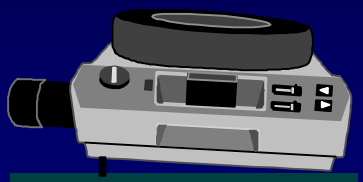
Indexed Files : Outline

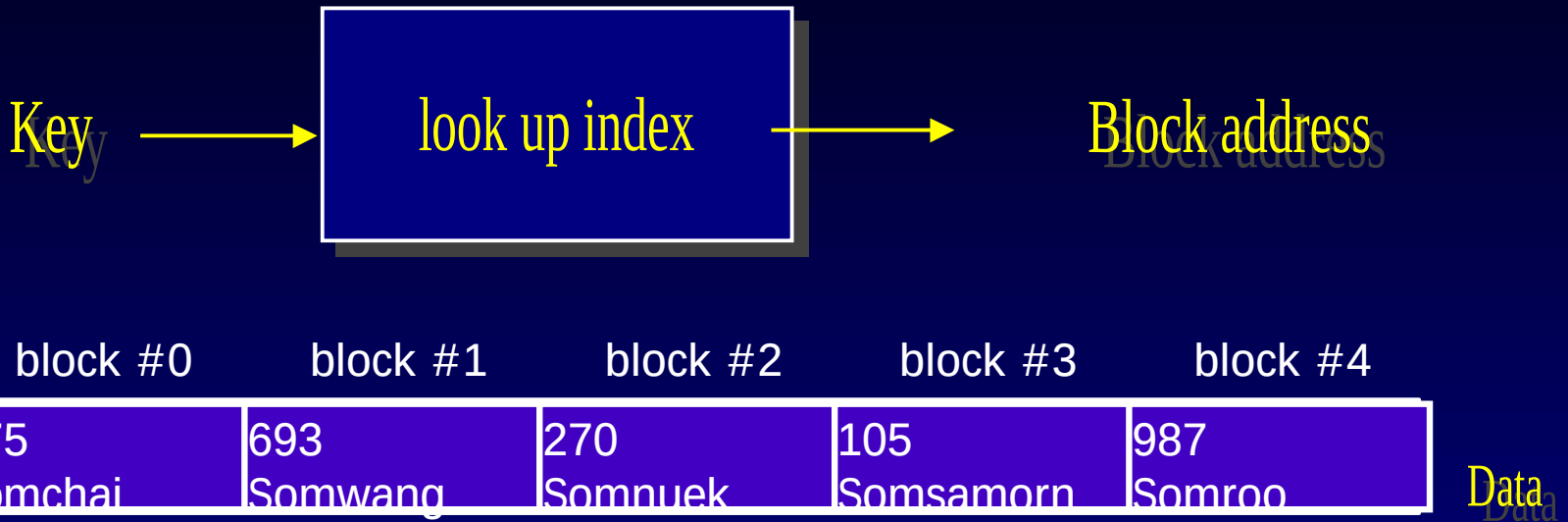
- Introduction
- Indexed Files
- Full Index Organization
- Indexed Sequential Files
- Multilevel Indexes
- Overflow Management
- Performance Analysis



