Bloom filters

Agenda

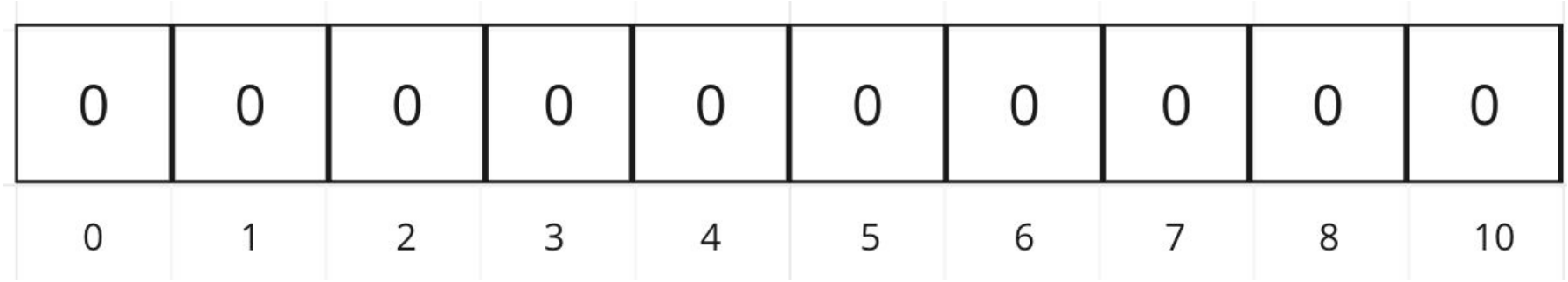
- Introduksjon og bakgrunn
- Teknisk gjennomgang
- Praktisk eksempel
- Space/Time Trade-offs in Hash Coding with Allowable Errors (1970)
- Konklusjon (fordeler og ulemper)

Introduksjon og bakgrunn

- Burton H. Bloom introduserte bloom filter i 1970 i den vitenskapelige artikkelen:
  - Space/Time Trade-offs in Hash Coding with Allowable Errors
- Sannsynlighetsbasert (probabilistic)
  - Kan finne ut om et element kanskje er i settet eller om elementet garantert ikke er i settet
  - Ikke mulig å finne ut om et element garantert er i settet
- Plass-effektivt
  - Tillater falske positive for å oppnå en effektiv plassutnyttelse.

Teknisk gjennomgang

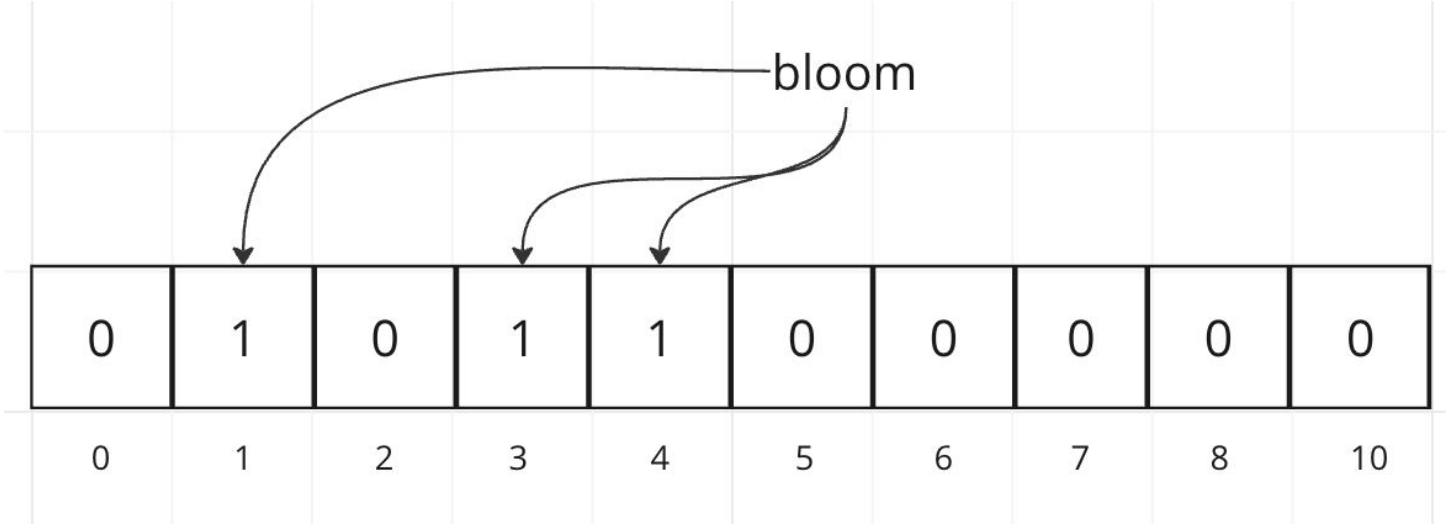
- Datastruktur som bruker flere hash-funksjoner for å legge til eller sjekke verdier i et bit-array
  - hvert element er en bit som kan være 0 eller 1
  - Alle bits i arrayet starter som 0



- For å legge til et element i filteret, hasher man elementet i en hash funksjon.

