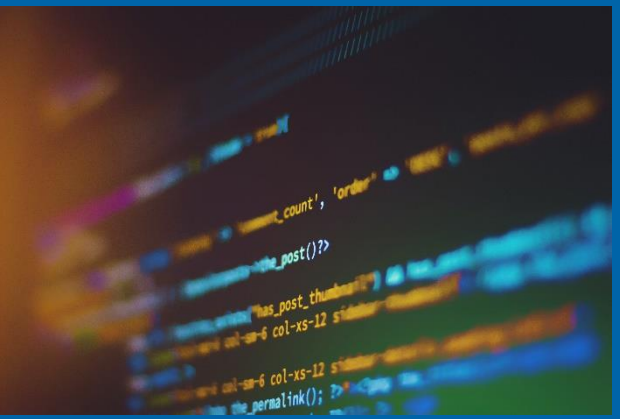
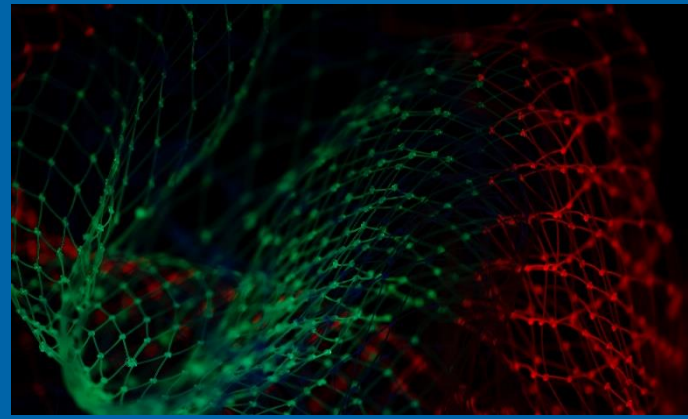
TECHNISCHE UNIVERSITÄT
BERGAKADEMIE FREIBERG
Die Ressourcenuniversität. Seit 1765.

BLOOM-FILTER

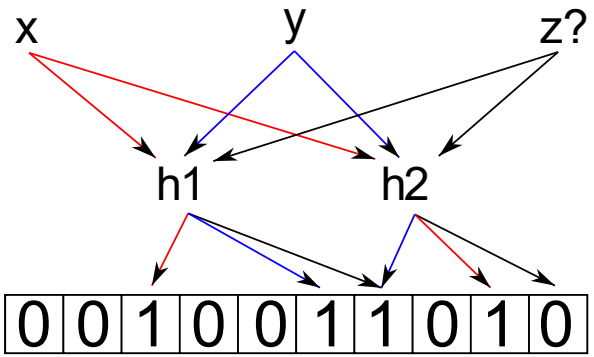
Eine probabilistische Datenstruktur





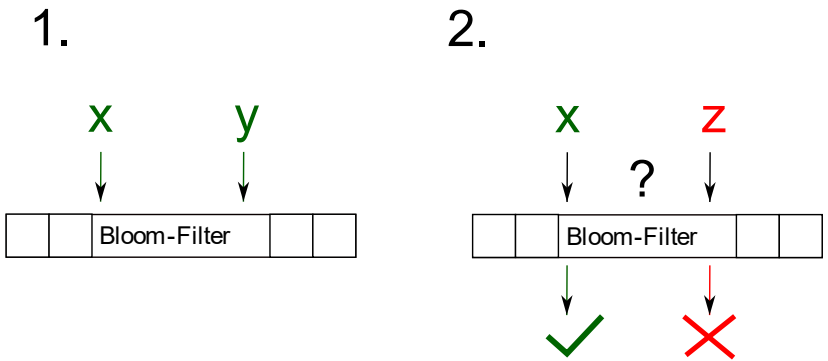
Gliederung

- Prinzipielle Idee
- Anwendungen
- Funktionsweise und Operationen
- Varianten
- Zusammenfassung
- Quellen



Prinzipielle Idee

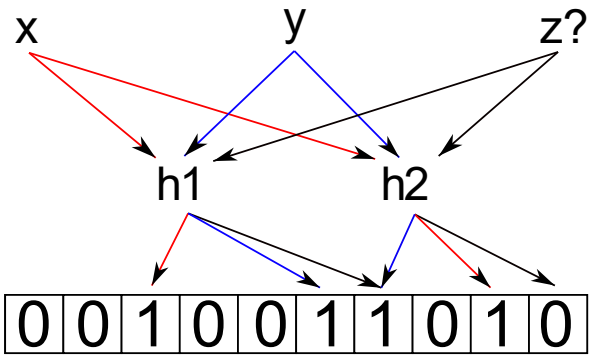
- Erfunden durch **Burton H. Bloom 1970** [1]
- Zur **Überprüfung**, ob **Elemente** in einer bestimmten Menge **enthalten** sind
- **Effizient** und **platzsparend**
- **Ohne** dabei Elemente selbst zu **speichern**
- **Probabilistische** Datenstruktur → **Falsch-Positiv-Rate**



