Informatik II: Algorithmen und Datenstrukturen SS 2015

Vorlesung 4a, Dienstag, 12. Mai 2015

(Assoziative Felder aka Maps)

Prof. Dr. Hannah Bast
Lehrstuhl für Algorithmen und Datenstrukturen
Institut für Informatik
Universität Freiburg

UNI FREIBURG

Blick über die Vorlesung heute

UNI FREIBURG

- Organisatorisches
  - Ihre Erfahrungen mit dem Ü3 (O-Notation)
  - Bitte ins Forum schauen
- Assoziative Felder aka Maps aka = also known as
  - Normale Felder vs. Assoziative Felder
  - Anwendungsbeispiel: MapCountingSort
  - Maps in Python / Java / C++

Erfahrungen mit dem Ü3 1/3

UNI FREIBURG

- Zusammenfassung / Auszüge Stand 12. Mai 12:00
  - Immer noch über 100 Leute dabei … sehr schön
  - Blatt sehr theoretisch, bitte mehr Praxis … Geduld bitte
  - Zeitaufwand war für die meisten diesmal ok
  - Ich hasse Integrale … schade, gab aber eine Alternative
  - Beim Layout der Lösungen von solchen Aufgaben gibt es noch einigen Nachholbedarf … Empfehlung: LaTeX
  - Einige haben sowas geschrieben wie $\Theta(f) = \log n$

    Das ist falsch rum, man schreibt: **$f = \Theta(\log n)$**

    Und mathematisch korrekt wäre: **$f \in \Theta(\log n)$**
  - Korrekturen sind Sonntag, spätestens Montag da

Θ(…) ist eine Menge von Fkt.

nicht symmetrisch

oder $\Theta(f) = \Theta(\log n)$

Erfahrungen mit dem Ü3 2/3

UNI FREIBURG

- Lösungsskizze Aufgabe 1

$$\sum_{i=1}^{n} \log i = \Theta(n \cdot \log n)$$

$$\text{„}O\text{“}: \quad \sum_{i=1}^{n} \underbrace{\log i}_{\leq \log n} \leq n \cdot \log n = O(n \cdot \log n) \qquad c=1,\ n_0=1$$

$$\text{„}\Omega\text{“}: \quad \sum_{i=1}^{n} \log i \geq \sum_{i=\lceil \frac{n}{2} \rceil}^{n} \underbrace{\log i}_{\geq \log \frac{n}{2}} \geq \frac{n}{2} \cdot \underbrace{\log \frac{n}{2}}_{\log n - 1 \geq \frac{\log n}{2},\ n \geq 2}$$

$$\geq \frac{n}{4} \cdot \log n = \Omega(n \cdot \log n) \qquad c=\frac{1}{4},\ n_0=2$$

Erfahrungen mit dem Ü3 3/3

$\Theta(f) = n \cdot \log n$

was $n \cdot O(\ldots)$ eben steht, immer so einfach wie möglich, also NICHT $O(2 \cdot n)$

- Lösungsskizze Aufgabe 3

```
i = 2; while (i < n) { i = i * 3; }
```

2, nach #1 $2 \cdot 3$, nach #2 $2 \cdot 3^2$, $2 \cdot 3^3$, ..., nach #k $2 \cdot 3^k$

L = Anzahl Iterationen / Durchläufe

(1) $2 \cdot 3^{L-1} < n$ weil noch nicht fertig

(2) $2 \cdot 3^{L} \geq n$ weil dann fertig

(1) $\Rightarrow L < \log_3 \frac{n}{2} + 1 = O(\log n)$ (exakt mit c, n_0)

(2) $\Rightarrow L \geq \log_3 \frac{n}{2} = \Omega(\log n)$ (dito)

ALTERNATIV: $L = \lceil \log_3 \frac{n}{2} \rceil$

↑ kürzeren Nachdenken

Forum

UNI FREIBURG

- Fragen zu den Übungsblättern
  - Es stehen öfter mal Kommentare in den erfahrungen.txt, dass Sachen unklar waren etc.
  - Fast immer wurden diese Sachen dann auch im Forum gefragt und dort auch beantwortet
  - Also bitte:

    Bei Unklarheiten immer gleich im Forum fragen

    Bzw. erstmal nachschauen, ob die Frage vielleicht schon gestellt und beantwortet wurde

Assoziative Felder 1/10

U = „Universum"

- Definition
  - Verwalten einer Menge von n Elementen, jedes mit einem eindeutigen Schlüssel S aus einer beliebigen Menge U

    Terminologie: Schlüssel = **Key**, Element = **Value**
  - Uns interessieren insbesondere folgende Operationen:

    insert(key, value) Einfügen von value mit Schlüssel key

    lookup(key) Ist key ∈ S und wenn ja mit welchem Wert

    erase(key) Falls key ∈ S, dann das Elemente löschen

    Wir schauen uns jetzt erstmal nur die ersten beiden Operationen an (insert und lookup)