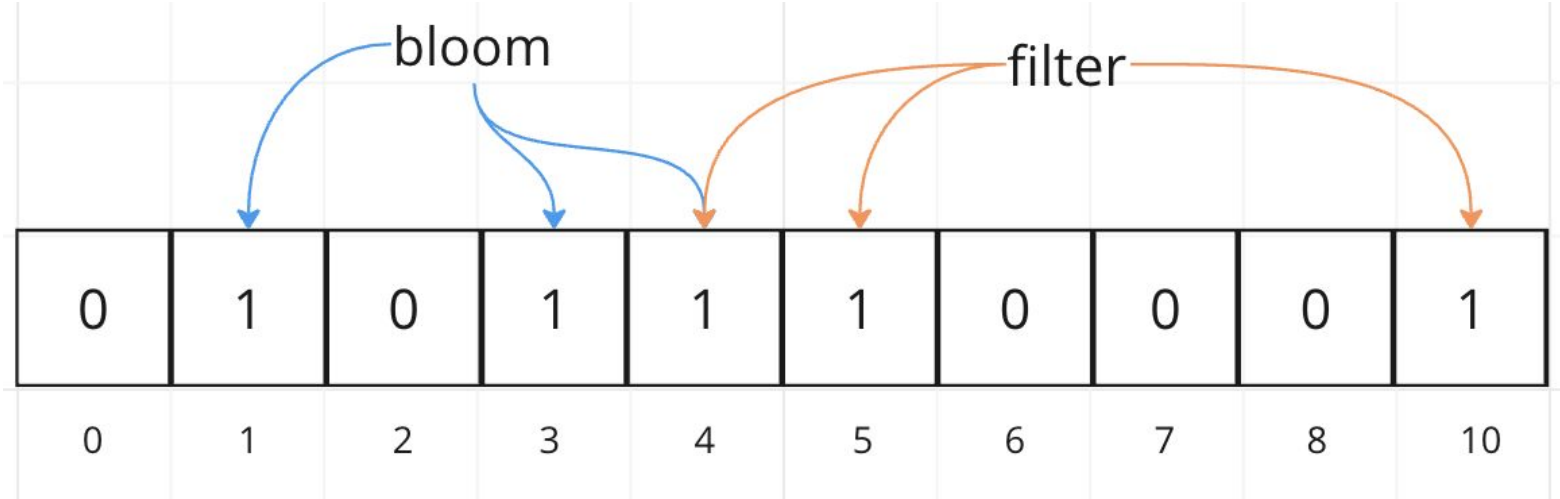
Legge til verdier i bloom filter

- La oss legge til “bloom” i filteret.
- Vi kalkulerer indeksene ved hash funksjonene.
  - h1(“bloom”) % 10 = 1
  - h2(“bloom”) % 10 = 3
  - h3(“bloom”) % 10 = 4



