ALGORITMI ANALIZE MASIVNIH PODATKOV

DOMEN MONGUS

P03 – Prilagoditve pomnilniški hierarhiji

Vsebina – V4Volume

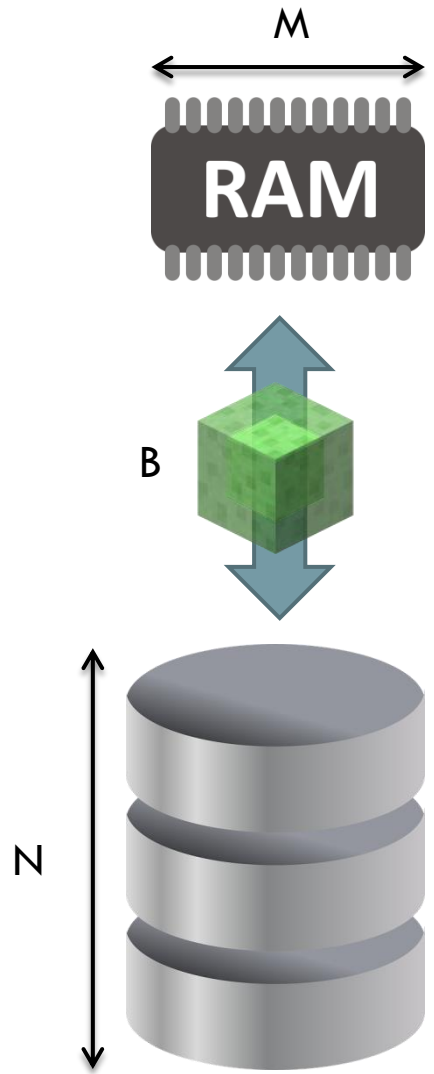
- DAM model (ang. Disk Access Model)
- Predpomnilniško-zavedni algoritmi

Ključne predpostavke:

- Podatki so preveliki za RAM
  - Podatkovne strukture so prevelike za RAM
- Operacije nad podatki so zelo enostavne
  - Časovna zahtevnost odvisna zgolj od števila dostopov do diska

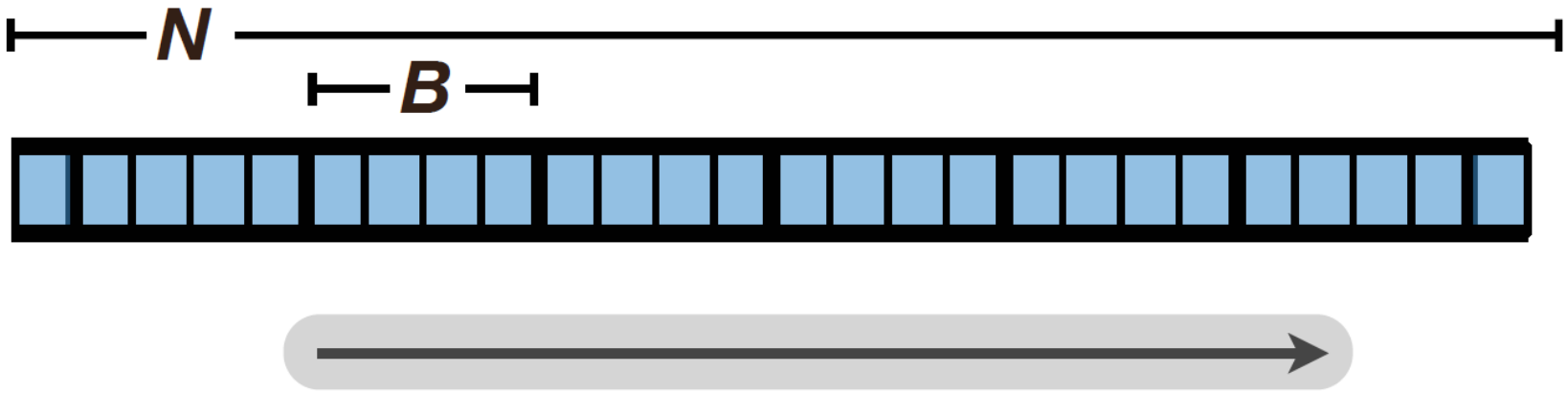
Model dostopov do diska (DAM)

- Velika količina podatkov
  - Podatki se prenašajo v blokih
  - V pomnilnik lahko shranimo nekaj blokov
  - Velikost diska je „neomejena“
- Cilj: Minimizacija prenosa podatkov
  - Parametri:
    - B = velikost bloka
    - M = velikost pomnilnika RAM
    - N = velikost podatkov



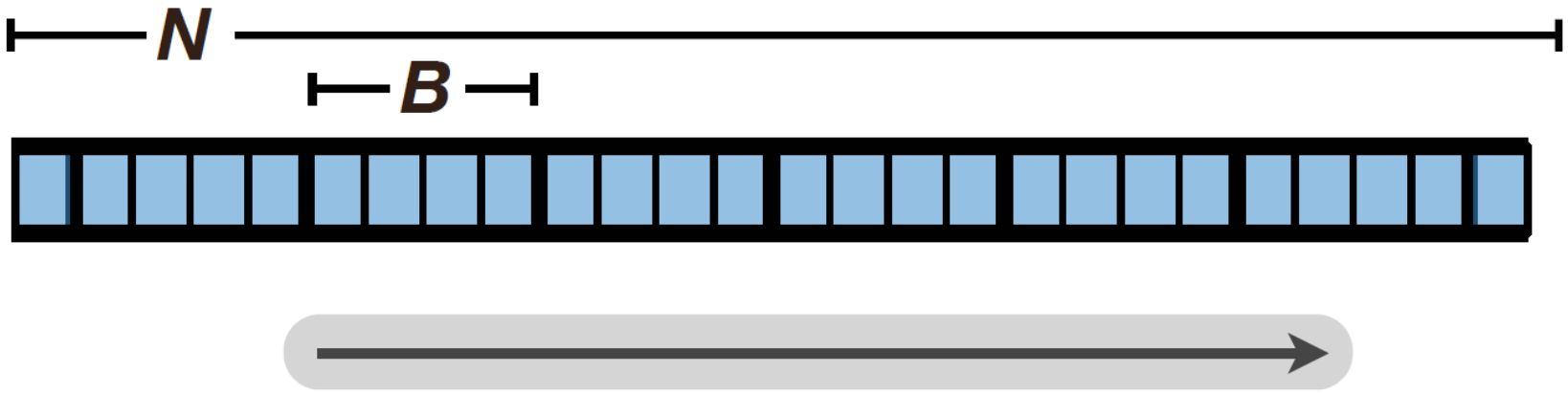
Prebiranje vrstice

- Koliko IO operacij je potrebno za branje?



