Proseminar
Algorithms and Data Structures
Einführungsveranstaltung

Joost-Pieter Katoen

Lehrstuhl für Informatik 2
(Software Modeling and Verification)

RWTH Aachen University

katoen@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/algds11/

21. Oktober 2011

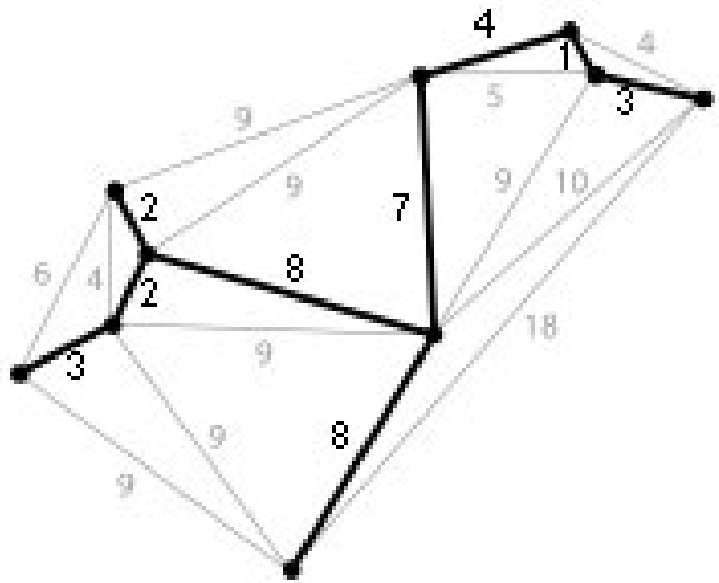
Outline

Seminarthema

Thema des Proseminars

Weiterführung und Vertiefung diverser Themen der Vorlesung *Datenstrukturen und Algorithmen*

- Algorithmen:
  - neue Algorithmen
  - andere Komplexitätsmaße, ...
- Datenstrukturen:
  - Bäume
  - Hashing, ...
- Inhalt:
  - Problemstellung
  - Arbeitsweise des Algorithmus bzw. der Datenstruktur
  - Effizienzeigenschaften
  - Anwendungen



Zielsetzung

Ziele des Proseminars

- Selbständiges Einarbeiten in ein neues Thema
- Literaturrecherche
- Darstellen des Inhalts in einer wissenschaftlichen Ausarbeitung
- Verständliches Präsentieren

Anforderungen zur Ausarbeitung

Ausarbeitung

- Selbständiges Verfassen einer ca. 15-seitigen Ausarbeitung
- Vollständiges Literaturverzeichnis
- Korrektes Zitieren
- Plagiarismus:
  Die nicht gekennzeichnete Übernahme fremder Inhalte führt zum sofortigen Ausschluss.
- Schriftgröße 11pt, übliche Seitenränder
- Titelseite mit Thema, Proseminar, Name, Matrikelnummer, Datum
- Sprache Deutsch oder Englisch
- Korrekte Sprache wird vorausgesetzt:
  $\geq$ 10 Fehler pro Seite $\Longrightarrow$ Abbruch der Korrektur

Anforderungen Vortrag

Vortrag

- 30-minütiger Vortrag
- Zielgruppengerechte Präsentation der Inhalte
- Übersichtliche Folien:
  - ≤ 15 Textzeilen
  - sinnvoller Einsatz von Farben
- Vortrag in Deutsch oder Englisch

Outline

Bibliothekseinführung

Einführung in die Literaturrecherche

- Einweisung in themenspezifische Literaturrecherche
- Dauer: ca. zwei Stunden
- Teilnahme für BSc-Studierende verpflichtend
- Termine: werden noch bekannt gegegen

