Informatik II: Algorithmen und Datenstrukturen SS 2013

Vorlesung 4, Dienstag, 7. Mai 2013

(Assoziative Arrays 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)
  - Feedback von Ihrem Tutor / Ihrer Tutorin
  - Morgen (Mittwoch, 8. Mai) KEINE Vorlesung

    Dafür ein etwas aufwändigeres Übungsblatt
- Assoziative Arrays aka Maps aka = also known as
  - Nutzen anhand eines typischen Beispieles
  - Alternative dazu: Sortieren
  - Ü4, Aufgabe 1: Variante des Beispiels aus der Vorlesung
  - Ü4, Aufgabe 2: Allgemeineres CountingSort mit einer Map

Erfahrungen mit dem Ü3 (O-Notation)

UNI FREIBURG

- Zusammenfassung / Auszüge Stand 7. Mai 16:00
  - Für die meisten gut machbar, für viele auch schnell
  - Das am wenigsten aufwändige Übungsblatt bisher

    Das sei ja gerade das Problem mit der durchschnittlichen Arbeits-belastung nach ECTS Punkten: die Belastung ist zu ungleich verteilt
  - Hoffentlich bald wieder mehr Implementierung ... JA !
  - Je später die Abgaben, desto schlechter die Erfahrung

    Früher anfangen + im Forum fragen !
  - Was ist log ohne was: ln oder $\log_2$ oder $\log_{10}$?

    Innerhalb von O, Θ, etc. spielt das keine Rolle !
  - Daphne war letztes Semester schöner

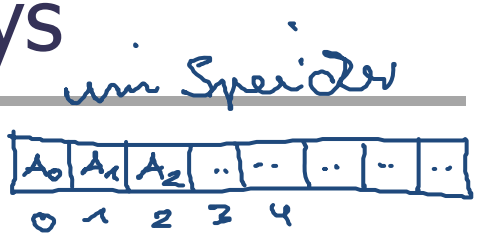
Feedback von Ihrem/r Tutor/in

UNI FREIBURG

- Machen Sie dazu svn update
  - ... spätestens bevor Sie ein neues Übungsblatt bearbeiten
  - Sie bekommen dann (hoffentlich) eine Datei

    uebungsblatt_XY/feedback-tutor.txt
  - Dort sehen Sie unter anderem
    - Wie viele Punkte und warum nicht
    - Verbesserungsvorschläge für Code / Beweise
    - Antworten auf Fragen aus Ihrer erfahrungen.txt
    - Und was Ihrem/r Tutor/in noch alles so einfällt ...
  - Wenn Sie meinen, dass Sie von einem Treffen profitieren würden, schreiben Sie Ihrem/r Tutor/in eine Mail !

Normale vs. Assoziative Arrays



- Normales Array
  - Zugriff auf eine (große) Anzahl von gleichartigen Elementen über einen fortlaufenden **Index**

    A_0, A_1, A_2, A_3, A_4, A_5, ...
  - Beispielanfrage: das Element mit Index 3
- Assoziatives Array
  - Zugriff auf eine (große) Anzahl von gleichartigen Elementen über einen beliebigen sog. **Schlüssel** (Key)
  - A_{Peter}, A_{Paul}, A_{Mary}, $A_{Elizabeth}$, ...
  - Beispielanfrage: das Element mit Schlüssel Elizabeth
  - Die Schlüssel können auch (nicht fortlaufende) Zahlen sein

Und wofür braucht man das?

UNI FREIBURG

- In der Praxis ...
  - ... gibt es kaum ein größeres Programm wo man nicht irgendwo ein assoziatives Array braucht
- Unser Beispiel für diese Vorlesung
  - Eine Liste von (bekannten) Personen aus FreeBase
  - Wir interessieren uns für die Vornamen
  - Insbesondere: welche Vornamen sind am häufigsten
  - Wir schauen uns zwei typische Lösungen dafür an
    - Mit Sortieren
    - Mit einem assoziativen Array aka Map

Lösung 1: Sortieren

TSV = TAB-separated values.

UNI FREIBURG

- Idee
  - Wir sortieren die Zeilen nach dem Vornamen
  - Dann stehen alle Personen mit dem selben Vornamen hintereinander
  - Die Größe von jedem Block können wir dann zählen
  - Und dann nach der Größe der Blöcke sortieren
- Nachteil
  - Sortieren braucht Zeit $\Theta(n \log n)$ und geht nicht in $O(n)$
  - Wir müssen **zweimal** über die ganzen Daten laufen
- Vorteil
  - Algorithmisch einfach
  - Kommandozeile: geht mit einfachen Unix/Linux-Befehlen

Lösung 2: Assoziatives Array / Map

UNI FREIBURG

- Idee
  - Ein assoziatives Array mit den Vornamen als Schlüssel
  - Der Wert ist entweder einfach ein Zähler, oder die Liste aller Leute mit dem Vornamen
- Vorteil
  - Laufzeit O(n)
  - Vorausgesetzt wir können in Zeit O(1) auf ein Element des Arrays zugreifen, wie bei einem normalen Array
  - Dass das geht, sehen wir **nächste Woche**

    Aber erst mal sollen Sie verstehen (und zwar durch eigene Erfahrung), wozu und wie man assoziative Arrays benutzt !

