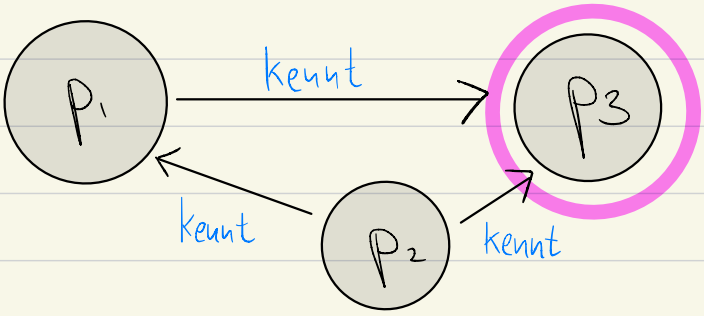
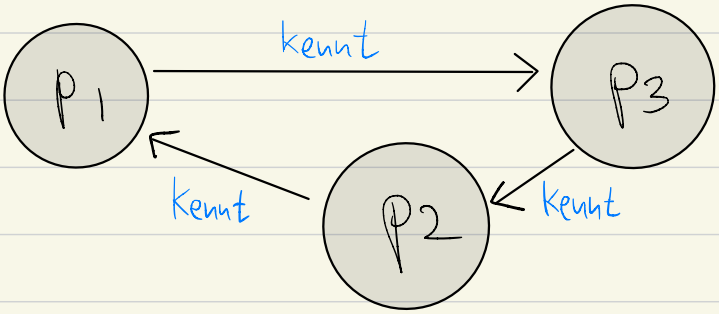
(wiederholt)

Star Suche

Star: 1. jede andere Person kennt Star

2. Star kennt keine andere Person





kein Star!

Ziel: unter n Personen $p_1, \dots, p_n$,

finde Star (falls da) mit

möglichst wenig Fragen $p_i \xrightarrow{?} p_j$

naiv: alle $n \cdot (n-1)$ Fragen

frage jede Person über jede andere

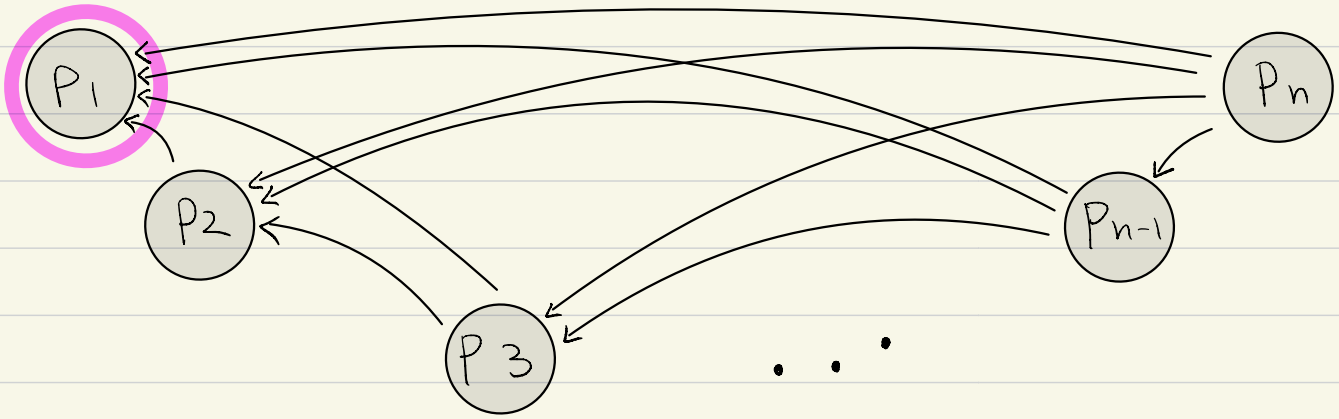
geht es besser?

rekursiver Ansatz ($n \geq 3$)

führe Lösung zurück auf Lösung für n-1 Personen

1. finde Star p_s unter $p_1, \dots, p_{n-1}$ (Rekursion)
2. teste ob p_s Star für $p_1, \dots, p_n$: 2 Fragen
3. — " — p_n — " — : $2 \cdot (n-1)$ — " —

