Informatik II

Übungsstunde 8

simon.mayer@inf.ethz.ch
Distributed Systems Group, ETH Zürich

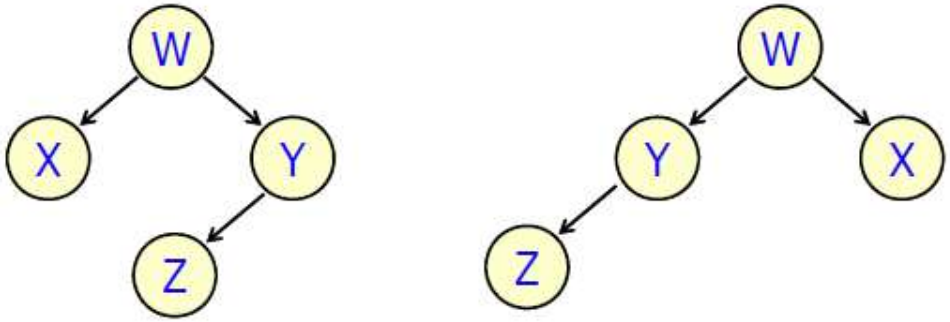
Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Backtracking, Entscheidungsbaeume
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

Mi a bináris keresőfa?

Zur Erinnerung: Binärbaum

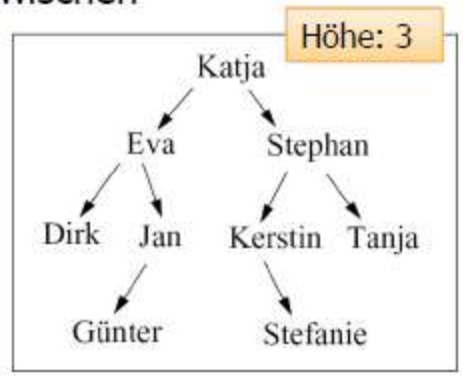
- Jeder Knoten hat *höchstens* zwei Nachfolger
- Unterscheide linken und rechten Nachfolger / Unterbaum



Höhe eines Binärbaums

- Die **Tiefe eines Knotens** ist sein Abstand (d.h. die Länge seines Weges) zur Wurzel
  - D.h. die Wurzel hat die Tiefe 0

- Die **Höhe eines Baums** ist die maximale Tiefe
  - D.h. die Länge eines längsten Weges von der Wurzel zu einem Blatt (Es kann mehrere solche – gleich langen – Wege geben!)
  - Mit anderen Worten: Der maximale Abstand zwischen einem Knoten und der Wurzel
  - Ein Baum, der nur aus einer Wurzel besteht, hat also die Höhe 0.



Zeitbedarf: Anzahl der Schritte als Funktion der Zahl der Elemente n

- Methode „insert" wird n Mal nicht-rekursiv aufgerufen
  - Aber: Wieviele Schritte benötigt ein insert-Aufruf?
    - → jedesmal wird ein Ast ganz durchlaufen
  - Aber wie lange ist ein Ast (im Mittel, maximal...)?
    - Im Extremfall (Eingabe sortiert!): n (bzw. 0.5n „gemittelt")
    - Im „Normalfall", wenn gutartige Bäume entstehen: etwa log(n)

→ O(n log n)

- Methode „inorder" wird 1 Mal von aussen und 2n Mal rekursiv aufgerufen – also linear zu n → O(n)

- Eine genaue mathematische Analyse liefert dann: Im häufigen Normalfall werden insgesamt nur ca. c n log(n) Schritte zum Sortieren benötigt

kleine Konstante

Mi a bináris keresés?

Binärsuche auf Arrays

- Problem: Feststellen, ob ein Element in einem sortierten Array vorkommt
- Lösungsprinzip:
  - Prüfen, in welcher Hälfte der gesuchte Wert liegen muss (wenn man ihn nicht zufällig genau in der Mitte getroffen hat!)
  - Verfahren dann iterativ (oder rekursiv) auf die „richtige" Hälfte anwenden → entweder linke oder rechte Grenze neu setzen
- Stopkriterien:
  - Wert gefunden → liefere Index des Arrays
  - Restbereich „kollabiert" → liefere '-1' als Fehl-Indikator

Binärsuche auf Arrays (2)

```
// A sei hier ein globales int array
int binSearch (int s) {
  int m;
  int li = 0; int re = A.length-1;

  while (re >= li) {
    m = (li+re)/2;
    if (s == A[m]) return m;
    if (s < A[m])
      re = m - 1;
    else
      li = m + 1;
  }

  return (-1);
}
```

Voraussetzung: A sei aufsteigend sortiert

Was passiert wenn
- li + re ungerade ist?
- und wenn li = re?

