Aufgabe 1: Drehfreudig?
DFS Lösung

Team-ID: 00001

Team-Name: Was?

Bearbeiter dieser Aufgabe:
Mathieu de Borman

28. November 2025

Inhaltsverzeichnis

1 Lösungsidee

Mithilfe einer Tiefensuche (DFS) wird rekursiv die Tiefe und Breite der Rechtecke ermittelt, um anschließend anhand dieser Daten Rückschlüsse über die Höhen der Rechtecke zu ziehen und somit die Drehfreudigkeit zu bestimmen.

1.1 Definitionen

Sei V die Menge der Knoten im Baum.
Sei $r \in V$ der Wurzelknoten.
Sei $V' \coloneqq V \setminus \{r\}$.
Sei B das Tupel aller Blattknoten im Baum von links nach rechts.
Sei $t : V \to \mathbb{N}_0$ eine Funktion, die angibt, wie tief ein Knoten ist.
Sei $w : V \to \mathbb{R}$ eine Funktion, die angibt wie breit das Rechteck eines Knotens ist.
Sei $h : V \to \mathbb{N}$ eine Funktion, die angibt, wie hoch das Rechteck eines Knoten ist.
Sei d die Distanz zwischen den beiden Bäumen.

Sei $p : V' \to V$ eine Funktion, die für einen Knoten den Elternknoten zurückgibt.
Sei $c : V \to 2^V$ eine Funktion, die für einen Knoten die Kinderknoten zurückgibt.

1.2 Interpretation der Aufgabenstellung

> „Jeder Knoten wird also durch ein Rechteck dargestellt, und für jeden Knoten u, der kein Blatt ist, gilt, dass die Rechtecke seiner Kinder die gleiche Breite haben und die Vereinigung ihrer Oberseiten die Unterseite des Rechtecks von u ergibt."
>
> — *Aufgabenstellung*

$$\Rightarrow \forall v \in V' : w(v) = \frac{w(p(v))}{|c(p(v))|}$$

Aus den Abbildungen:

$$\begin{aligned} &\forall v \in V' : t(v) = t(p(v)) + 1 \\ \Rightarrow &\forall v \in V : h(v) = 1 + \begin{cases} 0 & ,\text{für } c(v) \neq \emptyset \\ \max\limits_{v' \in V}(t(v')) - t(v) & ,\text{für } c(v) = \emptyset \end{cases} \end{aligned}$$

> „Unter einer Zeichnung eines Baums mit der Wurzel oben kann man den Baum zusätzlich um 180° gedreht mit der Wurzel unten zeichnen. Wenn man nun auch noch Rechtecke nach den obigen Regeln dazu zeichnet, kommt die Frage auf, ob erreicht werden kann, dass sich die Blätter in der Mitte passgenau treffen, das heißt, dort dieselben Rechtecke benutzen. Dann nennen wir den Baum *drehfreudig*."
>
> — *Aufgabenstellung*

Für einen *drehfreudigen* Baum gilt demnach:

$$\begin{aligned} &\Rightarrow \forall i : \text{die Rechtecke von } B_i \text{ und } B_{|B|-i+1} \text{ sind Gleich} \\ &\Rightarrow \forall i : w(B_i) = w\Big(B_{|B|-i+1}\Big) \wedge h(B_i) = h\Big(B_{|B|-i+1}\Big) \end{aligned}$$

Damit zwei Rechtecke genau aufeinanderliegen, müssen ihre Ober- und Unterseiten übereinstimmen. Damit das für zwei gegenüberliegende Knoten gilt, muss für d gelten:

$$\begin{aligned} \forall i : d &= t(B_i) + t\Big(B_{|B|-i+1}\Big) + h(B_i) \\ &= \Big(1 + \max_{v \in V}(t(v)) - h(B_i)\Big) + \Big(1 + \max_{v \in V}(t(v)) - h\Big(B_{|B|-i+1}\Big)\Big) + h(B_i) \\ &= 1 + \max_{v \in V}(t(v)) - h(B_i) + 1 + \max_{v \in V}(t(v)) - h(B_i) + h(B_i) \\ &= 2 + 2 \cdot \max_{v \in V}(t(v)) - h(B_i) \\ \Rightarrow \forall i, j : d &= 2 + 2 \cdot \max_{v \in V}(t(v)) - h(B_i) = 2 + 2 \cdot \max_{v \in V}(t(v)) - h\big(B_j\big) \\ &\Rightarrow h(B_i) = h\big(B_j\big) \\ &= 1 + \max_{v \in V}(t(v)) - t(B_i) = 1 + \max_{v \in V}(t(v)) - t\big(B_j\big) \\ &\Leftrightarrow t(B_i) = t\big(B_j\big) \end{aligned}$$