Deadlines

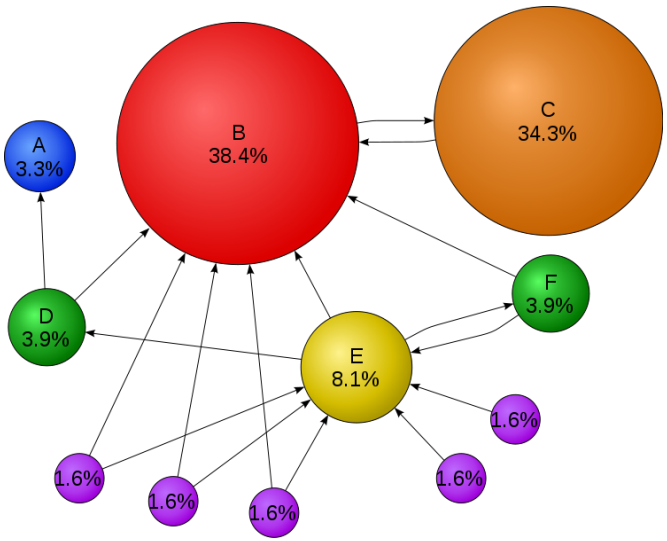
Deadlines

Folgende Termine sind einzuhalten:

- 11.11.2011: letzte Rücktrittsmöglichkeit
- 18.11.2011: Gliederung + erste Fassung 2 Kapitel
- 01.12.2011: erste Fassung der Ausarbeitung
- 19.12.2011: endgültige Fassung der Ausarbeitung
- 24.01.2012: endgültige Fassung der Folien
- 09./10.02.2011: Blockseminar

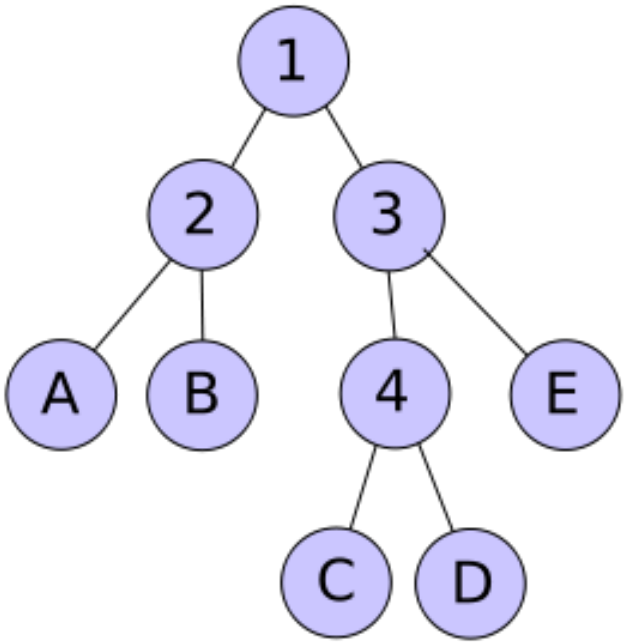
Outline

1. Pagerank-Algorithmus



- Algorithmus zur Analyse verlinkter Webseiten
- Allgemeiner: beliebige Graphen
- Basis: Bestimmung der Wahrscheinlichkeit, dass zufälliges Benutzerverhalten auf die Seite führt
- Benutzung in Google-Suchmaschine

2. Huffman-Kodierung

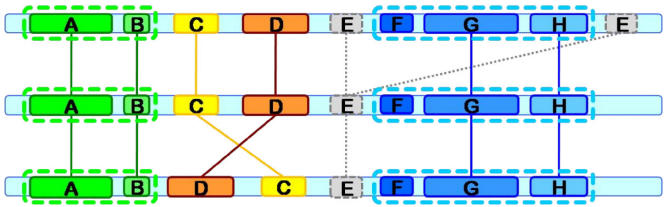


- Ziel: Erstellung einer Binärkodierung mit minimaler mittlerer Wortlänge benötigt
- Basis: Wahrscheinlichkeitsverteilung der einzelnen Zeichen
- Ansatz: Verwendung eines vollen Binärbaums zur Darstellung des Codes