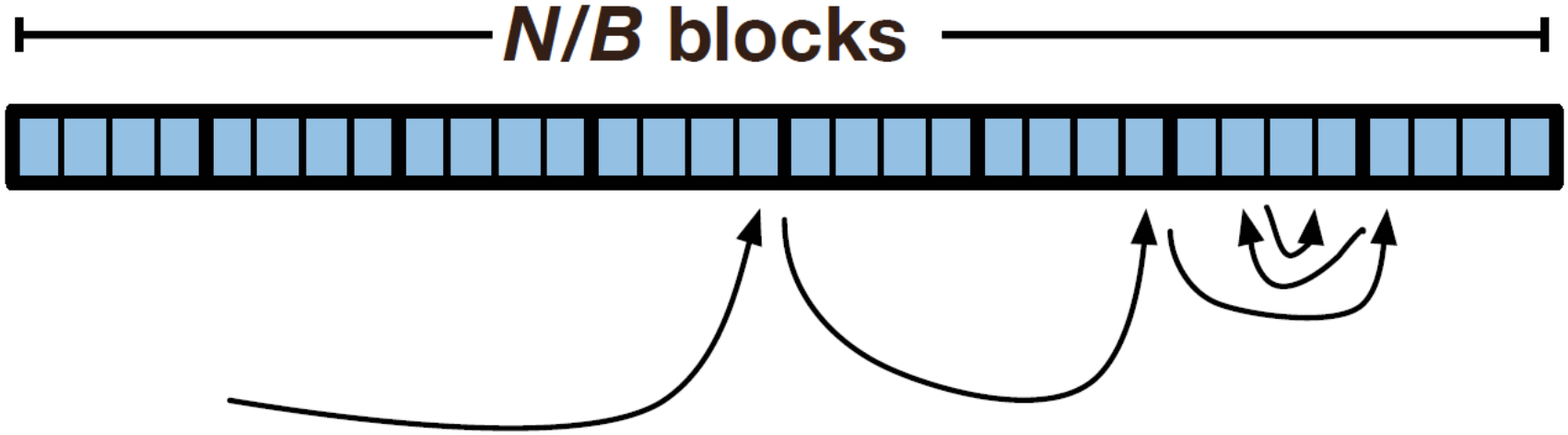
Prebiranje vrstice

- Koliko IO operacij je potrebno za branje?
  - O(N/B)



Koliko IO operacij je potrebno za iskanje elementa?

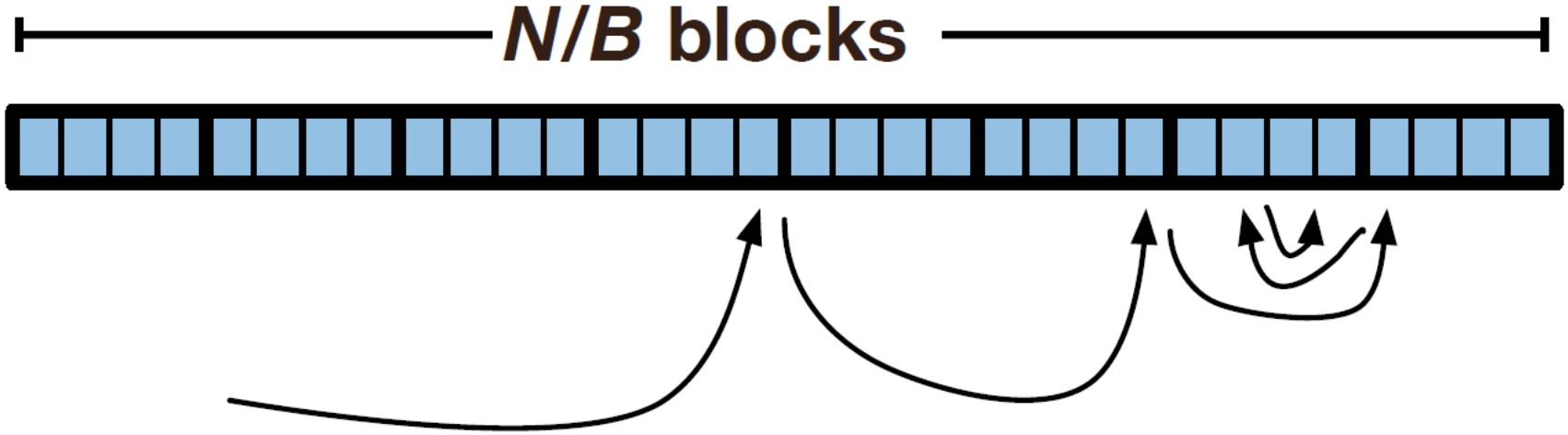
- Najslabši primer?



Koliko IO operacij je potrebno za iskanje elementa?

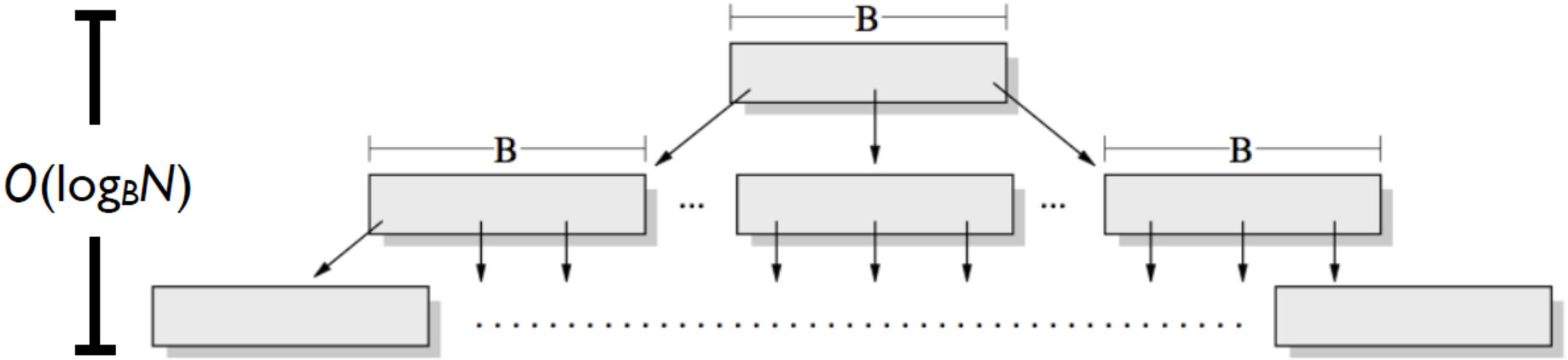
- Najslabši primer?

$$O\left(\log_2 \frac{N}{B}\right) \approx O(\log_2 N)$$



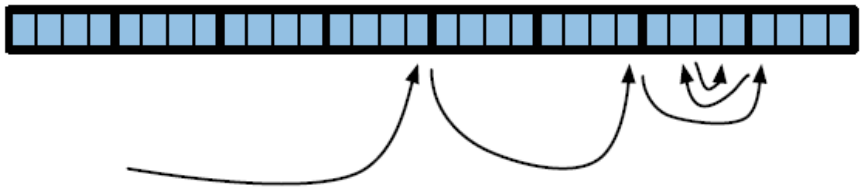
Koliko IO operacij je potrebno za iskanje elementa?