Ist UNSYMMETRISCH: wenn man auch über „value" zugreifen will, braucht man dasselbe nochmal in die andere Richtung

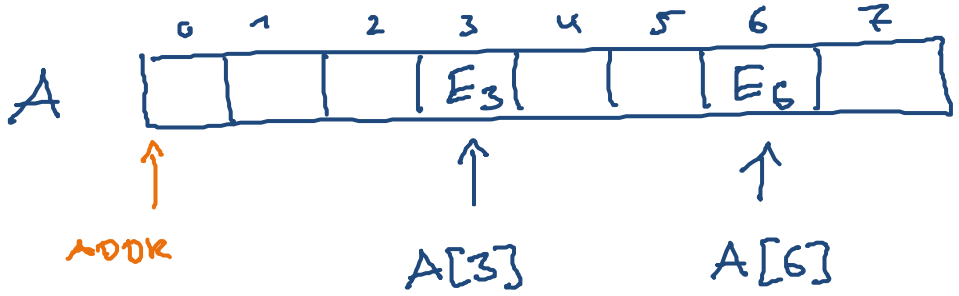
Assoziative Felder 2/10

UNI FREIBURG

- Beispiel 1
  - Das normale Feld ist ein Spezialfall mit $U = \{0, \ldots, n-1\}$

    Der Schlüssel von einem Element ist dann gerade die Position in dem Feld (Index)
  - Dann bekommt man

Laufzeit insert	O(1)	besser geht's nicht
Laufzeit lookup	O(1)	besser geht's nicht
Platzverbrauch	O(n)	besser geht's nicht



Adresse vom A[i] = ADDR + i · S

S = Größe vom einem Element

Assoziative Felder 3/10

UNI FREIBURG

- Beispiel 2
  - Datensätze aller Informatik Studierenden mit Matrikelnummer

1234567	Studi X
3276432	Studi Y
2334523	Studi Z

Matrikelnummer ist eindeutig pro Datensatz, von daher ein geeigneter Schlüssel

|U| = 10 Millionen

U = {0, …, 9.999.999}

Annahme: Matrnr. bis zu 7-stellig.

Die Schlüsselmenge S (Matrikelnummern der Infostudis) ist viel kleiner als die Menge U aller möglichen Matrikelnummern

Assoziative Felder 4/10

UNI FREIBURG

- Beispiel 3 12, 12, 17, 12, 12, 17, 4, 4, 4, 17, 4, 4, 12, 4, 4

  – Die verschiedenen Zahlen aus der Eingabe zu einem Sortieralgorithmus, und wie oft sie jeweils vorkommen

12	5 mal
17	3 mal
4	7 mal

U = {Menge aller möglicher Zahlen}

Die Schlüsselmenge S ist hier ebenfalls viel kleiner als U

Dazu schreiben wir später ein Programm

Assoziative Felder 5/10

UNI FREIBURG

- Beispiel 4
  - Suchanfragen aus der Log-Datei einer Suchmaschine mit Ihrer Häufigkeit

baby shower purple	5 mal
citroen cx	38 mal
fuji digital cameras	6 mal

    Ähnlich wie Beispiel 3, nur dass die Schlüssel hier Zeichenketten sind, dazu mehr in der Vorlesung morgen
  - Das wird Gegenstand der 2. Aufgabe vom Ü4 sein

Assoziative Felder 6/10

UNI FREIBURG

- Beispiel 5
  - Wie Beispiel 3, nur mit einem kleineren U und abstrakten Elementen, als handliches Beispiel für die nächsten Folien

Schlüssel Elemente

5	Element 1
14	Element 2
12	Element 3

|U| = 20

Die Schlüssel kommen aus der Menge U = {0, …, 19}

Die genau Beschaffenheit der Elemente wird auf den nächsten Folien keine Rolle spielen … es sind einfach nur Daten, die an der betreffenden Stelle gespeichert werden