3. Cocktail Sort (aka Shaker Sort)

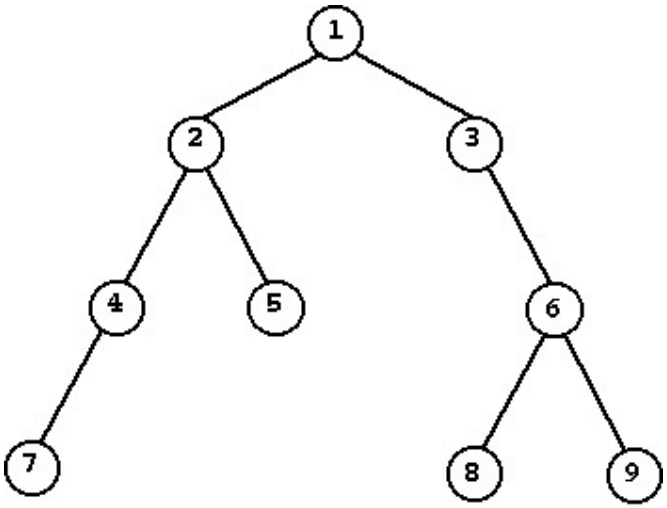
- Abwechselndes Durchlaufen des zu sortierenden Feldes von oben nach unten
- Vergleich und ggf. Tausch benachbarter Elemente
- Bidirektionalität $\Rightarrow$ schnelleres Absetzen von großer und kleiner Elemente
- Quadratische Worst-Case-Laufzeit

4. Longest Common Subsequence



- Definition: längste gemeinsame (nicht notwendigerweise zusammenhängende) Teilsequenz mehrerer Zeichenketten
- Anwendungen: `diff`, Bioinformatik
- NP-hart für beliebige Anzahl von Zeichenketten
- Quadratisch für zwei Zeichenketten (dynamische Programmierung)

5. Deutsch-Schorr-Waite Baumtraversierung



NLR: 1 2 4 7 5 3 6 8 9

LNR: 7 4 2 5 1 3 8 6 9

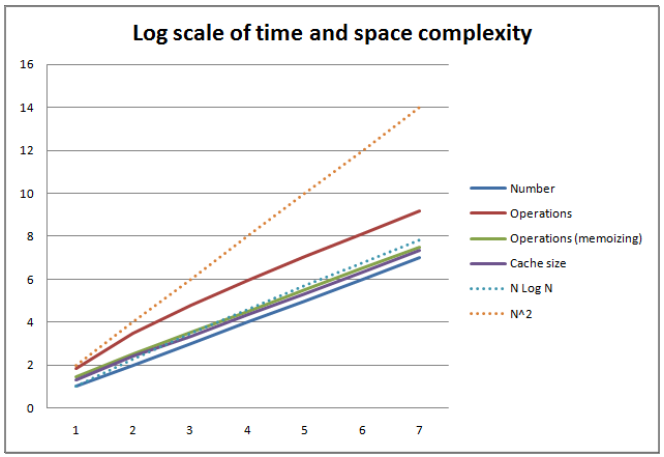
LRN: 7 4 5 2 8 9 6 3 1

- Klassische Lösung: Rekursion/Stack
- Ziel: Vermeidung des zusätzlichen Speicheraufwands
- Ansatz: Pointerrotation in der Heapdarstellung

6. Problem der K kürzesten Pfade

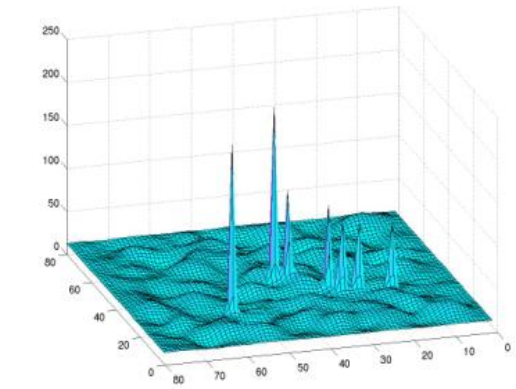
- Problem: Finden der K kürzesten Pfade zwischen Knotenpaar in einem gerichteten Graphen
- Ansatz: Verallgemeinerung der Bellman-Gleichung

