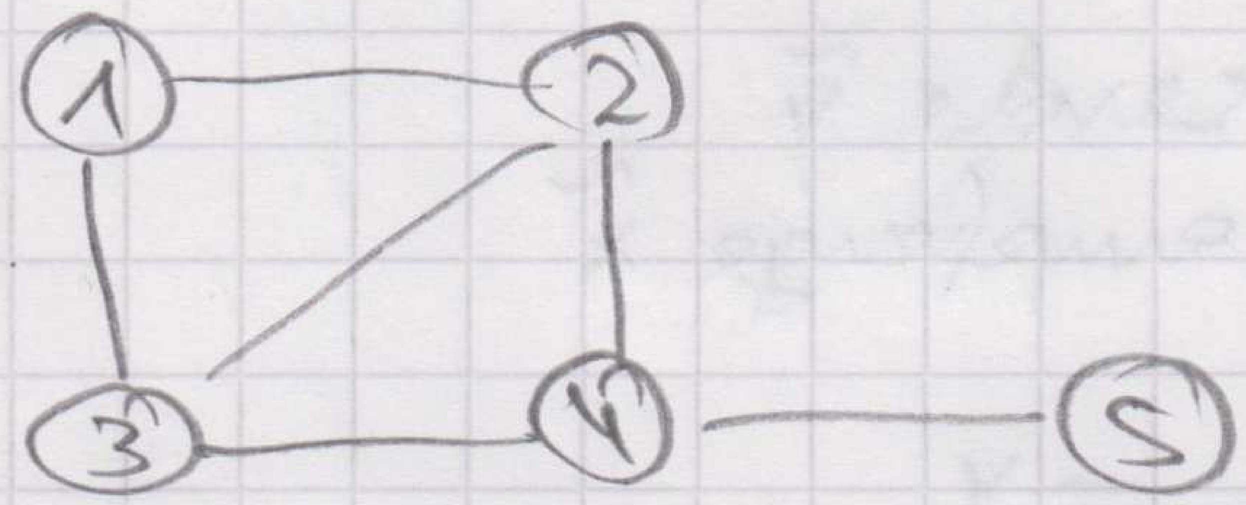
Bsp: Problem in NP

G1



G2

G1 ist isomorph zu G2

$$\pi : \left(\begin{array}{c|c|c|c|c} 1 & 2 & 3 & 4 & 5 \\ \hline 4 & 5 & 3 & 2 & 1 \end{array} \right) \begin{array}{l} G1 \\ G2 \end{array}$$

Kanten in G1:

$\{1, 2\}$

⋮

$\{2, 3\}$

Kanten in G2:

$\{\underbrace{\pi(1)}, \underbrace{\pi(2)}\}$

$\{4, 5\}$ ✓

$\{\underbrace{\pi(2)}, \underbrace{\pi(3)}\}$

$\{5, 3\}$ ✓

Problem-Größe wird gemessen über $\underbrace{|V|}_{\text{Anzahl Knoten}}$ und $\underbrace{|E|}_{\text{Anzahl Kanten}}$

Überprüfung der Lösung kostet Aufwand $O(|E_1|)$

Bestimmung einer Lösung kostet Überprüfung aller $|V_1|!$

Permutation, also $O(\underbrace{|V_1|!}_{\text{exponentielles Wachstum}} \cdot |E_1|)$, nicht polynomial.

Sichere Schlüssel-Vereinbarung:

Teilnehmer 1	$\xleftrightarrow{\text{unsichere}}$ Leitung	Teilnehmer 2
benötigt S zur Verschlüsselung	$a, p \in \mathbb{N}$ öffentlich bekannt	benötigt S zum Entschlüsseln

Teilnehmer 1:

↳ wähle Zufallszahl $x < p$

↳ bilde $\tilde{x} = a^x \bmod p$

Umkehrung: modulare Logarithmus-Fkt ist nicht polynomial berechenbar

↳ sende $\tilde{x}$
empfange $\tilde{y}$

↳ setze $S = \tilde{y}^x \bmod p$

$= (a^y \bmod p)^x \bmod p$

$= (a^y)^x \bmod p$

$= a^{x \cdot y} \bmod p$

umgekehrt

$S = \tilde{x}^y \bmod p$

Teilnehmer 2:

↳ wähle $y < p$

↳ bilde $\tilde{y} = a^y \bmod p$

↳ sende $\tilde{y}$
empfange $\tilde{x}$

$S = \tilde{x}^y$

$= (a^x \bmod p)$

$= (a^x \bmod p)^y \bmod p$

$= a^{x \cdot y} \bmod p$

Bsp: Polynomiale Reduktion

1) Problem a: Paarungsproblem

$$x \begin{pmatrix} 23 \\ 42 \\ 17 \\ 88 \\ 12 \\ 57 \end{pmatrix} \qquad y \begin{pmatrix} 48 \\ 59 \\ 11 \\ 12 \\ 64 \\ 34 \end{pmatrix}$$

Kleinster Wert in x mit kleinstem Wert in y

Problem b : Sortierproblem

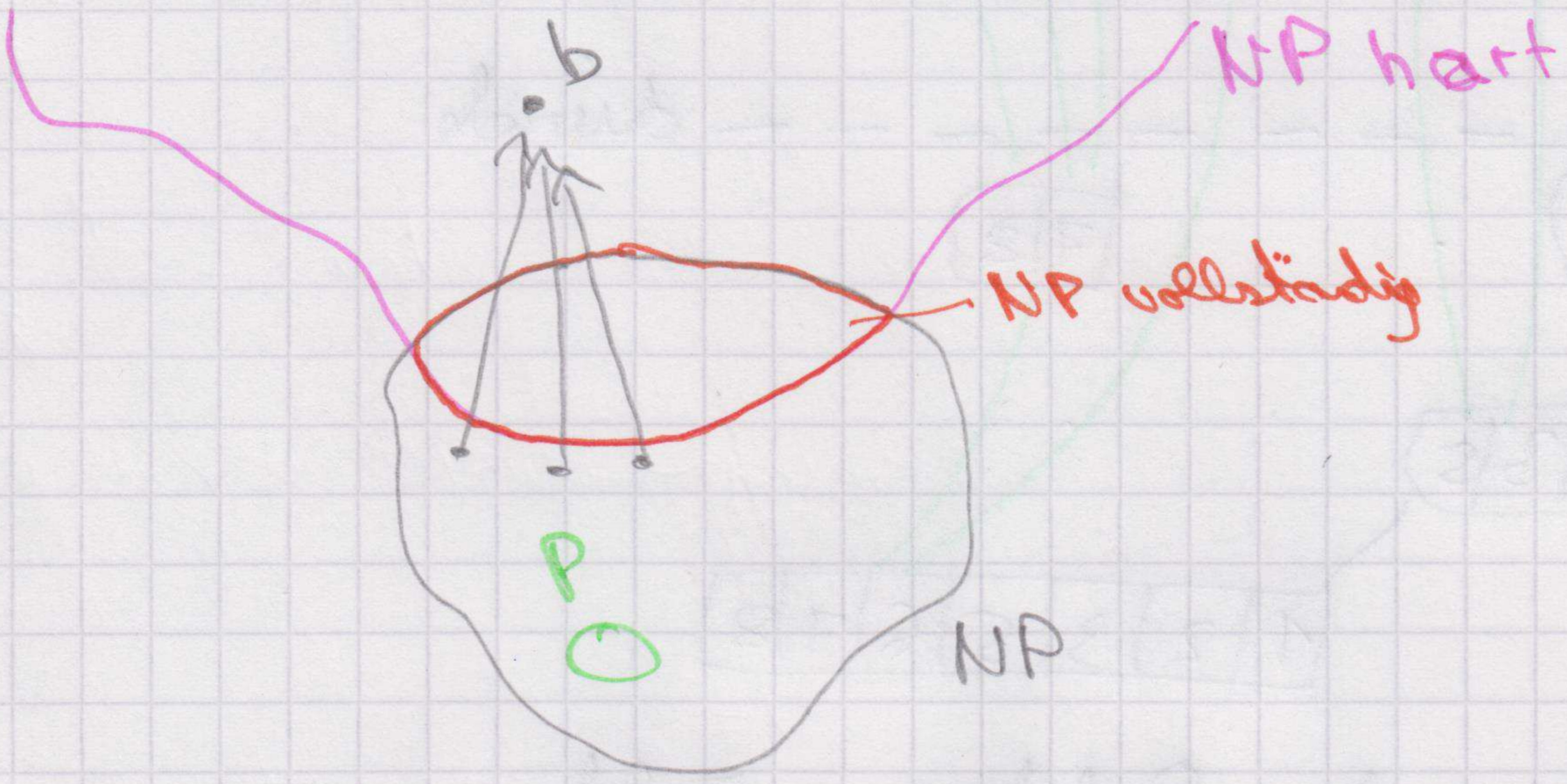
Eingabe Vektor x
Ausgabe sortierter Vektor x