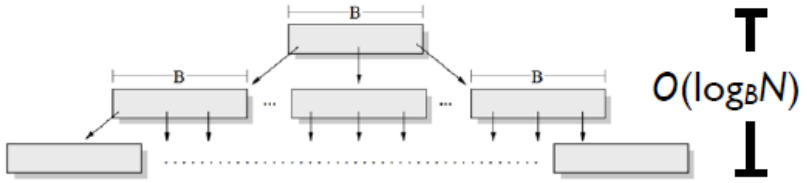
- PRIMER: Binarno drevo



Koliko IO operacij je potrebno za iskanje elementa?

- **Nauk zgodbe:** podatkovna struktura je ključ do učinkovite implementacije algoritma.

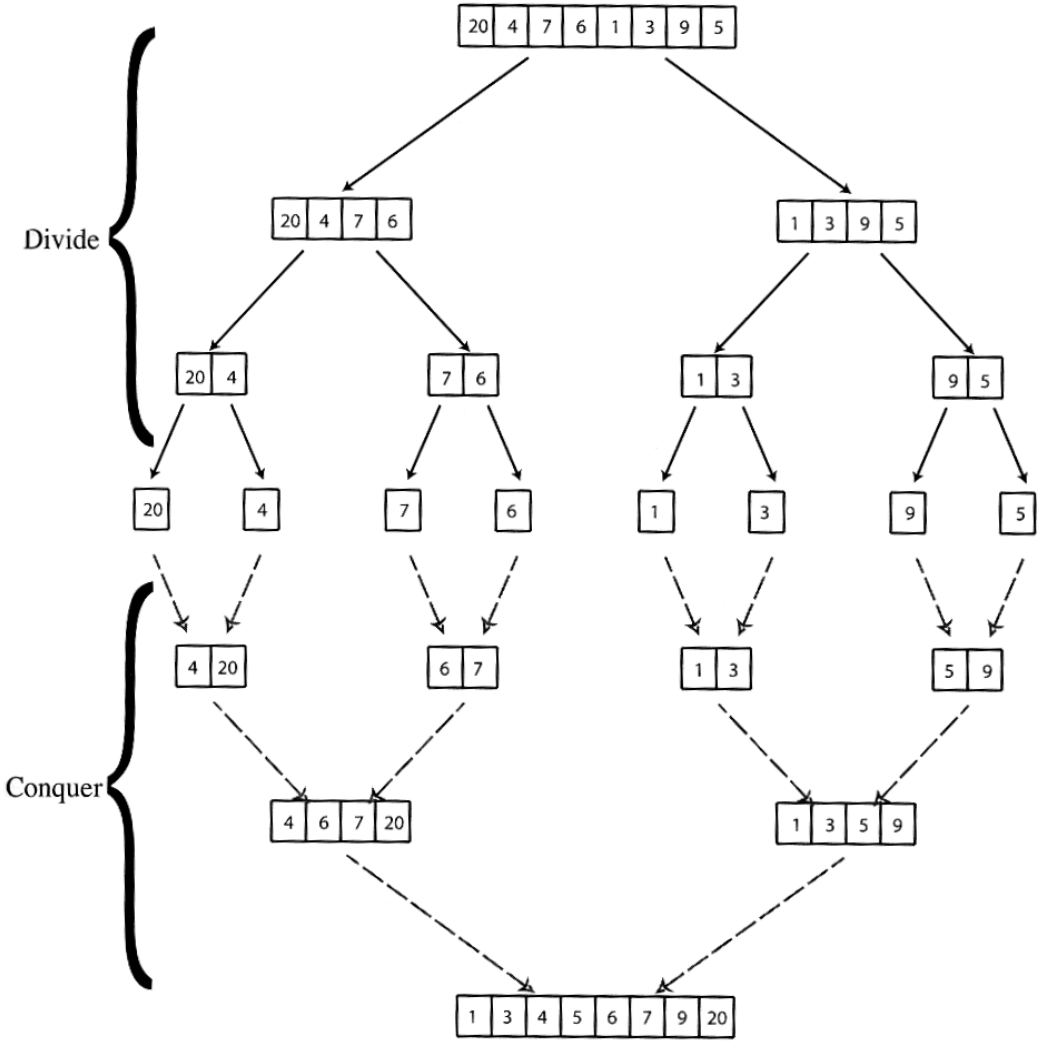




$$O(\log_2 N)$$

$$O(\log_B N) = O\left(\frac{\log_2 N}{\log_2 B}\right)$$

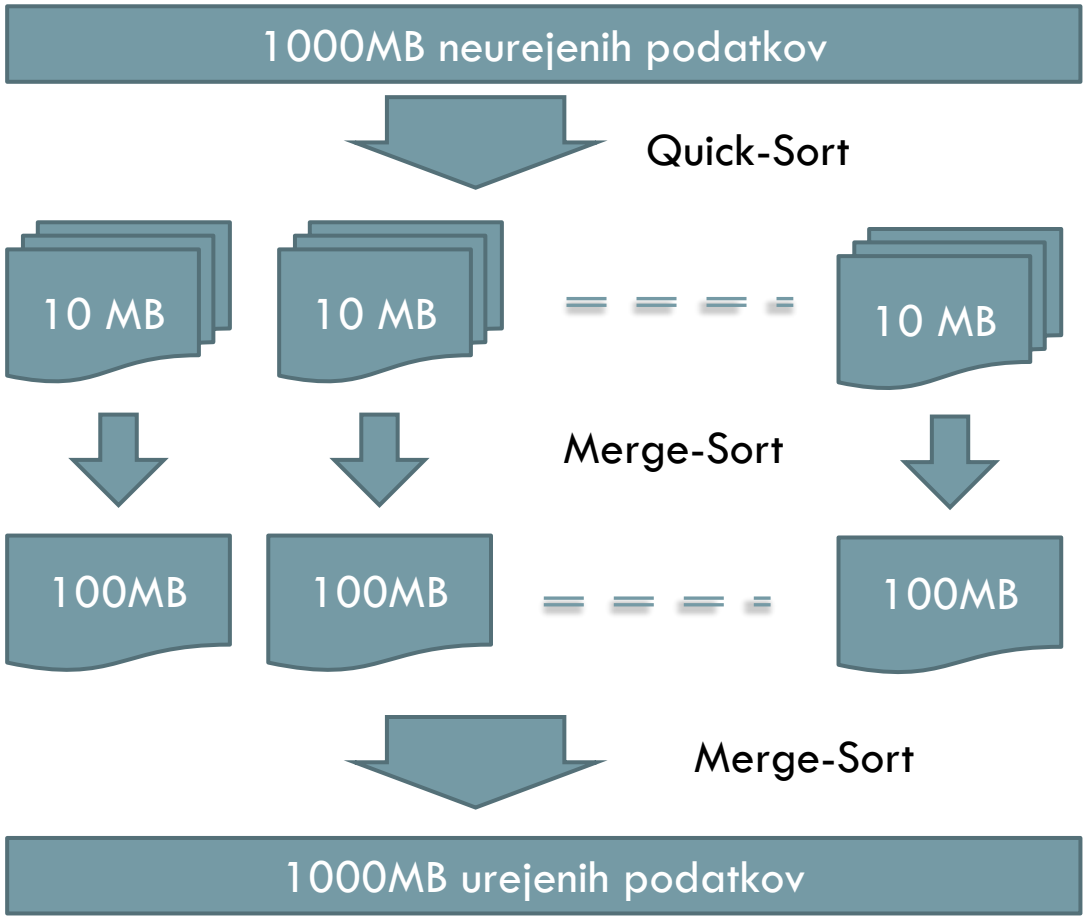