Ein Baum ist also *drehfreudig*, wenn:

$$\forall i, j : t(B_i) = t\big(B_j\big) \land w(B_i) = w\Big(B_{|B|-i+1}\Big)$$

1.3 Tiefensuche

Im Baum wird nun eine Tiefensuche (DFS) ausgeführt, um die Tiefe und Breite jedes Rechtecks rekursiv zu bestimmen. Anschließend werden die geordneten Daten auf Drehfreudigkeit überprüft und das Ergebnis ausgegeben.

2 Umsetzung

Die Lösung wurde in JavaScript umgesetzt.

2.1 Breitengleichheit

In dieser Lösung wird mit $t(r) = 0$ und $w(r) = 1$ gerechnet.
Daher gilt für alle $v \in V$:

$$\begin{aligned} w(v) &= \frac{w(p(v))}{|c(p(v))|} \\ &= \frac{w(p(p(v)))}{|c(p(p(v)))| \cdot |c(p(v))|} \\ &\vdots \\ &= \frac{w(r)}{|c(r)| \cdot \ldots \cdot |c(p(p(v)))| \cdot |c(p(v))|} \\ &= \frac{1}{|c(r)| \cdot \ldots \cdot |c(p(p(v)))| \cdot |c(p(v))|} \end{aligned}$$

Wir rechnen nun stattdessen mit:

$$\frac{1}{w(v)} = |c(r)| \cdot \ldots \cdot |c(p(p(v)))| \cdot |c(p(v))|$$

Diese Methode ermöglicht es, die Gleichheit der Breiten allein mit natürlichen Zahlen zu überprüfen. Dadurch werden mögliche Floating-Point-Fehler vermieden.

2.2 Tiefensuche

In der Funktion `turnable` wird ermittelt, ob der Baum *drehfreudig* ist. Da der Baum sehr tief werden kann, wird anstelle von Rekursion ein Stack verwendet, um Overflow-Fehler zu vermeiden.

3 Komplexitätsanalyse

Wir setzen voraus, dass die Grundrechenarten in konstanter Zeit durchführbar sind und dass Zahlen konstant viel Speicher belegen. Da bei dem DFS jeder Knoten nur einmal besucht wird, ergibt sich eine gesamte Laufzeitkomplexität von $\mathcal{O}(n)$.

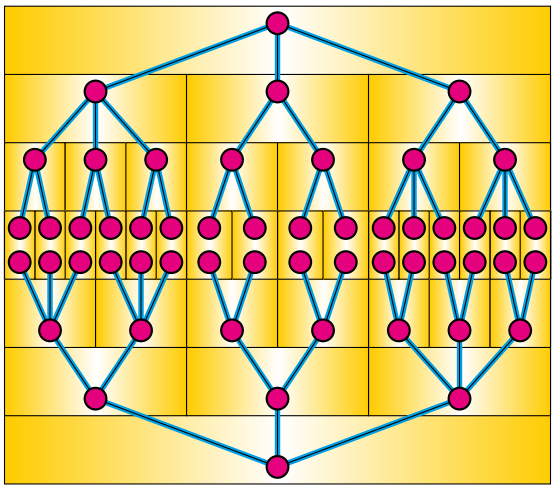
4 Werkzeuge

In dieser Arbeit wurden keine KIs, ILP-Solver oder LLMs verwendet. Zur Darstellung der Grafik wurde das SVG-Namespace von w3.org (http://www.w3.org/2000/svg) genutzt.