Funktionsweise und Operationen

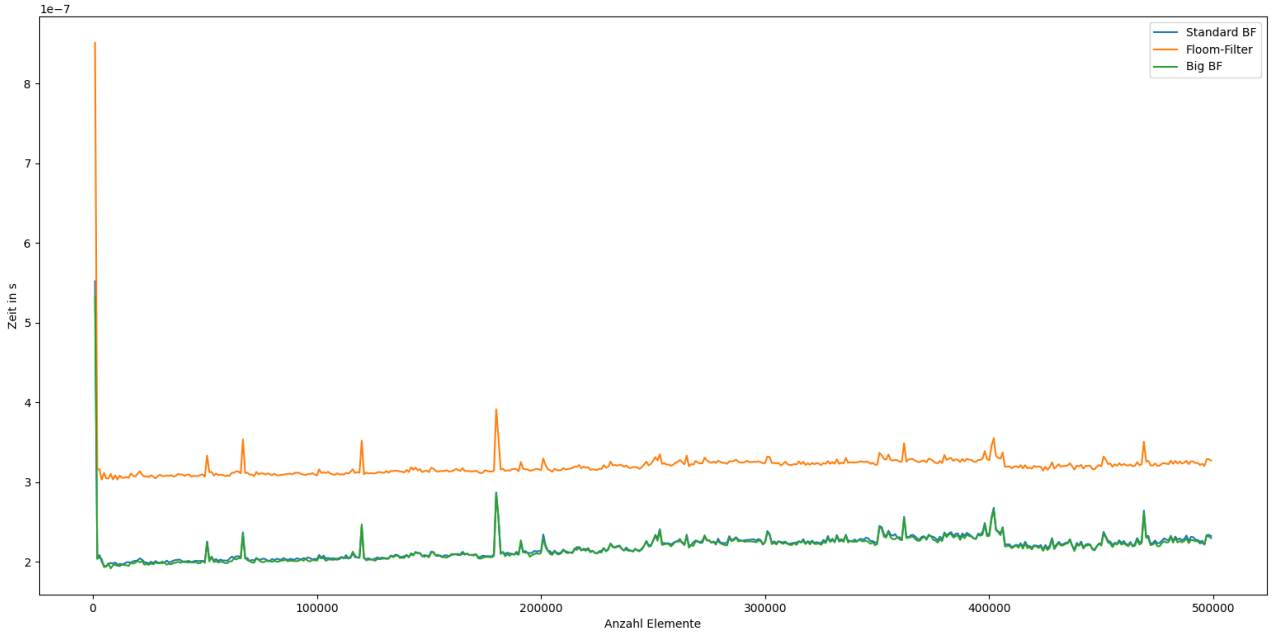
- Array der Länge **m** Bit voller Nullen [1]
- **n** einzufügende Elemente
- **k** unabhängige Hash-Funktionen (z.B. Murmur Hash [6])

- **Einfügen:** [1], [4]
  - **Einzufügende Elemente** werden auf Werte zwischen **0** und **n-1** gehasht
  - Bits an diesen Stellen auf Eins gesetzt
  - Zeit-/Rechenaufwand pro Element unabhängig von Arraygröße
  - Nur von **k** abhängig → $O(k)$



