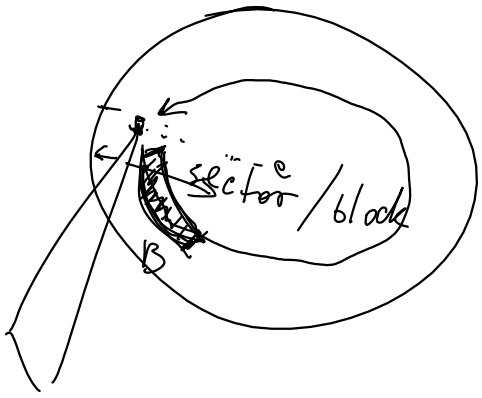
External Memory Algorithms

Cache-efficient Algorithms

nVM

NVMe

seek time



Disk Access Memory (DAM) model

External memory Model

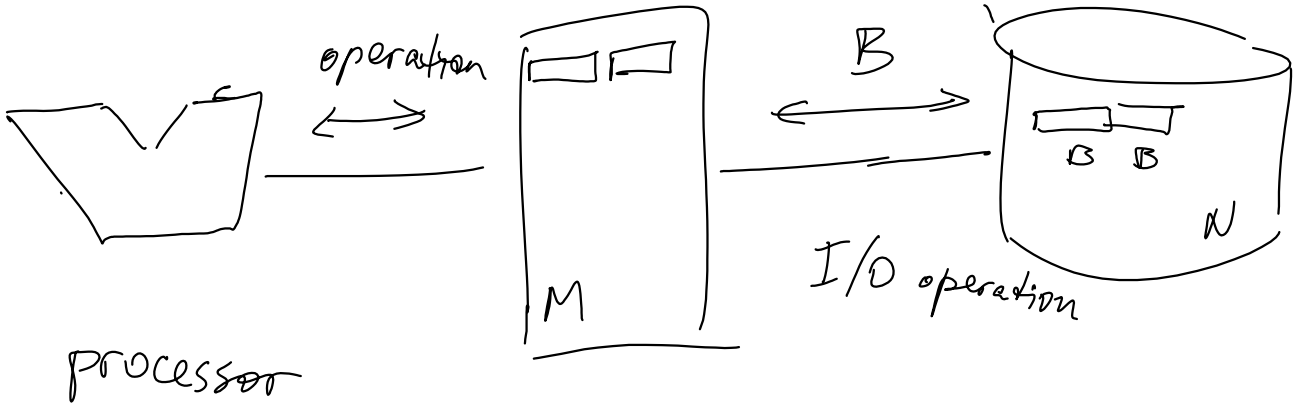
I/O Model

EM mode

$N >> M$

fast

slow

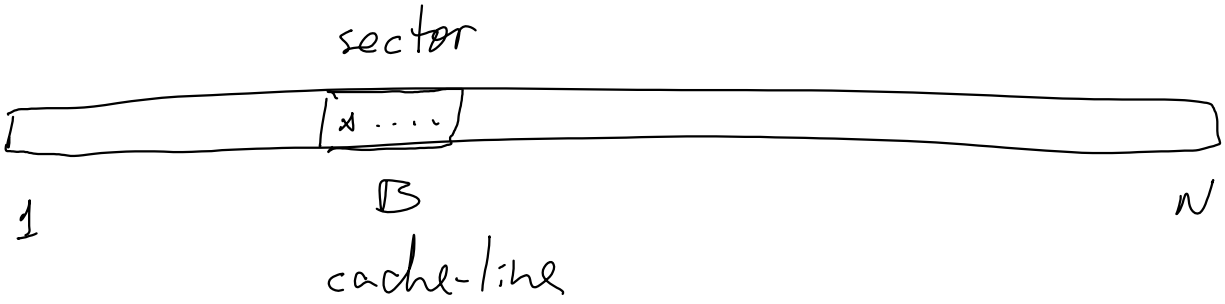


internal memory

External memory

cache

DRAM



E.g. On harddrive $B \simeq 4kB$ $M \simeq 4GB$

cache line $B \simeq 64B$ $M \simeq 4\text{-}8MB$

Complexity Metrics

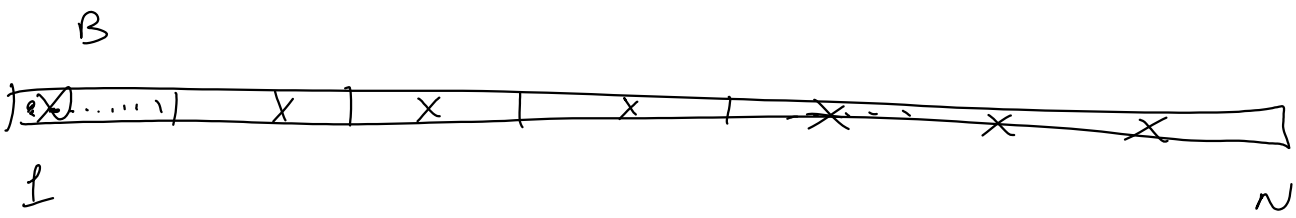
only I/O-compexity = # of I/Os

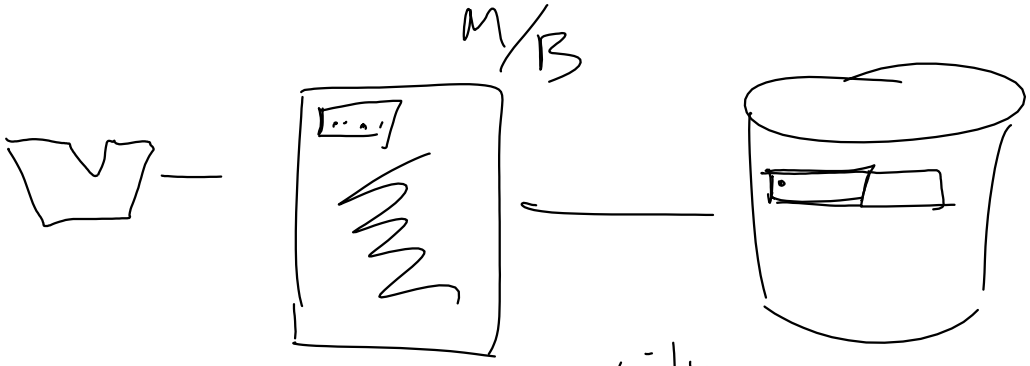