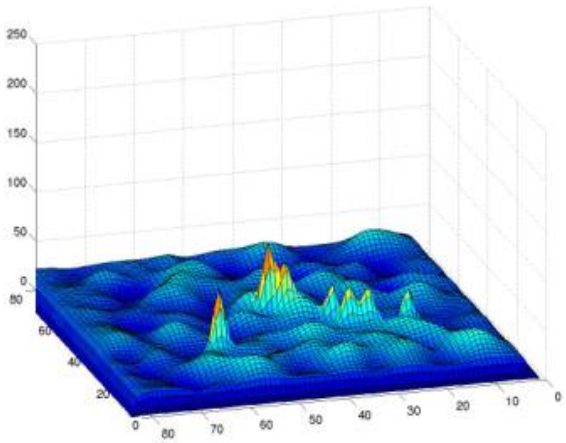
7. Amortisierte Laufzeitanalyse



- Allgemeinen Laufzeitanalyse: maximale Kosten der einzelnen Schritte
- Amortisierte Laufzeitanalyse: Worst Case aller Operationen im gesamten Durchlauf des Algorithmus
- Verbesserung der oberen Schranke bei seltenem Auftreten teurer Operationen
- Idee: der Worst Case ändert den Zustand der Datenstruktur so ab, dass er nicht wiederholt auftreten kann (z.B. dynamische Arrays)
- Drei unterschiedliche Berechnungsmethoden (Aggregat-, Account-, Potentialfunktion-Methode)

