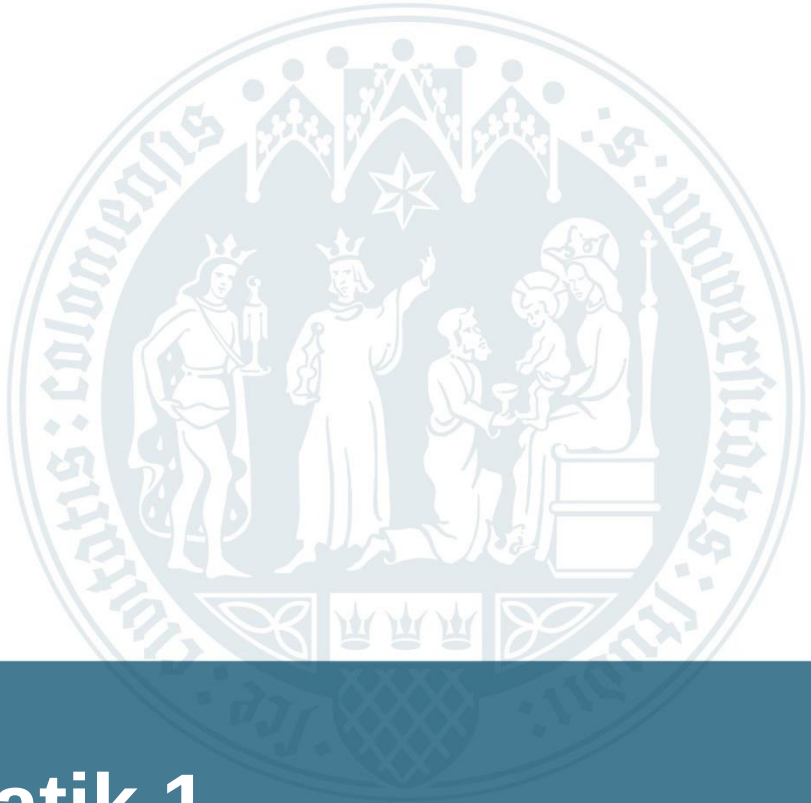
Universität zu Köln



Grundzüge der Informatik 1

Vorlesung 15 - flipped classroom

Datenstrukturen

Was ist eine Datenstruktur?

- Eine Datenstruktur ist eine Anordnung von Daten im Speicher eines Rechners, die effizienten Zugriff auf die Daten ermöglicht
- Datenstrukturen für viele unterschiedliche Anfragen vorstellbar

Datenstrukturen

Ein grundlegendes Datenverwaltungsproblem

- Speicherung von Datensätzen

Beispiel

- Kundendaten (Name, Adresse, Wohnort, Kundennummer, offene Rechnungen, offene Bestellungen,…)

Anforderungen

- Schneller Zugriff
- Einfügen neuer Datensätze
- Löschen bestehender Datensätze

Datenstrukturen

Zugriff auf Daten

- Jedes Objekt x hat einen Schlüssel key[x]
- Eingabe des Schlüssels liefert Datensatz
- Schlüssel sind vergleichbar (es gibt totale Ordnung der Schlüssel)

Beispiel

- Kundendaten (Name, Adresse, Kundennummer)
- Schlüssel: Name
- Totale Ordnung: Lexikographische Ordnung

Datenstrukturen

Grundlegendes Datenverwaltungsproblem

- Organisiere die Daten im Speicher eines Rechners so, dass folgende Operationen effizient durchgeführt werden können (S die aktuelle Menge der Objekte):
- Suchen(S,k):
  - Es wird ein Zeiger x auf ein Objekt mit Schlüssel k=key[x] zurückgegeben oder NIL, wenn es kein Objekt mit Schlüssel k in S gibt
- Einfügen(S,x):
  - Objekt x wird in S eingefügt
- Löschen(S,x):
  - Objekt x wird aus S entfernt

Datenstrukturen

Vereinfachung

- Schlüssel sind natürliche Zahlen
- Schlüssel sind eindeutig
- Eingabe nur aus Schlüsseln