(ignore actual operations)

```
for i = 1 to N
    a[i] = a[i] + 1
```

$O(n)$ time/operations

$scan(N) = O\left(\frac{n}{B}\right)$ I/Os

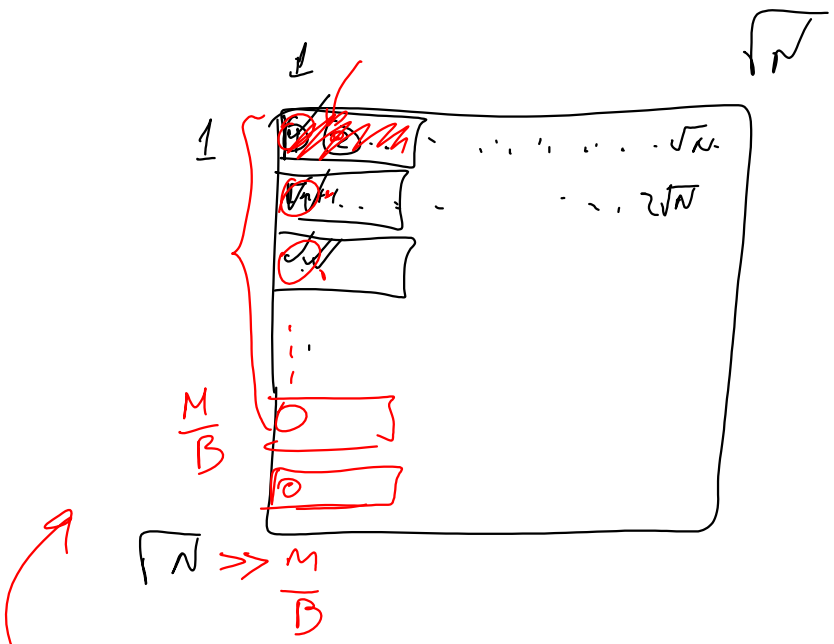




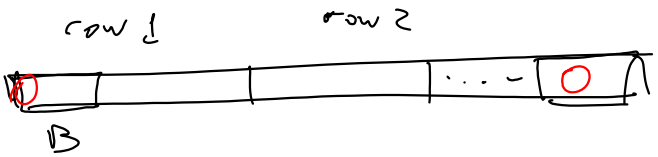
Reading B items ⇒ 1 I/O

writing B items ⇒ 1 I/O

Matrix Transpose

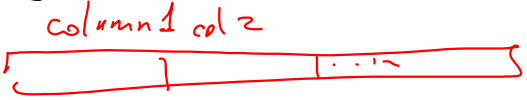


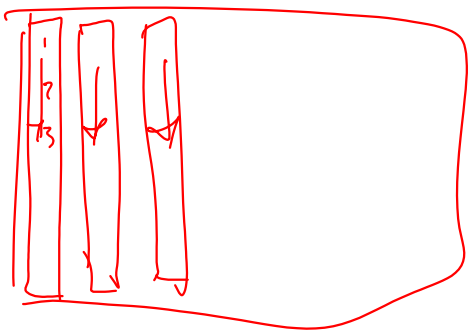
$\sqrt{N} >> \frac{M}{B}$



row-major layout

column-major layout






```
for i = 1 to √N          (i: row)
    for j = 1 to √N      (j: column)
        A[i][j]++
```