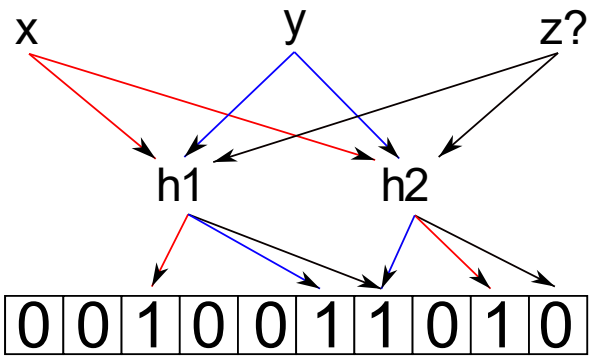
Funktionsweise und Operationen

Zeit pro Einfügen (in s) in Abhängigkeit der insgesamt einzufügenden Elemente



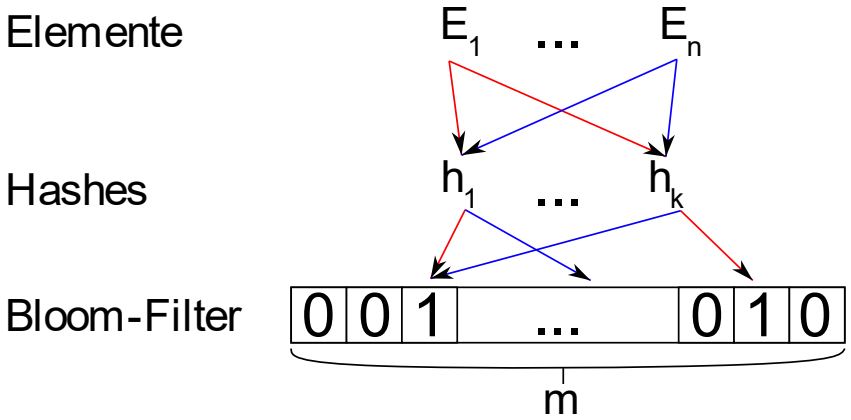
Funktionsweise und Operationen

- **Überprüfen:** [1], [4]
  - Zu **prüfendes Element** wird auf Werte zwischen **0** und **n-1** gehasht
  - Bits an diesen Stellen getestet
  - **Fall 1:** mindestens eine Stelle ist **Null** → **sicher** kein Mitglied
  - **Fall 2:** alle Stellen sind **Eins** → **vermutlich** ein Mitglied
  - **Falsch-Positiv-Rate** (FPP)
  - Wieder nur von **k** abhängig → $O(k)$



Funktionsweise und Operationen

- **Optimale Wahl der Parameter:**
  - **m** Bit Länge
  - **n** einzufügende Elemente
  - **k** unabhängige Hash-Funktionen
  - **FPP** Falsch-Positiv-Rate



Funktionsweise und Operationen

- **Optimale Wahl der Parameter:**
  - **m** Bit Länge
  - **n** einzufügende Elemente
  - **k** unabhängige Hash-Funktionen
  - **FPP** Falsch-Positiv-Rate

Wahrscheinlichkeit für ein Bit, noch Null zu sein: [5]

$$\left(1 - \frac{1}{m}\right)^{kn}$$

Somit ist die Wahrscheinlichkeit für k Einsen (FPP): [5]

$$FPP = \left(1 - \left(1 - \frac{1}{m}\right)^{nk}\right)^{k} \approx \left(1 - e^{-kn/m}\right)^{k}$$

Funktionsweise und Operationen

- **Optimale Wahl der Parameter:**
  - **m** Bit Länge
  - **n** einzufügende Elemente
  - **k** unabhängige Hash-Funktionen
  - **FPP** Falsch-Positiv-Rate

Da FPP möglichst klein sein soll ergibt sich: [6]

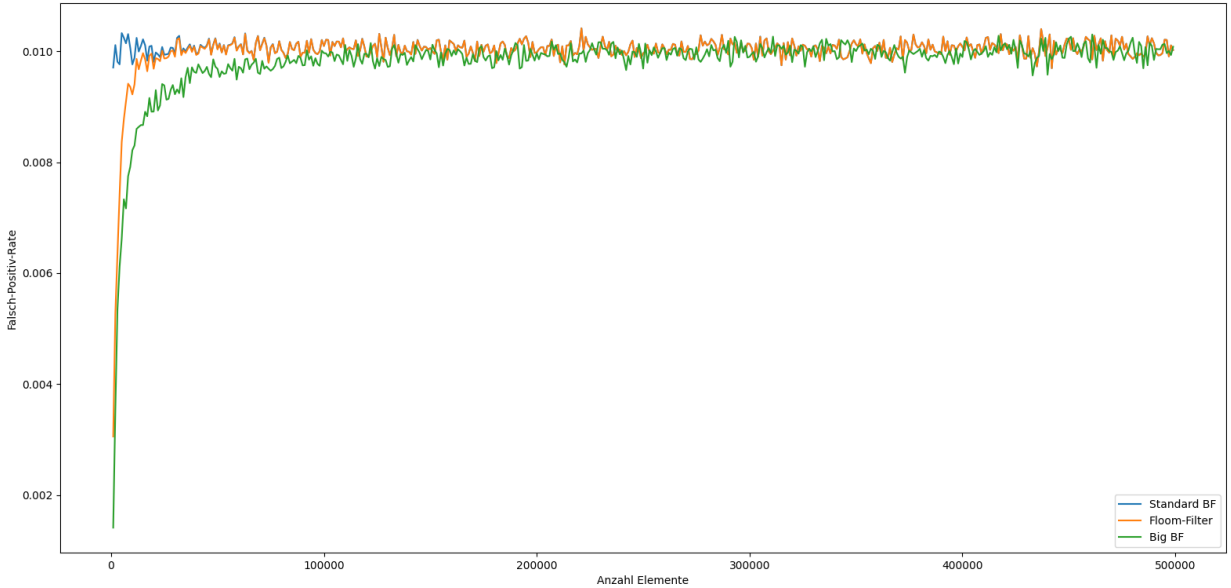
$$m = -\frac{n \ln FPP}{(\ln 2)^2}$$

Weiterhin ist die optimale Anzahl Hash-Funktionen k: [7]