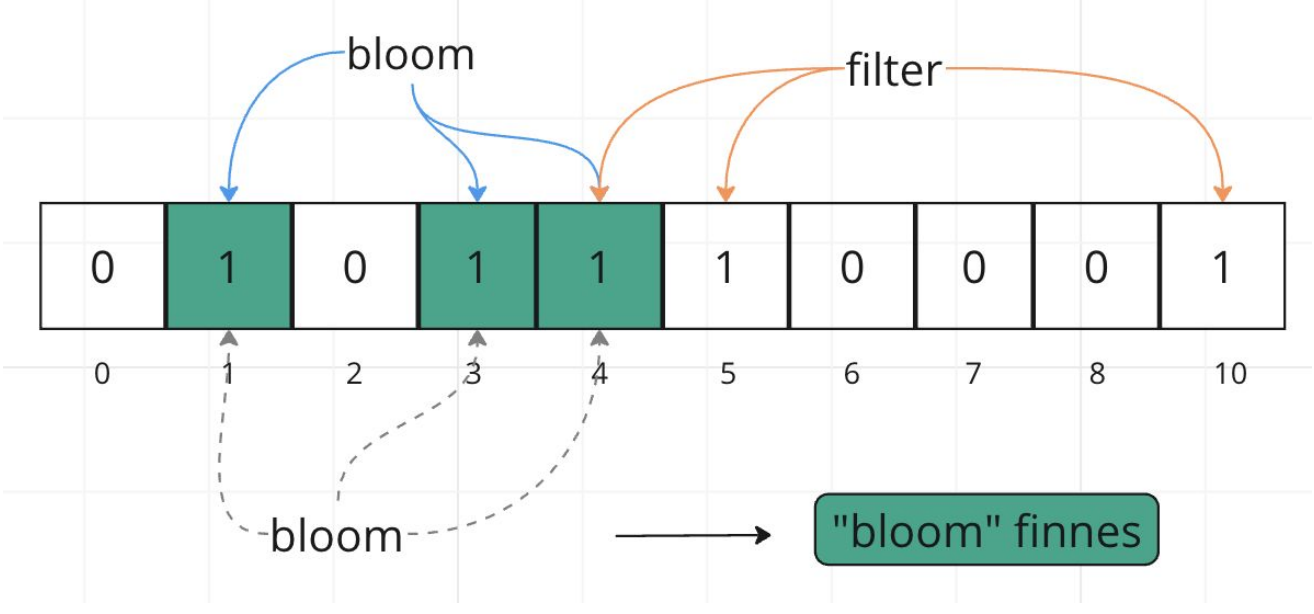
Legge til verdier i bloom filter

- Vi legger til ordet “filter”
- Hash funksjonene returnerer indeksene 4, 5 og 10.

