I. Labyrinth vom 1.12. zu Ende
II. Asymptotische Notation

<u>ad I.</u> hatten: Labyrinth

Kreuzungen ≙ Knoten
Weg. zwischen Kreuzungen ≙ Kanten
} Graph G

Aufgabe: Finde von geg. Startknoten s einen anderen Knoten t (Ausgang) in G
↳ wir wenden Tiefensuche an
+ zeigen, dass die in gewissem Sinne bestmöglich ist
↳ also (a) wieviele Schritte braucht DFS
(b) untere Schranke für bel. Strategie

<u>ad (a):</u>
letztes Mal: <u>Lemma 1:</u> keine Kante wird mehr als zweimal durchlaufen

<u>Korollar 2:</u> In einem Graph mit insgesamt n+1 Knoten (inkl. Start) wird der Ausgang t in höchstens 2n-1 Schritten gefunden

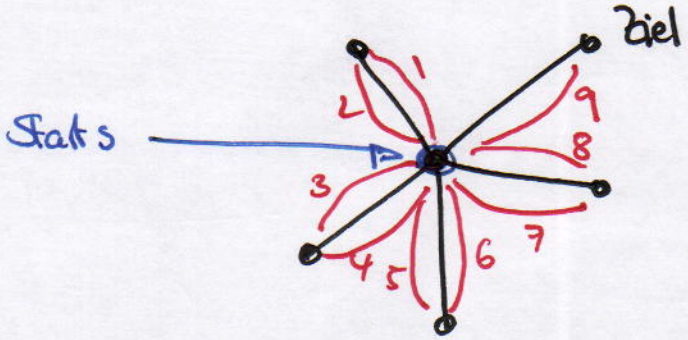
<u>Beweis:</u> Graph mit n+1 Knoten
⟹ aufspannender Baum (den die Tiefensuche konstruiert) hat genau n Kanten
Wenn jede Kante zweimal durchlaufen wird: 2n Schritte
Aber: wenn Ausgang gefunden → letzte Kante nicht zurück
⟹ 2n-1 Schritte

ad (b): sollten zeigen, dass „bestmöglich"

↳ noch überlegen, dass es keine Strategie geben kann, die für <u>jedes</u> Labyrinth mit $n+1$ Kreuzungen weniger als $2n-1$ Schritte benötigt.

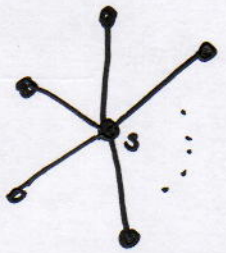
[für jede Strategie ex. Labyrinth, für das mind. $2n-1$ Schritte fürs Finden des Ausgangs benötigt werden]

⇝ extremal Beispiel: $n+1=6$



$2n-1=9$

<u>Allgemein:</u>



Graph mit Knoten s und Kanten zwischen s und allen anderen Knoten.

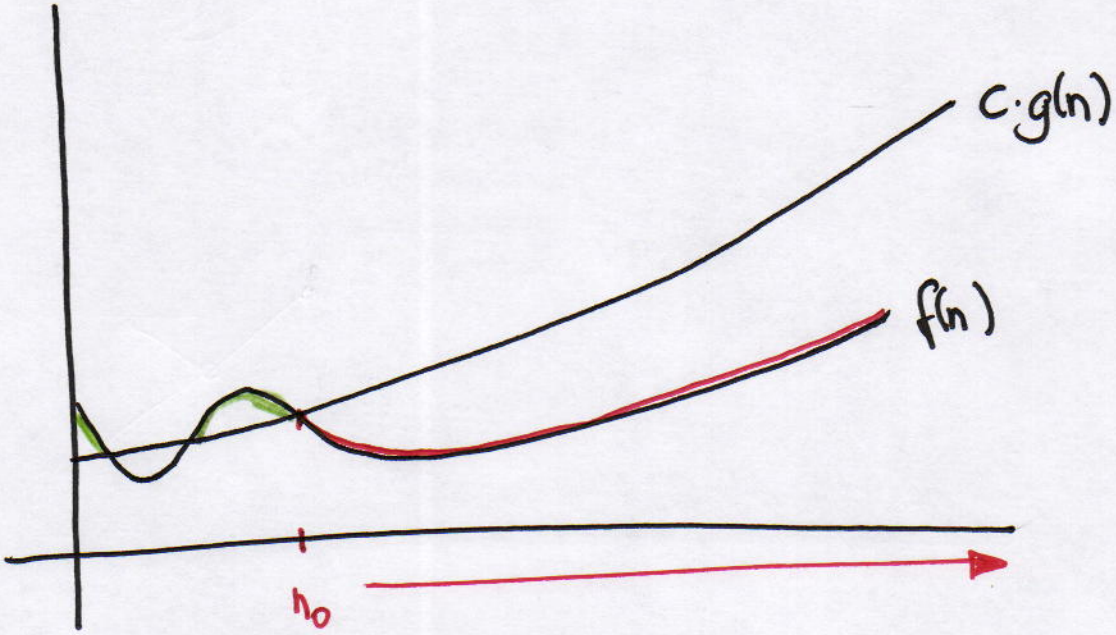
Der Knoten, den die Strategie zuletzt besucht ist der Ausgang.

