Algorithmen und Datenstrukturen (für ESE) WS 2010 / 2011

Vorlesung 1, Montag, 18. Oktober 2010
(Einführung, Organisatorisches, Dynamische Felder)

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

UNI FREIBURG

Allgemeines zu dieser Vorlesung 1/2

UNI FREIBURG

- Thema
  - Letztes Semester in der C++ Vorlesung ging es darum, wie man überhaupt ordentlich programmiert
  - Fragen der Performanz spielten noch kaum eine Rolle
  - Genau darum geht es jetzt in dieser Vorlesung
    - Wie schnell ist mein Programm
    - Wie kann ich es schneller machen
    - Wie kann ich beweisen, dass es immer so schnell läuft
    - Manchmal geht es auch um Sparsamkeit im Platzverbrauch u.ä., aber meistens um Zeit
  - Wie in der C++ Vorlesung kein Anspruch auf Vollständigkeit
  - Aber sie sollen die BASICS lernen, wenn Sie die beherrschen, können Sie sich den Rest selber aneignen

Allgemeines zu dieser Vorlesung 2/2

- Art und Weise
  - Wieder großer Praxisbezug (wie in der C++ Vorlesung)
  - Aber auch Theorie (sofern sie der Praxis nutzt)
  - Auch die Übungsblätter halb theoretisch, halb praktisch
  - Ablauf der praktischen Übungen wie in der C++ Vorlesung
  - Nach soviel Praxis im letzten Semester fangen wir aber heute mit etwas Theorie an (praktisch motiviert natürlich)
- Aufwand
  - Es gibt für die Veranstaltung 4 ECTS Punkte
  - Das entspricht 120 Arbeitsstunden für das ganze Semester
  - 14 Vorlesungen à 6 Stunden Arbeit + Klausur am Ende
  - Also ca. 4 Stunden / Übungblatt (es gibt jede Woche eins)

Übungsgruppen

- Es gibt EINE zentrale Übung zur Veranstaltung
  - Wird abwechselnd von einem der TutorInnen gehalten:
    - Ina Baumgarten (Info BSc), Mirko Brodesser (Info MSc), Marjan Celikik (Info PhD), Sebastian Sester (ESE BSc)
  - Wird aufgenommen (wenn auch vielleicht nicht ganz so professionell geschnitten wie die Vorlesung)
  - Keine Anwesenheitspflicht (weder in der Vorlesung noch in den Übungsgruppen)
  - Aber mindestens 60% der Punkte in den Übungsblättern ist Teilnahmevoraussetzung für die Klausur
  - Punkte aus den Übungsblättern können Endnote verbessern, siehe nächste Folie
  - Abgabe / Korrektur Übungsblätter wieder über SVN

Note

- Die Endnote ergibt sich folgendermaßen
  - Die Punkte aus der ersten Hälfte der Übungsblätter werden auf max. 10 Punkte skaliert → ÜH1
  - Die Punkte aus der zweiten Hälfte der Übungsblätter werden auf max. 10 Punkte skaliert → ÜH2
  - In der Abschlussklausur wird es 5 Aufgaben à jeweils 10 Punkte geben → K1, K2, K3, K4, K5
  - Von ÜH1, ÜH2, K1, K2, K3, K4, K5 werden die besten 5 genommen und aufsummiert (maximal 50 Punkte)
  - Aus dieser Summe wird die Endnote ermittelt
  - Schummeln / Plagiat bei den Übungsblätter bzw. bei der Klausur hat wie immer Nicht-Bestehen zur Folge

    (einmal reicht, das tut man ja nicht aus Versehen)

Themen (kann sich im Detail noch ändern)

- Grundlegende Algorithmen und Datenstrukturen
  - Dynamische Felder
  - Sortieren
  - Hashing
  - Prioritätswarteschlangen
  - Listen
  - Kürzeste Wege / Dijkstra's Algorithmus
  - Suchbäume / Skip Lists
- Methodologisches
  - Asymptotische Analyse, O-Notation
  - Rekursionsgleichungen
  - Korrektheitsbeweise

Dynamische Felder

- Was?
  - Das was in der STL der std::vector ist
  - Ein Feld, das mit dem Bedarf wächst
  - Es gibt kaum ein Programm wo man das nicht braucht
- Wie?
  - Dazu gab es schon eine Übung in der C++ Vorlesung (5. Übungsblatt, Aufgabe 1 & 2)
  - Da haben wir, wenn ein neues Element kam, neuen Platz alloziert und umkopiert
  - Wir haben dabei immer doppelt soviel Platz alloziert wie vorher, und dann erst wieder wenn der auch voll war
  - Warum gerade doppelt soviel?
  - Was tun wenn das Feld wieder kleiner wird?