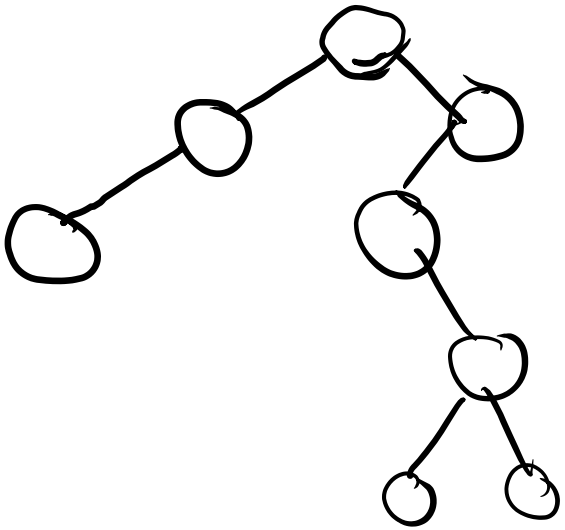
Analyse von Datenstrukturen

- Platzbedarf in O-Notation
- Laufzeit der Operationen in O-Notation

Gierige Algorithmen

Aufgabe 1

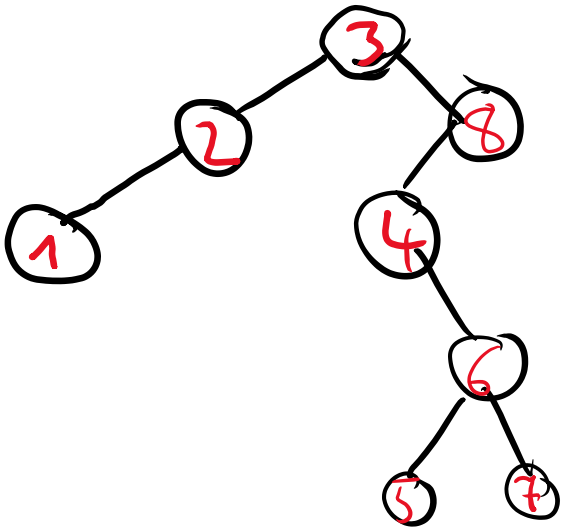
- Ordnen Sie die Schlüssel 1,2,3,4,5,6,7,8 den Knoten des unten stehenden Suchbaums zu



Gierige Algorithmen

Aufgabe 1

- Ordnen Sie die Schlüssel 1,2,3,4,5,6,7,8 den Knoten des unten stehenden Suchbaums zu



Datenstrukturen

Aufgabe 2

- Entwickeln Sie eine Datenstruktur, die die folgenden Operationen in O(1) Zeit unterstützt.
- Einfügen(x): Fügt $x \in \{1,\ldots D\}$ in die Datenstruktur ein. Mehrfache Vorkommen von x sind möglich.
- Löschen(x): Löscht $x \in \{1,\ldots,D\}$ aus der Datenstruktur (sofern vorhanden)
- Anzahl(x): Gibt die Anzahl Kopien von x an, die in der Datenstruktur abgespeichert sind
- Ihre Datenstruktur soll dabei O(D) Speicher benutzen. Beschreiben Sie Ihre Datenstruktur und geben Sie Pseudocode für die Operationen Einfügen, Löschen und Anzahl an.

Datenstrukturen

Beschreibung der Datenstruktur

- Die Datenstruktur besteht aus eine Feld A[1…D]. Der Eintrag A[i] bezeichnet die Anzahl Kopien von i, die aktuell in der Struktur abgespeichert sind. Zur Initialisierung wird A[i] für alle Einträge auf 0 gesetzt.

Einfügen(x)

1. A[x] = A[x]+1

Löschen(x)

1. A[x] = A[x]-1

Anzahl(x)

1. **return** A[x]

Gierige Algorithmen

Aufgabe 3

- Die Datenstruktur „einfaches Feld“ hat den Nachteil, dass eine maximale Größe zu Beginn festgelegt werden muss. Entwickeln Sie eine Datenstruktur „dynamisches Feld“ mit folgenden Eigenschaften:
  - Eine Sequenz von r Einfüge- und Löschoperationen benötigt Laufzeit O(r)
  - Zu jedem Zeitpunkt benötigt das Feld O(n) Speicher, wobei n die Anzahl der Elemente ist, die aktuell im Feld gespeichert sind
- Analysieren Sie die Laufzeiten der Operationen Einfügen und Löschen

Gierige Algorithmen

Idee

- Wenn das Feld voll ist, verdoppele die Größe des Feldes und kopiere alle Daten
- Wenn das Feld zu weniger als ¼ voll ist, halbiere die Feldgröße und kopiere alle Daten

Datenstrukturen

Dynamisches Feld

- Feld A[1,…,m]
- $n \geq 1$ bezeichnet aktuelle Anzahl Elemente in Datenstruktur
- Invariante: $m/4 \leq n \leq 2m$
- Initialisierung: m=4

Datenstrukturen

Einfügen(x**)**

```
1.   n=n+1
2.   A[n] = x
3.   if m=n then
4.      m=2m
5.      B = new array[1..m]
6.      for i=1 to n do
7.         B[i] = A[i]
8.      delete A
9.      A=B
```

Datenstrukturen

Löschen(i**)** ** i ist Index des zu löschenden Objekts im Feld*

1. A[i] = A[n]
2. A[n] = **nil**
3. n = n-1
4. **if** n=m/4 **and** m>4 **then**
5. m= m/2
6. B= **new array**[1..m]
7. **for** i=1 **to** n **do**
8. B[i] = A[i]
9. **delete** A
10. A=B

Datenstrukturen

Laufzeit

- Einfügen läuft in O(1) Zeit, außer es muss ein neues Feld angelegt werden
- Direkt nachdem ein neues Feld angelegt wurde, gilt n=m/2
- Bevor eine neue Einfügeoperation zu einer Verdoppelung der Feldgröße führt, müssen n weitere Elemente eingefügt werden
- Bevor eine neue Löschoperation zu einer Halbierung der Feldgröße führt, müssen n/2 weitere Elemente gelöscht werden
- Die Kosten dieser n bzw. n/2 Operationen sind $\Theta(n)$
- Die Kosten für das Anlegen des Feldes kann auf diese Operationen umgelegt werden (man kann sich vorstellen, dass bei jeder Operation O(1) Zeiteinheiten auf ein Zeitkonto eingezahlt werden und diese genutzt werden, um die Verdoppelung bzw. Halbierung zu bezahlen)
- Damit benötigt jede Operation im Durchschnitt O(1) Zeit (man bezeichnet dies auch als amortisierte Laufzeit)