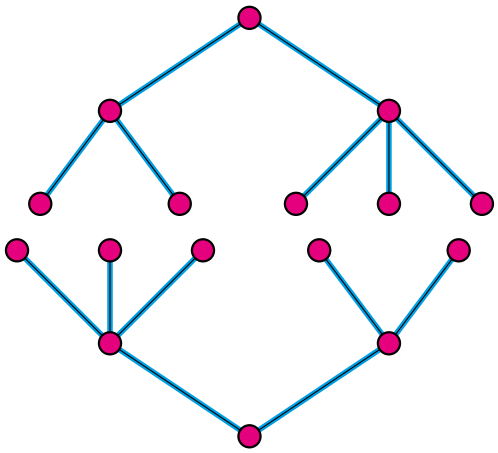
5 Beispiele

Wir rufen nun das JavaScript-Programm mit den verschiedenen BwInf-Eingabedateien auf. Das Programm wird mithilfe des Browsers ausgeführt. Über den Knopf „Datei hochladen“ können Beispieleingaben ausgewählt werden. Die Ausgabe erscheint auf der Webseite. Genau dann, wenn der Baum mit Rechtecken angezeigt wird, ist der Baum *drehfreudig*. Das Programm terminiert für alle BwInf- Beispieleingabedateien in weniger als 20 ms auf einem gewöhnlichen PC. Alle Ergebnisse sind auch im Ordner der Programmdatei zu finden.