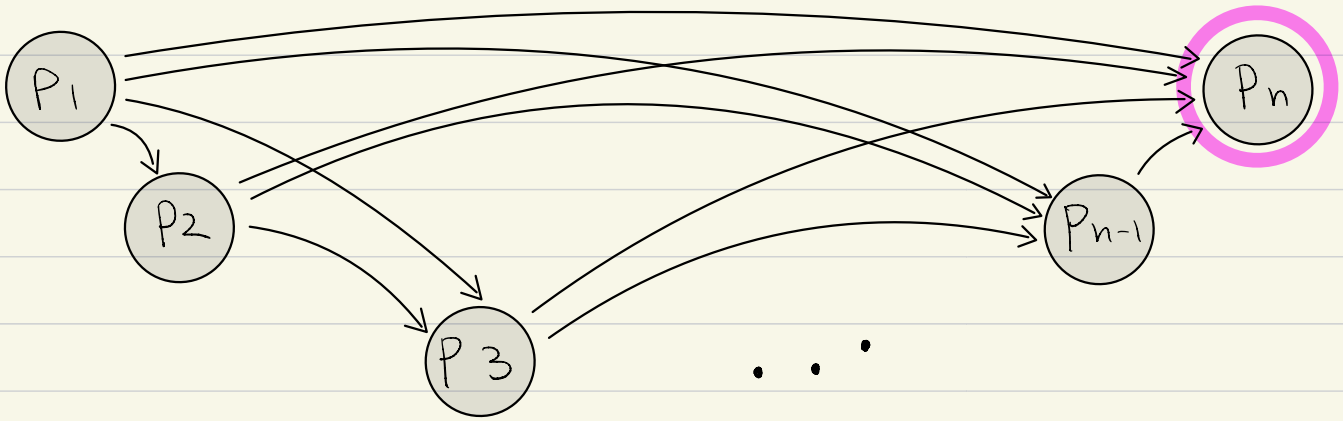
best-case: nie Schritt 3



n	2	3	4	...	n	Total
Fragen	2	+2	+2	...	+2	$2 \cdot (n-1)$

Rekursionsbasis

Worst-case: immer Schritt 3



n	2	3	4	...	n	Total
Fragen	2	+2·2	+2·3	...	+2·(n-1)	n·(n-1)

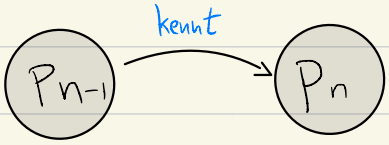
geht es besser?

Idee: stelle sicher, dass p_n kein Star (auch in Rekursion)

$\rightsquigarrow$ nie Schritt 3

wie?

vor Schritt 1:

vertausche p_{n-1} und p_n falls 

: +1 Fragen

n	2	3	4	5	...	n	Total
Fragen	2	+3	+3	+3	...	+3	$\underbrace{2+3\cdot(n-2)}$

$= 3n-4$

wichtiger als endgültiger Algorithmus:
Strategie um naheliegenden Alg. zu verbessern

Algorithmen vergleichen / bemessen

Qualitätsmass für Algorithmen

was heisst "besser"?

wichtige Kriterien: (diese Vorlesung)

- Korrektheit — Ergebnis richtig oder zumindest nützlich
- Laufzeit — mehr Daten pro Tag auswerten; mehr Kunden pro Tag bedienen

Herausforderungen

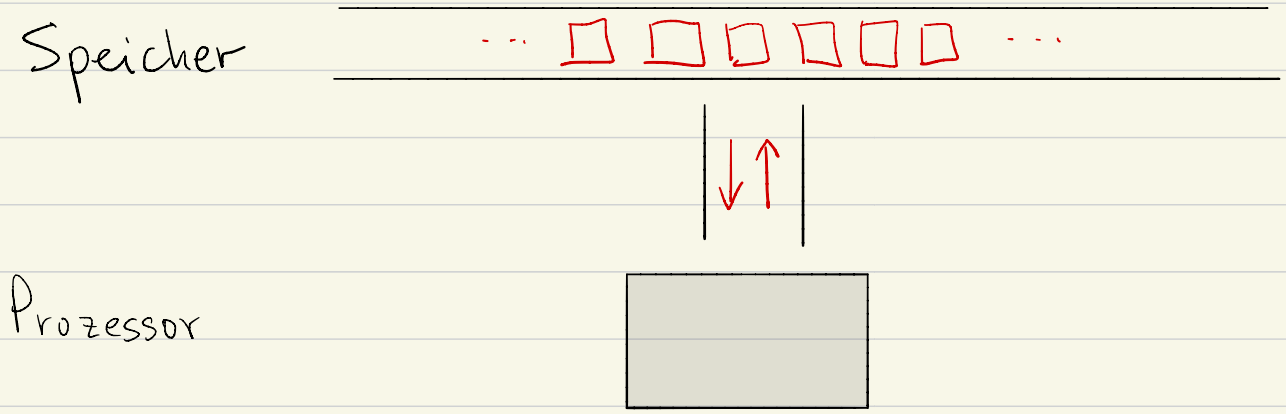
- welche Eingabe? (zukünftige Eingaben vorhersehbar?)
- welcher Computer? (Mobiltelefon, Laptop, Supercomputer)
- welche Implementierung? (Java, Python, ...)