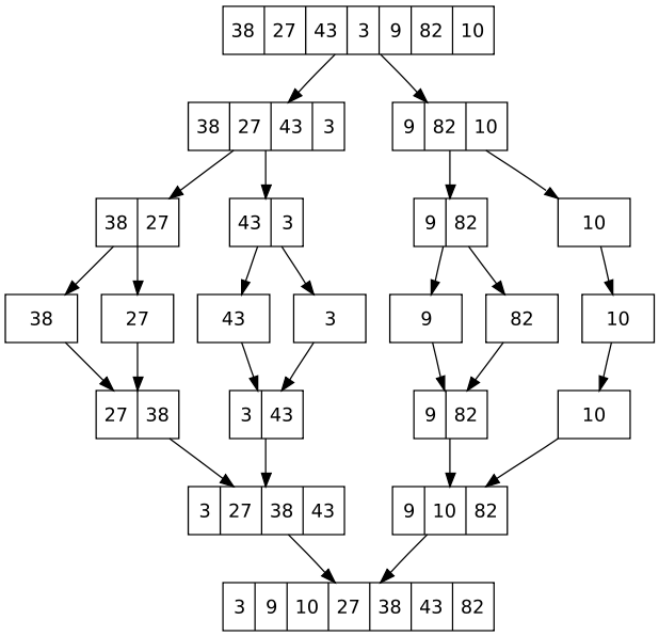
8. Smoothed Analysis





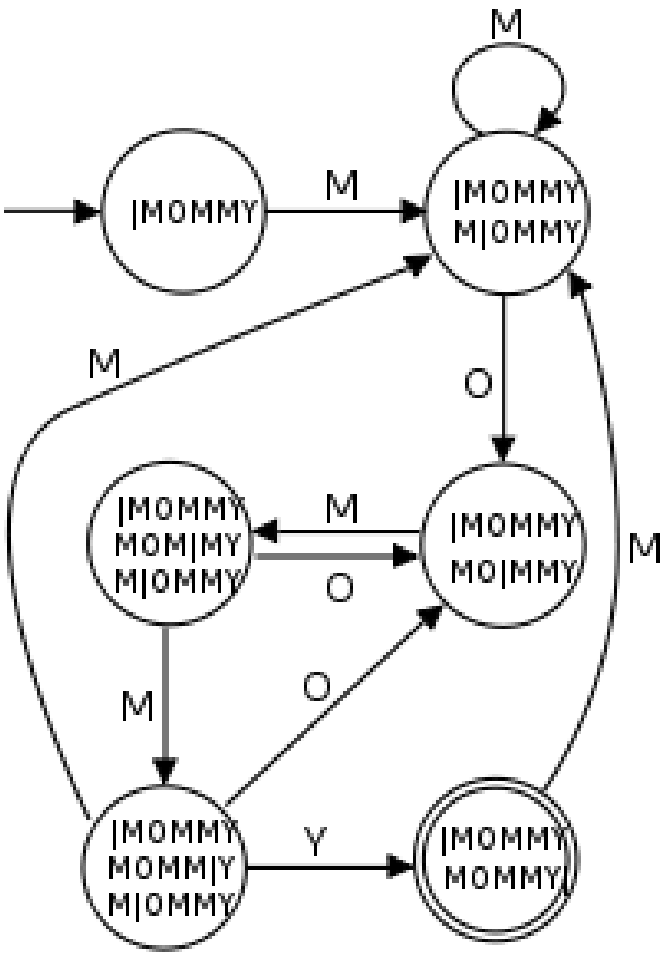
- Ziel: realistische Abschätzung des Laufzeitverhaltens von Algorithmen
- Kombination von Worst-Case- und Average-Case Analyse
- Liefert „geglättete“ Effizienzergebnisse

9. Externes Sortieren



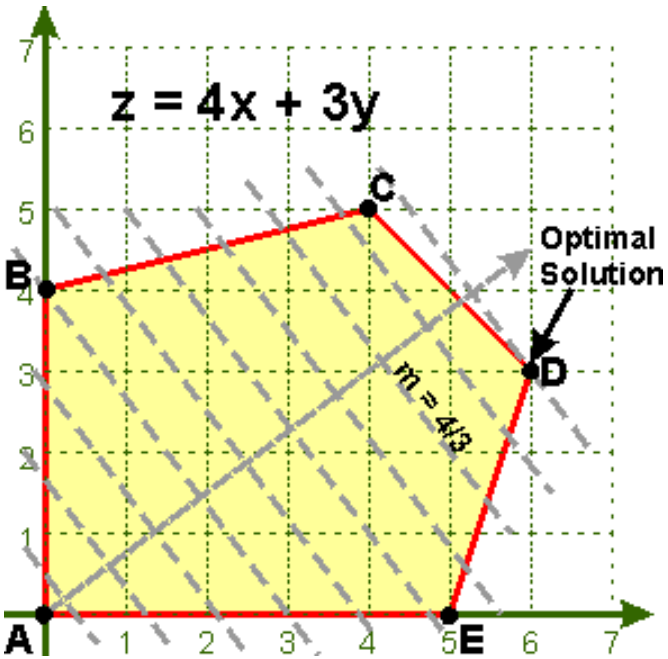
- Ziel: Sortierung großer Datenmengen unter Verwendung externer Speicher
- Typischer Ansatz: divide-and-conquer (sort-merge-Strategien)

10. String Matching



- Ziel: Suche von Vorkommen eines Strings in einem (großen) Text
- Komplexität des naiven Algorithmus: $|\text{Text}| \cdot |\text{String}|$
- Verbesserung durch Vorverarbeitung des Suchstrings

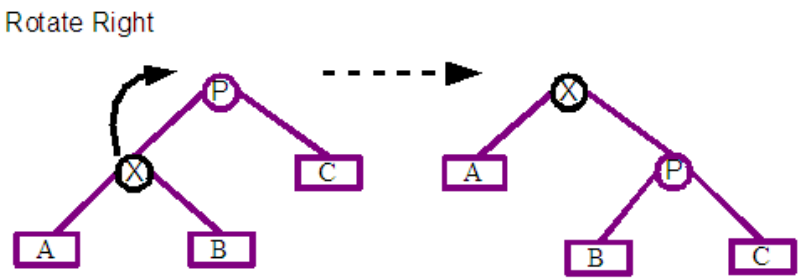
11. Lineare Optimierung (aka Lin. Progr.)