PONOVITEV: Urejanje z zlivanjem



Učinkovito urejanje velikih podatkov

- N = 1000 MB
- M = 10 MB
- B = 1MB

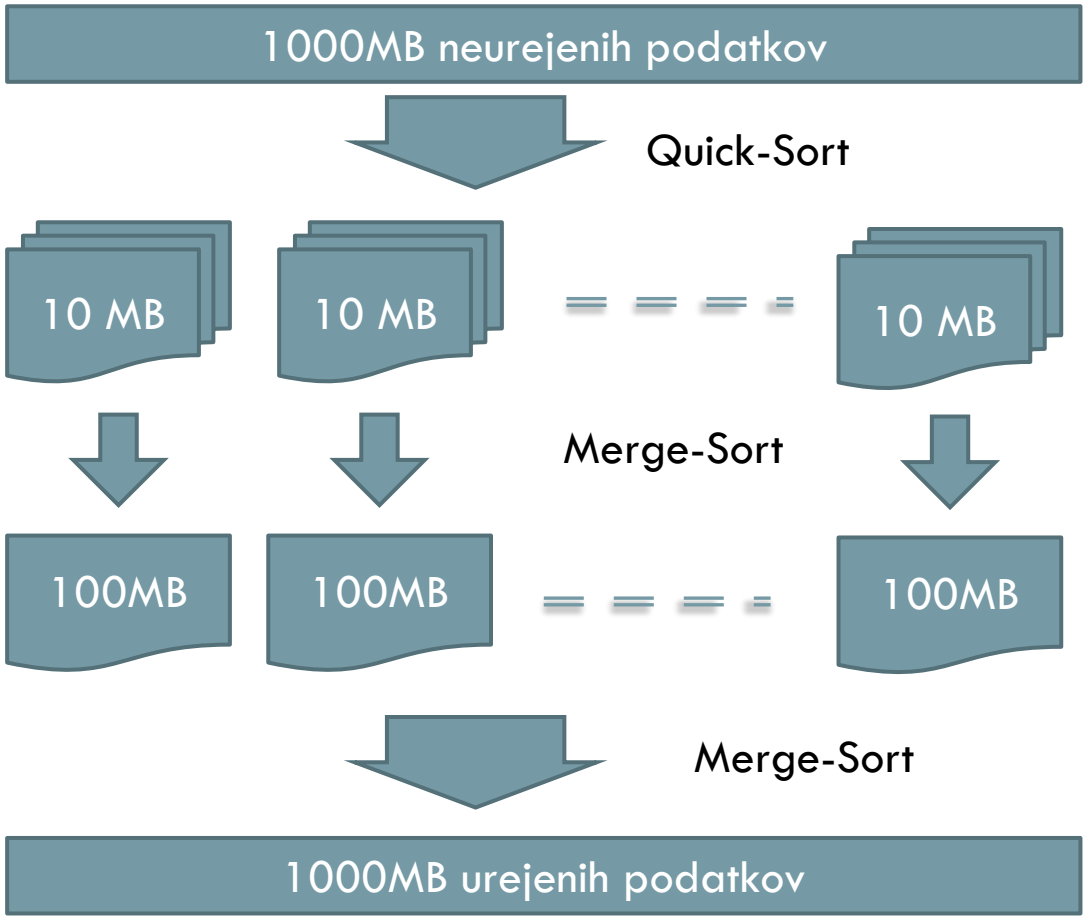
- Zakaj merge v dveh korakih?



Učinkovito urejanje velikih podatkov

- N = 1000 MB
- M = 10 MB
- B = 1MB

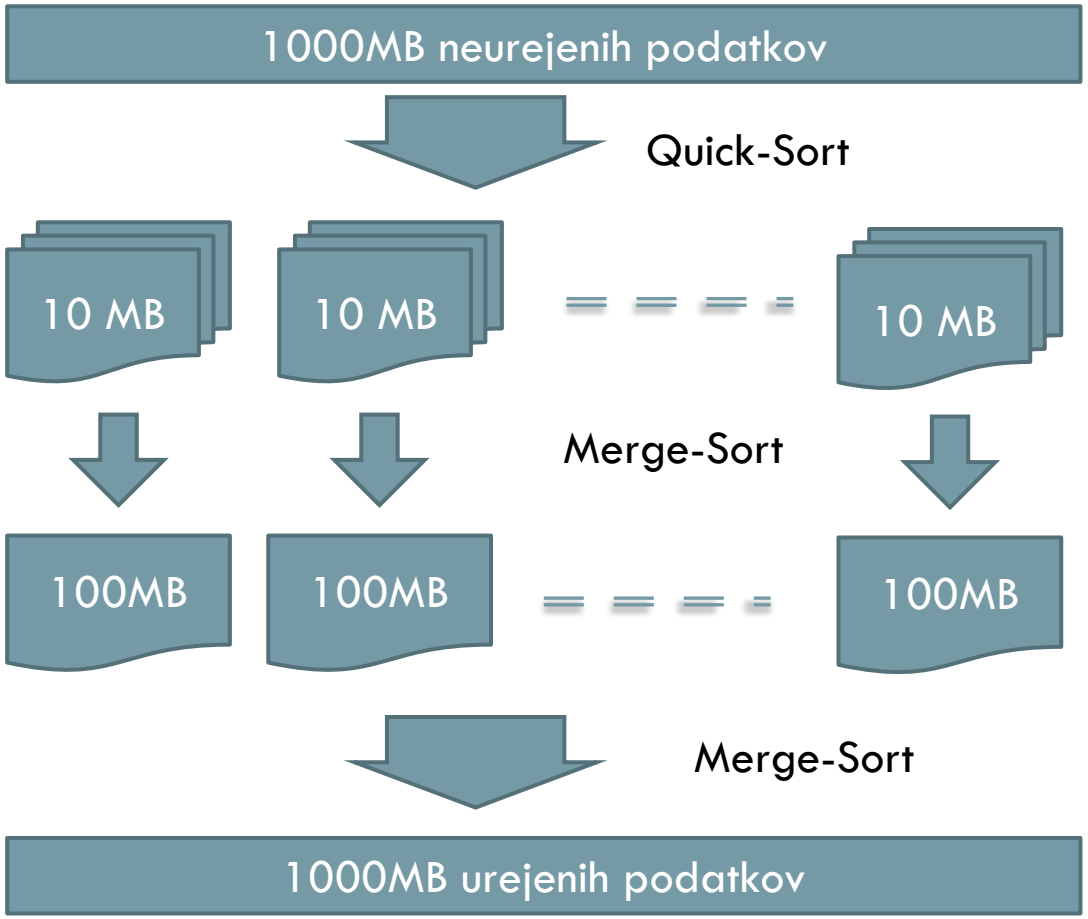
- Zakaj merge v dveh korakih?
  - Vedno beremo 1 blok!