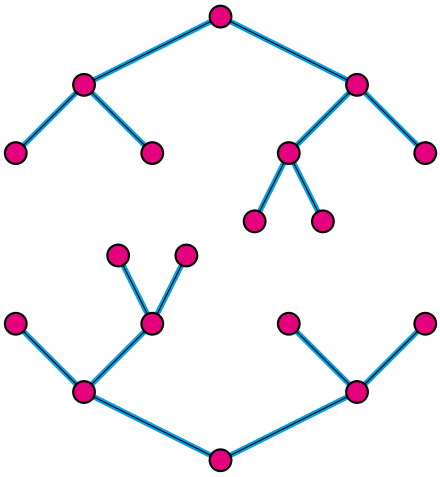
5.1 Standardbeispiele



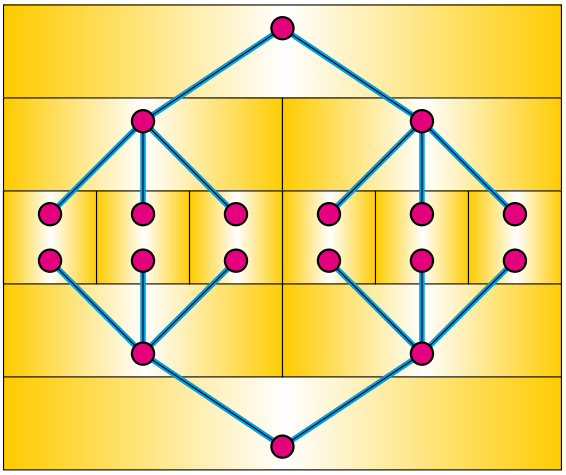
drehfreudig01



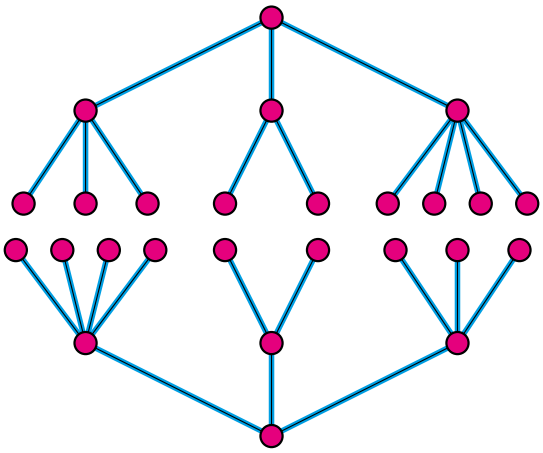
drehfreudig02



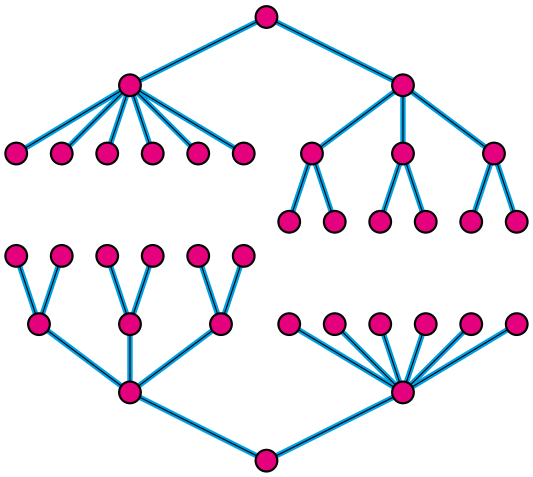
drehfreudig03



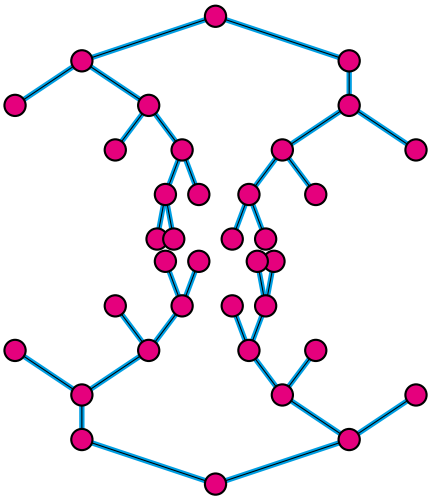
drehfreudig04



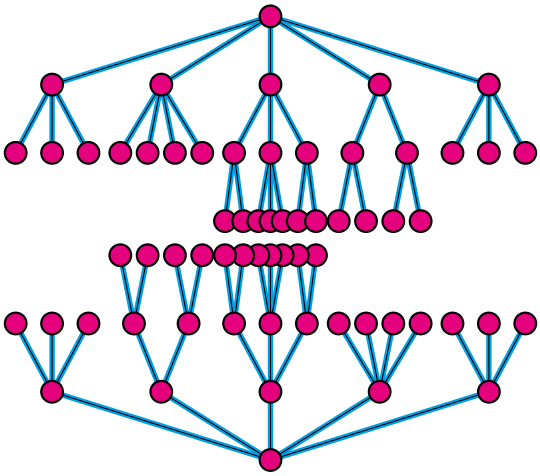
drehfreudig05



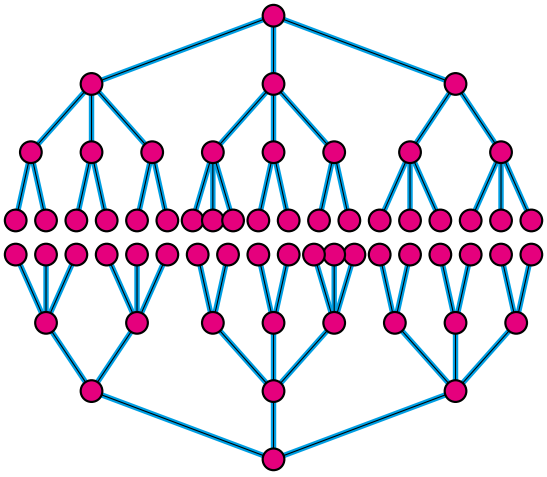
drehfreudig06



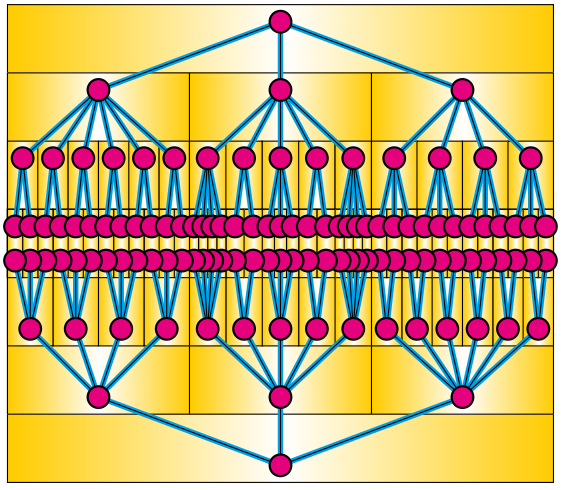
drehfreudig07



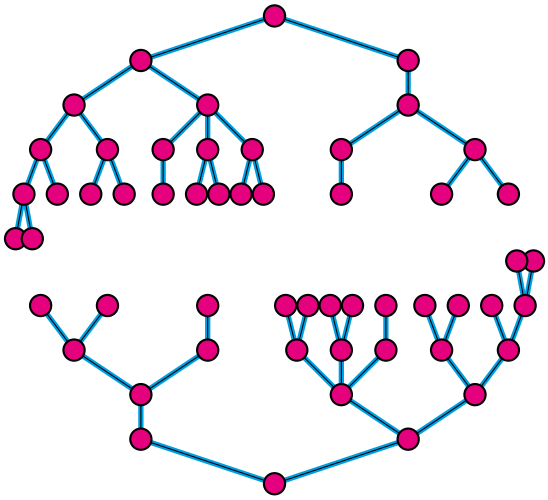