- Ziel: Optimierung linearer Zielfunktionen über einer Menge, die durch lineare (Un-)Gleichungen eingeschränkt ist
- Simplexverfahren
- Ellipsoid-Methode

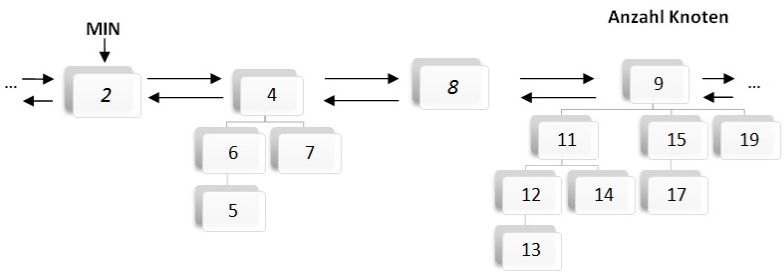
Outline

12. Splay Trees



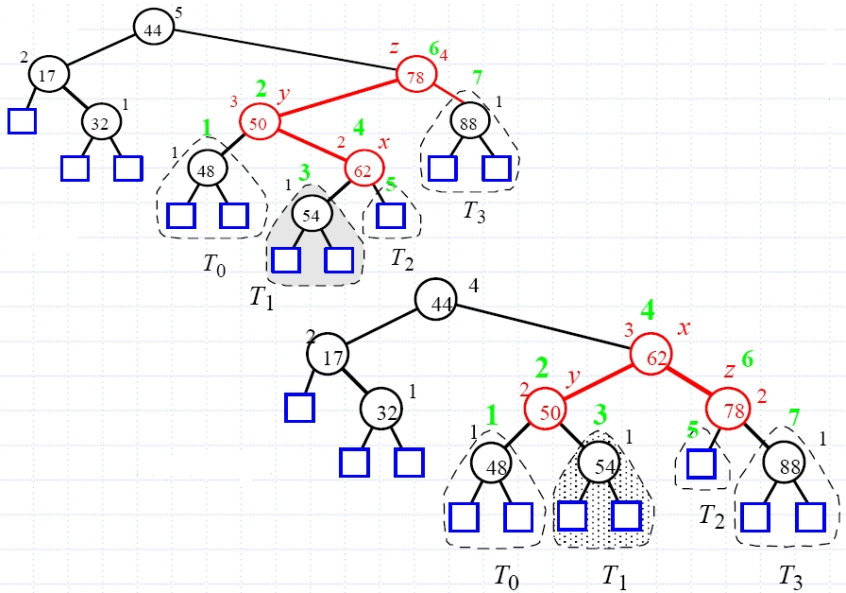
- Baumorganisierte Datenstruktur
- Effizient Einfügen, Suchen und Löschen von Elementen
- Grundlegende Operation: *splay* (rotieren eines Elements bis zur Wurzel)