II. Asymptotische Notation

(i) O-Notation „obere Schranke"

$f \in O(g(n)) \iff$ Es gibt $c > 0$, $n_0 \in \mathbb{N}$, so dass
$0 \leq f(n) \leq c \cdot g(n) \quad \forall n \geq n_0$ gilt.

Was soll das?
Man will Aussage über Wachstum von f treffen.
Dabei interessieren „kleine n" nicht so – asymptotisch soll es gelten $\rightsquigarrow$ Abschätzung soll für $\forall n \geq n_0$ gelten



„f wächst nicht schneller als g"

Beispiel:

1. $f(n) = 70n^3 + 125n^2 + 17$
   $g(n) = n^3$

Dann:

$$f(n) = 70n^3 + 125n^2 + 17 \underset{n \geq 1}{\leq} 70n^3 + 125n^3 + 17$$

$$\underset{n \geq 1}{\leq} 70n^3 + 125n^3 + 17n^3$$

$$= 212n^3 \underset{\text{für } c=212}{=} c \cdot g(n)$$

Damit:

Für $c = 212$ und $n \geq n_0 = 1$ gilt

$$f(n) \leq c \cdot g(n)$$

Zudem: $f(n) = 70n^3 + 125n^2 + 17 \geq 0$

<u>Also:</u> $0 \leq f(n) \leq c \cdot g(n) \quad \forall n \geq n_0 = 1.$

<u>Anmerkung:</u> c und n_0 sind nicht eindeutig:

$$f(n) = 70n^3 + 125n^2 + 17 \underset{\substack{125n^2 \leq n^3 \\ \Leftrightarrow 125 \leq n}}{\leq} 70n^3 + n^3 + 17$$

$$\underset{n \geq 1}{\leq} 70n^3 + n^3 + 17n^3$$

$$= 88n^3 \underset{\text{für } c=88}{=} c \cdot g(n)$$

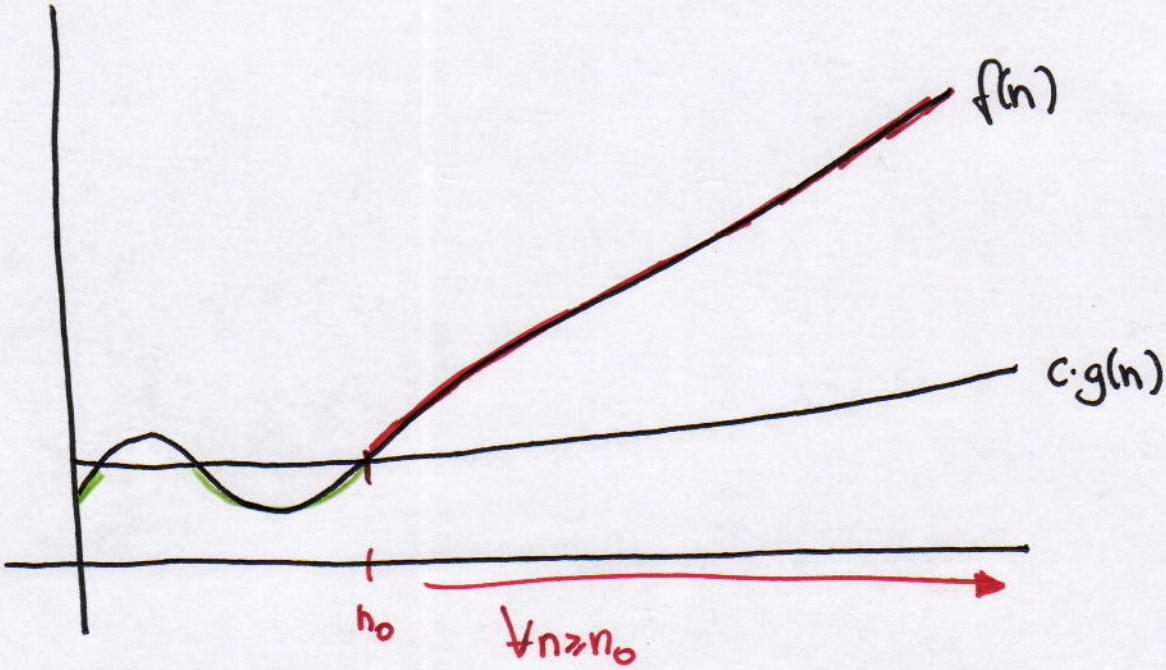
Also: für $c = 88$ und $n \geq n_0 = 125$ gilt

$$0 \leq f(n) \leq c \cdot g(n)$$

(ii) <u>Ω-Notation</u> „untere Schranke"