Učinkovito urejanje velikih podatkov

- N = 1000 MB
- M = 10 MB
- B = 1MB

- Koliko IO operacij?



Učinkovito urejanje velikih podatkov

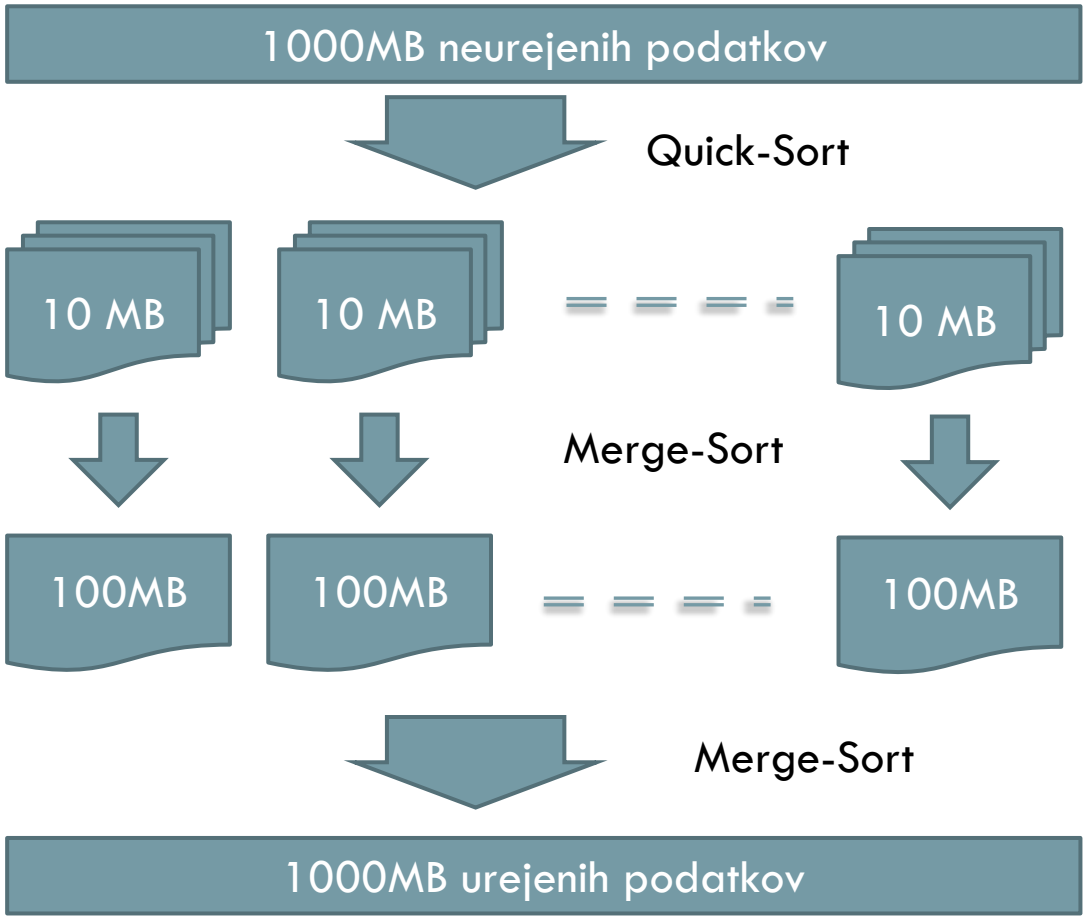
- N = 1000 MB
- M = 10 MB
- B = 1MB

- Koliko IO operacij?