$a \leq_p b$

dann, sortiere x und y, finde Paarung durch gleichen Index.

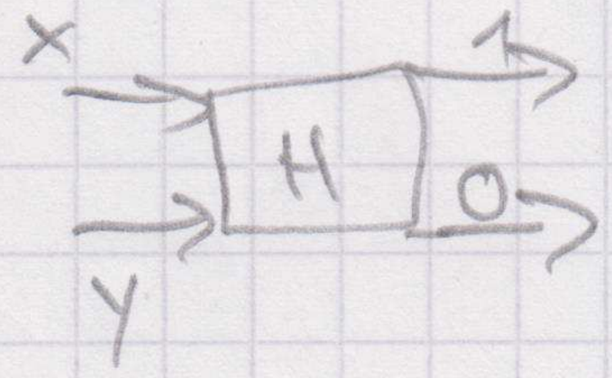
gilt auch $b \leq_p a$?

↳ Ja, löse Paarungsproblem für originales x mit
$y = (0, 1, 2, \dots, n-1)$. Für jedes Paar (x_i, y_i) speichere x_i
im Ergebnisfeld unter Index y_i.

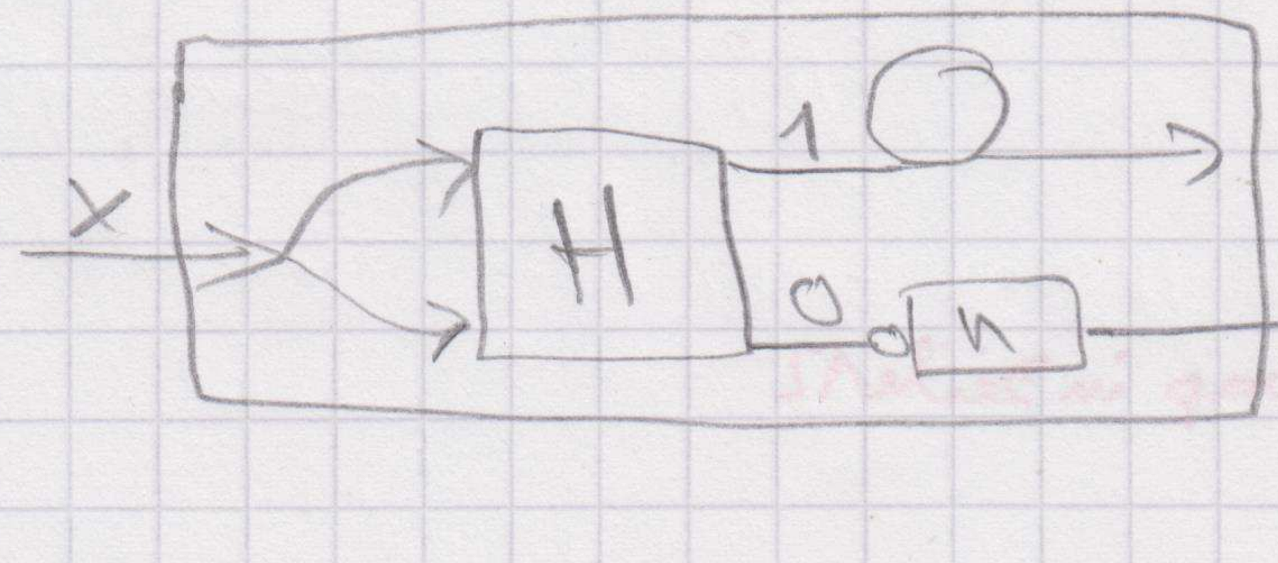


Dieser Satz ist eine Lüge ??