$f \in \Omega(g(n)) \Leftrightarrow$ Es gibt $c > 0$, $n_0 \in \mathbb{N}$, so dass

$$0 \leq c \cdot g(n) \leq f(n) \quad \forall n \geq n_0 \text{ gilt.}$$



<u>Beispiel:</u>

(2.) $f(n) = 70n^3 + 125n^2 + 17$

$g(n) = n^3$

Dann:

$$g(n) = 1 \cdot n^3 \underset{n \geq 1}{\leq} 70n^3 \underset{n \geq 1}{\leq} 70n^3 + 125n^2 + 17 = f(n)$$

Damit: Für $c = 1$, $n_0 = 1$ gilt: $c \cdot g(n) \leq f(n) \quad \forall n \geq n_0$

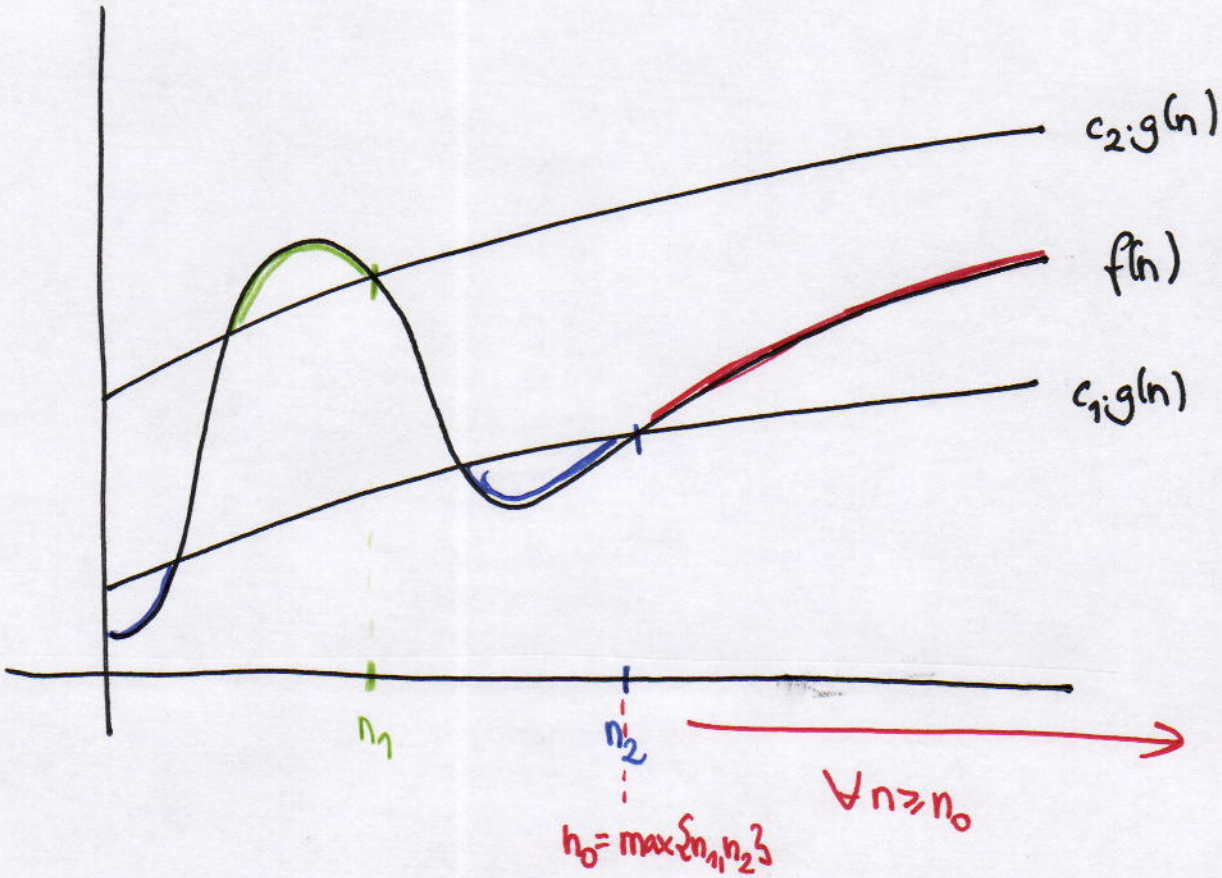
Zudem: $g(n) \geq 0$

Also: $0 \leq c \cdot g(n) \leq f(n)$ für $c = 1$, $n \geq n_0 = 1$

(iii) Θ - Notation
„Teta"

