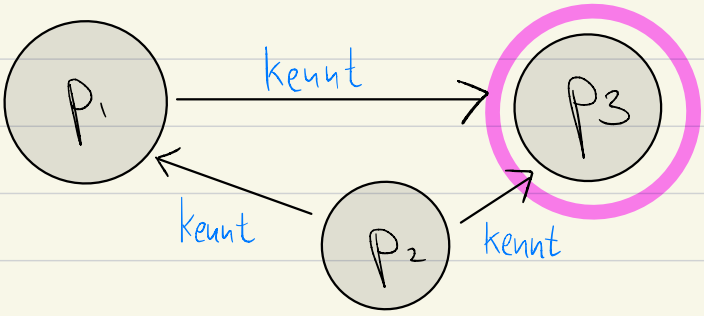
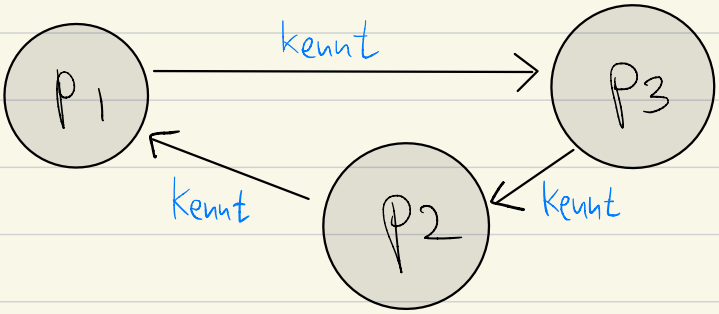
(wiederholt)

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

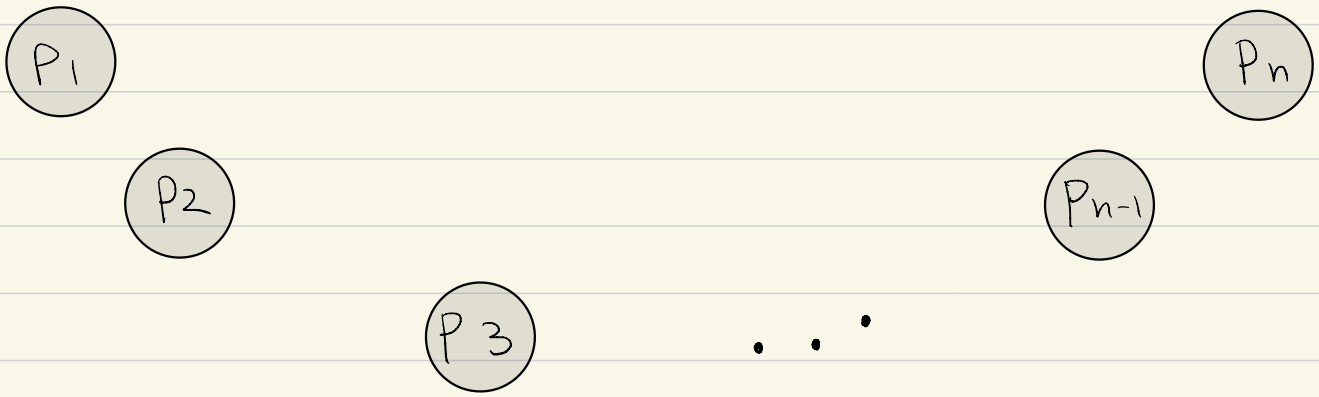
geht es besser?

[ ] Ansatz

führe Lösung zurück auf Lösung für n-1 Personen

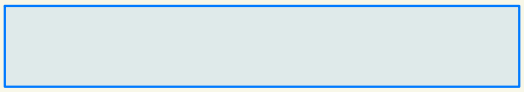
1. finde Star p_s unter $p_1, \dots, p_{n-1}$ (Rekursion)
2. teste ob p_s Star für $p_1, \dots, p_n$: [ ] Fragen
3. — " — [ ] — " — : [ ] — " —

best-case:



n	2	3	4	...	n	Total
Fragen	2	+2	+2	...	+2	[ ]

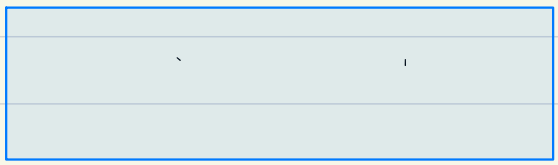
Worst-case :



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

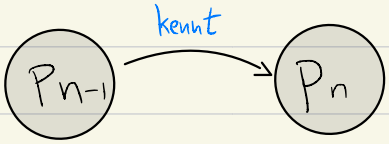
geht es besser?

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



wie?

vor Schritt 1:

vertausche pn-1 und pn falls 

: +  Fragen

n	2	3	4	5	...	n	Total
Fragen	2	+3	+3	+3	...	+3	

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

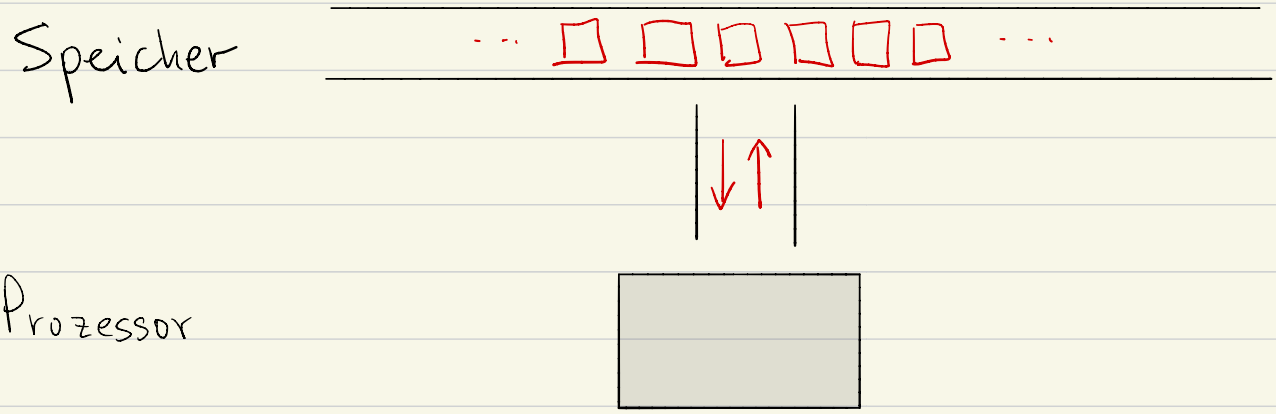
- welche Eingabe? (zukünftige Eingaben vorhersehbar?)
- welcher Computer? (Mobiltelefon, Laptop, Supercomputer)
- welche Implementierung? (Java, Python, ...)

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
- 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

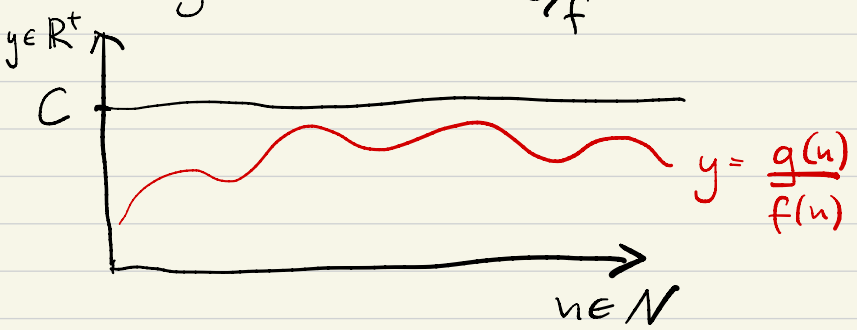
Definition: Für $f: \mathbb{N} \to \mathbb{R}^+$, $\mathbb{R}^+ = \{y \in \mathbb{R} \mid y > 0\}$,

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

meist: $\mathbb{N} = \{n_0, n_0+1, \dots\}$, $n_0 \in \mathbb{N}$, $n_0 \geq 1$

"Ordnung von f"

also: $g \in O(f) \iff g/f : X \to \mathbb{R}^+$ beschränkt



Beispiele: ($n_0 = 1$)

$\underbrace{10 \cdot n^{1.5g}}_{g(n)} + 1000 \cdot n + 100 \in O(\overbrace{n^{1.5g}}^{f(n)})$

da $\frac{g(n)}{f(n)} = 10 + 1000 \cdot n^{-0.5g} + 100 \cdot n^{-1.5g} \leq 10 + 1000 + 100 = 1110$

$0.0001 \cdot n^2 \in O(n^{1.5g})$

da $\frac{g(n)}{f(n)} = 0.0001 \cdot n^{0.41}$ unbeschränkt

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

Kursveränderungen einer Aktie:

7 -11 18 10 -23 -3 27 -1

Wann kaufen? Wann wieder verkaufen?

Gewinn:

Problem: Maximum Subarray

Eingabe: $a_1, \dots, a_n \in \mathbb{Z}$ (ganze Zahlen), $n \geq 1$

Ausgabe: grösstmöglich Teilsumme

$S^* = a_i + \dots + a_j \quad (i, j \in \{1, \dots, n\}, \; i \leq j)$

$S^* = 0$ falls alle Zahlen negativ

naiver Algorithmus: berechne alle Teilsummen

$$S_{i,j} := a_i + \cdots + a_j$$

elementare Operationen: Addition, Vergleichen, ...

Anzahl Additionen: $T(n) := \sum_{i=1}^{n} \sum_{j=i}^{n}$ [ ]

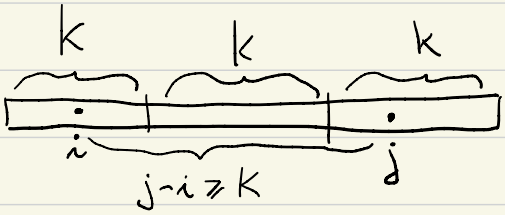
[ ] Anzahl Additionen für $S_{i,j}$

Obere Schranke:

$$T(n) = \sum_{i=1}^{n} \sum_{j=i}^{n} (j-i) \leq \sum_{i=1}^{n} \sum_{j=1}^{n} n = \bigcirc$$

Untere Schranke: $(n = 3k \in \mathbb{N})$

$$T(n) = \sum_{i=1}^{n} \sum_{j=i}^{n} (j-i) \geq \sum_{i=1}^{k} \sum_{j=2k+1}^{n} \underbrace{(j-i)}_{\geq \bigcirc} \geq \sum_{i=1}^{k} \sum_{j=2k+1}^{n} k = \quad [\quad]$$



Also: $T(n) \leq O(n^3)$ und $n^3 \leq O(T(n))$

Schreibweise dafür: $T(n) = \Theta(n^3)$ "gleiche Ordnung"

Geht es besser?

Idee: Teilsummen nicht unabhängig

z.B.: $S_{3,7} = S_{3,6} + a_6$

Pseudo code:

For $i = 1..n$:

$S_{ii} \leftarrow a_i$

For $j = i+1 .. n$:

$S_{ij} \leftarrow S_{i,j-1} + a_j$

} 2 nested loops

Anzahl Additionen:

i	1	2	3	...	n-1	n	Total
Additionen	n-1	n-2	n-3	...	1	0	

geht es besser?

müssen wir alle [ ] Teilsummen berechnen?