Beweis : H ist nicht berechenbar
Ansatz : H ist berechenbar dann existiert Program



und weiter Program P

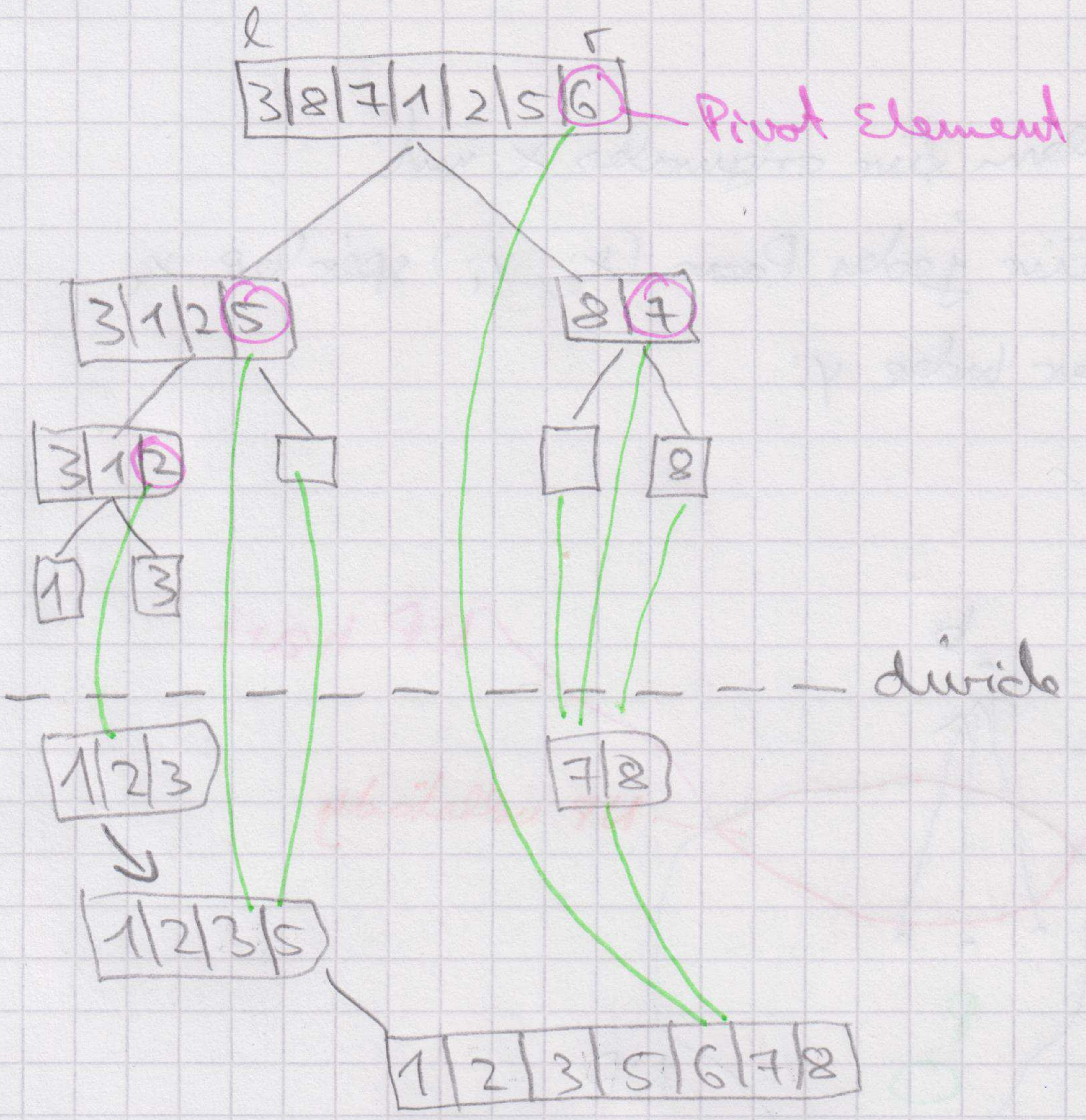


Hält P bei Eingabe von P? $p(P) = 1$?

