Randomisierte Algorithmen und probabilistische Analyse

S.Seidl, M.Nening, T.Niederleuthner

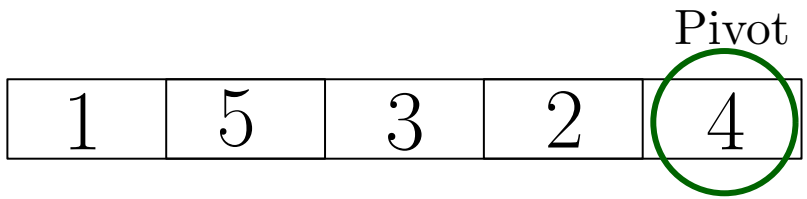
Inhalt

Inhalt

Quicksort - kurze Wiederholung

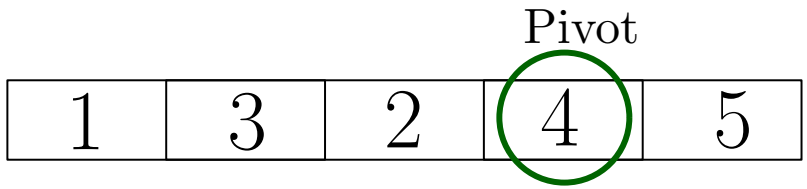
Ausgangssituation:

Wähle als Pivotelement p das letzte Element:

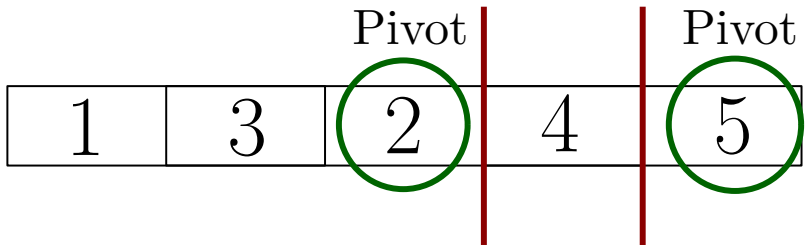


Quicksort - kurze Wiederholung

Tausche alle Elemente $< p$ nach links, alle $\geq p$ nach rechts:



Rufe Quicksort rekursiv für die Teilarrays links und rechts von p auf:



Eigenschaften

- Average-Case Laufzeit: $\Theta(n\ log(n))$
- Worst-Case Laufzeit: $\Theta(n^2)$, beispielsweise bei bereits sortiertem Input.

Problem: Die Laufzeit von Quicksort ist abhängig von der Form des Inputs. Wie kann man das vermeiden?

Lösung: Randomisierte Algorithmen!

Inhalt

Randomisierte Algorithmen

Was sind randomisierte Algorithmen?

- zufällige Auswahl des nächsten Schrittes
- bei gleicher Eingabe unterschiedliche Laufzeit
- Laufzeit und Korrektheit nur mit gewisser Wahrscheinlichkeit bestimmbar

Motivation für randomisierte Algorithmen

- Einfacher
- Effizienter
- leichter zu Implementieren
- Praktische Sicht

Algorithmustypen

- Las-Vegas-Algorithmen
  - immer korrekt
  - können auch versagen
- Monte-Carlo-Algorithmen
  - nicht immer korrekt
  - Fehlerarten:
    - Zweiseitige Fehler
    - Einseitige Fehler

Techniken

- Überfluss an Zeugen
  (abundance of witnesses)
- Vereitelung gegnerischer Angriffe
- Fingerabdruck und Hashing
- Zufallsstichproben
  (random sampling)

Inhalt

Eigenschaften

- Las-Vegas-Algorithmus
- wählt Pivotelement zufällig
- $\rightarrow$ macht gegnerische Angriffe unmöglich!
  Wiederum gilt:
- Average-Case Laufzeit: $\Theta(n\ log(n))$
- Worst-Case Laufzeit: $\Theta(n^2)$

Der große Unterschied

- Laufzeit hängt vom Zufall ab und nicht vom Input!
- Wahrscheinlichkeit für Worst-Case im Randomized Quicksort:

$$\frac{2^{n-1}}{n!}$$

Inhalt

Miller-Rabin-Test

- Monte-Carlo-Algorithmus
- Eingabe: Ungerade natürliche Zahl p
- Ausgabe: p ist Prim oder p ist Zusammengesetzt
- Wahrscheinlichkeit für ein falsches Ergebnis ist $\leq \frac{1}{4}$, in den allermeisten Fällen deutlich geringer.

