max planck institut informatik

Ideen und Konzepte der Informatik

Suchen und Sortieren

[Ordnung ist das halbe Leben]

Kurt Mehlhorn

(viele Folien von Kostas Panagiotou)

Suchen

- Welche Telefonnummer hat Kurt Mehlhorn?
- Wie schreibt man das Wort „Gerechtigkeit“?
- Welche Webseiten enthalten die Wörter „Uni Saarland“? Welche ist die „wichtigste“?

Web hat mehrere Billionen Seiten.

Suche nach der Nadel im Heuhaufen.

Bedeutung von Suchen

- Menschen verbringen viel Zeit mit Suchen und Ordnen (Sortieren), Computer auch.
- Suchen und Sortieren sind Hauptanwendungen von Computern.
- Es gibt hocheffiziente Suchverfahren: Suche im Web in weniger als 1 Sekunde.
- Sortieren hilft beim Suchen: Ordnung ist das halbe Leben.

Gibt es ein X in der Buchstabensuppe?

Bild von Ursus Wehrli, 2011.

Aha!

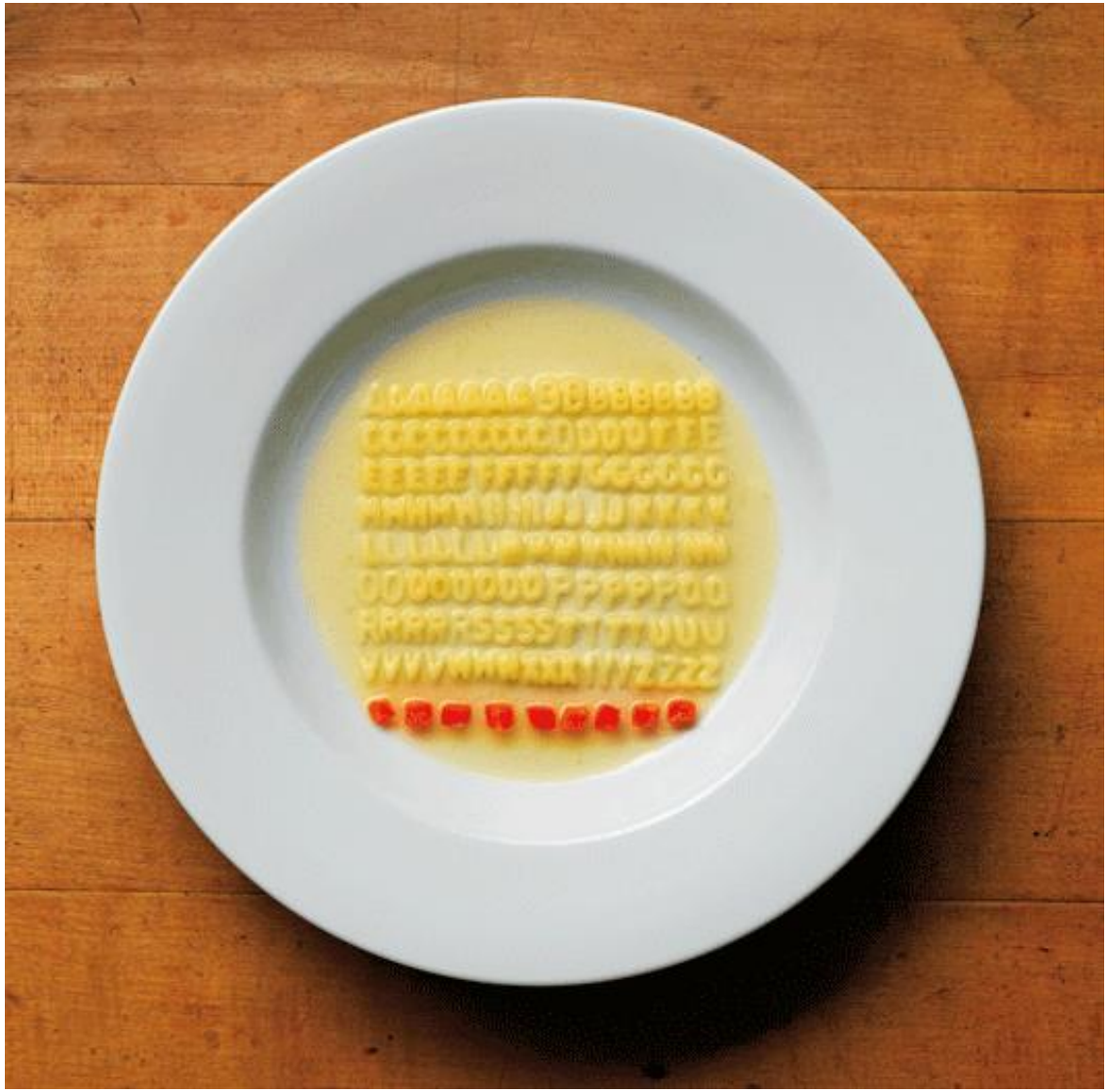


Bild von Ursus Wehrli, 2011.

Ein grünes Badetuch?

Bild von Ursus Wehrli, 2011

Aha!

Bild von Ursus Wehrli, 2011

Konzepte der heutigen Vorlesung

- Suchen = Information ablegen und verarbeiten, so dass man sie schnell wiederfindet (oder sagen kann, dass sie nicht vorhanden ist).
- Ordnung erleichtert das Suchen.
- Sortieren = nach einem Kriterium ordnen.
- Datenstruktur = Organisation einer Menge, so dass Suchen schnell geht.