$$k = \frac{m}{n} \ln 2$$

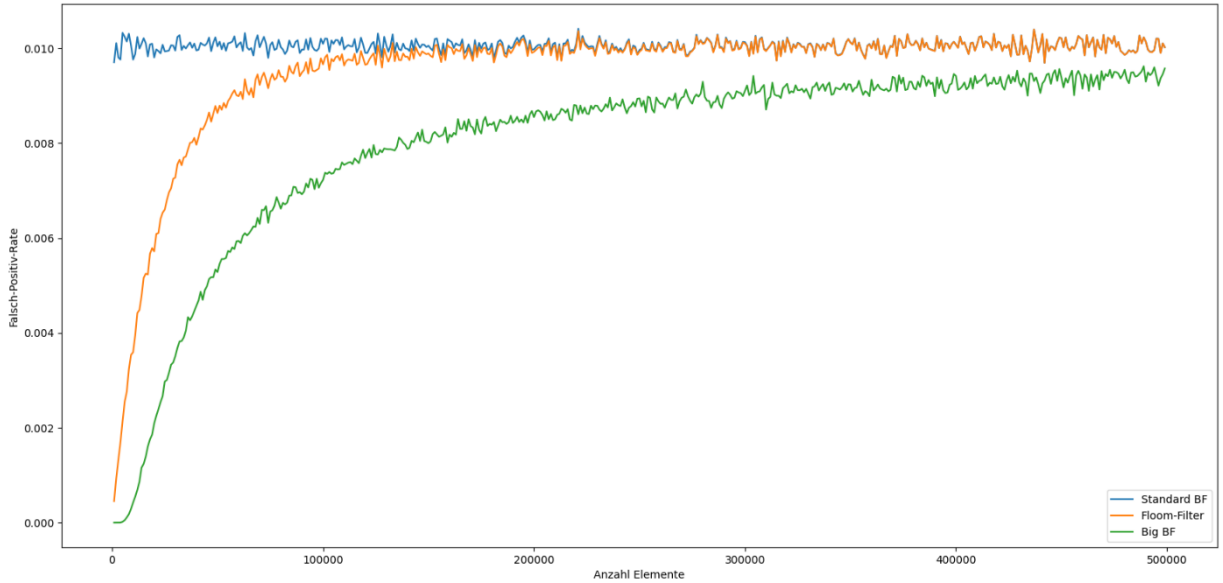
Vergleich der Varianten

Falsch-Positiv-Rate (Soll: 0.01) in Abhängigkeit der insgesamt einzufügenden Elemente
Floom-Filter mit 12 Bereichen



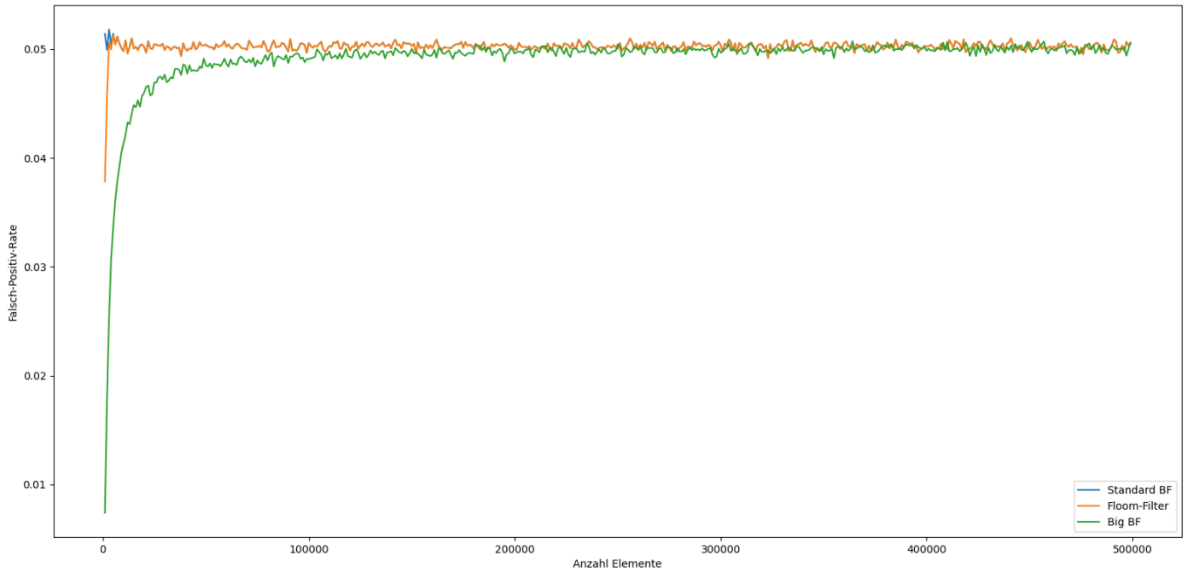
Vergleich der Varianten

Falsch-Positiv-Rate (Soll: 0.01) in Abhängigkeit der insgesamt einzufügenden Elemente
Floom-Filter mit 16 Bereichen