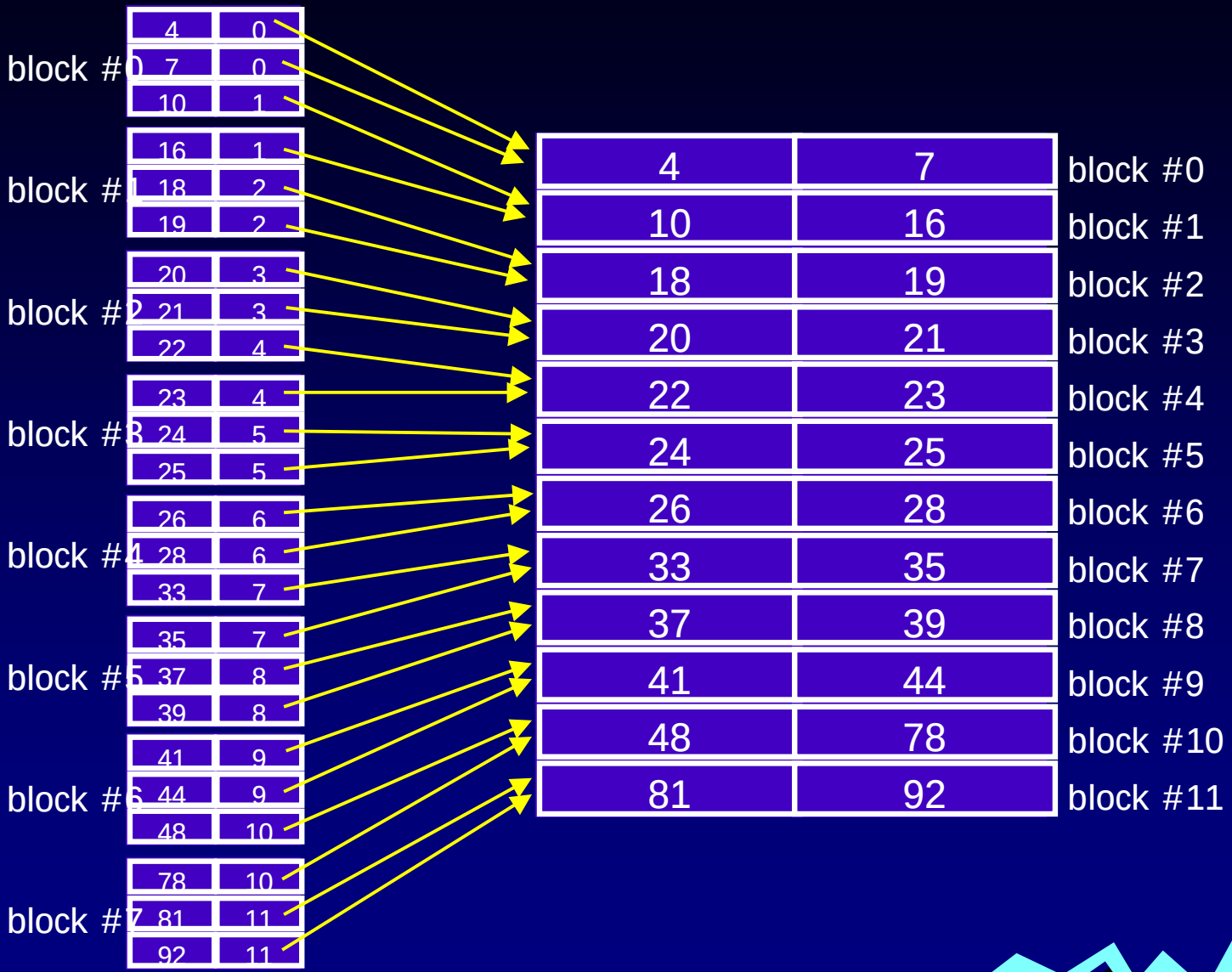
Indexed Files

	Ordered sequential processing	Random access
Sequential file structure	fast	slow
Direct file structure	slow	fast
Indexed file structure	fast	fast

Indexed file



Full Index Organization

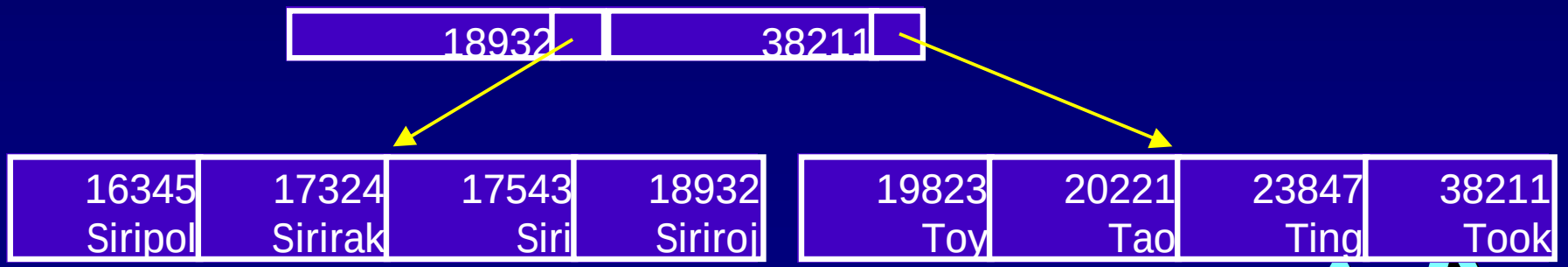


Indexed File Structure

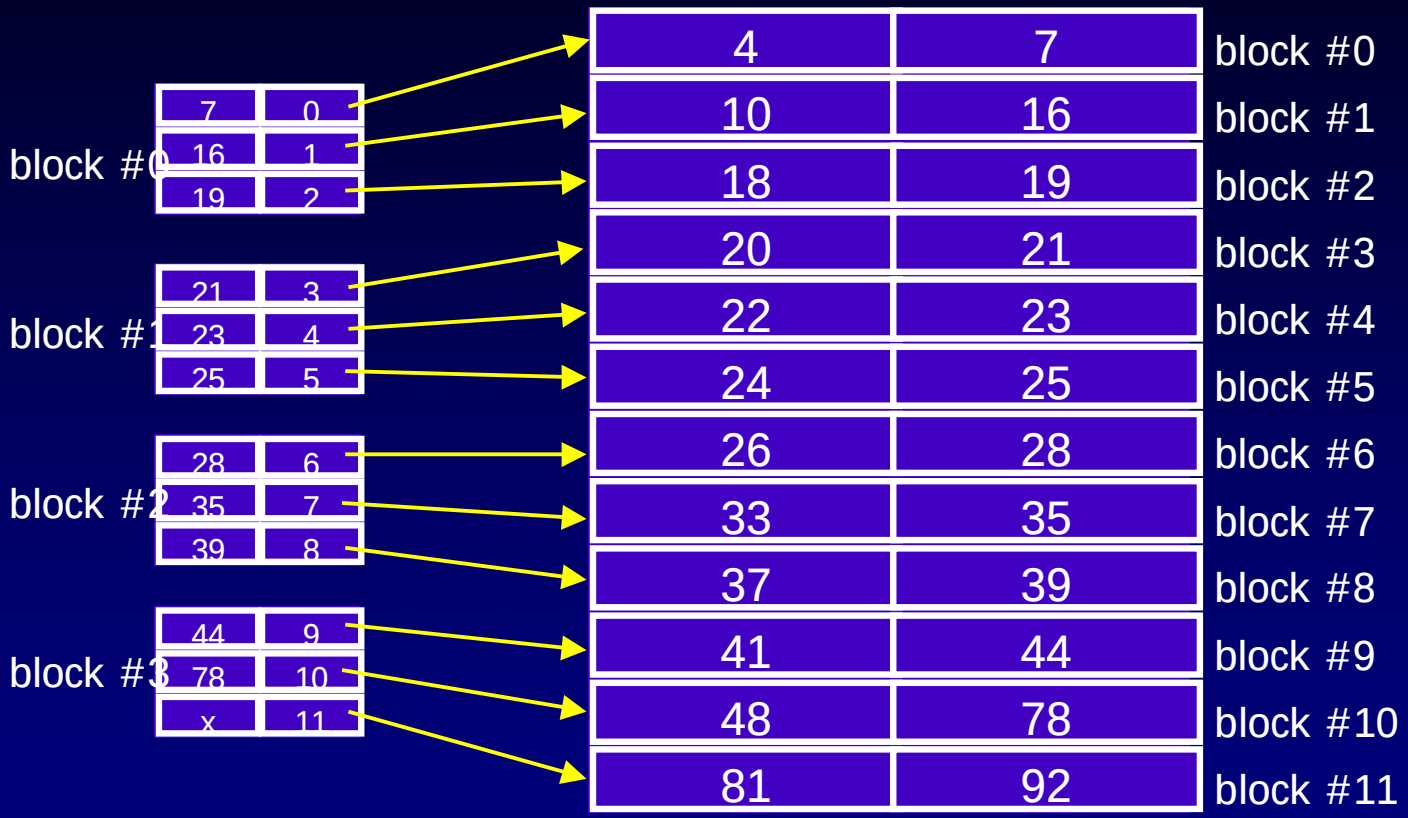
- Binary search for the target key in the index
- *RetrieveOne* : *SL*[*BinarySearch*] + 1 *rba*
- *RetrieveAll* : 1 *rba* + *m2 sba*
- *DeleteOne* : *SL*[*RetrieveOne*] + 2 *sba*
- *InsertOne* : *SL*[*RetrieveOne*] + 2 *sba*