row column

$O(n)$ time

$O\left(\frac{n}{B}\right)$ I/Os

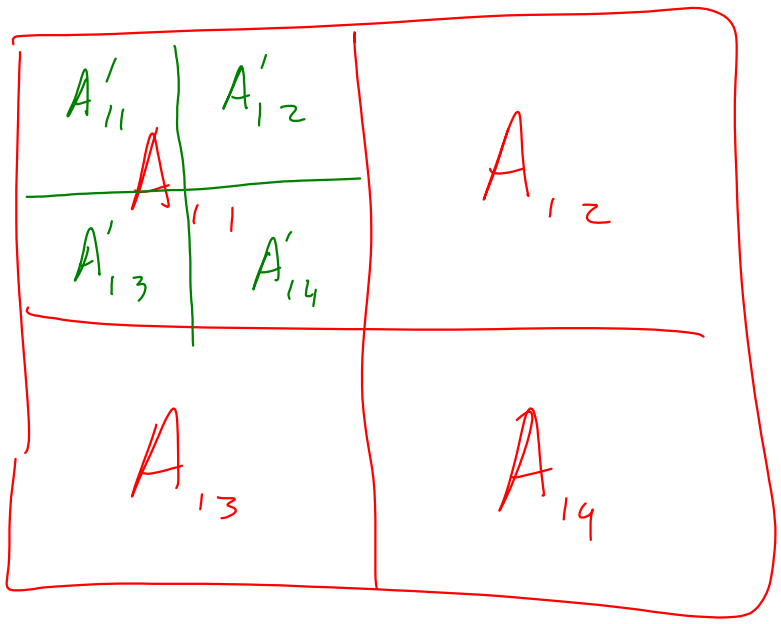
```
for row = 1 to √n
    for col = 1 to √n
        A[col][row]++
```