Map in Java und C++ 1/3

UNI FREIBURG

- In Java und C++ heißen die assoziativen Arrays **Map**
  - Der Index heißt dort **key**, das Element heißt **value**
  - Eine Map unterstützt u.a. die folgenden Operationen
    - **Einfügen** von Element value mit Schlüssel key
      - in Java put(key, value) ... in C++ insert(key, value)
    - **Zugriff** auf das Element mit Schlüssel key
      - in Java get(key) ... in C++ operator[](key)
    - **Löschen** des Elementes mit Schlüssel key
      - in Java remove(key) ... in C++ erase(key)
    - **Fragen** ob Element mit Schlüssel key da ist
      - in Java containsKey(key) ... in C++ count(key)

Map in Java und C++ 2/3

UNI FREIBURG

- Effizienz
  - Hängt von der Implementierung ab, es gibt insbesondere
    - In Java: java.util.HashMap und java.util.TreeMap
    - In C++: __gnu_cxx::hash_map und std::map

      in C++11 ist die hash map std::unordered_map
  - Was das genau ist, sehen wir in der nächsten Vorlesung
    - Da bauen wir uns unsere eigene **Map**
    - Und analysieren sie dann

Map in Java und C++ 3/3

UNI FREIBURG

- **Vorsicht** bei folgendem Feature der Map in C++
  - Nehmen wir an, wir haben eine map mit Schlüsseln vom Typ std::string und Elementen vom Typ int

    ```
    std::map<std::string, int> M;
    ```

  - Dann kann man auf Elemente einfach mit dem [] Operator zugreifen wie bei einem normalen Array

    ```
    M["peter"] = 54;
    M["paul"] = 52;
    ```

  - Aber Vorsicht, wenn das Element vorher nicht in der Map war, wird es durch [] angelegt, mit dem Defaultwert für den Typ von value, für int ist das 0.

    ```
    if (M["mary"] > 0) ...            // Inserts "mary" with value 0.
    if (M.count("mary") > 0) ...   // Only asks if "mary" is there.
    ```

Erweitertes CountingSort 1/3

UNI FREIBURG

- In Vorlesung 2b hatten wir CountingSort gesehen
  - Damit konnte man in Zeit O(n) und Platz O(m) sortieren, 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 17 3 12 12

Erweitertes CountingSort 2/3

UNI FREIBURG

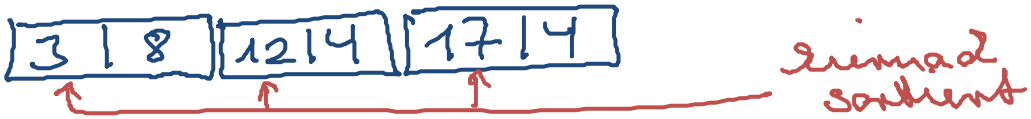
- MapCountingSort, Algorithmus + Beispiel

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

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

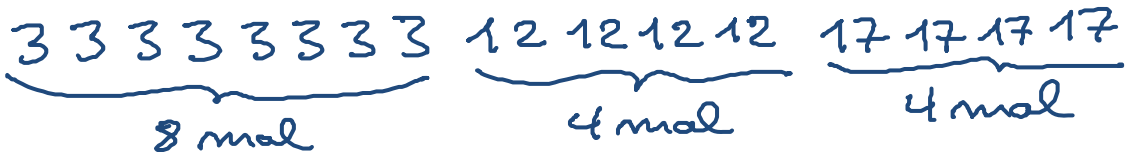
  Beispiel: 17 : 4 mal , 3 : 8 mal , 12 : 4 mal

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

  Beispiel: 

  Hinweis: Zum Sortieren in ein Feld schreiben, dessen Elemente Paare von Schlüssel, Anzahl sind. Dieses Feld dann nach den Schlüssel sortieren.

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

  Beispiel: 

Erweitertes CountingSort 3/3

UNI FREIBURG

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

    Schritt 1: $\Theta(n)$ falls Map Operationen in $\Theta(1)$

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

    Schritt 3: $\Theta(n)$
  - Das sollte also dann gut funktionieren, wenn $m << n$

    Insbesondere linear, wenn $m = O(n / \log n)$

ein Beispiel $n = 16$, $m = 3$

Literatur / Links

UNI FREIBURG

- Assoziative Arrays
  - In Mehlhorn/Sanders:

    4 Hash Tables and Associative Arrays (das führt schon weiter)
  - In Cormen/Leiserson/Rivest

    12 Hash Tables (das ebenso)
  - In Wikipedia

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

    http://en.wikipedia.org/wiki/Associative_array
- Map in Java und in C++

  http://download.oracle.com/javase/1.4.2/docs/api/java/util/Map.html
  - und ...HashMap bzw. ...TreeMap

  http://www.cplusplus.com/reference/stl/map/