Indexed Sequential Files

- If records in the data file are ordered,
  - ordered sequential is fast.
  - do not have to be full indexed
    (keep only max index value of the data block)
  - # indexes decreases, # index blocks decreases search length decreases, improve performance



Indexed Sequential File

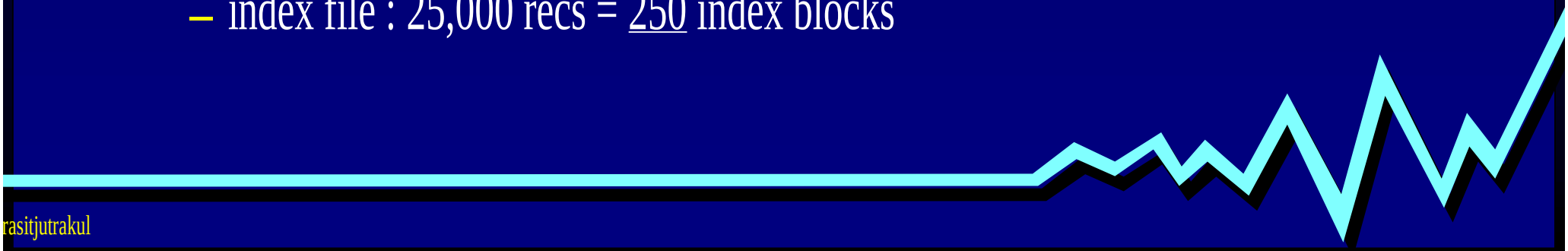


Indexed Sequential Files

- 100,000 records, each of size 500 bytes
- index record size = 20 bytes
- block size = 2000 bytes

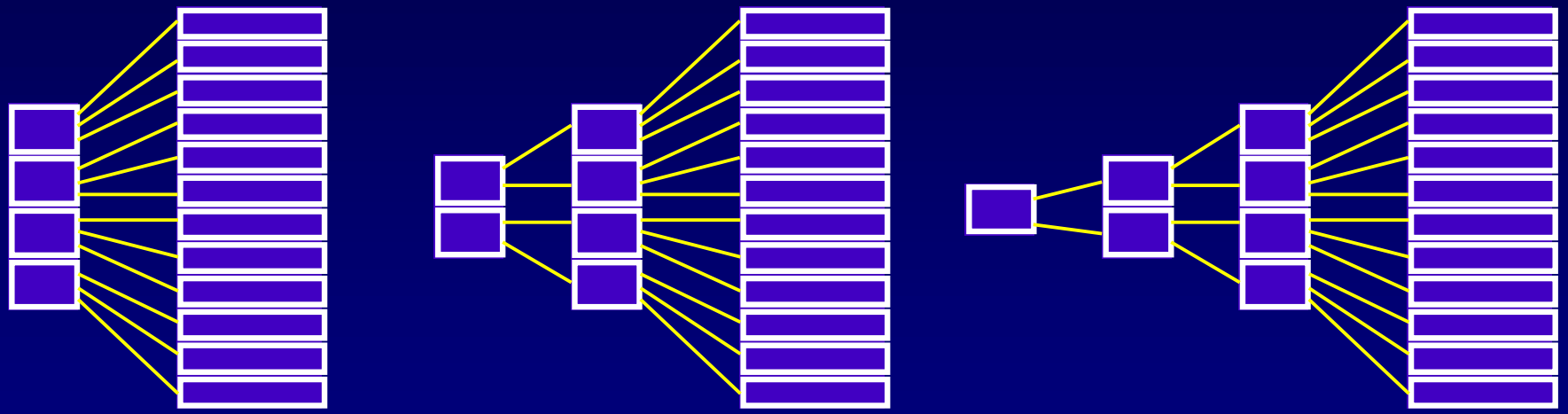
  1 block = 4 data recs, 1 block = 100 index recs.