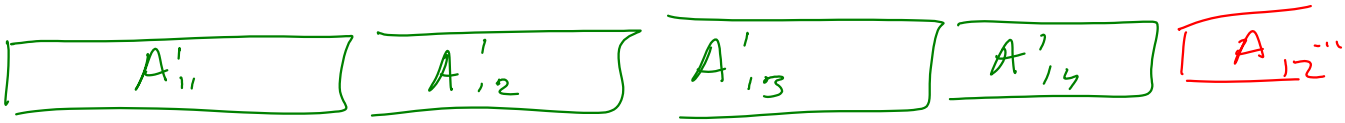
$O(n)$ time

$O(n)$ I/Os if $\sqrt{n} > \frac{n}{B}$

$O\left(\frac{n}{B}\right)$ I/Os if $\sqrt{n} \leq \frac{n}{B}$

Morton Z-order

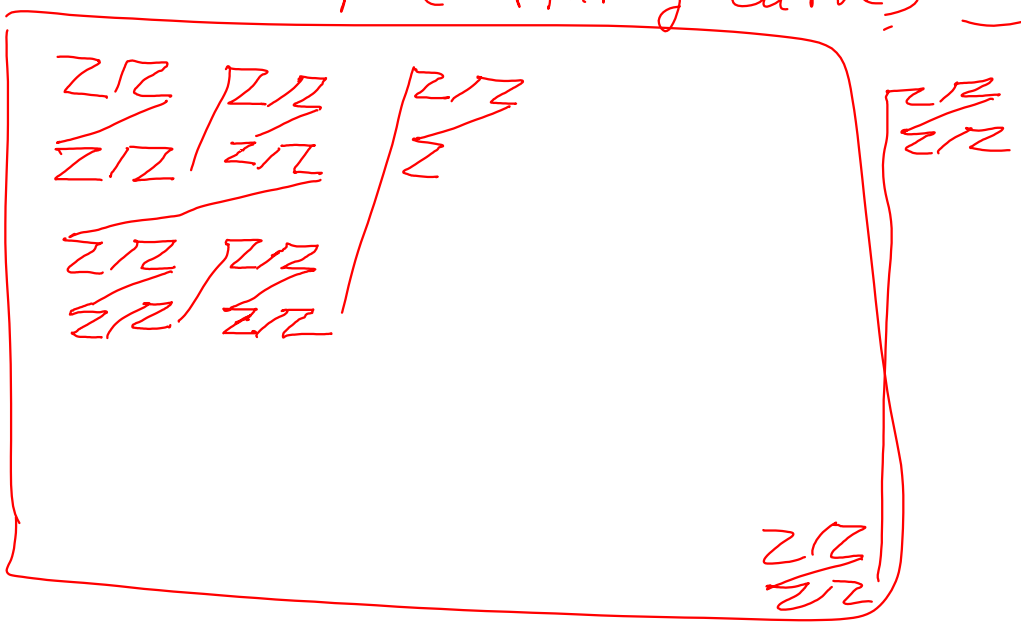




$A[i][j]$ $\quad addr = \cancel{offset} + i \cdot \sqrt{n} + j$

$A[\ A + i\sqrt{n} + j\ ]$

Morton Z-order
Space-filling curves



Matrix transposition

$A =$

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

$\Rightarrow$

a	e	i	m
b	f	j	n
c	g	k	o
d	h	l	p

$= A^T = C$

$\Rightarrow$

```
for i = 1 to √n
    for j = 1 to √n
        C[j,i] = A[i,j]
```

$O(n)$ → $O(\frac{n}{B})$

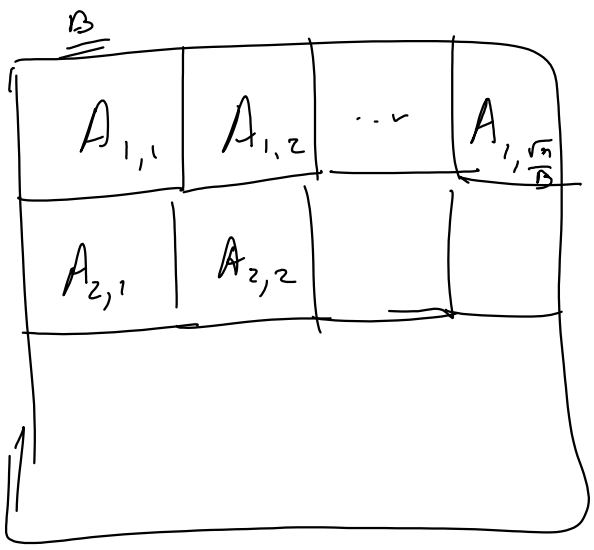
$\Rightarrow O(n) + O(\frac{n}{B}) = O(n)$ I/Os

$M \geq B^2$

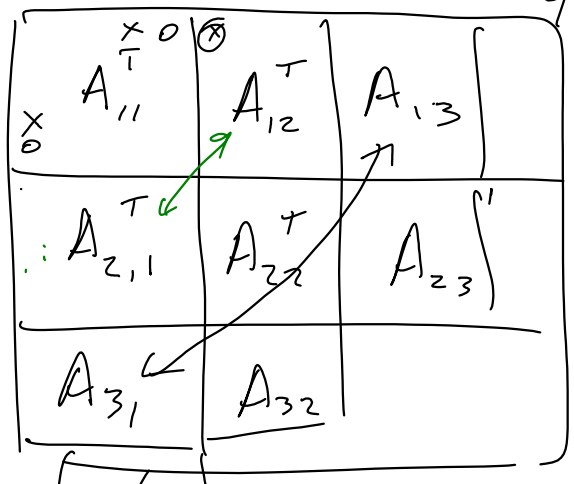
# of submatrices $= \frac{n}{n'} = \frac{n}{B^2}$

each takes B I/Os

A

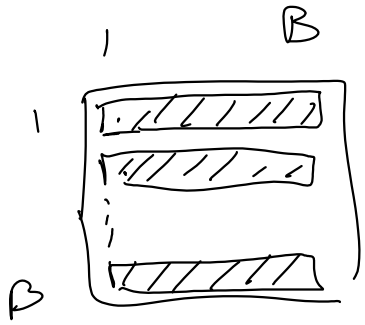


$\Rightarrow$



$\neq A^T$

$\Rightarrow O\left(\frac{n}{B^2} \cdot B\right) = O\left(\frac{n}{B}\right)$ I/Os



$n' = B^2$ elements

```
for i = 1 to B
    for j = 1 to B
        A^T[j][i] = A[i][j]
```

$O(B)$ I/Os.

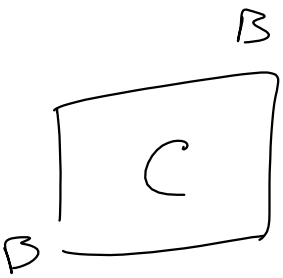
$\Rightarrow O\left(\frac{B^2}{B}\right)$ I/Os

$\Rightarrow O\left(\frac{n'}{B}\right)$ I/Os

for $i = 1$ to $\frac{\sqrt{n}}{B}$

for $j = 1$ to $\frac{\sqrt{n}}{B}$

$A_{ji} \Leftrightarrow A_{ij}$



copy (A, C):

for $i = 1$ to B

for $j = 1$ to B

$C[i][j] = A[i][j]$

$$O\left(\frac{B^2}{B}\right) = O(B)$$

$$\Rightarrow O\left(\frac{|A|}{B}\right) = O\left(\frac{n}{B}\right)$$

# of submatrices $= \frac{n}{n'}$

Total I/Os for copying $= O\left(\frac{n}{n'}\right) O\left(\frac{n'}{B}\right) = O\left(\frac{n}{B}\right)$

---

without any assumptions

to transpose $k \times m$ matrix

takes $\Theta\left(\frac{N}{B} \cdot \log_{\min\{k, m, M, \frac{N}{B}\}}\right)$

Cache-oblivious Algorithms.