drehfreudig08



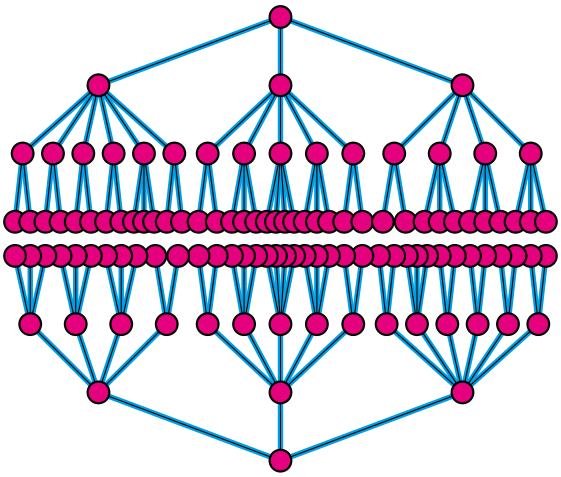
drehfreudig09



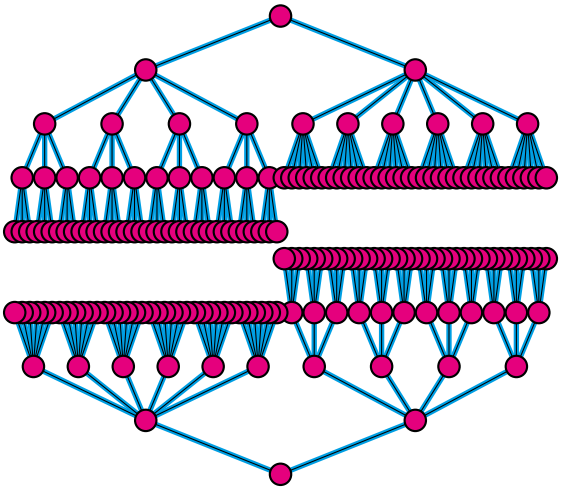
drehfreudig10



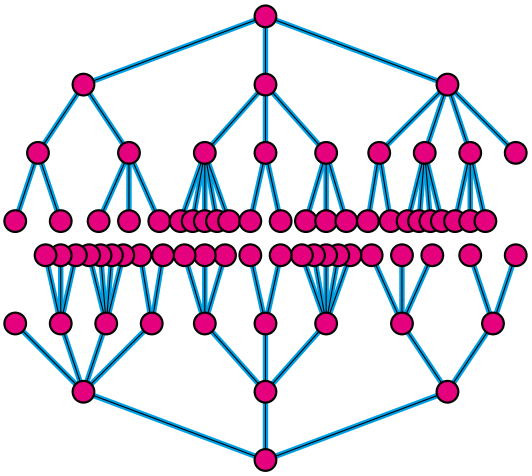
drehfreudig11



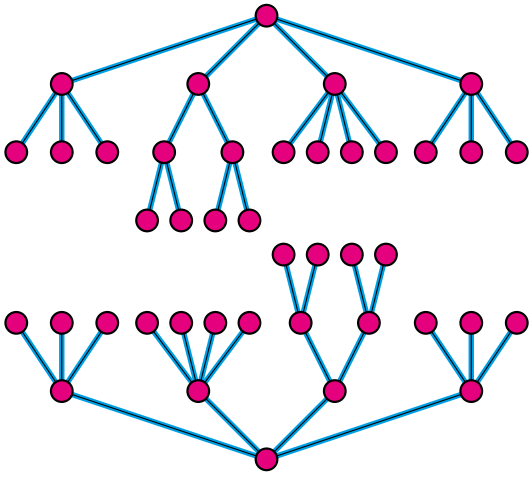
drehfreudig12



drehfreudig13

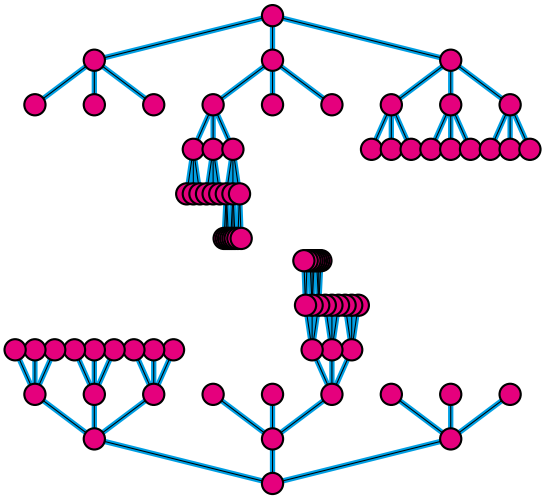


drehfreudig14

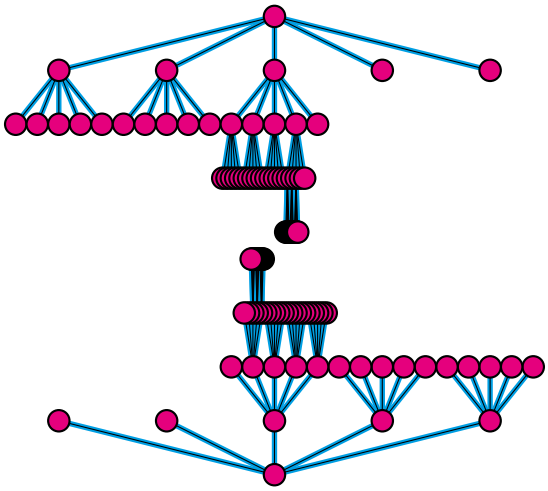


drehfreudig15

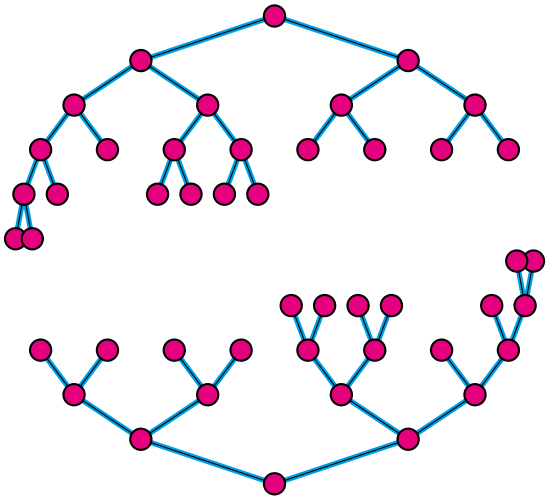
5.2 Zusatzbeispiele



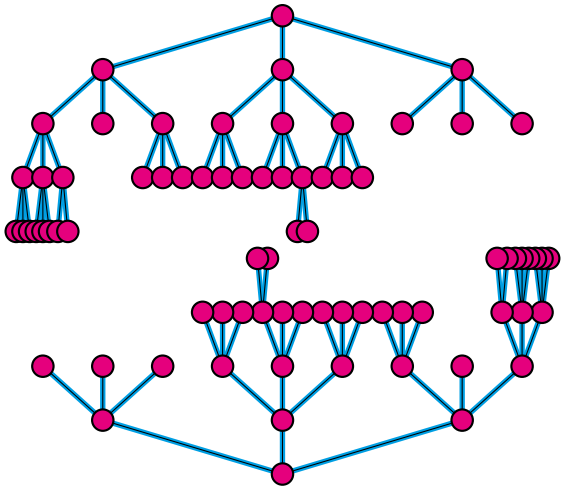