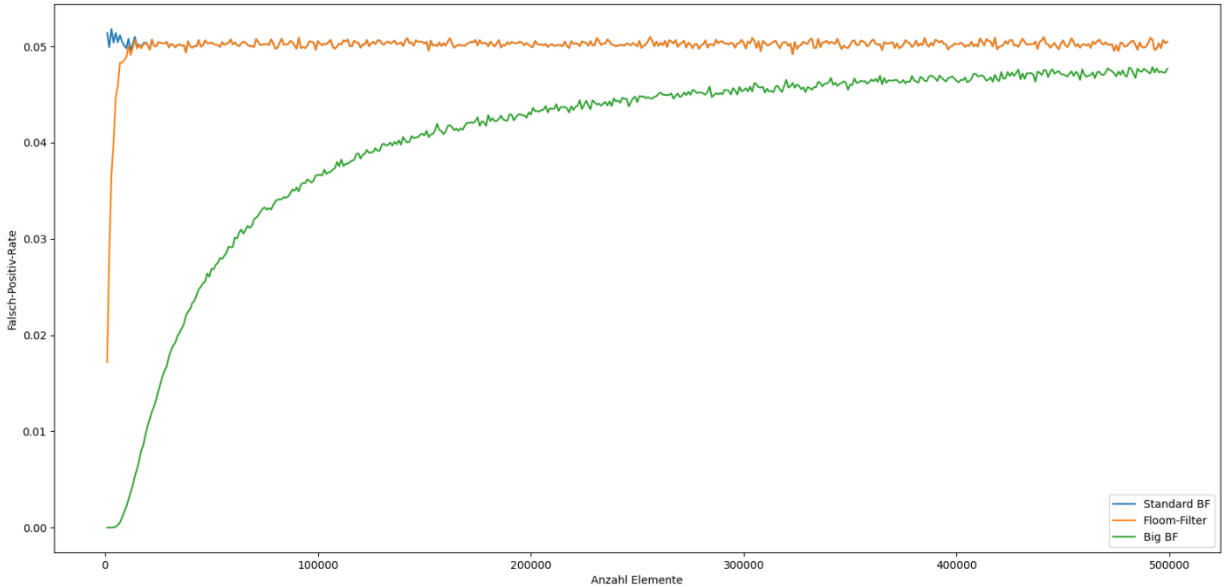
Vergleich der Varianten

Falsch-Positiv-Rate (Soll: 0.05) in Abhängigkeit der insgesamt einzufügenden Elemente
Floom-Filter mit 12 Bereichen



Vergleich der Varianten

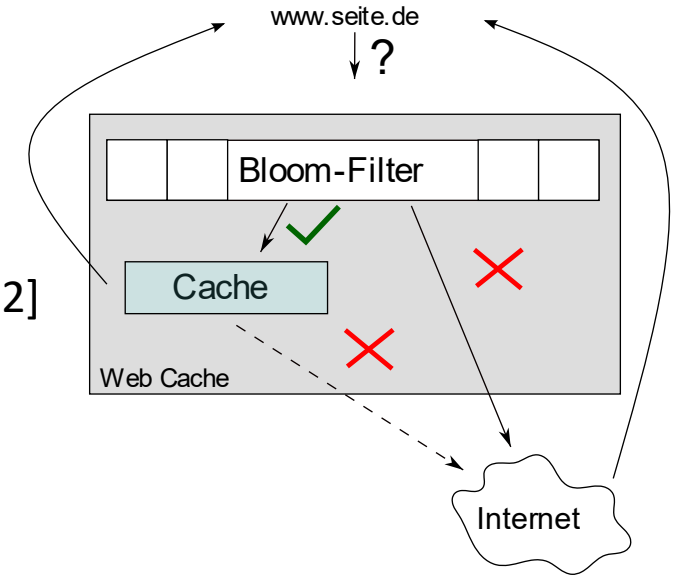
Falsch-Positiv-Rate (Soll: 0.05) in Abhängigkeit der insgesamt einzufügenden Elemente
Floom-Filter mit 16 Bereichen



Anwendungen

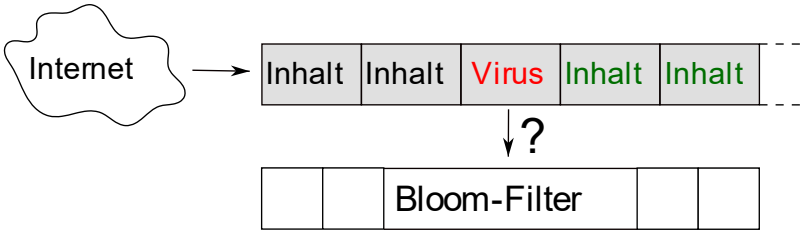
- **Allgemein:** Überall, wo in **kurzer Zeit** eine Aussage über **Mitgliedschaft** eines Elements in einer vorhandenen Menge Elementen gefragt ist [2]