Fragen:
1) Dürfen gleiche Elemente mehrfach vorkommen?

2) Was geschieht, wenn A doch nicht sortiert ist?
(Wie könnte dies in linearer Zeit überprüft werden?)

Terminiert das?

Wie schnell?

- Und wovon haengt das ab?

Bactracking

- Ziel: Systematisches Durchmustern eines (sehr) grossen Zustandsraumes, um zu einem Problem eine Lösung zu finden

Passende, gute, optimale, alle Lösungen...

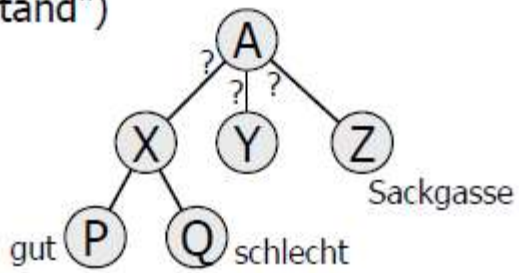
- Systematisches „trial and error"
- Beispiel: Ausgang in einem Labyrinth suchen:
  - Sich für eine Richtung entscheiden
  - In diese Richtung weitergehen
  - Wenn „letztendlich" erfolglos: zurückkehren und eine andere Richtung wählen

Backtracking

Falls bereits alle Richtungen ausprobiert → noch weiter zurück

Durchmustern des Entscheidungsbaums

- Idee: Den Baum (→ keine Zyklen!) aller möglichen Entscheidungen systematisch (rekursiv) durchlaufen:
  - Knoten = Entscheidungssituation („Zustand")
  - Kante = Entscheidung
  - Nachfolgeknoten = Situation nach der Entscheidung
  - Blatt = Endsituation



- Nützliche Strategie: Sackgassen möglichst früh entdecken
  - Ist abhängig vom konkreten Problem
  - Spart evtl. viel Suchaufwand (ganze Unterbäume!)
  - Beispiel: Analyse eines Spielbaums: „Dieser Zug ist hoffnungslos"

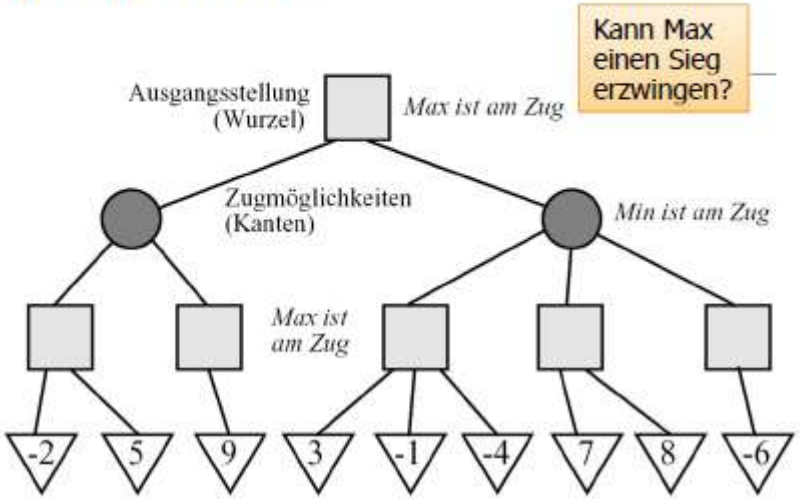
„Game Playing", Computerschach

- Schach ist prominentester Vertreter des „automatischen strategischen Spielens"
  - Faszination: Verbindung von uraltem Spiel und moderner Technik
  - Herausforderung: Eine „intelligente" Maschine bauen
  - Reiz: kleine, abgeschlossene Welt (klare Spielregeln)
- Hoffnung (trügerisch!): Wenn man ein Spiel wie Schach beherrscht, dann auch andere Probleme (aus Wirtschaft, Politik...), zu deren Lösung „Intelligenz" nötig ist