Q_{opt} - I/O complexity of an algorithm under optimal cache-replacement policy

LRU = least recently used

Q_{LRU} - I/O complexity of an algorithm under LRU replacement policy

Theorem: $Q_{LRU}(N, M, B) \leq$

$\leq 2 \cdot Q(N, \underline{2M}, B)$

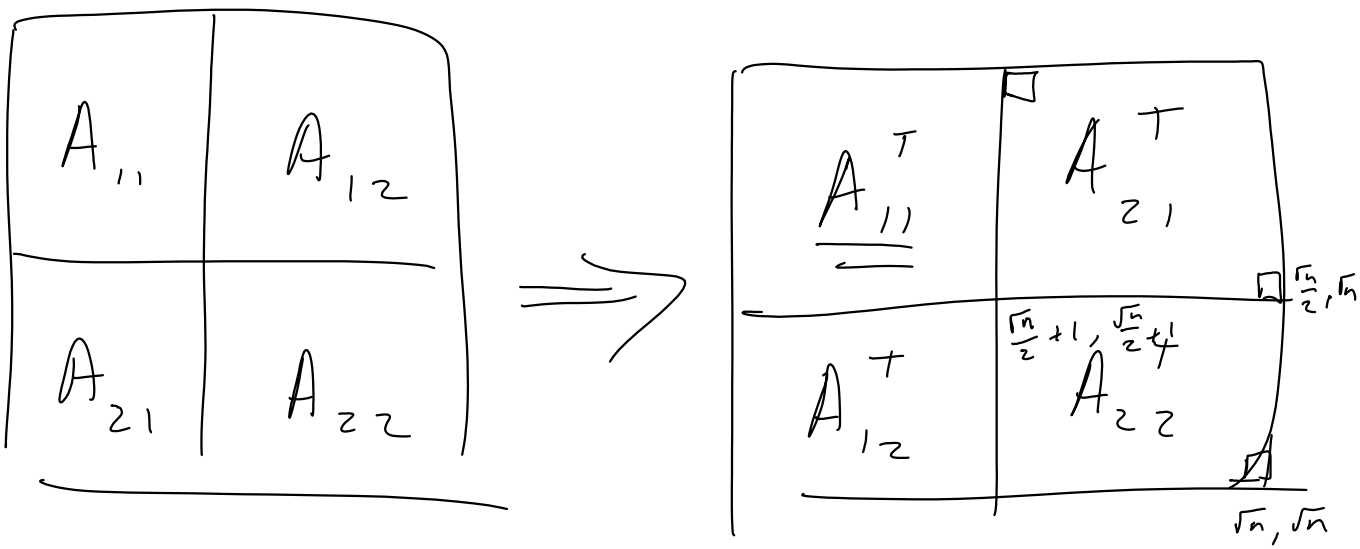
Flush-when-full policy

Cache-Oblivious Matrix transposition.

$M \geq B^2$

A

$C = A^T$



Each submatrix is transposed recursively

Transpose$(A, 1, 1, \sqrt{n}, \sqrt{n})$

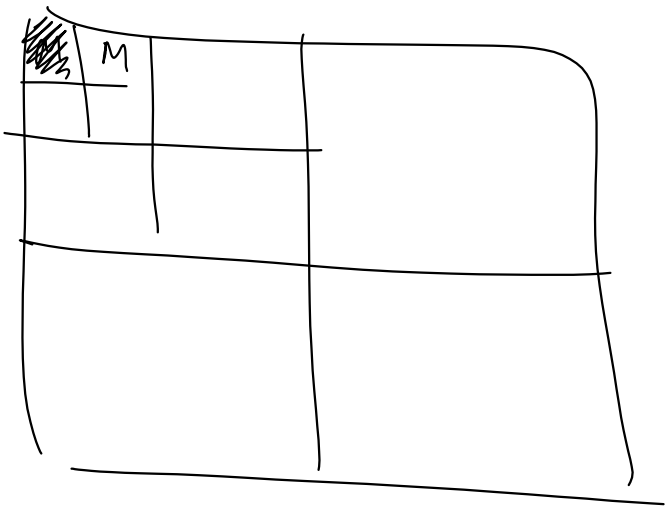
Transpose $(A, 1, 1, \frac{\sqrt{n}}{2}, \frac{\sqrt{n}}{2})$