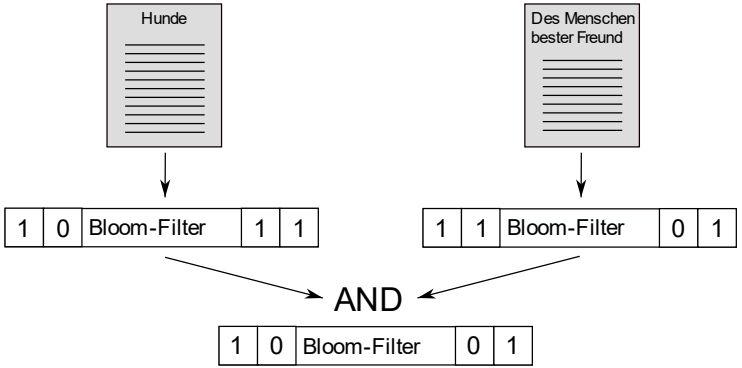
- **Netzwerkanwendungen:**
  - Routing [10]
  - Verhinderung von Loops [11]
  - IP-Routenverfolgung [12]
  - Verhindern von DDoS-Angriffen [3], [12]
  - Web-Caches [13]



Anwendungen

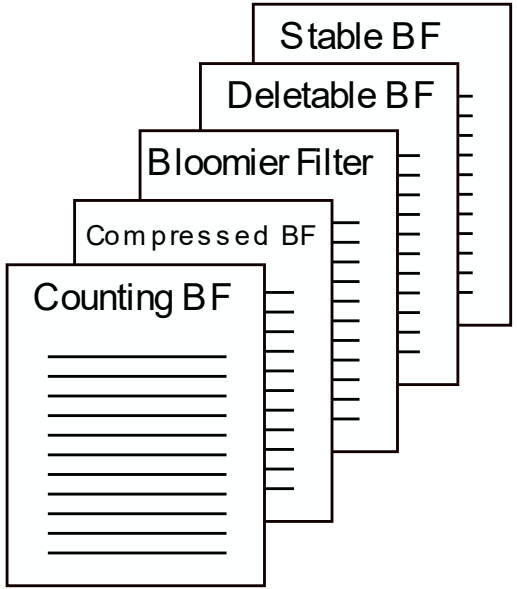
- **Sicherheitsanwendungen:**
  - Intrusion Detection Systems [14]
  - Verschlüsselte Suche [2]
  - Datenbanken [15]
- **Weitere Anwendungen:**
  - Rechtschreibprüfung [16]
  - Longest Prefix Matching [17]
  - Suchmaschinen [18]





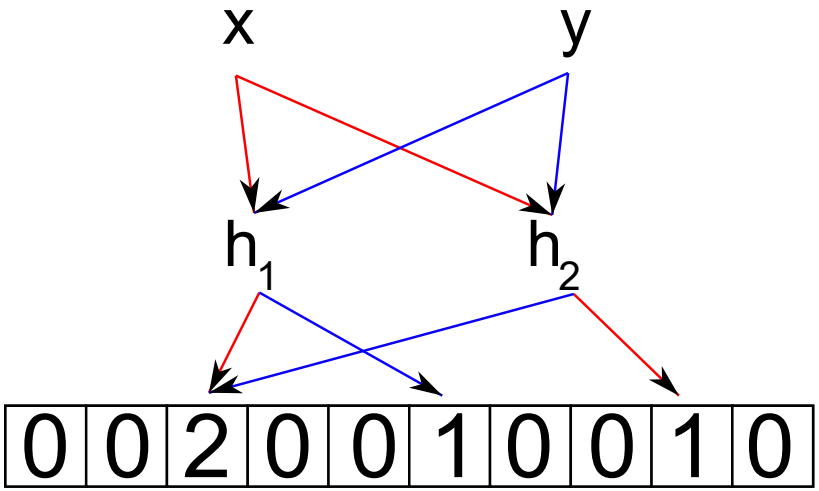
Varianten

- **Dutzende** Varianten, je nach Anwendung
- Setzen an unterschiedlichen Stellen und Problemen an, um für jeweilige **Bedürfnisse** zu **optimieren**
- Beispiele: Counting BF, Variable Increment BF, Compressed BF, Scalable BF, Generalized BF, Bloomier Filter, Stable BF, Weighted BF, Deletable BF, Spectral BF, Robust BF, …