Miller-Rabin-Test - Pseudocode

```
1  finde u ungerade und k ≥ 1, sodass p − 1 = u · 2^k;
2  wähle a zufällig aus {2, ..., p − 2};
3  setze b = a^u mod p;
4
5  wenn b == 1 oder b == p − 1
6     return "Prim";
7
8  repeat k − 1 mal
9     b = b^2 mod p
10    wenn b == p − 1
11       return "Prim";
12    wenn b == 1
13       return "Zusammengesetzt";
14
15 return "Zusammengesetzt";
```

Einsatzbereiche

- Sortieren & Suchen in Datenstrukturen
- Randomisierte Graphalgorithmen: Minimum-Cut, Shortest Path, minimale Spannbäume
- Pattern-Matching
- und viele mehr.

Inhalt

Limit: Theorie vs. Realität

- worst-case Komplexität schlecht
- Performance in Realität durchaus gut
  - z.B Rucksackproblem
  - worst-case Input äußerst selten
- Verhalten bei "typischem" Input interessant
  - "typischer Input" anwendungsabhängig

Lösung: Wahrscheinlichkeitstheorie um das Verhalten zu analysieren

Inhalt

Probabilistische Analyse

- Wahrscheinlichkeitsverteilung über Inputmenge
- Erwartungswerte werden gebildet

Voraussetzung: Kenntnisse/Annahmen über Verteilung

Grundbegriffe Wahrscheinlichkeitstheorie

- Wahrscheinlichkeitsraum S
  - Menge von Elementarereignissen
- Ereignis A
  - Teilmenge eines Wahrscheinlichkeitsraumes
- Zufallsvariable X
  - Funktion $X : S \rightarrow \mathbb{R}$
  - Indikatorvariable (nur 0 oder 1)
- Wahrscheinlichkeitsverteilung $Pr[\,]$
  - ordnet jedem Ereignis eine Wahrscheinlichkeit zu

Wahrscheinlichkeitsverteilung beschreiben

- Erwartungswert $\mathbb{E}[\,]$ für Zufallsvariable
  - Gewichtete Durchschnittswert von X
  - Allgemein:

$$\mathbb{E}[X] = \sum_{x \in S} x * Pr[X = x]$$

  - Indikator:

$$\mathbb{E}[X] = Pr[X = 1]$$

Wahrscheinlichkeitsverteilung beschreiben

- Tail Distribution
  - Wahrscheinlichkeit, dass X einen Wert weit ab vom Erwartungswert annimmt
  - loose und tight Bounds
  - Markov's Ungleichung (X nicht negativ)

$$\forall a > 0 \,.\, Pr[X \geq a] \leq \frac{\mathbb{E}[X]}{a}$$

Inhalt

Quicksort Analyse - Idee

- Input-Elemente aus uniformer Verteilung
- Erwartete Anzahl der Vergleiche X: $2n\,ln\,n$
  - X als Summe von X_{ij}(Indikator)
  - $X_{ij} = 1$, wenn zwei Elemente verglichen werden

$$Pr[X_{ij} = 1] = \frac{2}{i - j + 1}$$

  - Betrachte $\mathbb{E}[X]$ als Summe $\mathbb{E}[X_{ij}]$
    - Harmonische Reihe

Quellen

- T. Ottmann, P. Widmayer: Algorithmen und Datenstrukturen, 2011, Springer-Verlag
- P. Gamper, Randomisierte Primzahlentests, 2007, rwth Aachen
- T. Worsch, Skript der VO Randomisierte Algorithmen 2015/16, Universität Karlsruhe
- Carnegie Mellon School of Computer Science, Lecture 3 - Probabilistic Analysis and Randomized Quicksort
- Dr. H. Röglin, Skript der VO Probabilistic Analysis of Algorithms, Universität Bonn, 2016

Vielen Dank für die Aufmerksamkeit!