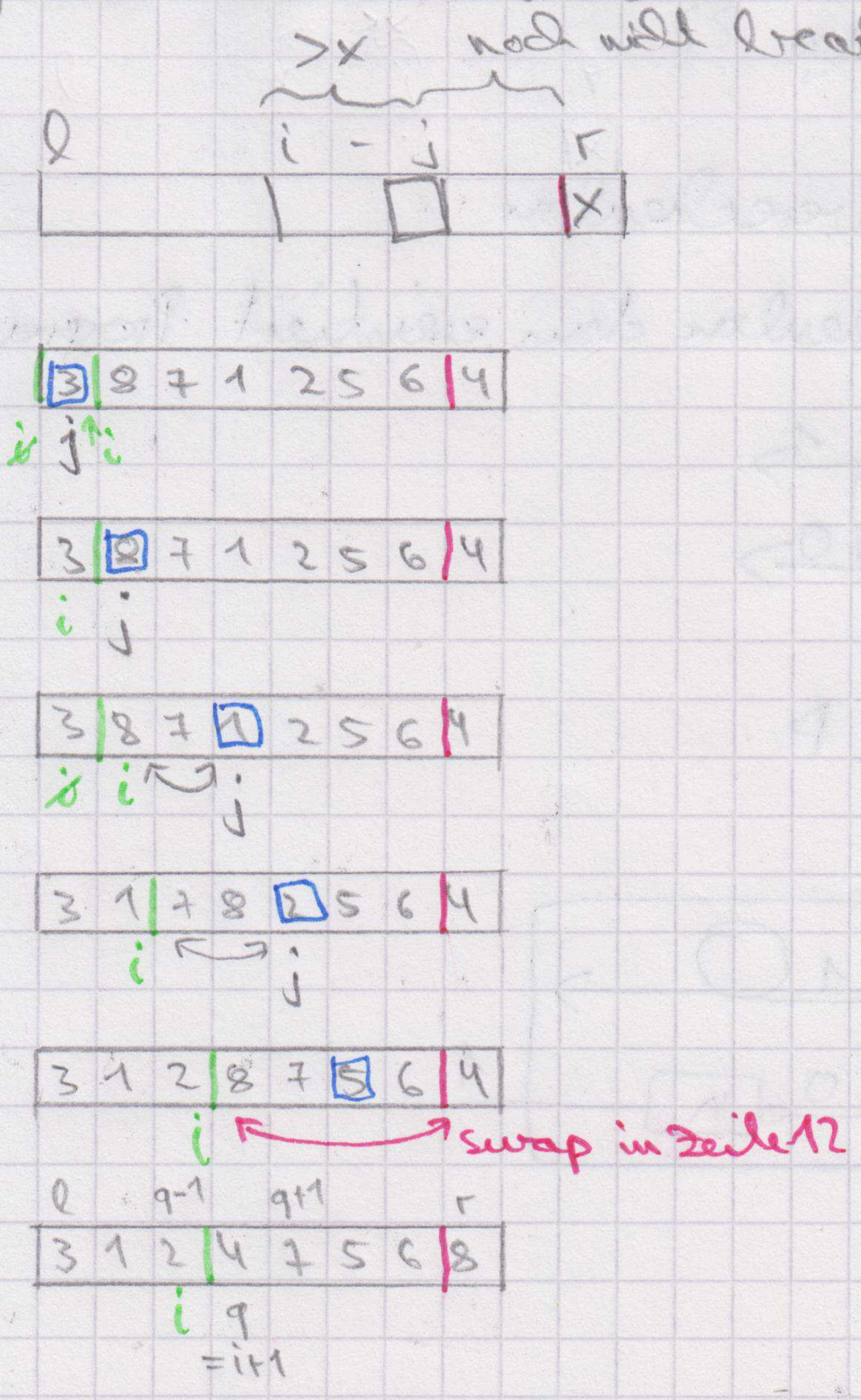
1) Ja: $H(P,P) = 0$ d.h. P hält nicht bei Eingabe von P. ↯