Assoziative Felder 7/10

5 14 12
E_1 E_2 E_3

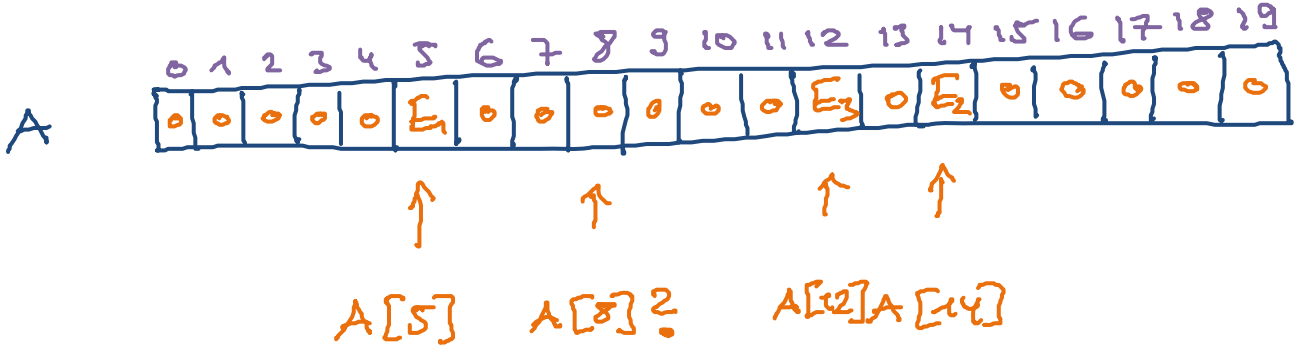
$U = \{0, \ldots, 19\}$

- Realisierung 1
  - Feld A der Größe |U|, für jedes i ∈ S steht an A[i] das zugehörige Element, sonst ein spezieller Wert, z.B. 0
  - Dann bekommen wir:

    Laufzeit insert: $\in \mathcal{O}(1)$ perfekt

    Laufzeit lookup: $\in \mathcal{O}(1)$ perfekt

    Platzverbrauch: $\in \mathcal{O}(|U|)$ schlecht, wenn $n \ll |U|$



Falls $n = |U|$, dann normales Feld

Assoziative Felder 8/10

5 14 12
E_1 E_2 E_3

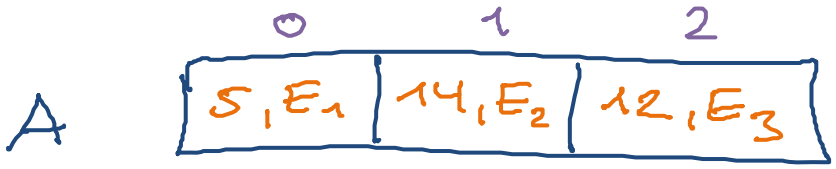
$U = \{0, \dots, 19\}$

- Realisierung 2
  - Feld der Größe |S|, an Stelle i steht einfach das i-te Elemente zusammen mit seinem Schlüssel
  - Dann bekommen wir:

    Laufzeit insert: $\Theta(1)$ perfekt

    Laufzeit lookup: $\Theta(n)$ grauenvoll

    Platzverbrauch: $\Theta(n)$ perfekt



Assoziative Felder 9/10

3 14 12
E_1 E_2 E_3

$U = \{0, \ldots, 19\}$

- Realisierung 3
  - Wie Realisierung 2, aber sortiert

    Dann geht lookup schneller, aber man muss beim Einfügen schauen, dass es sortiert bleibt

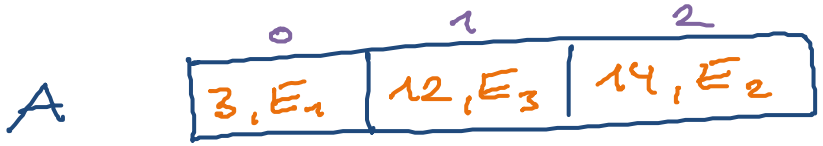
    Das machen wir in Vorlesung 8a und 8b (Suchbäume)
  - Dann bekommen wir:

    Laufzeit insert O(log n)

    Laufzeit lookup O(log n)

    Platzverbrauch $\Theta(n)$

siehe V8a u. 8b
binäre Suche



Assoziative Felder 10/10

UNI FREIBURG

- Realisierung 4 (Wunsch)
  - Wir hätten gerne eine Datenstruktur, mit der wir für eine Menge von n Elemente mit beliebigen Schlüsseln n aus einer beliebigen Menge U bekommen:

Laufzeit insert:	$\Theta(1)$	besser geht's nicht
Laufzeit lookup:	$\Theta(1)$	besser geht's nicht
Platzverbrauch:	$\Theta(n)$	besser geht's nicht

Das geht mit einer HashMap, dazu morgen mehr, heute wollen wir einfach erstmal schauen, wie man sie benutzt