- Laufzeit (Komplexität) von Algorithmen.

Wichtige Ordnungen

- Zahlen ordnet man der Größe nach, etwa

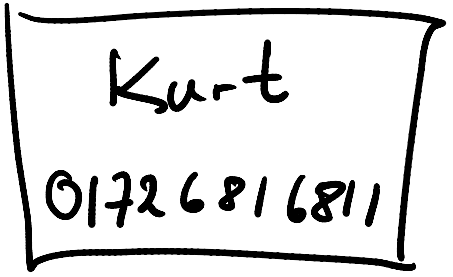
$$2 < 5 < 13 < 45$$

- Worte ordnet man alphabetisch, etwa

Kurt < Michael < Stefan < Stefanie

Suchen

- Daten können alles Mögliche sein:
  - Zeichenketten, Zahlen, Bilder, …



- Hier: (Name + Telefonnummer)
- Haben einen Karteikasten: auf jeder Karteikarte steht ein Name und eine Nummer.
- Frage: Wie viele Zettel muss man anschauen, bis man die Nummer zu einem Namen hat?

Grundoperation

- Zwei Objekte a und b vergleichen.
- Das Resultat ist
  - a < b, a ist kleiner als b, a steht vor b
  - a = b, a ist gleich b
  - a > b, a ist größer als b, a steht nach b
- Wir messen Effizienz in Anzahl von Vergleichen.
- Sagt auch tatsächliche Laufzeit gut vorher.

Zettelkasten ist ungeordnet

- Wir müssen **alle Karten** anschauen, um sicher zu sein, dass ein gesuchter Name nicht da ist.
- Falls ein Name da ist, im Mittel die Hälfte der Karten.

Anzahl der Vergleiche im schlechtesten Fall

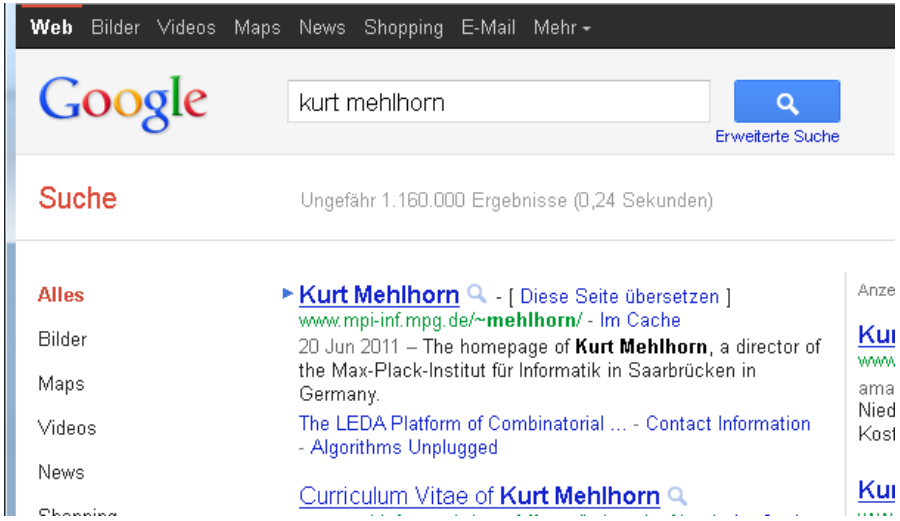
= Anzahl der Karten

Ein Beispiel

- Das Internet hat mehrere Billionen Webseiten
- 1 Billion = 1.000.000.000.000
- Optimistisch: Pro Sekunde können wir 1.000.000.000 Seiten durchsuchen

Dann brauchen wir 1.000 Sekunden!