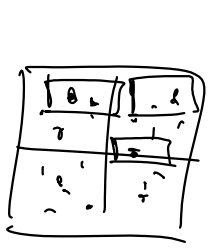
Transpose $(A, 1, \frac{\sqrt{n}}{2}+1, \frac{\sqrt{n}}{2}, \sqrt{n})$

$\vdots$

$\vdots$

$$Q(n, M, B) = \begin{cases} 4Q\left(\frac{n}{4}, M, B\right) + O\left(\frac{n}{B}\right) \\ O(1) \quad \text{if } n \leq 1 \end{cases}$$

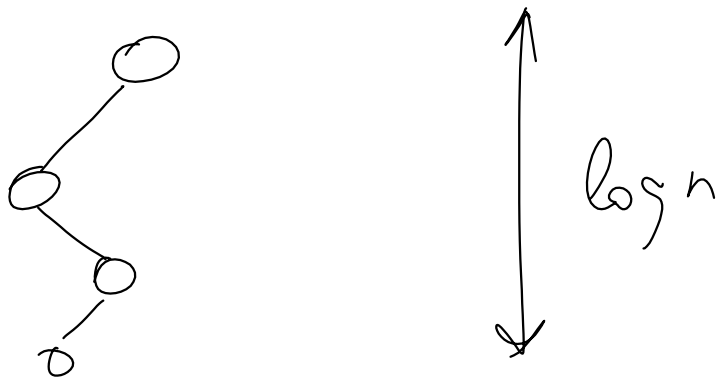




$$Q(n) = \begin{cases} 4Q\left(\frac{n}{4}\right) + O\left(\frac{n}{B}\right) & \text{if } n > M \\ O\left(\frac{n}{B}\right) & \text{if } n \leq M \end{cases}$$

$$\Rightarrow O\left(\frac{n}{B}\right) \qquad M \geq B^2$$

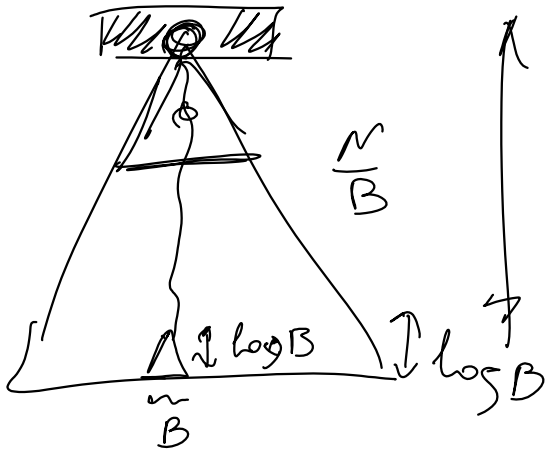
Binary Search



$$Q(n) = \begin{cases} Q\left(\frac{n}{2}\right) + O(1) & \text{if } n \geq 1 \\ O(1) & \text{if } n \leq B \end{cases}$$