$$O\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$$

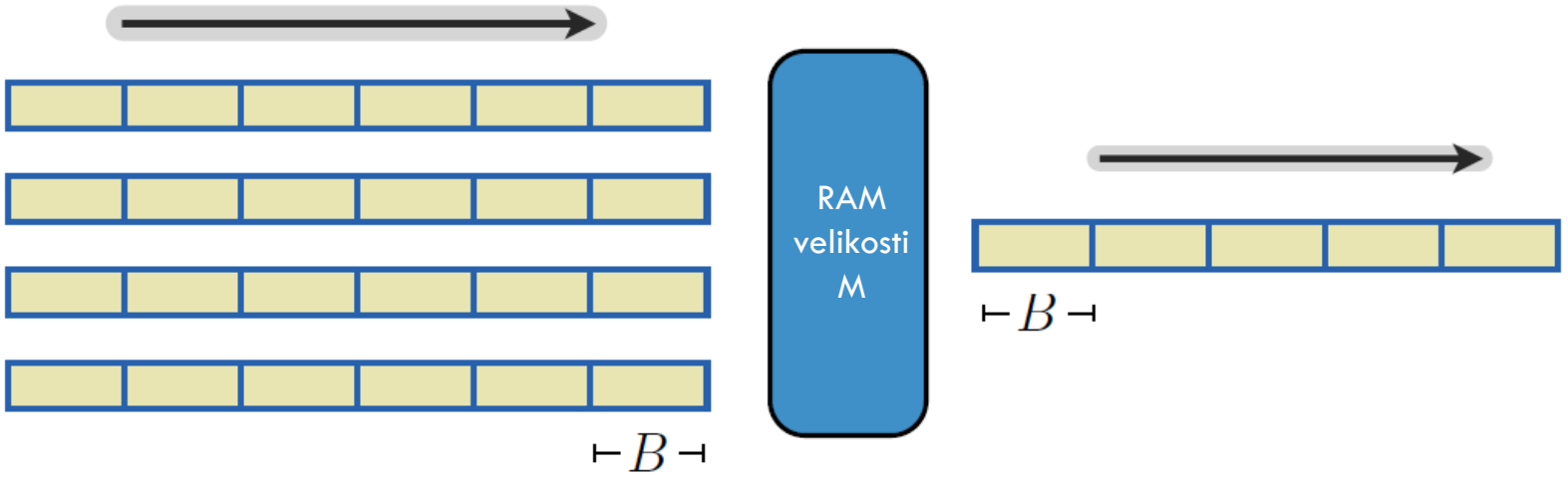
Cena prehoda

Število prehodov



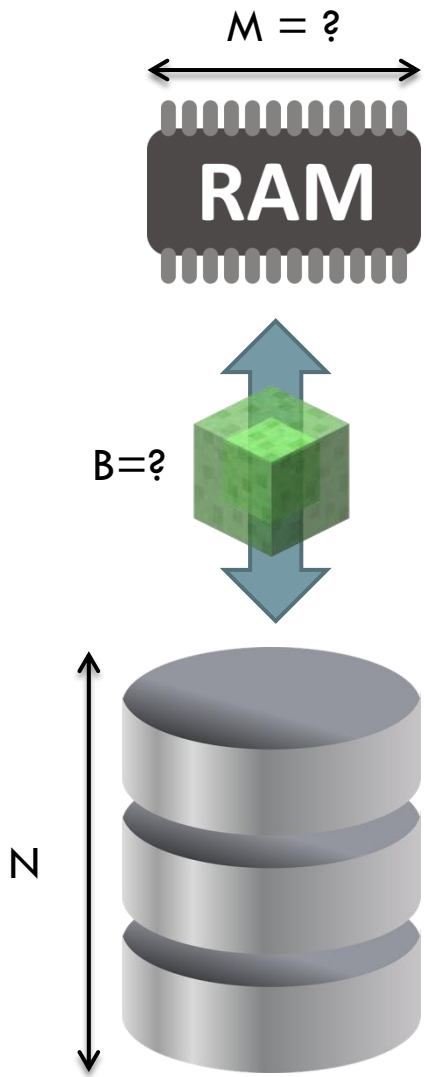
Učinkovito urejanje velikih podatkov

1. Uredi bloke velikosti M
2. Ustvari M/B tokov
3. Zlij tokove



Predpomnilniško nezavedni algoritmi

- Algoritem v naprej ne pozna parametrov B in M
- Cilj enak kot prej:
  - minimizirati število prenosov
- Optimizacija za vse možne M in B
  - Seveda pa ne najhitreje pri vseh M in B
  - Konsistentnost
- Je naš algoritem sortiranja predpomnilnoško zaveden?



Predpomnilniško nezavedni algoritmi

- Ang: Cache-oblivious algorithm
- Def: Algoritmi, ki izkoriščajo pomnilniško hierarhijo brez eksplicitnega znanja o velikosti pomnilnika.

- PRIMER: Transponirana matrika:
  - A velikosti n×m in B velikosti m×n
  - Kako izvedemo operacijo:

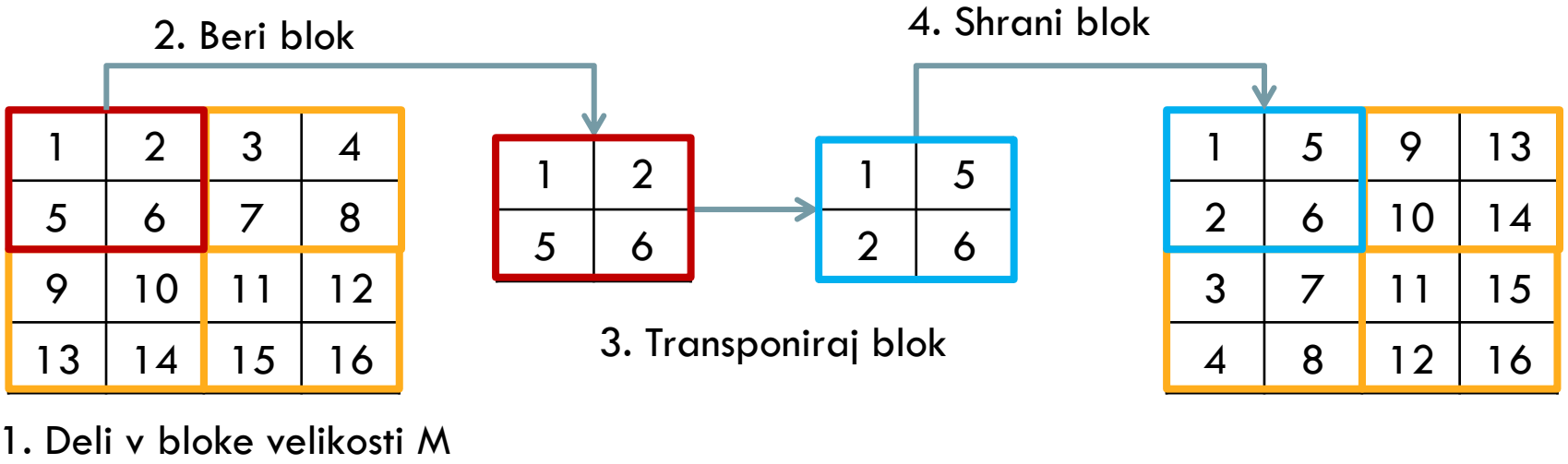
    $B = A^T$?
  - It's all about divide and conquer!