Ziel: universelle Garantieen (alle Eingaben, alle Computer, alle Implementierungen)

wie:

- mathematische Beweise
- Abstraktionen und Modelle ← heute

Berechnungsmodell



Speicher: Liste von Speicherzellen

- Eingabe in ersten n Zellen (n = Eingabegrösse)
- jede Zelle leer oder enthält ganze Zahl ($\leq n^{100}$)
  genaue Schranke spielt oft keine Rolle
- Zellen frei addressierbar ("random access")

Prozessor: führt elementare Operationen aus

- lesen / schreiben von Speicherzellen
- vergleichen ($=, >, <$)
- rechnen ($+, -, \cdot, \div$)

in diesem Modell:

Laufzeit = Anzahl elementarer Operation

Vergleich mit Praxis

- mehr elementare Operationen (instruction set architecture)
- Laufzeit für Operation nicht festgelegt

Was nutzt das Modell?
Modelle sind nie genau können aber trotzdem nützlich sein

Wie weit auseinander ist Laufzeit im Modell und in Praxis?

höchstens konstanter Faktor!

hängt ab von Computer Hardware
aber nicht von Eingabe grösse

Konsequenz für Laufzeitanalyse:

- ignoriere konstante Faktoren
- beachte Wachstum mit Eingabegrösse (z.B. linear, quadratisch, ...)

Formal: asymptotische Notation
geeignete Abstraktion für Laufzeit

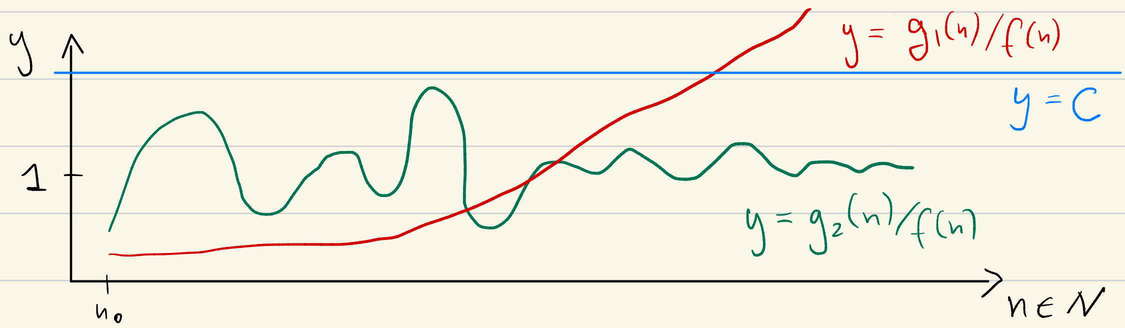
N = Menge möglicher Eingabegrössen (meist $N = \{n_0, n_0+1, \dots\}$, $n_0 \in \mathbb{N}$)

Definition: Für $f: N \to \mathbb{R}^+$, (z.B. Laufzeitfunktion) ($\mathbb{R}^+$ = positive reele Zahlen)

$$O(f) := \{ g: N \to \mathbb{R}^+ \mid \exists C > 0.\ \forall n \in N.\ g(n) \leq C \cdot f(n) \}$$

"Ordnung von f"

also: $g \in O(f) \iff \frac{g}{f}$ beschränkt (auf N)



Grenzwerte:

$\lim_{n \to \infty} \frac{g(n)}{f(n)} = \infty \implies g \notin O(f)$

$\quad$ — " — $\in \mathbb{R} \implies g \in O(f)$

wenn Grenzwert unbestimmt, beides möglich

Implikationen gelten jeweils nur in eine Richtung

Beispiele: (a) $10 \cdot n^{1.6} + 1000 \cdot n + 100 \in O(n^{1.6})$ $\quad (n_0 = 1)$

da $g(n)/f(n) = 10 + 1000 \cdot n^{-0.6} + 100 \cdot n^{-1.6} \leq 10 + 1000 + 100 = 1110$

b) $0.001 \cdot n^2 \notin O(n^{1.6})$ da $g(n)/f(n) = 0.001 \cdot n^{0.4}$ unbesch.

Schreibweise: $g(n) \leq O(f(n))$ statt $g \in O(f)$