$f \in \Theta(g(n)) \Leftrightarrow$ Es gibt $c_1, c_2 > 0$, $n_0 \in \mathbb{N}$, so dass
$0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \quad \forall n \geq n_0$ gilt

„f wächst wie g"



Beispiel:

3. $f(n) = 70n^3 + 125n^2 + 17$
$g(n) = n^3$

Wie in Beispiel 1 und 2 gezeigt, gilt:

Für $c_1 = 1$ und $c_2 = 212$, sowie $n_0 = 1$

$$0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$$

4. $f(n) = 2n^2 - \cancel{880}\ 25$

$g(n) = 4n^2$

(i) $g(n) \geq 0 \quad \forall n \geq 1$

(ii) $f(n) = 2n^2 - \cancel{880}\ 25 \underset{n \geq 1}{\leq} 2n^2 \underset{n \geq 1}{\leq} 4n^2 = g(n) \Rightarrow c_2 = 1$

(iii) $f(n) = 2n^2 - 25 \underset{\substack{n^2 \geq 25 \\ n \geq 5}}{\geq} n^2 = \frac{1}{4}(4n^2) = \frac{1}{4} g(n) \Rightarrow c_1 = \frac{1}{4}$

Insgesamt haben wir

$$0 \leq c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n) \quad \forall n \geq n_0$$

für $c_1 = \frac{1}{4}$, $c_2 = 1$, $n_0 = 5$

Also: $f \in \Theta(g(n))$

III. „Kodierung" von Zahlen im Computer

- Zahl $n \in \mathbb{N}$ im Computer speichern
- Binärsystem $\rightsquigarrow$ Zahl zur Basis 2

Beispiel:

$n=50$

$$\begin{aligned} n &= 5\cdot 10 + 0\cdot 1 \\ &= 0\cdot 100 + 5\cdot 10 + 0\cdot 1 \\ &= 0\cdot 10^2 + 5\cdot 10^1 + 0\cdot 10^0 \end{aligned}$$

Ziffern einer Zahl geben an, wie oft bestimmte Zehner-Potenz verwendet wird, z.B.:

$$\underset{10^3}{7}\,\underset{10^2}{5}\,\underset{10^1}{4}\,\underset{10^0}{3} = 7\cdot 10^3 + 5\cdot 10^2 + 4\cdot 10^1 + 3\cdot 10^0$$

$\hookrightarrow$ Man nennt 10 die „Basis" und wir verwenden nur die Ziffern 0,1,...,9

Das heißt:

Im Binärsystem ist die Basis 2 und wir verwenden nur die Ziffern 0 und 1

Und wie rechnen wir das um??

$$(50)_{10} = (?)_2$$

„zur Basis 10"

$2^0 = 1 \longleftarrow 0$
$2^1 = 2 \longleftarrow 1$
$2^2 = 4 \longleftarrow 0$
$2^3 = 8 \longleftarrow 0$
$2^4 = 16 \longleftarrow 1$
$2^5 = 32 \longleftarrow 1$
$2^6 = 64$

d.h. $(50)_{10} = (\underset{2^5}{1}\underset{2^4}{1}\underset{2^3}{0}\underset{2^2}{0}\underset{2^1}{1}\underset{2^0}{0})_2$

$\uparrow$ pro 0 oder 1 ein „Bit"

Hier brauchen wir also 6 Bits.

$n=32$: $(32)_{10} = (100000)_2$

$\uparrow$ 6 Bits

$\log_2 32 = 5$, d.h. man brauch $\log_2 32 + 1$ Bits

Allgemein: Für eine Zahl n braucht man höchstens

$$\log n + 2 \text{ Bits}$$

Begründung:

Für jedes n gibt es ein k mit

$$2^{k-1} < n \leq 2^k \iff \underbrace{k-1 < \log_2 n}_{k < \log_2 n + 1} \leq k$$

Für n braucht man also höchstens so viele Bits wie für 2^k.

Für 2^k braucht man $k+1$ Bits, d.h. für n höchstens

$$k+1 \leq (\log_2 n + 1) + 1 = \log_2 n + 2 \quad \text{Bits.}$$