2) Nein: $H(P,P) = 1$ d.h. P hält bei Eingabe von P. ↯

Bsp: Quicksort



Durchführung der Partition im Feld



Laufzeit-Analyse:

-> Worst case: in jedem Zerlegeschritt ist eine Teilliste leer: man benötigt n $\underbrace{\text{Partitionen}}_{\text{im Schritt } i \text{ Aufwand } O(i)}$

Gesamt: $O\left(\sum_{i=1}^{n-1}(n-i)\right) = O\left(\sum_{i=1}^{n-1} i\right)$

Kleiner Gauss $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$

$= O\left(\frac{(n-1)\cdot n}{2}\right)$

$= O(n^2)$

-> Best case: Teillisten sind in jedem Schritt gleich groß

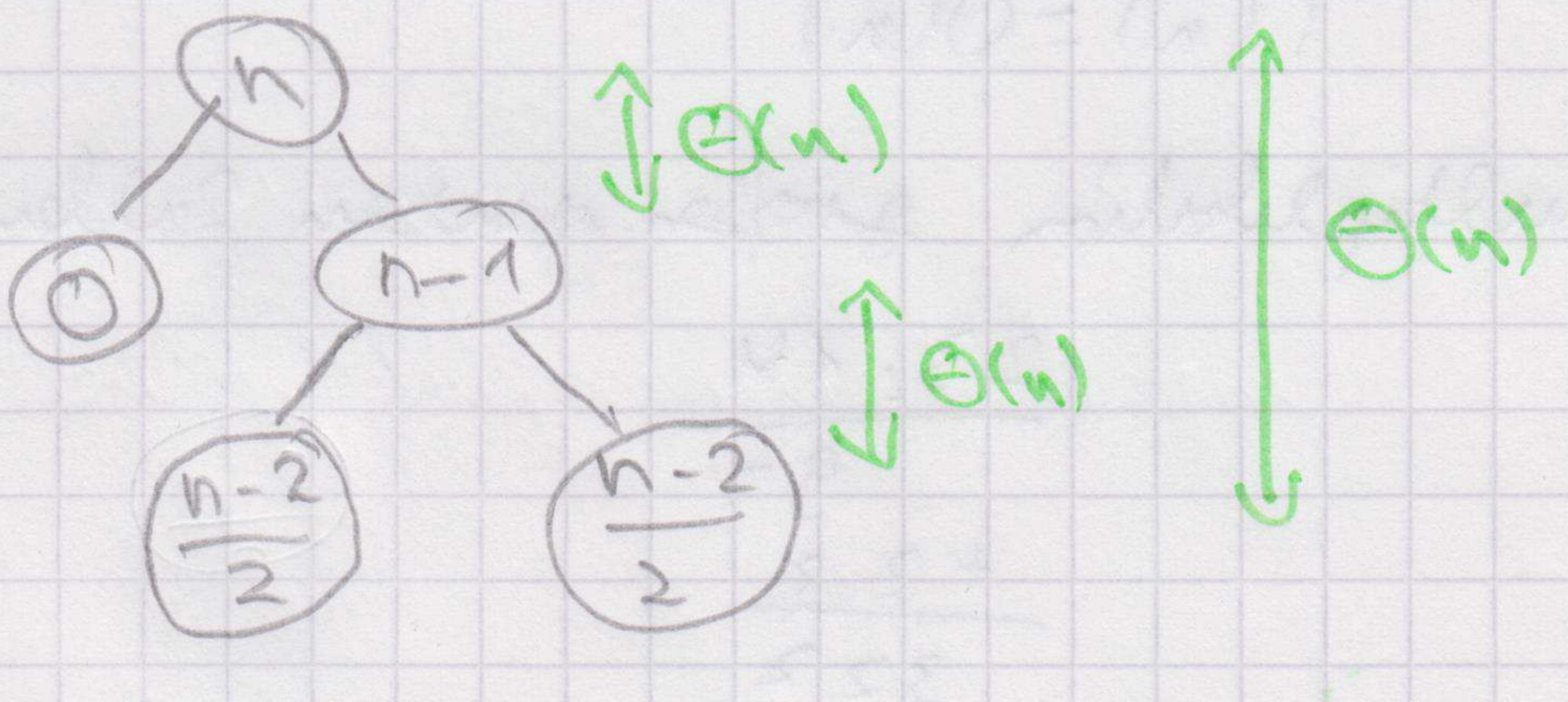
$T(n) = 2\cdot T\left(\frac{n}{2}\right) + \underbrace{c\cdot n}_{\Theta(n^1)}$