Aber Google ist viel schneller



Wie machen wir das besser?

- Wir sortieren unsere Karteikarten nach Name.
  - Also wie in einem Telefonbuch
- Wir suchen nach X.
- Wir ziehen eine Karte, darauf steht Y.
- X kommt vor / nach Y in der alphabetischen Ordnung der Namen oder X = Y.
- Was wissen wir nun? Welche Karte nehmen wir für den ersten Vergleich? Wie geht es weiter?

Binärsuche

Konzept:
Divide and Conquer

- Gegeben:
  - Liste mit ***N*** Elementen.
  - Aufsteigend sortiert: Der Nachfolger ist größer als sein Vorgänger.
- Frage: Enthält die Liste ein Element ***x***?
- Algorithmus:

Der Algorithmus Binärsuche

- Karteikasten: L[1], L[2], …, L[N] N Karten

ℓ ← 1; r ← N;
Solange ℓ ≤ r
  // falls x vorkommt, dann x in L[ℓ],…, L[r]
  Sei *m* der mittlere Index, also *m* = floor((ℓ + r)/2)
  // es gilt ℓ ≤ m ≤ r
  falls x = L[*m*] dann fertig, x ist gefunden
  falls x < L[*m*] dann r ← m – 1;
  falls x > L[*m*] dann ℓ ← m + 1;

// x kommt nicht vor

Rekursive Version von Binärsuche

- Karteikasten: L[1], L[2], …, L[N] N Karten

Suche(Karteikasten, x) bestimmt ob x im Kasten vorkommt

falls Kasten leer dann fertig, x nicht vorhanden

Sei *m* der mittlere Index

falls x = L[*m*] dann fertig, x ist gefunden

falls x < L[*m*] dann Suche(Teilkasten der Elemente vor L[*m*], x)

falls x > L[*m*] dann Suche(Teilkasten der Elemente nach L[*m*], x)

Komplexität (Anzahl der Vergleiche)

- N = 1: # Vergleiche = 1
- N = 3: # Vergleiche = 2
- N = 7: # Vergleiche = 3
- N = 15: # Vergleiche = 4
- N = 31: # Vergleiche = 5

$2^{40} = 1.099.511.627.776$

- $N = 15 = 2 \cdot 2 \cdot 2 \cdot 2 - 1 = 2^4 - 1$ # Vergleiche = 4
- $N = 31 = 2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 - 1 = 2^5 - 1$ # Vergleiche = 5
- N = …. $= 2^{40} - 1$ # Vergleiche = 40

Das ist irre!!!!!

- Anzahl Vergleiche = Zweierlogarithmus(N + 1)

Lineare Suche vs. Binärsuche

- Binärsuche funktioniert, wenn man die gegebenen Daten ordnen kann:
  - (Name, Telefonnummer)
  - Webseiten?
- ***Lineare Suche: Aufwand = Anzahl der Elemente.***
- ***Binärsuche: Aufwand = Logarithmus der Anzahl der Elemente.***
- ***Binärsuche ist rasend schnell.***

Wie sortiert man?

Mischen zweier sortierter Folgen

- Zwei aufsteigend sortierte Folgen von je n Elementen kann man mit höchstens 2n – 1 Vergleichen zu einer sortierten Folge mischen
- Strategie: Vergleiche die beiden ersten Elemente und bewege das kleinere zur Resultatfolge. Wenn eine Folge erschöpft ist, bewege alle Elemente der anderen.

Mischen Pseudocode

Seien A[0] bis A[n – 1] und B[0] bis B[n – 1] sortierte Folgen.