MapCountingSort 1/3

UNI FREIBURG

- Unterschied zu CountingSort
  - Mit CountingSort: sortieren mit Zeit O(n) und Platz O(m) wenn die n Zahlen aus dem Bereich 1..m waren
  - Mit einer Map können wir das jetzt verallgemeinern zu:

    n **beliebige** Zahlen, aber höchstens m verschiedene

    Beispiel: 17 17 3 3 17 3 3 3 3 12 12 3

    Lösung 3,3,3,3,3,3,3,12,12,17,17,17

MapCountingSort 2/3

UNI FREIBURG

- Algorithmus am Beispiel $n = 12$, $m = 3$

  Beispiel: 17 17 3 3 17 3 3 3 3 12 12 3

  – **Schritt 1:** Mit einer Map die Anzahl Vorkommen zählen

  Beispiel: 17: 3 mal, 3: 7 mal, 12: 2 mal

  – **Schritt 2:** Diese Anzahlen nach den Werten sortieren

  Beispiel: (3, 7), (12, 2), (17, 3) ← danach sortiert

  Zum Sortieren in ein Feld von Paaren Wert, Anzahl schreiben; dieses Feld dann nach den Werten sortieren

  – **Schritt 3:** Dann die Ausgabe schreiben wie gehabt

  Beispiel: 3, 3, 3, 3, 3, 3, 3 (7 mal), 12, 12 (2 mal), 17, 17, 17 (3 mal)

MapCountingSort 3/3

UNI FREIBURG

- Laufzeitanalyse
  - Die Laufzeit ist $\Theta(n \cdot M + m \cdot \log m)$

    wobei M = durchschnittliche Kosten einer Map Operation

    Schritt 1: $\Theta(n \cdot M)$ n Map Operationen

    Schritt 2: $\Theta(m \cdot \log m)$ z.B. mit QuickSort

    Schritt 3: $\Theta(n)$ n Werte ausgeben
  - Also gut, wenn M klein und $m << n$
  - Insbesondere linear, wenn $M = O(1)$ und $m = O(n / \log n)$

$m \leq \frac{n}{\log n} \Rightarrow m \cdot \log m \leq n$

$\leq \log n$

$\leq \frac{n}{\log n}$

z.B. $n = 2^{40} \approx 10^{12}$

$m \leq \frac{10^{12}}{40}$

Ass. Felder in Python, Java, C++ 1/5

UNI FREIBURG

- Python ... da heißt ein assoziatives Feld **Dictionary**
  - Grundoperationen

	keine Typinformation nötig
map = {}	Erzeuge leere Map
map[key] = value	Einfügen von key mit Wert value
value = map[key]	Wert für key
if key in map: ...	Fragen, ob key enthalten ist
del map[key]	Löschen von key
maps.items()	Alle key, value Paare

in Python3 : list(map.items())

Ass. Felder in Python, Java, C++ 2/5

UNI FREIBURG

- Java ... da heißt ein assoziatives Feld **Map**
  - Grundoperationen

K = key type, V = value type	
Map<K, V> map = …	Erzeuge leere Map
map.put(key, value);	Einfügen von key mit Wert value
value = map.get(key);	Wert für key
if (map.containsKey(key)) …	Fragen, ob key enthalten ist
map.remove(key);	Löschen von key
map.entrySet();	Alle key, value Paare

Ass. Felder in Python, Java, C++ 3/5

UNI FREIBURG

- C++ ... da heißt ein assoziatives Feld ebenfalls **Map**
  - Grundoperationen

	K = key type, V = value type
std::map<K, V> map;	Erzeuge leere Map
map[key] = value;	Einfügen von key mit Wert value
value = map[key];	Wert für key
if (map.count(key)) …	Fragen, ob key enthalten ist
map.erase(key)	Löschen von key
for (auto item : map) …	Iteration über alle key, value Paare

C++11

Ass. Felder in Python, Java, C++ 4/5

UNI FREIBURG

- C++
  - Achtung bei folgendem Nebeneffekt:

    `if (map[key]) ...` — Fügt key mit Wert 0 ein, falls key bisher nicht in map war

    `if (map.count(key) > 0) ...` — Fragt nur, ob key in map ist

  - Das ist gefährlich, kann aber auch nützlich sein, z.B.

    `map[key]++;` — Erhöht den Wert von key; falls key noch nicht in map, wird vorher mit 0 intialisiert

Ass. Felder in Python, Java, C++ 5/5

UNI FREIBURG

- Effizienz
  - Hängt von der Implementierung ab; effizient sind

    Hashtabellen Vorlesungen 4b, 5a, 5b

    Suchbäume Vorlesungen 8a und 8b
  - In den diversen Programmiersprachen:

    Java: java.util.HashMap und java.util.TreeMap

    C++11: std::unordered_map und std::map

    Python: ist dem Compiler überlassen

Literatur / Links

UNI FREIBURG

- Assoziative Arrays
  - In Mehlhorn/Sanders:

    4 Hash Tables and Associative Arrays (das führt schon weiter)
  - In Wikipedia

    http://de.wikipedia.org/wiki/Assoziatives_Feld

    http://en.wikipedia.org/wiki/Associative_array
  - In Python, Java, C++

    http://docs.python.org/tutorial

    http://docs.oracle.com/javase

    http://www.cplusplus.com/reference