  25,000 data blocks
- Full index :
  - index file : 100,000 recs = 1000 index blocks
- Indexed sequential :
  - index file : 25,000 recs = 250 index blocks

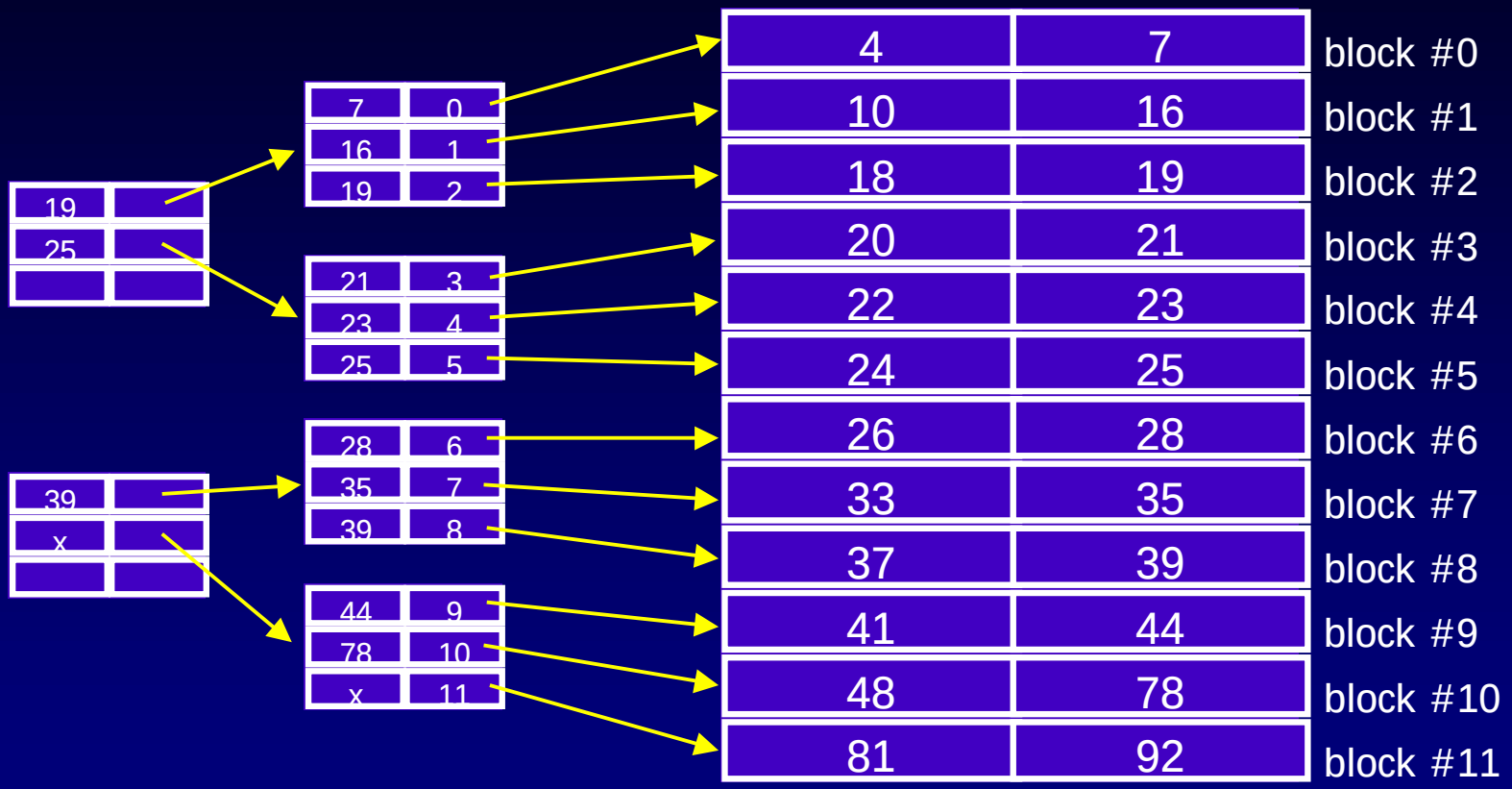


Multilevel Indexed Sequential

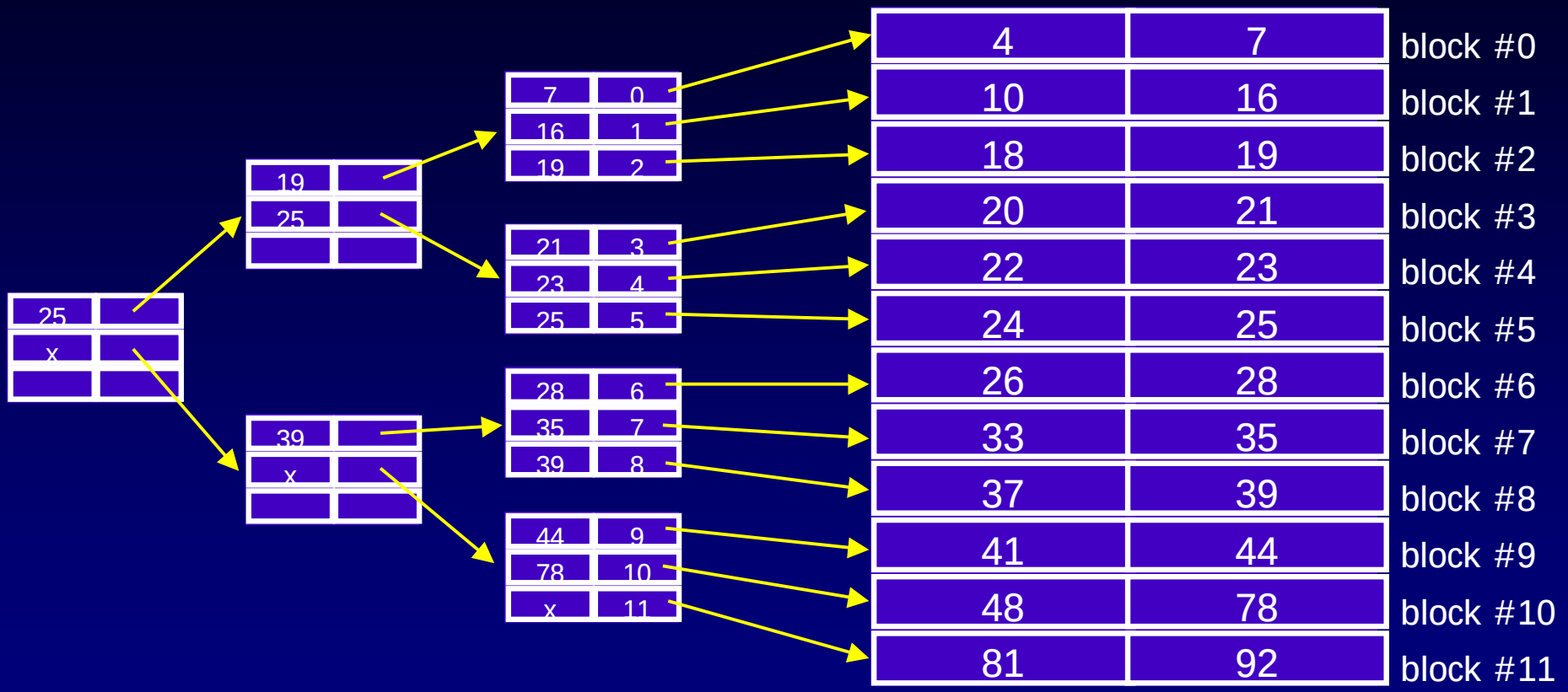
- Trimming search length = better performance
- Modify the logical structure of the index file