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

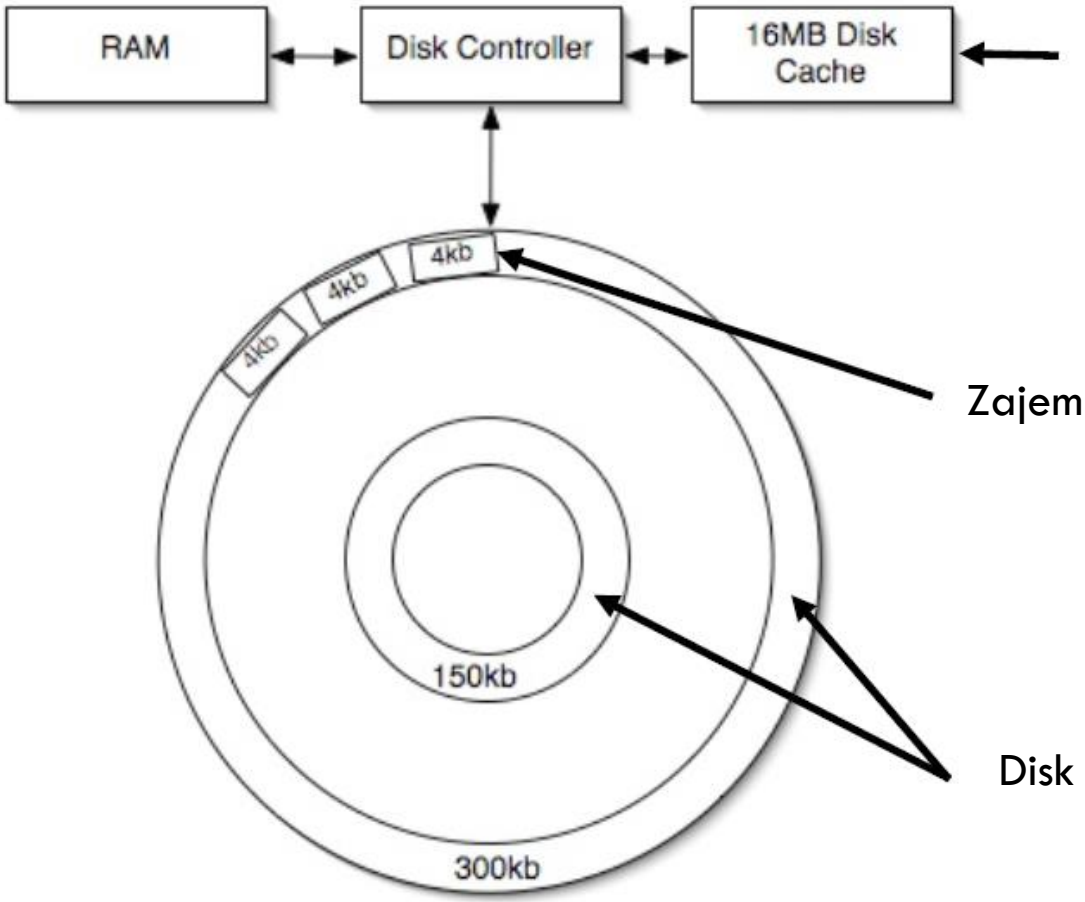
Predpomnilniško nezavedni algoritmi

- Učinkovit pomnilniško zaveden pristop:



- Če to idejo implementiramo rekurzivno, bomo vedno prišli do bloka velikosti M
  - Vse nadaljnje rekurzije ne zahtevajo več branja!

Resnično življenje



Disk zajame sledi v predpomnilnik - okoli 100 sledi

Zajem bloka konstantne velikosti

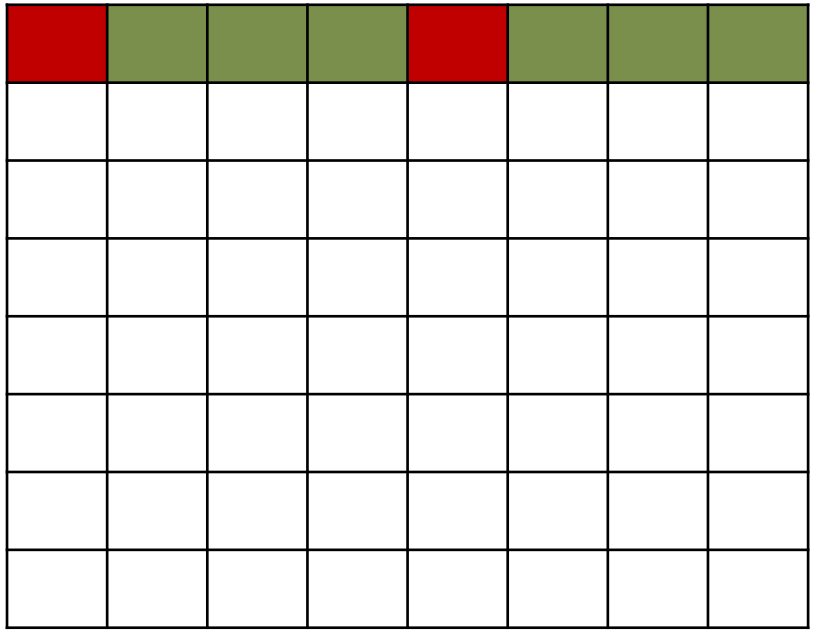
Disk je organiziran v sledi

Branje zaporednih blokov je cca. 10x hitrejše!