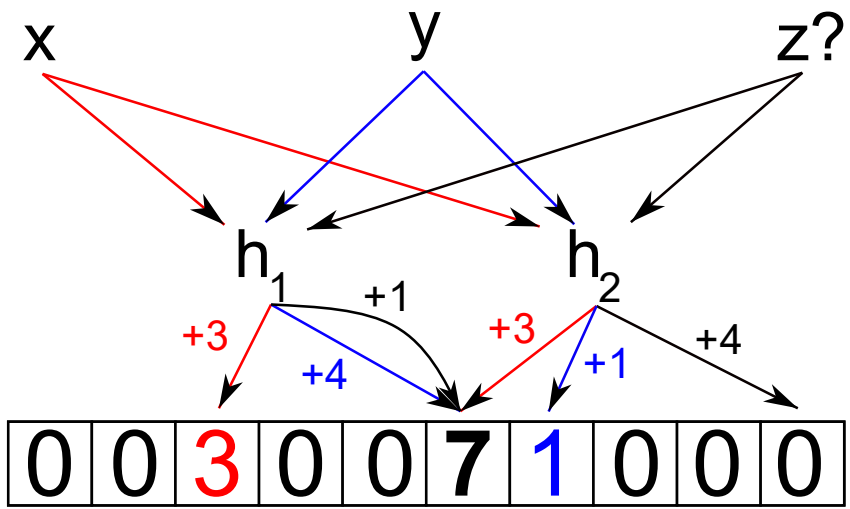
Varianten

- **Counting BF**: [7]
  - Statt je nur ein Bit, werden mehrere als **Zähler** verwendet
  - Pro Eintrag Zähler um Eins **erhöht**
  - So **Löschen** ermöglicht als neue Operation → Zähler um Eins **verkleinern**
  - Erhöhter Platzbedarf



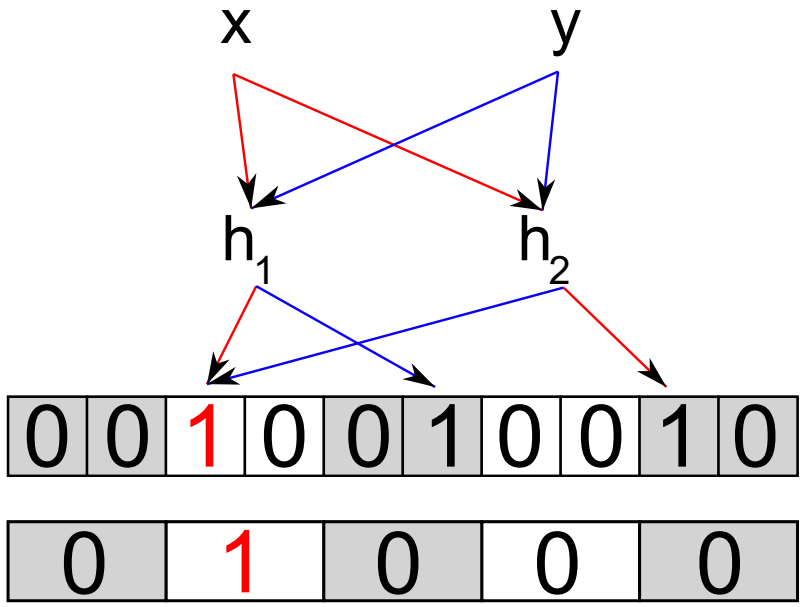
Varianten

- **<u>Variable Increment BF</u>**: [8]
  - Baut auf Counting BF auf
  - Erhöht um speziell bestimmte Zahlen aus B_h**-Reihen**
  - Summen sind **distinkt**
  - Somit zusätzliche **Überprüfung** über Wert, nicht nur anhand der Position im Array
  - Erhöhter Platzbedarf



Varianten

- **Deletable BF**: [9]
  - Teilt Array in **b Bereiche**
  - Zusätzlich b Bit **Extraspeicher**
  - Wenn bereits auf Eins stehendes Feld wieder gesetzt werden soll, wird der **Bereich markiert**
  - So manchmal **löschen** ermöglicht → wenn Bereich nicht markiert, kann Bit darin entfernt werden
  - Erhöhter Platzbedarf, aber **weniger** als bei Counting BF



Zusammenfassung

- In **Praxis** und **Literatur** seit Erfindung (1970) oft thematisiert
- Beliebt aufgrund der **Effizienz** und **Vielseitigkeit**
- Ständige **Weiterentwicklung**, neue **Varianten**
  → Welche Variante ist am **besten**?
- Benötigt meist **weitere Datenstruktur**, die Daten wirklich speichert

- **Ist eine Datenstruktur mit den Eigenschaften des Bloom-Filters möglich, die zudem die Elemente selbst speichert?**

Quellen

- [1] Burton H. Bloom, 1970, Space/time trade-offs in hash coding with allowable errors, *Communications of the ACM* 13, Issue 7, 422–426. DOI: https://doi.org/10.1145/362686.362692.
- [2] Saibal Kumar Pal and Puneet Sardana, 2012, BLOOM FILTERS & THEIR APPLICATIONS, *International Journal of Computer Applications Technology and Research* 1, 25–29. DOI: https://doi.org/10.7753/2012.1006 https://ui.adsabs.harvard.edu/link_gateway/2012IJCAT...1...25P/doi:10.7753/2012.1006
- [3] Ripon Patgiri, Sabuzima Nayak, and Samir Borgohain. 2018. Preventing DDoS using Bloom Filter: A Survey. *ICST Transactions on Scalable Information Systems* 5, 19, 155865. DOI: https://doi.org/10.4108%2Feai.19-6-2018.155865