$$= O\left(\log \frac{N}{B}\right) = O(\log N - \log B)$$

$=$



B-tree – a I/O-efficient search tree

$(2, 4)$-tree — each node has between 2 & 4 children

(a, b)-tree — each node ---- a & b children

$2 \le a \le \frac{b+1}{2}$

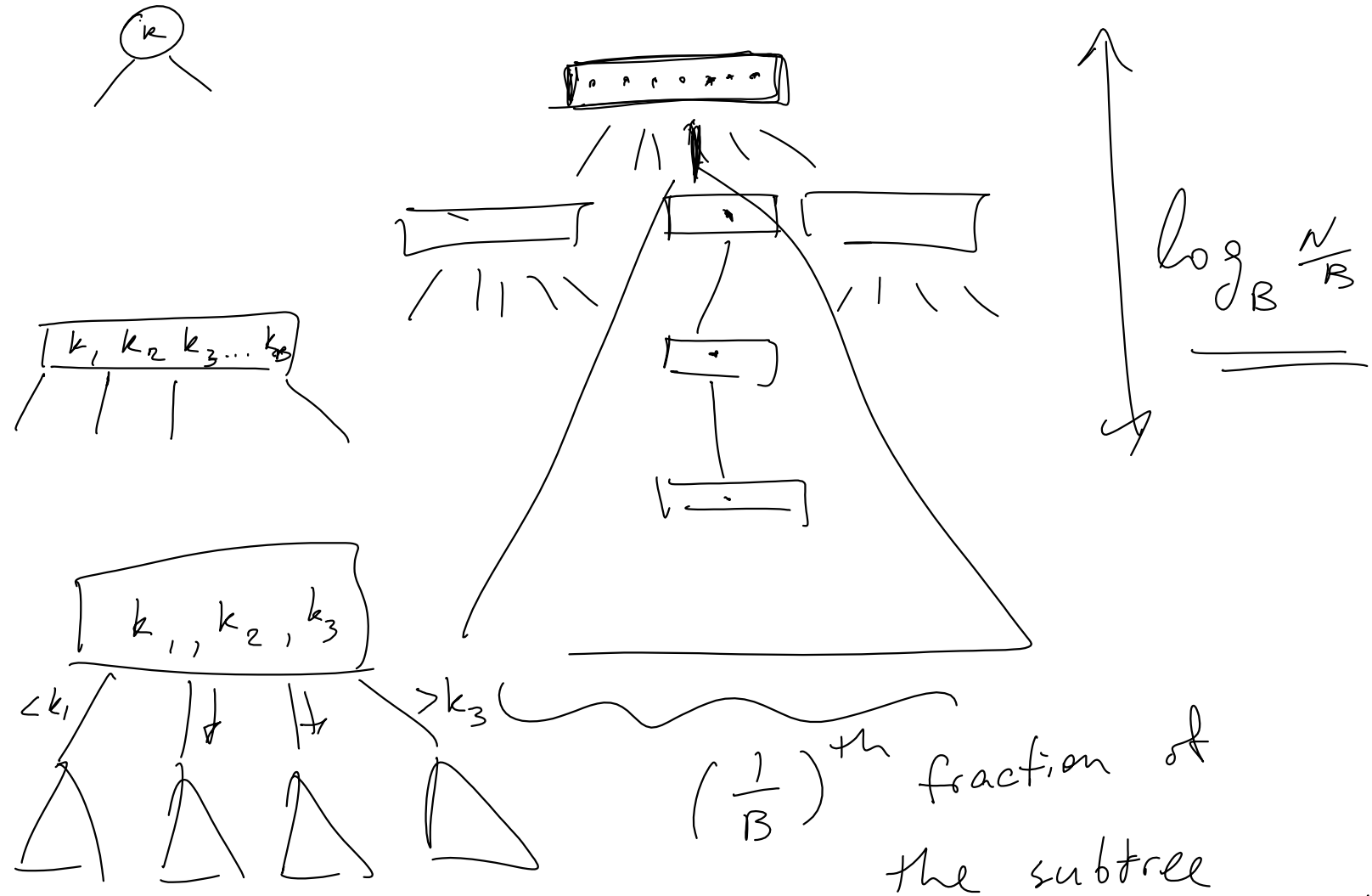
all leaves are on the same level

B-tree is an (a, b)-tree, where

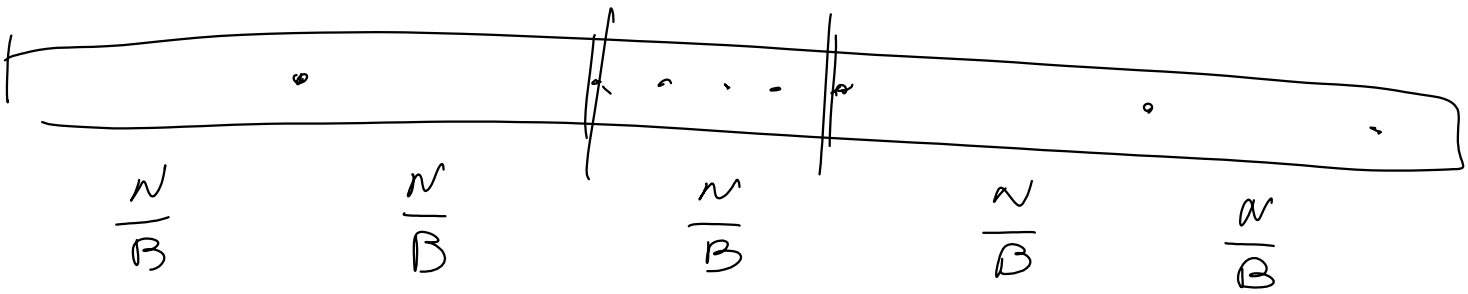
$a = \frac{B}{4}$ & $b = B$

$\left(\frac{B}{4}, B\right)$-tree