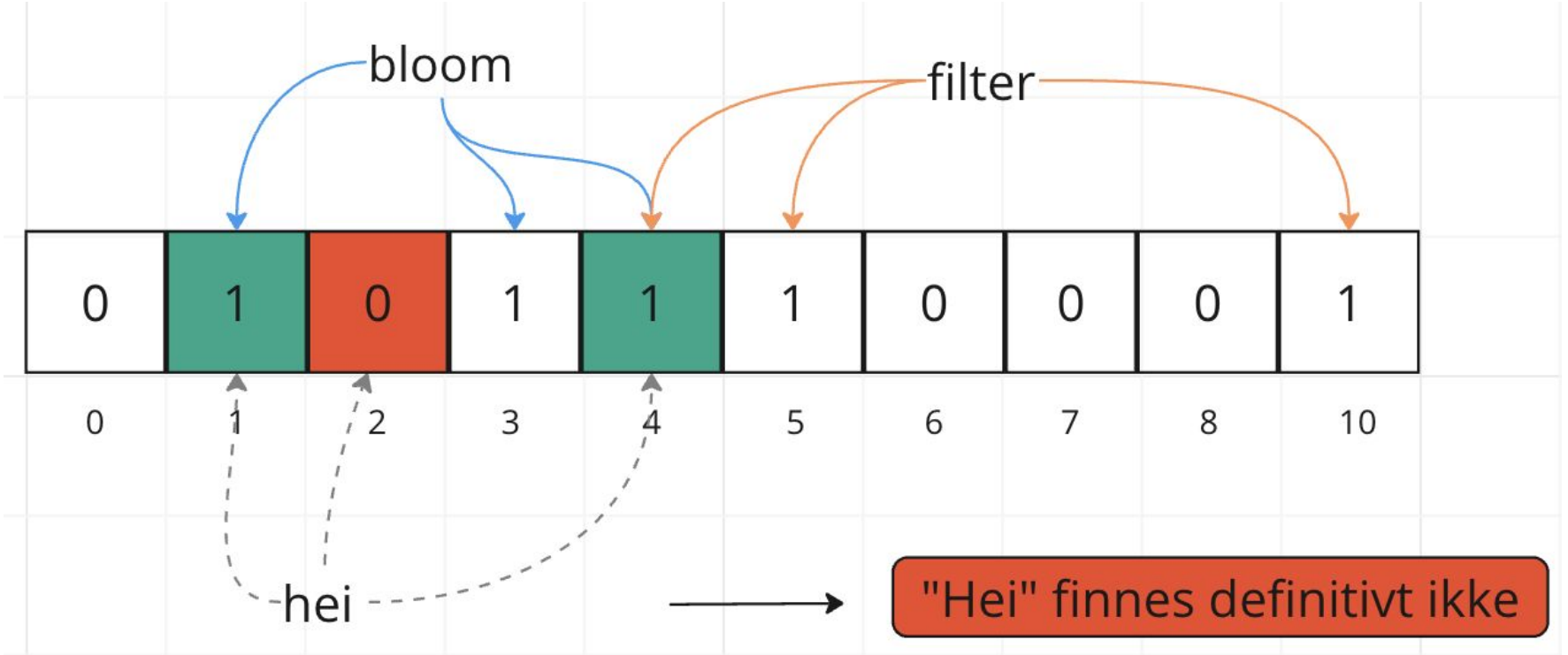


Sjekke verdier i bloom filteret

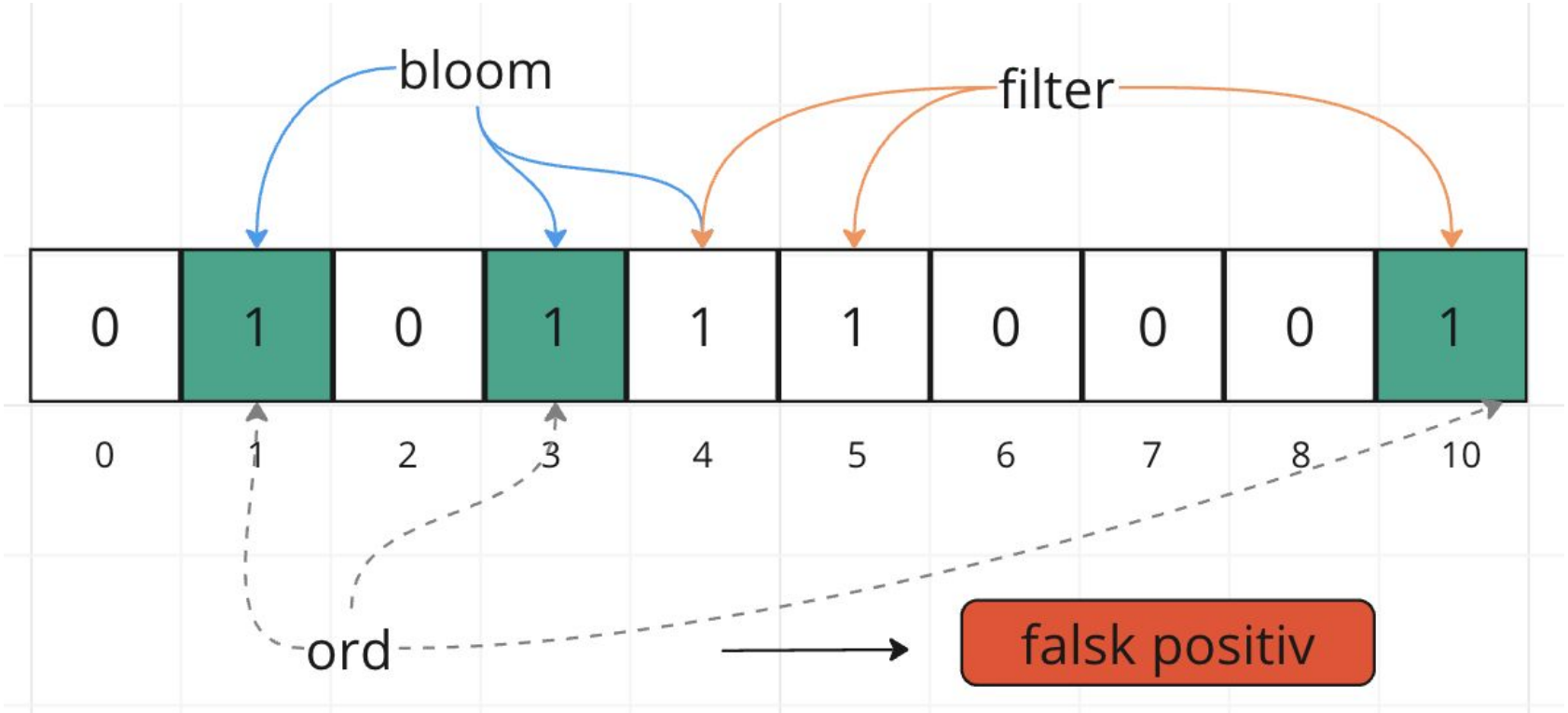
- Sjekker ordet “bloom”
- indeksen kalkuleres ved bruk av de samme hash funksjonene.