Stelle C[0] bis C[2·n – 1] für das Ergebnis bereit

Setze i und j auf Null

Solange (i < n oder j < n)

Falls (i < n und j < n und A[i] < B[j]) oder j = n

dann bewege A[i] nach C[i + j] und erhöhe i um eins

sonst bewege B[j] nach C[i + j] und erhöhe j um eins

Sortieren durch Mischen

- Wir haben $n = 2^k$ Elemente und damit n sortierte Folgen der Länge 1
- Paare die sortierten Folgen und mische je zwei zu einer Folge der doppelten Länge
- Solange noch mehr als eine Folge, wiederhole

Sortieren durch Mischen, Analyse

1) Wir haben $n = 2^k$ Elemente und damit n sortierte Folgen der Länge 1.

2) Paare die sortierten Folgen und mische je zwei zu einer Folge der doppelten Länge.

3) Solange noch mehr als eine Folge, wiederhole.

Jede Ausführung von 2) kostet nicht mehr als n Vergleiche.

Wir wiederholen 2) k-mal.

Also nicht mehr als $n \cdot k = n \cdot \log n$ Vergleiche.

Sortieren durch Mischen

Sortieren durch Mischen

sortiert n Elemente

mit nicht mehr als n log n Vergleichen

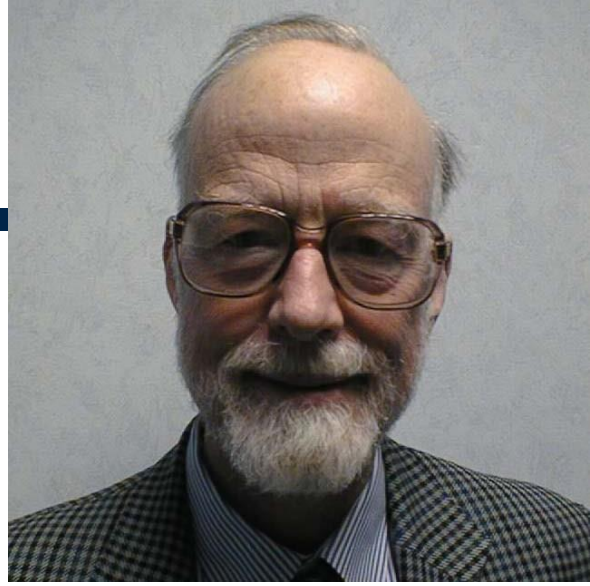
Was bedeutet das praktisch?

Tatsächliche Laufzeit auf KMs Rechner

- $n = 2^{22}$ 1.09 Sekunden
- $n = 2^{25}$ 9.94 Sekunden
- $n = 2^{29}$ 183 Sekunden
- $n = 2^{30}$ 1240 Sekunden

- Beachte $\frac{2^{29} \log 2^{29}}{2^{25} \log 2^{25}} = 16 \cdot \frac{29}{25} = 18.56$
- $\frac{183}{9.94} = 18.41$
- Analyse sagt Laufzeitwachstum gut vorher
- Aber letzte Zeile: Rechner ist in anderem Regime, benutzt Platte

Quicksort



Tony Hoare
Turing Award 1980

- S = Menge, die zu sortieren ist
- Wähle ein Element s in S
- Teile S in
  - $S_<$ = Elemente kleiner s
  - $S_>$ = Elemente größer s
- Gib aus

$$Sort(S_<) \quad s \quad Sort(S_>)$$

Rekursion endet, wenn $S_<$ und $S_>$ leer sind

Quicksort, Beispielausführung

Beim Teilen kann man Glück oder Pech haben

Laufzeit wie	$n \log n$	n^2
$n = 10^6$	0.1 sec	500 sec

Kann man das Glück erzwingen?

- Bis 1980: immer raffiniertere deterministische Strategien
- Seit 1980: wähle das Teilungselement zufällig

Randomisierter Algorithmus

- Urne mit roten und schwarzen Kugeln. Wie findet man eine rote Kugel ohne hinsehen?

Tony Hoare (1934 –

„Ich stelle fest, dass es zwei Wege gibt, ein Software-Design zu erstellen, entweder so einfach, dass es offensichtlich keine Schwächen hat, oder so kompliziert, dass es keine offensichtlichen Schwächen hat. Die erste Methode ist weitaus schwieriger.“

Tony Hoare, Dankesrede für den Turingpreis 1980

Zusammenfassung

- Binärsuche ist rasend schnell: 40 Vergleiche für Suche in einer Billion Elemente
- Sortieren ist billig: eine Million Elemente in 0.1 sec auf diesem Notebook
- Nächste Vorlesung: Websuche