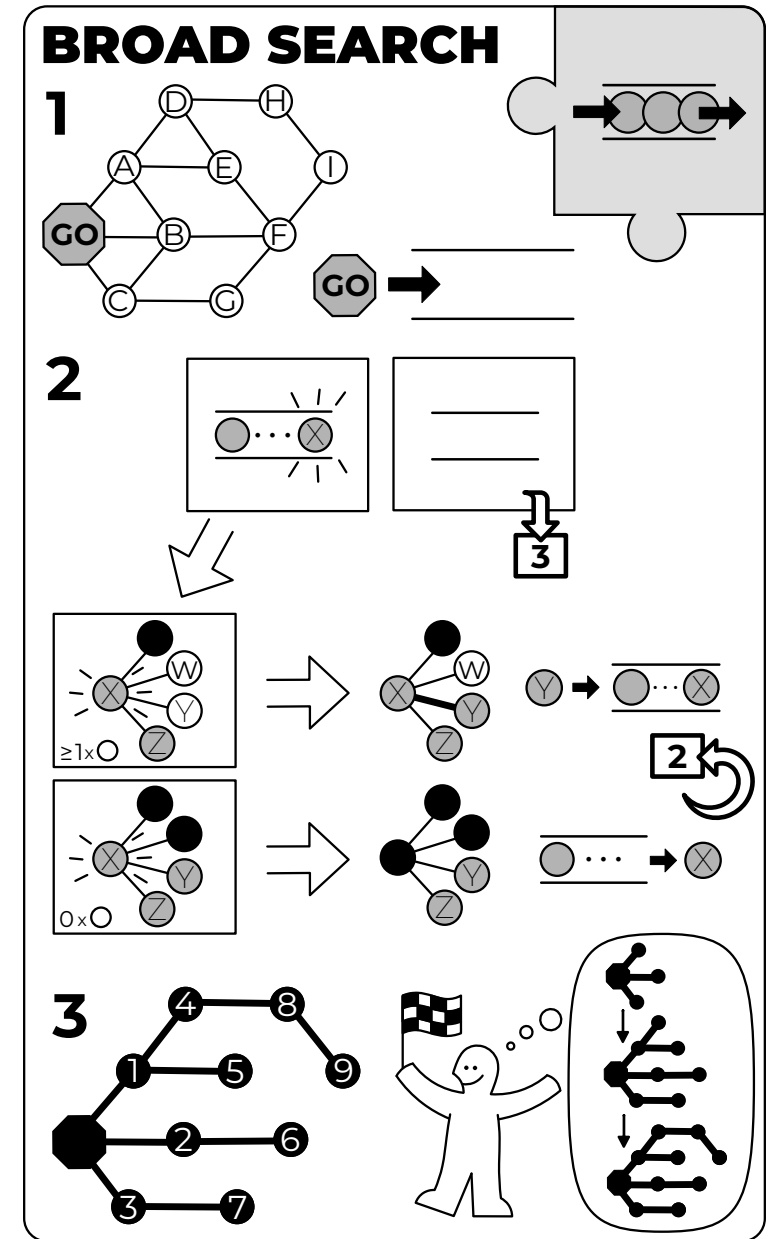
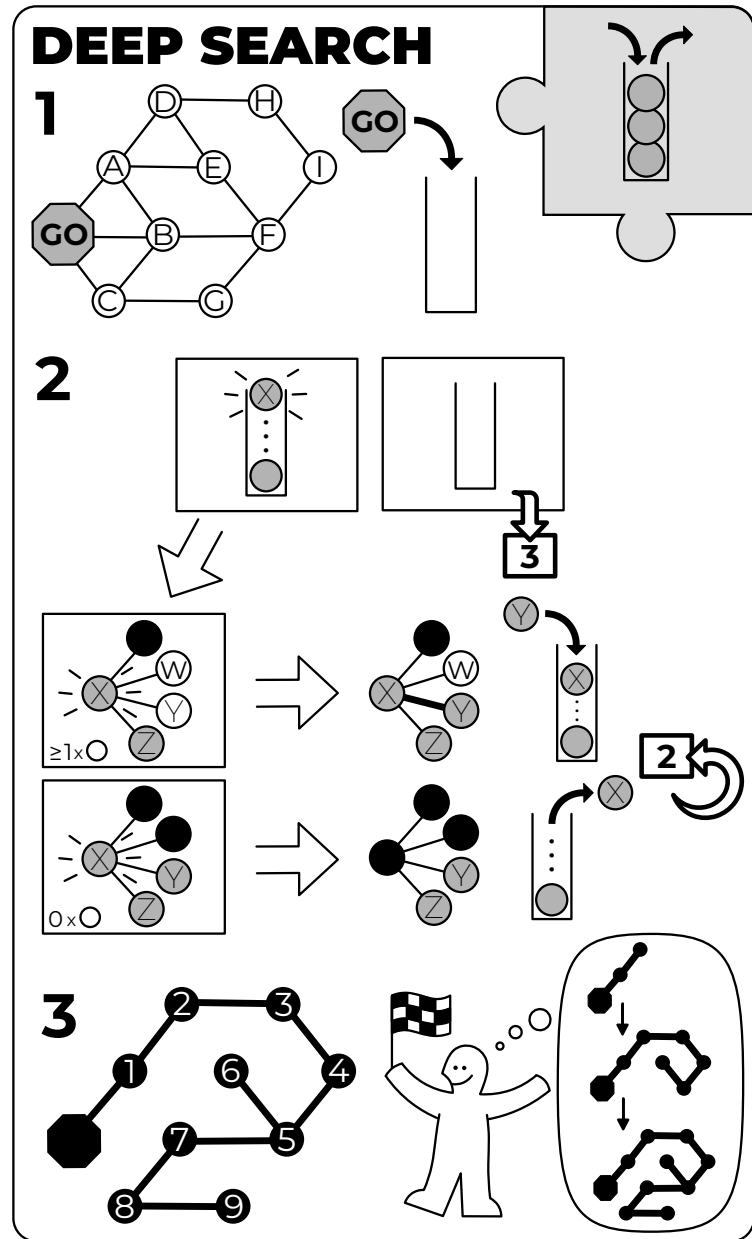
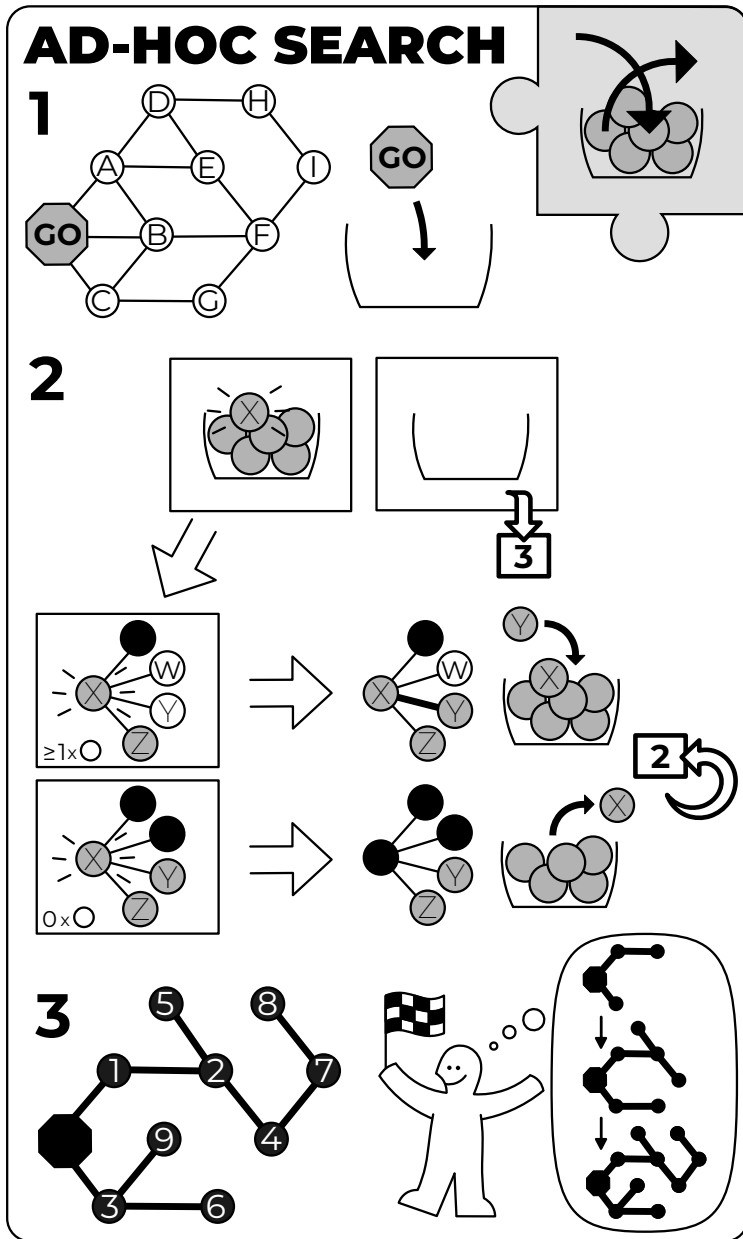
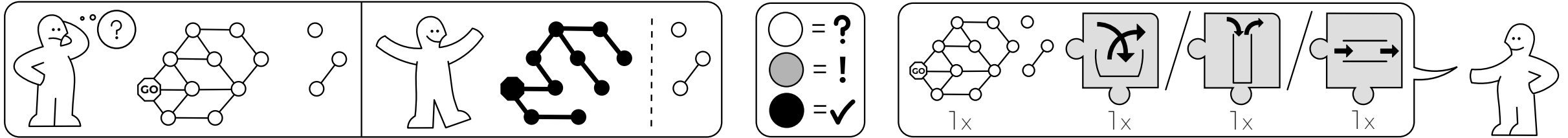
3



BROAD SEARCH



GRÅPH SCÄN



Kapitelende!

s.fekete@tu-bs.de