Anwendung Master Theorem: $a=2, b=2, k=1$

=> Fall 2) gilt: $2 = 2^1$

-> $T(n) = O(n \log n)$

-> Average case: Auf worst case Partition folgt immer eine best case Partition



=> $T(n) = O(n \log n)$

Vermeidung von WC:

6	3	2	7	8	4	5	1	9

optimal: Pivot = Median ← erfordert sortieren

Bsp: Spielpläne

2^1 Spieler:

$T_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, T_1^{+2} = \begin{pmatrix} 4 \\ 3 \end{pmatrix}$

$Z_1 = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}, S_1 = \begin{pmatrix} 3 & 4 \\ 4 & 3 \end{pmatrix}$

$$T_2 = \left(\begin{array}{c|c} T_1 & S_1 \\ \hline T_1^{+2} & Z_1 \end{array}\right) = \left(\begin{array}{c|cc} 2 & 3 & 4 \\ 1 & 4 & 3 \\ \hline 4 & 1 & 2 \\ 3 & 2 & 1 \end{array}\right)$$

Lohnt sich Divide & Conquer?

Bsp: Berechung von $S(1 \dots n) = \sum_{k=1}^{n} a_k$

a) mit Schleife: $n-1$ Additionen $O(n)$

b) Divide & Conquer:

$$S(1 \dots n) = S(1 \dots \tfrac{n}{2}) + S(\tfrac{n}{2}+1 \dots n)$$

Aufwand

$T(n) = 2 \cdot T(\frac{n}{2}) + 1$

$\Rightarrow$ Master Theorem: $a = 2, b = 2, k = 0$

$2 > 2^0 = 1 \Rightarrow$ Fall 1

$T(n) = O(n)$

Multiplikation großer ganzen Zahlen:

$$\begin{array}{r} 23 \cdot 14 \\ \hline 92 \\ +23 \\ \hline 322 \end{array}$$

Wieviel Ziffern-Multiplikationen? $\rightarrow$ 4

Allgemein bei n Ziffern: $O(n^2)$

$a = (a_1, a_0) \qquad b = (b_1, b_0)$

$= a_1 \cdot 10^1 + a_0 \cdot 10^0 \quad = b_1 \cdot 10^1 + b_0 \cdot 10^0$

$c = (c_2, c_1, c_0) = a \cdot b$

$c_2 = (a_1 \cdot b_1) \cdot 10^2$ $c_1 = (a_1 \cdot b_0 + a_0 \cdot b_1) 10^1$ $c_0 = (a_0 \cdot b_0) 10^0$

Trick: $c_1 = (a_1 + a_0) \cdot (b_1 + b_0) - (c_2 + c_0)$

$\Rightarrow$ jetzt 3 Multiplikation

D&C für n-stellige Multiplikation:

$a = (a_1 \cdot 10^{\frac{n}{2}}) + a_0$

$b = b_1 \cdot 10^{\frac{n}{2}} + b_0$

$c = a \cdot b = c_2 \cdot 10^n + c_1 \cdot 10^{\frac{n}{2}} + c_0$

$c_0 = a_0 \cdot b_0$ $c_1 = (a_1 + a_0) \cdot (b_1 + b_0) - (c_2 + c_0)$

$c_2 = a_1 \cdot b_1$

Aufwand Ziffern-Multiplikation:

$M(n) = 3\, M(\frac{n}{2}) + O \cdot 1$

$\hookrightarrow$ Master-Theorem: $a = 3, b = 2, k = 0$

$3 > 2^0 = 1$ Fall 1

$T(n) = O(n^{\log_2(3)})$

$= O(n^{1,585})$ $\Rightarrow$ besser als $O(n^2)$