Indexed Sequential File : 2 levels



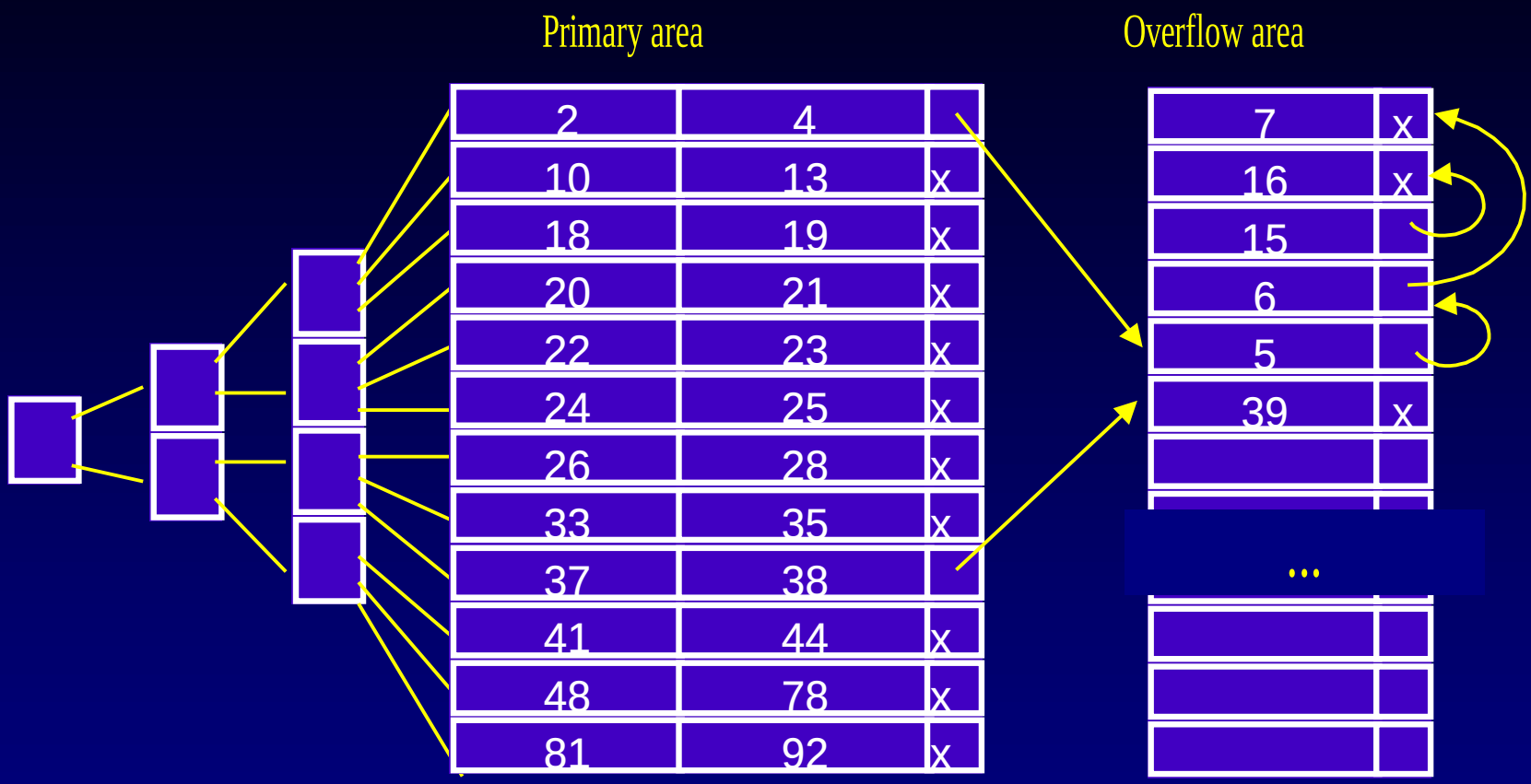
Indexed Sequential File : 3 levels



Overflow Records

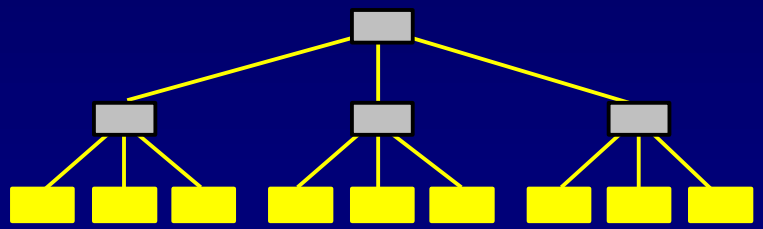
- Insertion generates overflow records
- Allocate empty slots for each blocks
- Reorganizing the file if needed
- Allocate extra overflow blocks (overflow area)
  - overflow records are linked in a logical, ordered, chained fashion with the primary block to which they belongs
  - overflow recorded are not blocked

Overflow Records



Performance Analysis

- Number of *rba*'s needed to retrieve a target depends the height of the index tree.
- The height depends on the *NBLK* and *BF* of index.
- Let *k* be the avg. # of indexes per index block
- Let the index tree be a *h* level tree.