Quellen

- [4] Bloom Filter. *Brilliant.org*. Retrieved 18:02, May 12, 2022, from https://brilliant.org/wiki/bloom-filter/
- [5] Fabio Grandi. 2018. On the analysis of Bloom filters. *Information Processing Letters* 129, 35–39. DOI: https://doi.org/10.1016/j.ipl.2017.09.004
- [6] Nayak, S. and Patgiri, R. 2021. *RobustBF: A High Accuracy and Memory Efficient 2D Bloom Filter*. DOI: https://doi.org/10.48550/arxiv.2106.04365
- [7] Fan, L., Cao, P., Almeida, J., and Broder, A. Z. 1998. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. In *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*. SIGCOMM '98. Association for Computing Machinery, New York, NY, USA, 254–265. DOI: https://doi.org/10.1145/285237.285287

Quellen

- [8] Rottenstreich Ori, Kanizo Yossi, and Keslassy Isaac. 2014. The Variable-Increment Counting Bloom Filter. *IEEE/ACM Trans. Networking* 22, 4, 1092–1105. DOI: https://doi.org/10.1109/TNET.2013.2272604
- [9] Rothenberg, C., Macapuna, C., Verdi, F., and Magalhaes, M. 2010. The deletable Bloom filter: a new member of the Bloom family. *IEEE Commun. Lett.* 14, 6, 557–559. DOI: https://doi.org/10.1109/LCOMM.2010.06.100344
- [10] Rhea S.C. and Kubiatowicz J. 2002. Probabilistic location and routing. In *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, 1248-1257 vol.3. DOI: https://doi.org/10.1109/INFCOM.2002.1019375

Quellen

- [11] Whitaker A. and Wetherall D. 2002. Forwarding without loops in Icarus. In *2002 IEEE Open Architectures and Network Programming Proceedings. OPENARCH 2002 (Cat. No.02EX571)*, 63–75. DOI: https://doi.org/10.1109/OPNARC.2002.1019229
- [12] Laufer Rafael P., Velloso Pedro B., Cunha Daniel de O., Moraes Igor M., Bicudo Marco D.D., Moreira Marcelo D.D., and Duarte Otto Carlos M.B. 2007. Towards Stateless Single-Packet IP Traceback. In *32nd IEEE Conference on Local Computer Networks (LCN 2007)*, 548–555. DOI: https://doi.org/10.1109/LCN.2007.15
- [13] Wang, J. 1999. A Survey of Web Caching Schemes for the Internet. *SIGCOMM Comput. Commun. Rev.* 29, 5, 36–46. DOI: https://doi.org/10.1145/505696.505701
- [14] Dharmapurikar S., Krishnamurthy P., Sproull T., and Lockwood J. 2003. Deep packet inspection using parallel Bloom filters. In *11th Symposium on High Performance Interconnects, 2003. Proceedings*, 44–51. DOI: https://doi.org/10.1109/CONECT.2003.1231477

Quellen

- [15] Gremillion, L. L. 1982. Designing a Bloom Filter for Differential File Access. *Commun. ACM* 25, 9, 600–604. DOI: https://doi.org/10.1145/358628.358632
- [16] Murugan, S., Bakthavatchalam, T. A., and Sankarasubbu, M. 2020. SymSpell and LSTM based Spell-Checkers for Tamil. http://uttamam.org/papers/20_17.pdf
- [17] Dharmapurikar, S., Krishnamurthy, P., and Taylor, D. E. 2003. Longest Prefix Matching Using Bloom Filters. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. SIGCOMM '03. Association for Computing Machinery, New York, NY, USA, 201–212. DOI: https://doi.org/10.1145/863955.863979
- [18] Jain, N., Dahlin, M., and Tewar, R. 2005. Using Bloom Filters to Refine Web Search Results. In *Eighth International Workshop on the Web and Databases (WebDB '05)*. https://www.microsoft.com/en-us/research/wp-content/uploads/2017/01/webdb-167.pdf

Quellen

- Bilder:
  - Folie 1 links: Photo by Shahadat Rahman on Unsplash
  - Folie 1 Mitte: Photo by Carlos Muza on Unsplash
  - Folie 1 rechts: Photo by Pietro Jeng on Unsplash
  - Folie 2 unten links: Photo by FLY:D on Unsplash

Diskussion

- Prinzipielle Idee
- Anwendungen
- Funktionsweise und Operationen
- Varianten
- Zusammenfassung
- Quellen

tu-freiberg.de

TU Bergakademie Freiberg bergakademie_freiberg TUBergakademie TUBergakademie

TU BERGAKADEMIE FREIBERG
Universitätskommunikation
Prüferstr. 2
09599 Freiberg
Tel. +49(0)3731 39-2711, -3461
kommunikation@tu-freiberg.de

WELTOFFENE HOCHSCHULEN GEGEN FREMDENFEINDLICHKEIT

FAMILIE IN DER HOCHSCHULE

Europäische Union

Europa fördert Sachsen. EFRE Europäischer Fonds für regionale Entwicklung