13. Fibonacci Heaps



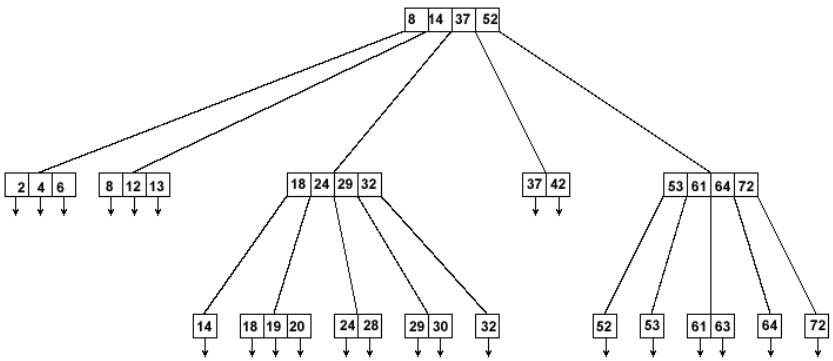
- Datenstruktur zur Implementierung einer Vorrangwarteschlange (*priority queue*)
- Liste von geordneten Bäumen
- Heap-Bedingung: Priorität jedes Knotens mindestens so groß wie Priorität seiner Kinder
- (Fast) alle Operationen haben amortisiert konstant Laufzeit

14. AVL-Bäume



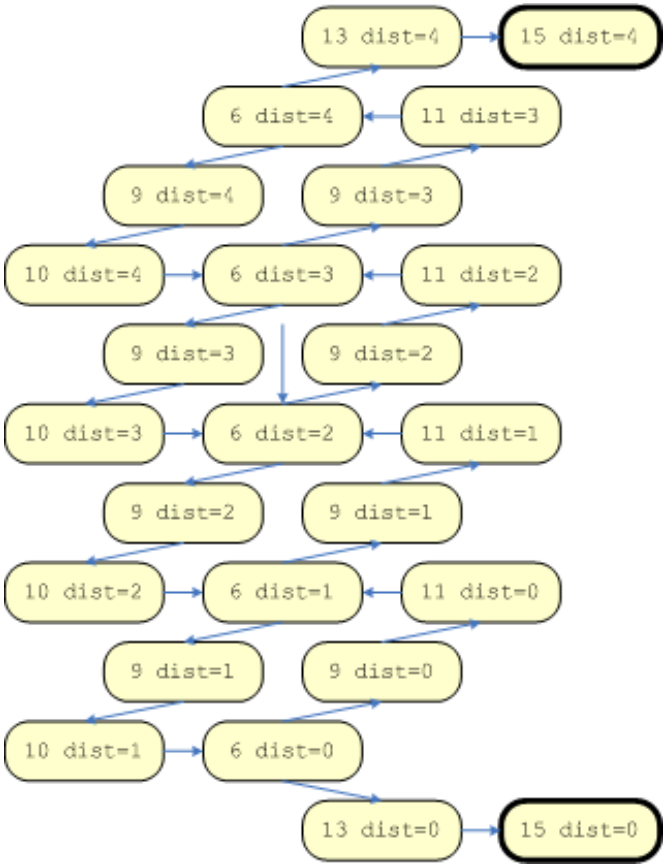
- Balancierter binärer Suchbaum
- Invariante: maximaler Höhenunterschied der Teilbäume jedes Knotens ist 1
- Worst-Case-Komplexität der üblichen Operationen: $O(\log n)$