$\Rightarrow$ Each node has $\Theta(B)$ keys



$\left(\frac{1}{B}\right)^{th}$ fraction of the subtree rooted at parent

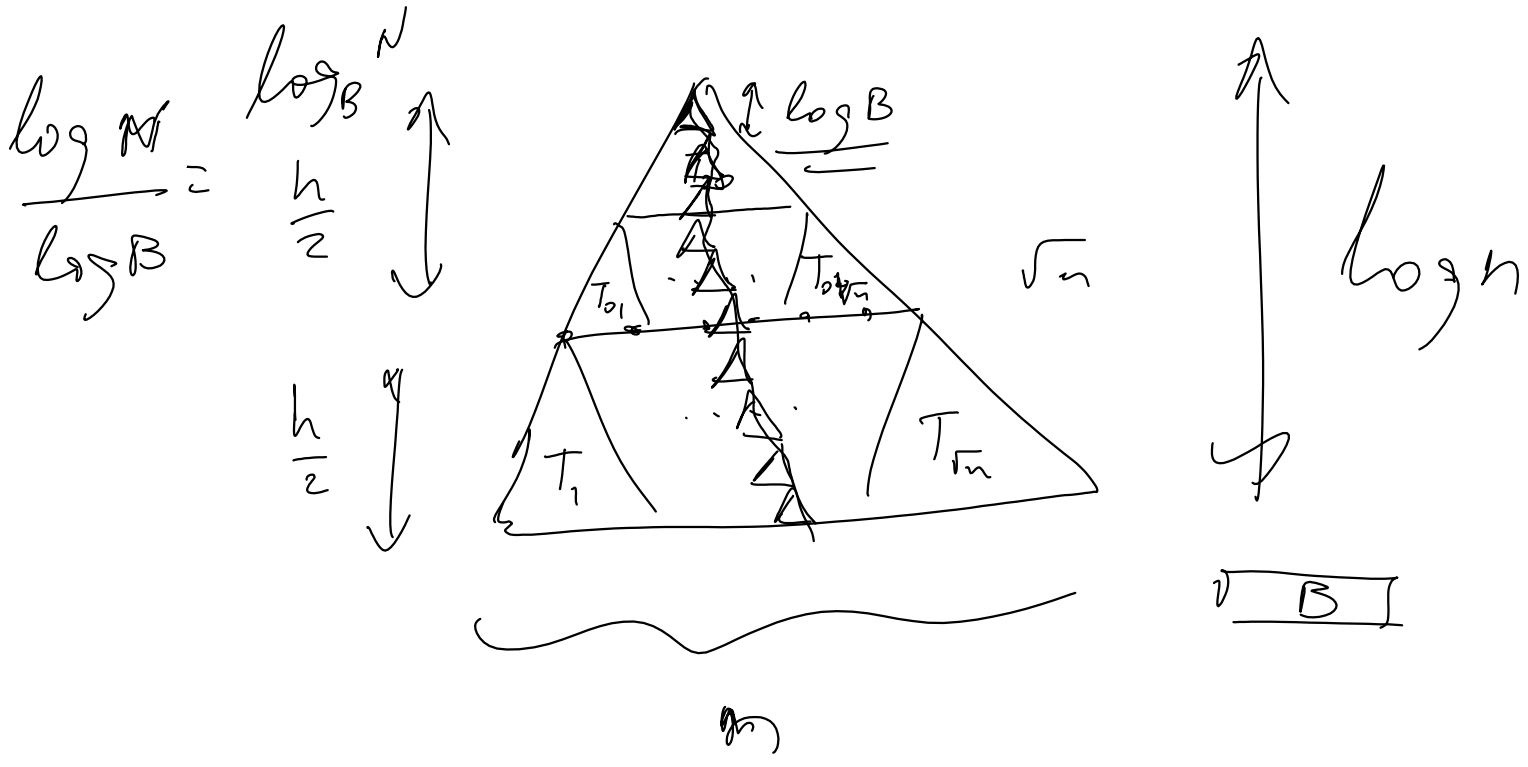


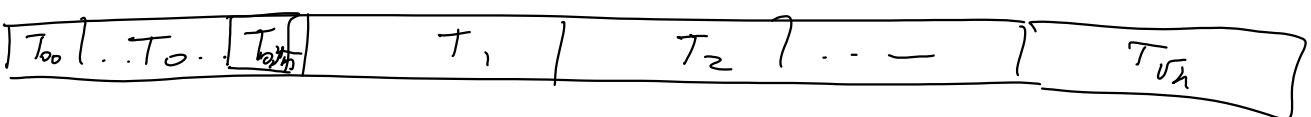
$$Q(N) = \begin{cases} Q\left(\frac{N}{B}\right) + \Theta(1) & \text{if } N > B \\ \Theta(1) & \text{if } N \le B \end{cases}$$

$$= \Theta\left(\log_B \frac{N}{B}\right).$$

van Emde Boas layout

$$\frac{\log N}{\log B} = \frac{\log_B N}{\frac{h}{2}}$$





$\Rightarrow O(\log_B N)$