$$NBLK_{index} = 1 + k + k^2 + \cdots + k^{h-1} = \frac{k^h - 1}{k - 1}$$

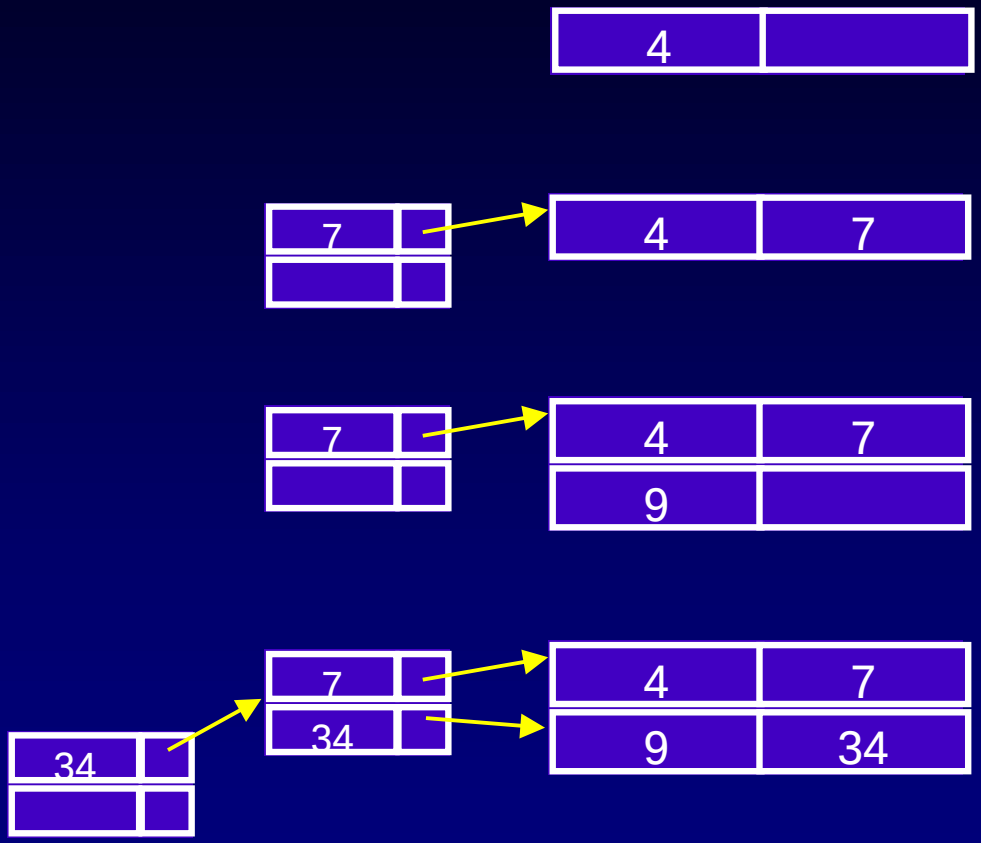
$$NBLK_{data} = k^h$$

$$h = \lceil \log_k (NBLK_{data}) \rceil$$

Performance : *RetrieveOne*

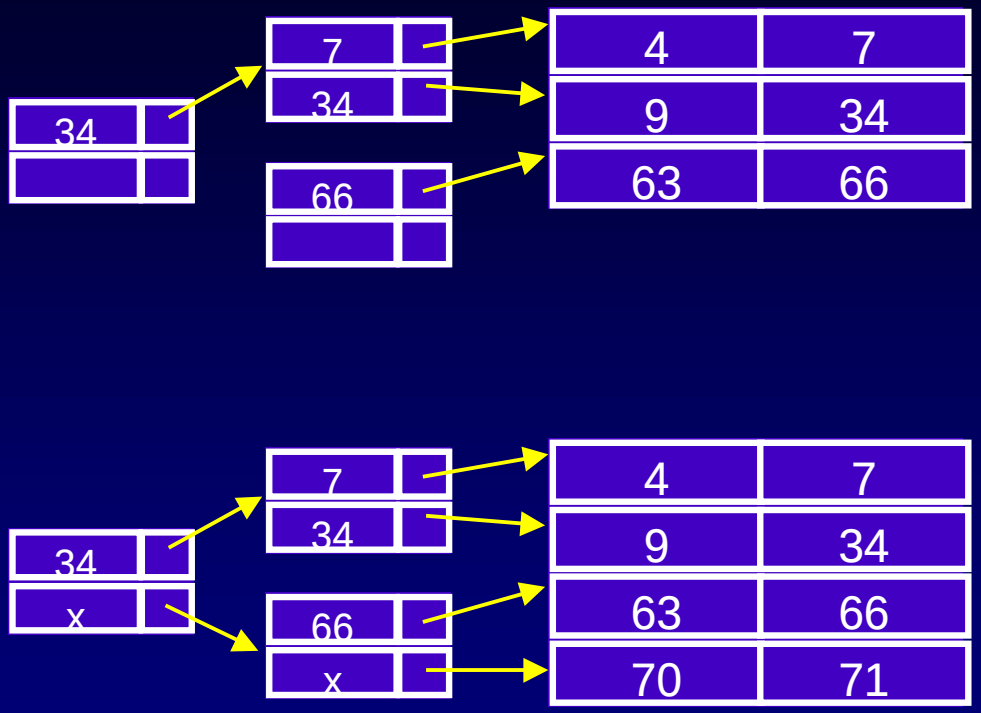
- 100,000 records, each of size 500 bytes
- index record size = 20 bytes
- block size = 2000 bytes
- Full index : 1000 blocks : log 1000 ≈10 *rba*
- Indexed seq (1 level) : 250 blocks
  - 1 + log 250 ≈9 *rba*
- Indexed seq (multilevel) : BF = 2000/20 = 100