---

- Es sind (nur noch) einige wenige menschliche Spieler (manchmal) besser als die besten Schachprogramme
  - Es gibt aber andere strategische Spiele, wo Maschinen den Menschen noch nicht ganz so überlegen sind

Spielbäume

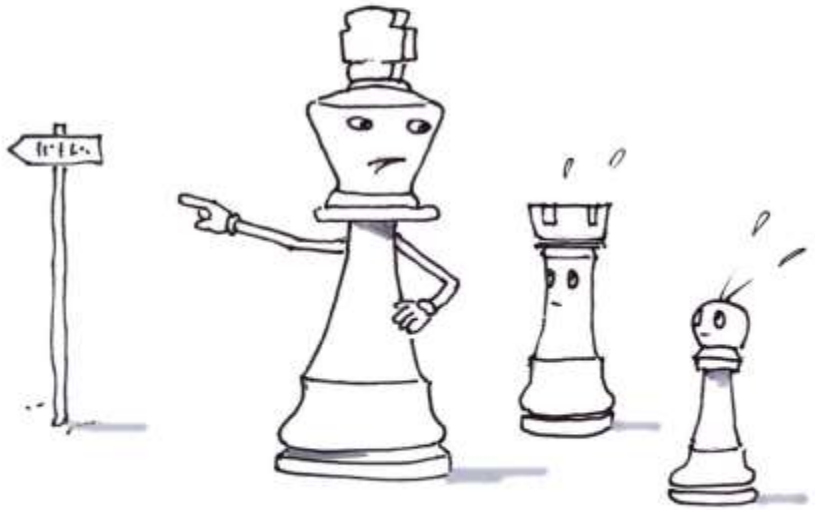


Spieler „**Max**" und „**Min**" ziehen abwechselnd
- Konvention: Max beginnt
- Niveauweise abwechselnd Max-Knoten / Min-Knoten

- Blätter beschreiben eine Endsituation
  - Sind mit dem Wert der Auszahlungsfunktion für Max markiert
- Max wird bestrebt sein, ein Blatt mit hohem Wert zu erreichen
  - Entsprechend wird Min einen möglichst kleinen (negativen) Wert anstreben

Strategie

- Vollständiger Verhaltensplan: Handlungsvorschrift (d.h. Zug) für jede Situation, in die der Spieler kommen könnte
  - In der Praxis aber meist zu viel, um alles im Voraus zu berechnen
  - Daher oft nur heuristische Regeln (z.B.: wenn der Gegner zwei Steine in einer Reihe hat, dann...)



Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Backtracking, Entscheidungsbaeume
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

Übung 8

1. Binaersuche
   - Entscheidungsbaum zeichnen & Ueberlegungen anstellen
     - Uebereinanderlegen, Faktoren
   - Implementierung:
     - `find(List<Unit<Key, Value>> haystack, Key needle)`
     - `setFactor(int factor)`
       - Verallgemeinern der Suche -> neu auch unbalancierte Suchbaeume
     - `getNumberOfCalls()`
       - Benchmarking mit verschiedenen Faktoren
       - Durschnittliche # rekursiver Aufrufe fuer verschiedene Faktoren

Übung 8

2. Tic-Tac-Toe & Spielbaeume

- Ueberlegungen zu Spielbaeumen…
- Ueberlegen Sie sich, wie das Attribut eines Knotens auf Grund der Attribute der Nachfolger berechnet wird, wenn **Sie bzw. Ihr Gegner an der Reihe sind**.

Übung 8

3. Reversi 2

a. Implementieren von `ICheckMove` ohne Framework-Funktion.
   Ideen?

b. Implementieren eines Spielers, der unter allen moeglichen Zuegen den besten auswaehlt
   Bester Zug: *Zug, nach dessen Durchfuehrung man maximal mehr Steine besitzt als der Gegner: «Denktiefe = 1»*
   d.h.: Kein Spielbaum noetig!

   Ermitteln des besten Zugs: Board kopieren (clone), Zug ausfueheren, zaehlen...

Ablauf

- Besprechung der Vorlesung
- Uebungsbezogene Themen:
  Backtracking, Entscheidungsbaeume
- Zeit zum Programmieren...
  und fuer noch mehr Fragen

Projekt: Nicht so ganz klar…

Informatik II

Übungsstunde 8

simon.mayer@inf.ethz.ch
Distributed Systems Group, ETH Zürich