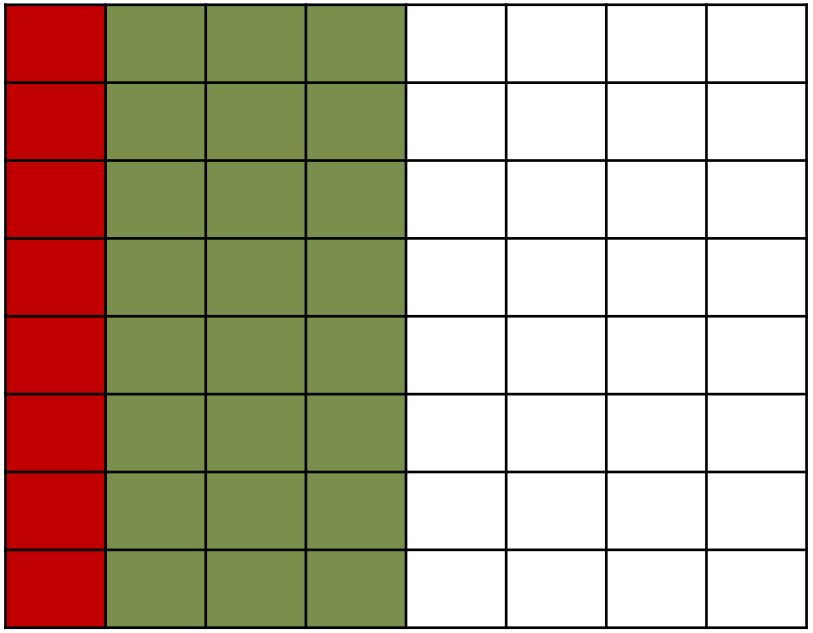
Resnično življenje

Množenje matrik

A

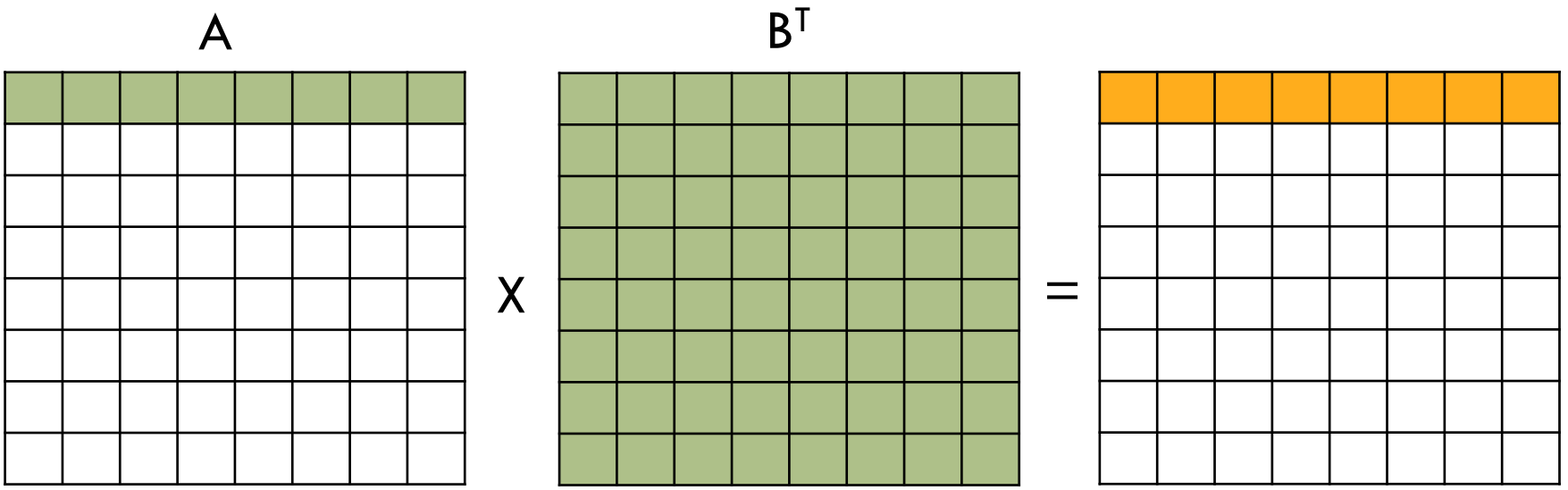


B



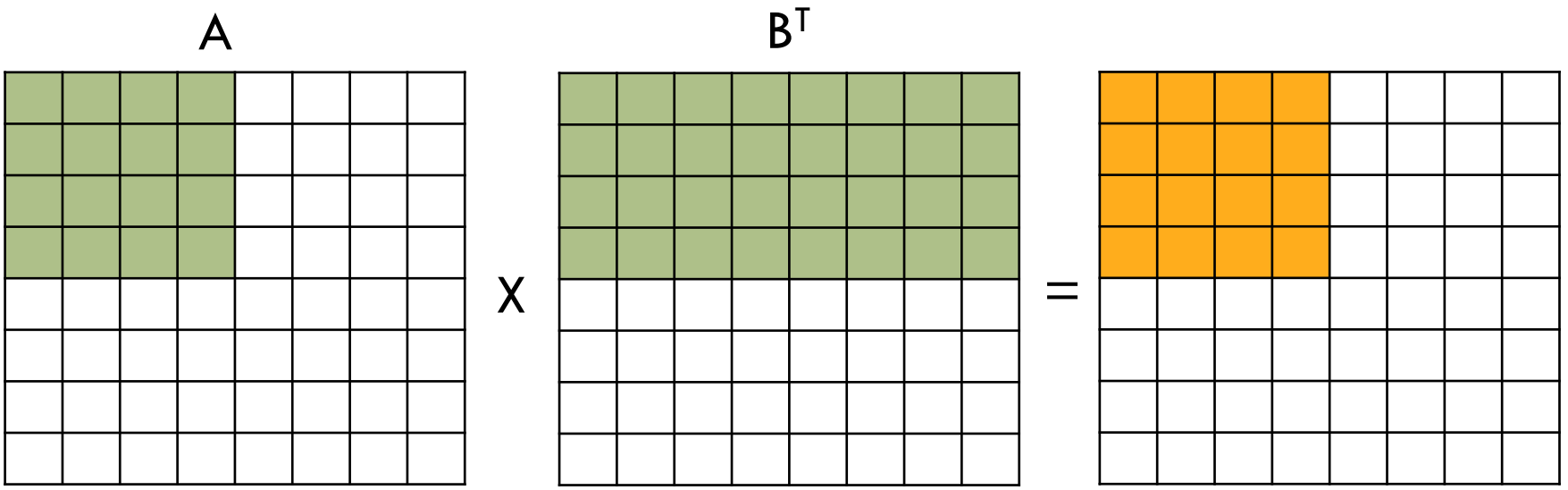
Resnično življenje

Množenje matrik (transponiran B)



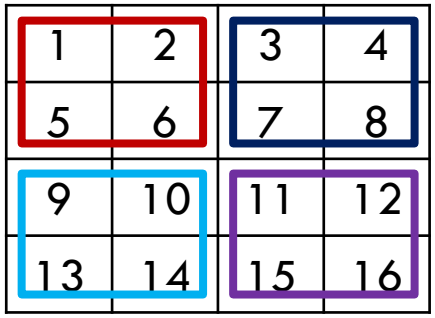
Resnično življenje

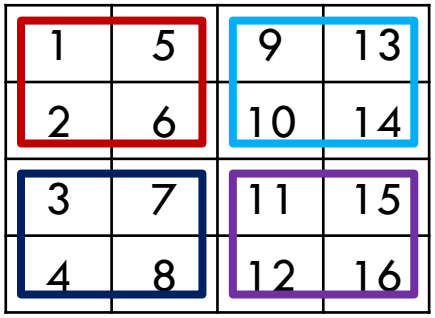
Množenje matrik (transponiran B)

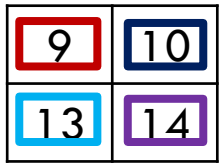


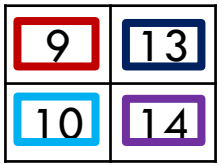
Predpomnilniško nezavedni algoritmi

- Pomnilniško nezaveden pristop:
  - PREDPOSTAVKE PODATKOVNE STRUKTURE:
    - Hrani toliko podatkov, kot jih lahko
    - Ko dostopamo do vrednosti, ki je ne hrani, prebere blok



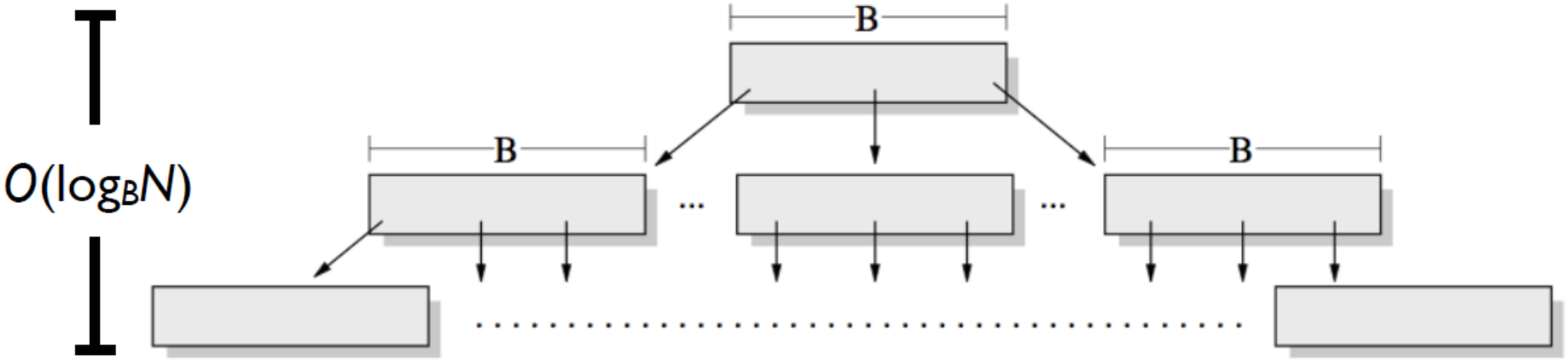






Koliko IO operacij je potrebno za iskanje elementa?

- PRIMER: Binarno drevo
  - Ne glede na velikost bloka, stvar deluje enako!



Lijačno (funnel) zlivanje

- Definicija strukture zlivanja, ki deluje učinkovito neglede na M
  - *k*-lijak – podatkovna struktura, ki zlije k urejenih vhodnih tokov
- Rekurzivno zlivanje: N^(1/3) tokov z N^(2/3) elementov