  - $h = \lceil \log_{100} 100000 \rceil = 3$
  - $1 + h = 4$ *rba*

Initial Loading



4, 7, 9, 34, 63, 66, 70, 71

Initial Loading



4, 7, 9, 34, 63, 66, 70, 71

Reorganization Point

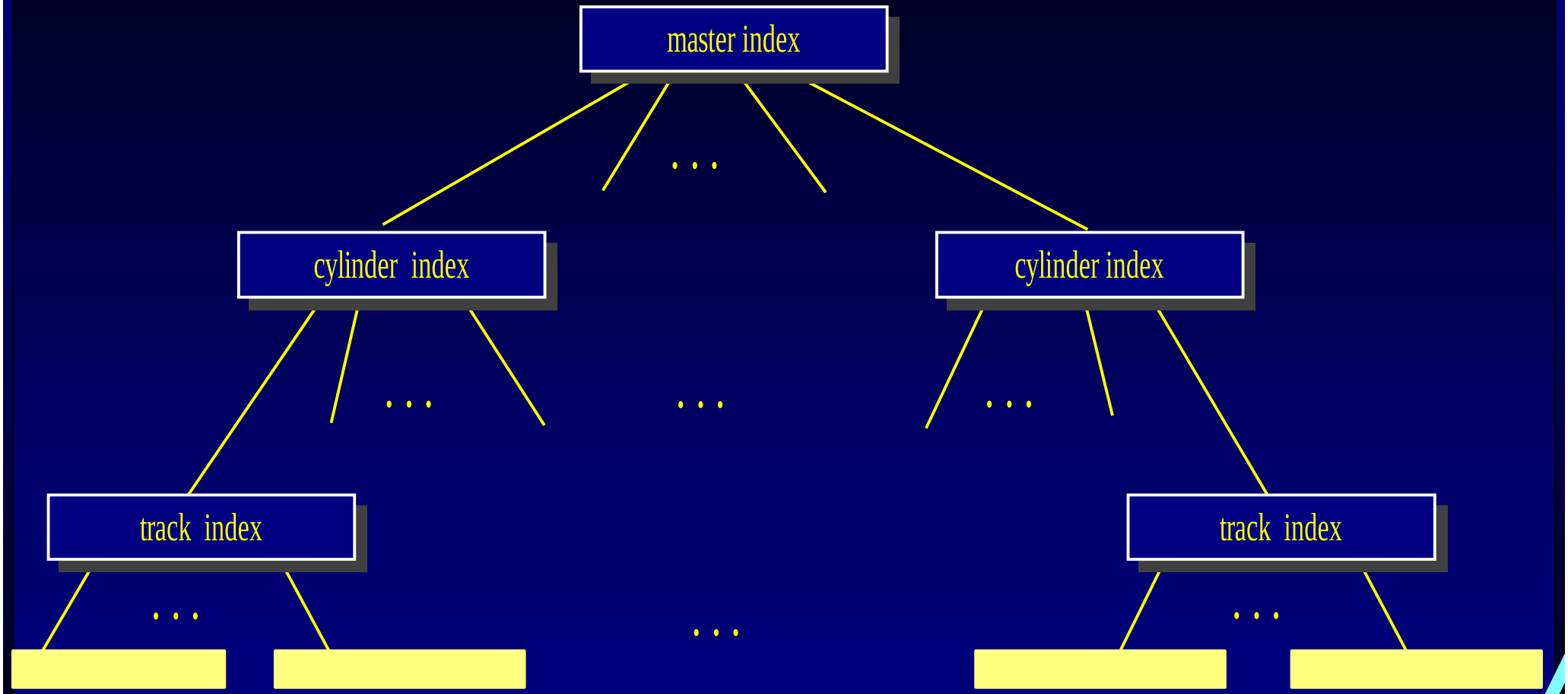
- Reorganize when performance has deteriorated by 50% from the performance just after (initial loading).
- Let n_1 be # of *RetrieveOne* in a unit time
- Let n_2 be # of *RetrieveAll* in a unit time
- Let L be the average length of overflow recs.