Sjekke verdier i bloom filteret



Falsk positiv



Space/Time Trade-offs in Hash Coding with Allowable Errors (1970)

- Burton H.Bloom 1970
- Introduserer bloom filtre
- Går gjennom og sammenligner dette med konvensjonell hashing i tillegg til en annen metode for hashkoding med tillatte falske positiver

Space/Time Trade-offs in Hash Coding with Allowable Errors (1970) (fortsettelse)

- Sammenligner to ulike metoder for hashkoding med tillatte falske positiver
  - Metode 1 er bygget på den konvensjonelle metoden for “feilfri” hashkoding
  - Metode 2 er bloom filtre

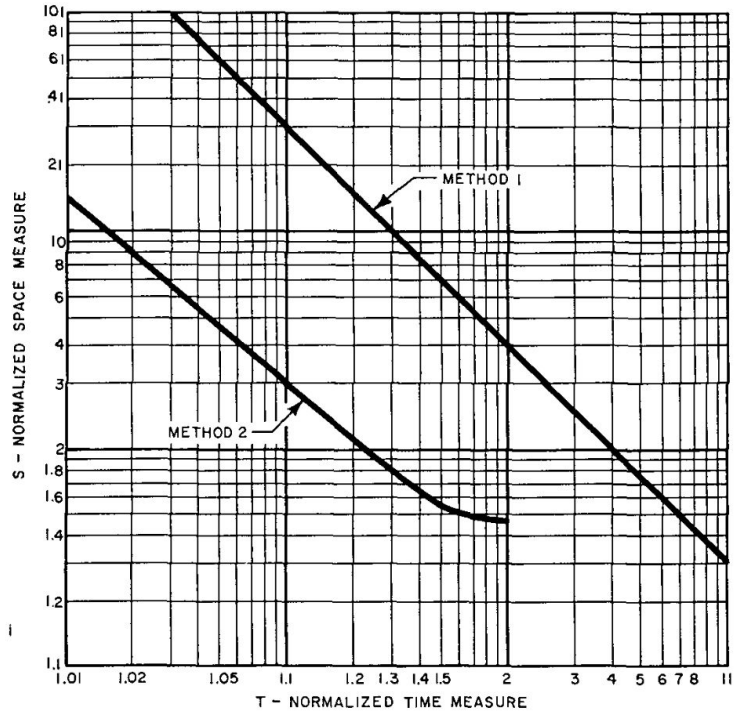


FIG. 1

Space/Time Trade-offs in Hash Coding with Allowable Errors (1970) (fortsettelse)

- Sammenligner bloom filtre med konvensjonell hashing
- Bloom filtre sparer mange disk-aksesser, men tillatter også falske positiver

TABLE I. SUMMARY OF EXPECTED PERFORMANCE OF HYPHENATION APPLICATION OF HASH CODING USING METHOD 2 FOR VARIOUS VALUES OF ALLOWABLE FRACTION OF ERRORS

P = Allowable Fraction of Errors	*N = Size of Hash Area (Bits)*	*Disk Accesses Saved*
$\frac{1}{2}$	72,800	45.0%
$\frac{1}{4}$	145,600	67.5%
$\frac{1}{8}$	218,400	78.7%
$\frac{1}{16}$	291,200	84.4%
$\frac{1}{32}$	364,000	87.2%
$\frac{1}{64}$	509,800	88.5%

Hvor brukes bloom filter

- Mange NoSQL-databaser bruker Bloom-filtre for å redusere diskavlesninger for nøkler som ikke eksisterer
- Nettlesere som chrome pleide å bruke et bloom-filter for å identifisere ondsinnede nettadresser
  - Dette brukes imidlertid ikke lenger ettersom antallet ondsinnede URL-er vokser til millioner, og en mer effektiv, men komplisert løsning er nødvendig
- Brukt i stavekontroll
- Brukt i søkemotorer

Variabler / parametere

- Størrelse på bit array (m)
- Antall elementer som blir lagt til i arrayet (n)
- Optimalt antall hashfunksjoner (k)
- Sannsynlighet for falske positiver (P)

$$m = -\frac{n \ln P}{(ln2)^2}$$

$$k = \frac{m}{n} ln2$$

$$P = \left(1 - \left[1 - \frac{1}{m}\right]^{kn}\right)^k$$

Konklusjon

- I brukstilfeller der vi kan tolerere falske positiver, men ikke noen falske negativer, kan et bloom filter være veldig nyttig fordi det er svært tid- og plasseffektivt sammenlignet med konvensjonell hashing

Ressurser

- Space/Time Trade-offs in Hash Coding with Allowable Errors: https://github.com/aohrn/in3120-2023/blob/main/papers/space-time-trade-offs-in-hash-coding-with-allowable-errors.pdf
- Bloom filter wikipedia: https://en.wikipedia.org/wiki/Bloom_filter#:~:text=Another%20alternative%20to%20classic%20Bloom,efficient%20variants%20of%20cuckoo%20hashing.
- https://mightguy.medium.com/bloomfilter-in-searchengine-part-i-2f34814894b9
- https://brilliant.org/wiki/bloom-filter/