15. B-Bäume



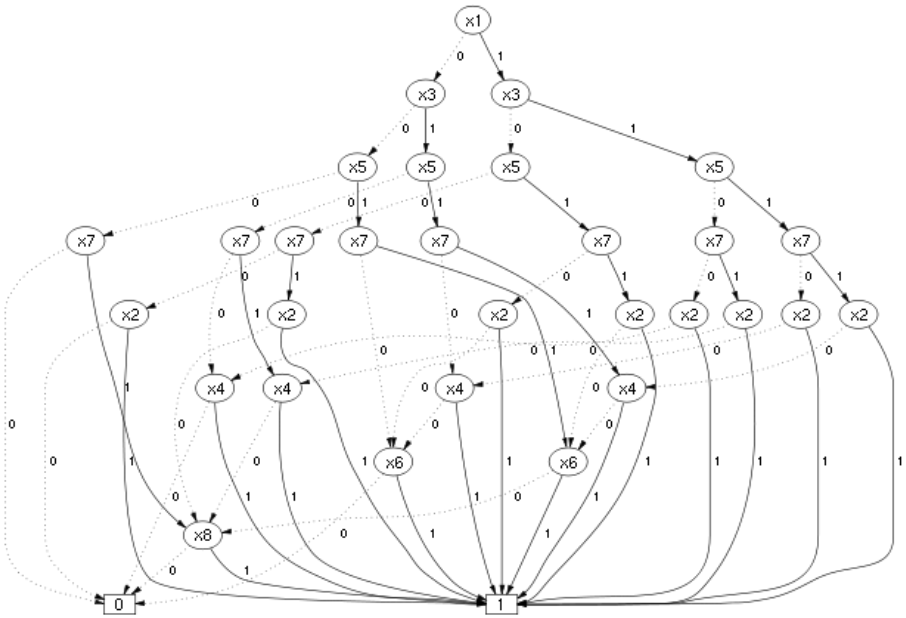
- Immer vollständig balanciert
- Besonderheit: Anzahl der Knotennachfolger variabel (mit Obergrenze)
- Einsatz vor allem in Datenbanken und Dateisystemen

16. Bitstate Hashing



- Effiziente Darstellung von Zustandsräumen
- Problem: Erkennung von Zykeln
- Idee: Benutzung einer Hashfunktion, Speicherung von 0/1 im Hasharray
- Bis zu 98% Speicherersparnis im SPIN-Tool

17. Binäre Entscheidungsdiagramme

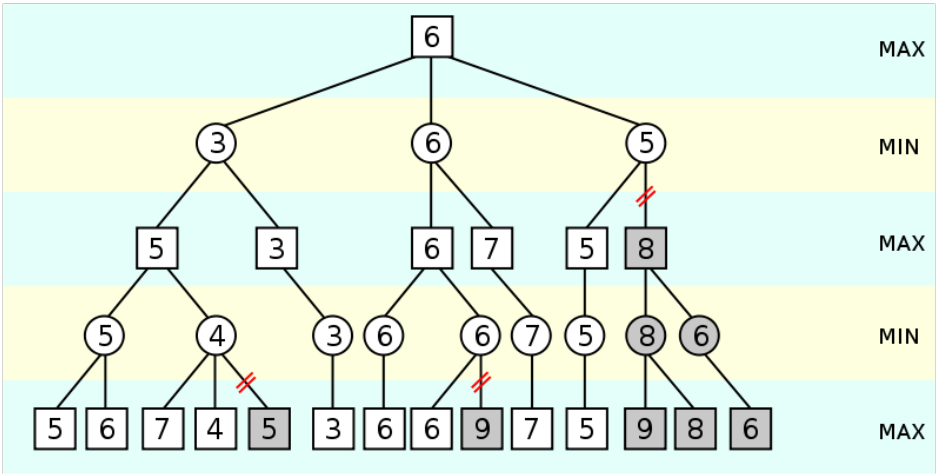


- Datenstruktur zur Darstellung Boolescher Funktionen
- Einsatz: Verifikation von (Hardware-)Systemen
- Problem: Finden geeigneter Variablenordnungen

18. RAID: Redundant Array of Independent [Inexpensive] Disks

- Organisation mehrerer physischer Festplatten als ein logisches Laufwerk (mit Redundanz)
- Vorteile: höhere Datenverfügbarkeit bei Ausfall einzelner Festplatten, größerer Datendurchsatz
- Verschiedene Level (0/1/5)

19. Spielbäume



- Darstellung von 2-Personen-Spielen mit abwechselnden Zügen (z.B. Schach, Go, Reversi, Dame, oder Vier gewinnt)
- Ziel: Bestimmung optimaler Strategien
- Minimax-Verfahren
- α-β-Pruning