$$NBLK_{index} = 1 + k + k^2 + \cdots + k^{h-1} = \frac{k^h - 1}{k - 1}$$

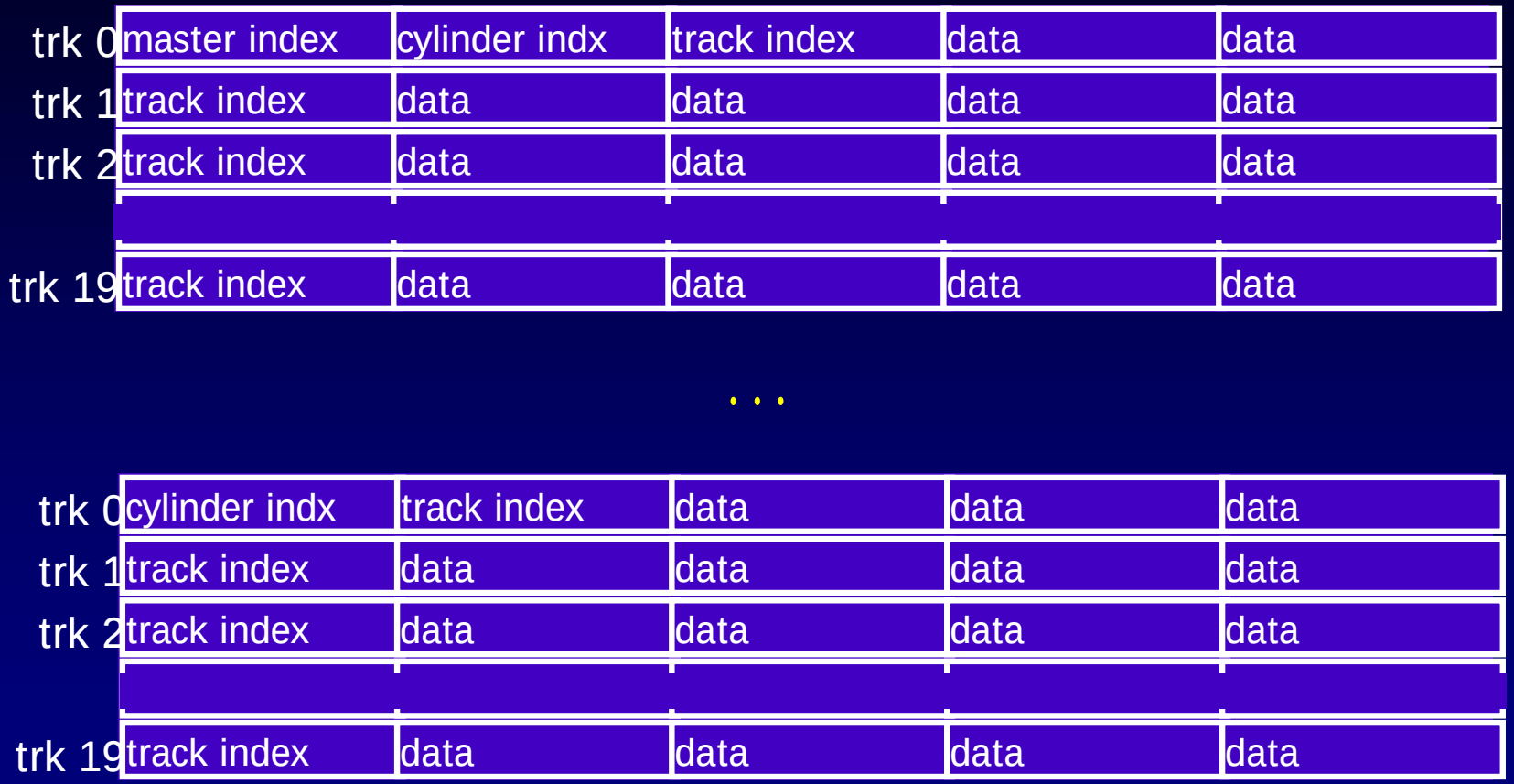
$$NBLK_{data} = k^h$$

$$h = \lceil \log_k (NBLK_{data}) \rceil$$

Physical Structure



Physical Structure



Physical Structure

- Faster access
  - Mingling the data and index blocks : locality
  - Keep master index (level 0 index) in RAM

Example

- 10,000 records, 160 bytes/record, key is 16 bytes, pointer is 4 bytes
- HP7925 - 256 bytes/sector, 64 sectors/track, 9 tracks/cylinder, 815 cylinder
- Choose BF = 6, utilization=(160x6)/(256x4) = 93.8%
- 1 block = 1024, 1024/(16+4) = 51 index entries
- 1 track = 64/4 = 16 blocks
- 10000 records, 10000/6 = 1667 blocks
- number of cylinders = 1667/(16x9-10) = 13 cyl.

16 block / tracks, 9 tracks/cylinder
(9 track index block + 1 cylinder index block)