schmuck00



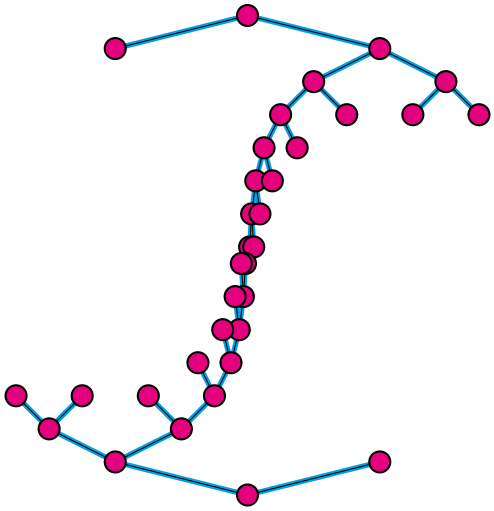
schmuck01



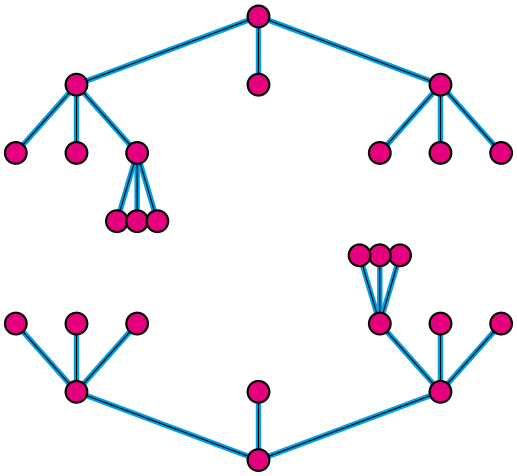
schmuck0



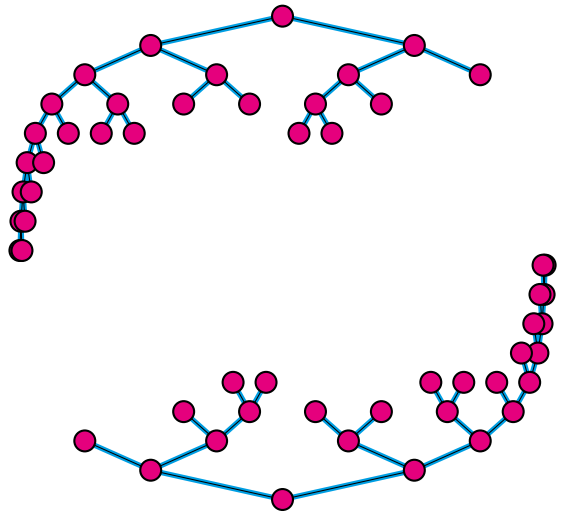
schmuck1



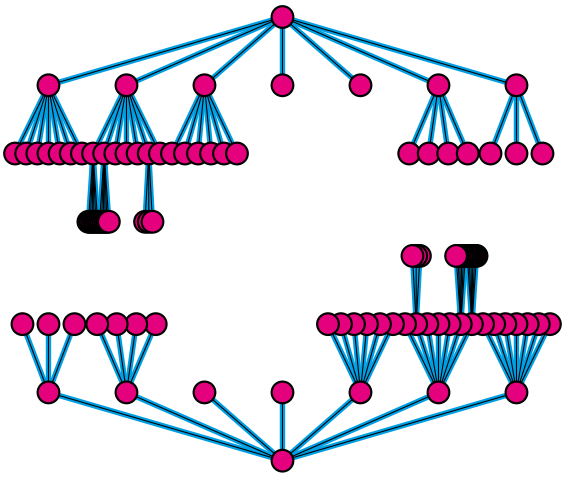
schmuck2



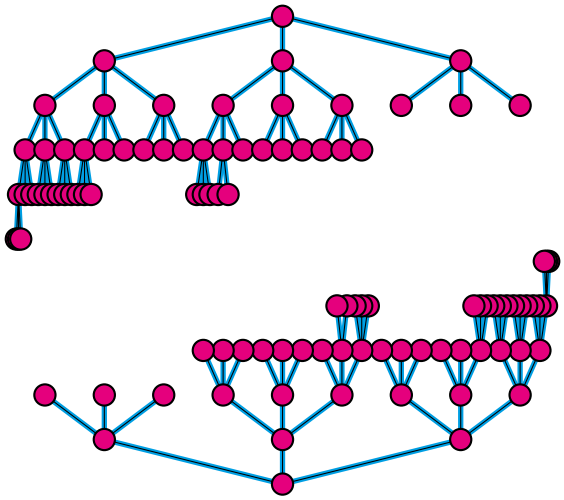
schmuck3



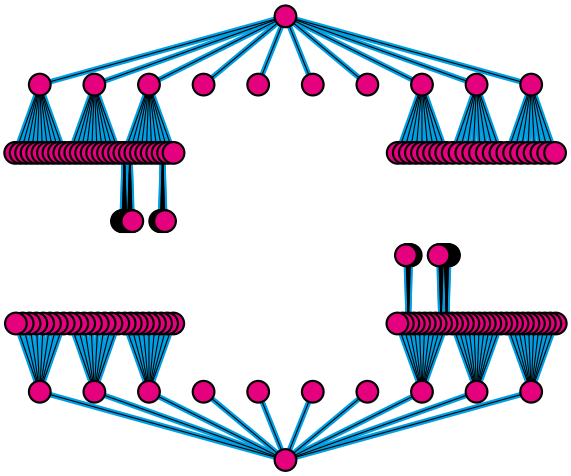
schmuck4



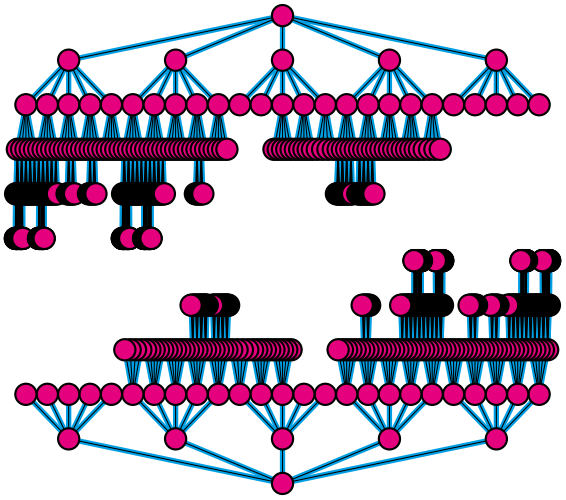
schmuck5



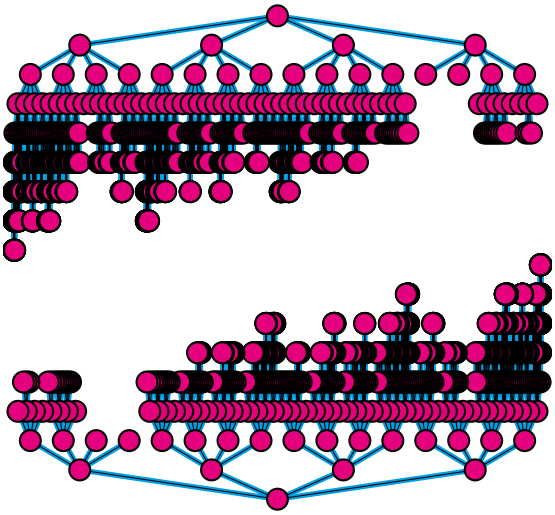
schmuck6



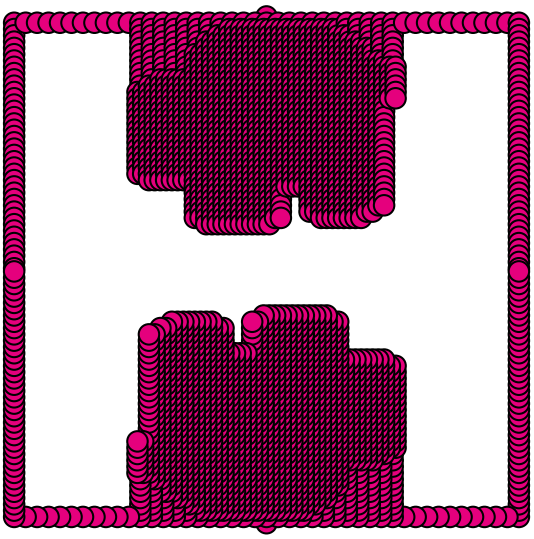
schmuck7



schmuck8



schmuck9



amogus

6 Quelltext

script.js JavaScript

```
193 //Ermittle, ob der Baum drehfreudig ist
194 function turnable(n, parents, kids) {
195  let widths = [1]
196  let leafWidths = []
197  let depths = [0]
198  let leafDepths = []
199  for (let i = 1; i <= n; i++) {
200   let width = widths[parents[i]] * kids[parents[i]].length
201   let depth = depths[parents[i]] + 1
202   widths.push(width)
203   depths.push(depth)
204   if (!kids[i].length) {
205    leafWidths.push(width)
206    leafDepths.push(depth)
207   }
208  }
209  let turns = 1
210  for (let i = 0; i < leafWidths.length; i++) {
211   turns &= leafWidths[i] == leafWidths[leafWidths.length - i - 1]
212   turns &= leafDepths[i] == leafDepths[0]
213  }
214  return [turns, widths, depths]
215 }
216 //Ermittle, ob der Baum drehfreudig ist und visualisiere das Ergebnis
217 function main([n, parents, kids]) {
218  let [turns, widths, depths] = turnable(n, parents, kids)
219  drawResult(n, parents, kids, turns, widths, depths)
220 }
```