Dynamische Felder — Ziel

- Wir wollen ...
  - ... dass jede Operation in (durchschnittlich) konstanter Zeit geht, also in Zeit unabhängig von der Größe des Feldes
  - ... dass immer ungefähr soviel Platz alloziert ist, wie wir auch Elemente im Feld haben
  - Heute in der Vorlesung: höchstens **dreimal** so viel
  - **Übungsblatt 1:** beliebiger Faktor, z.B. 2 oder 1.5 oder 1.1

Dynamische Felder — Programm

- Erstmal informal
  - bei pushBack Reallokation falls die Größe des Feldes nicht mehr ausreicht
    - dann Vergrößerung auf 3/2 mal die Anzahl Elemente
  - bei popBack Reallokation falls das Feld mehr als **dreimal** groß ist wie die Anzahl der Elemente
    - dann Verkleinerung auf 3/2 mal die Anzahl Elemente
  - damit ist das Feld zu keinem Zeitpunkt mehr als **dreimal** so groß wie die Anzahl der Elemente
- Schreiben wir gerade richtigen Code dazu
  - steht dann nach der Vorlesung wie gehabt im SVN

Dynamische Felder — Analyse 1/3

- Notation
  - Gegeben n Operationen $O_1, \ldots, O_n$
  - Sei s_i die Größe des Feldes (#Elemente) nach Operation O_i
  - Sei c_i die Kapazität des Feldes nach Operation O_i
  - Sei $cost(O_i)$ die Zeit für Operation O_i
    - wir nehmen an, die ist $\leq A$ falls keine Reallokation
    - und sonst $\leq A + B \cdot s_i$
    - für irgendwelche Zahlen A und B unabhängig von n
- Wir wollen beweisen
  - Lemma: Wenn bei O_i eine Reallokation stattfindet und dann erst wieder bei O_j, dann ist $j - i > s_i/2$
  - Satz: $cost(O_1) + \ldots + cost(O_n) \leq (A + 3B) \cdot n$

Dynamische Felder — Analyse 2/3

- Beweis Lemma

  [Wenn bei O_i eine Reallokation stattfindet und dann erst wieder bei O_j, dann ist $j - i > s_i/2$]

  floor(x) = ⌊x⌋

  – Nach O_i ist die Kapazität genau $\text{floor}(3/2 \cdot s_i)$

  „abgerundet x"

  – Betrachte eine Operation O_j nach O_i mit $j - i \leq s_i/2$

  ceil(x) = ⌈x⌉

  – Es können bis dahin höchstens $j - i$ Elemente dazu gekommen sein, also $s_j \leq s_i + j - i \leq s_i + s_i/2$

  „aufgerundet x"

  – Also $s_j \leq \text{floor}(3/2 \cdot s_i)$ und damit keine Vergrößerung

  – Es können bis dahin höchstens $j - i$ Elemente weg gekommen sein, also $s_j \geq s_i - (j - i) \geq s_i - s_i/2$

  – Also $\text{floor}(3/2 \cdot s_i) \leq 3 \cdot s_j$ und damit keine Verkleinerung

  – QED

Dynamische Felder — Analyse 3/3

- Beweis Satz

  [$cost(O_1) + \ldots + cost(O_n) \leq (A + 3B) \cdot n$]

  – Seien die Reallokationen bei $O_{i1}, \ldots, O_{il}$

  – Die Kosten dafür sind B mal $s_{i1} + \ldots + s_{il}$

  – Nach Lemma ist $i2 > i1 + s_{i1}/2$, $i3 > i2 + s_{i2}/2$ usw.

  – Also $s_{i1} < 2 \cdot (i2 - i1)$, $s_{i2} < 2 \cdot (i3 - i2)$ usw.

  – Also $s_{i1} + \ldots + s_{il} < 2 \cdot (i2 - i1) + 2 \cdot (i3 - i2) + \ldots + s_{il}$

  – Also $s_{i1} + \ldots + s_{il} < 2 \cdot il + s_{il} \leq 3 \cdot n$

  – Die Kosten für die anderen Operationen sind $\leq A \cdot n$

  – Also insgesamt $\leq 3B \cdot n + A \cdot n = (A + 3B) \cdot n$

  – QED

Literatur / Links

- Allgemein zur Vorlesung
  - Cormen / Leiserson / Rivest: Introduction to Algorithms

    [Klassisches Lehrbuch zu Algorithmen und Datenstrukturen. Nur das Inhaltsverzeichnis ist online, muss man kaufen oder ausleihen.]

    http://mitpress.mit.edu/algorithms/
  - Mehlhorn / Sanders: Algorithms and Data Structures, The Basic Toolbox

    [Neueres Lehrbuch zu Algorithmen und Datenstrukturen, mit etwas praktischere Ausrichtung als der Cormen/Leiserson/Rivest. Das ganze Buch steht online!]

    http://www.mpi-inf.mpg.de/~mehlhorn/Toolbox.html
- Dynamische Felder
  - In Mehlhorn/Sanders: 3.2 Unbounded Arrays
  - In Cormen/Leiserson/Rivest: 18.4 Dynamic Tables