GATE CRASH COURSE

COMPUTER SCIENCE

Database Management System

File Structures, Indexing,

Lecture_07

Vijay Agarwal sir

(TODAY GOAL)

FD & Normalization

Transaction & Concurrency Control

Query Language

File Org & Indexing

File Org & Indexing :

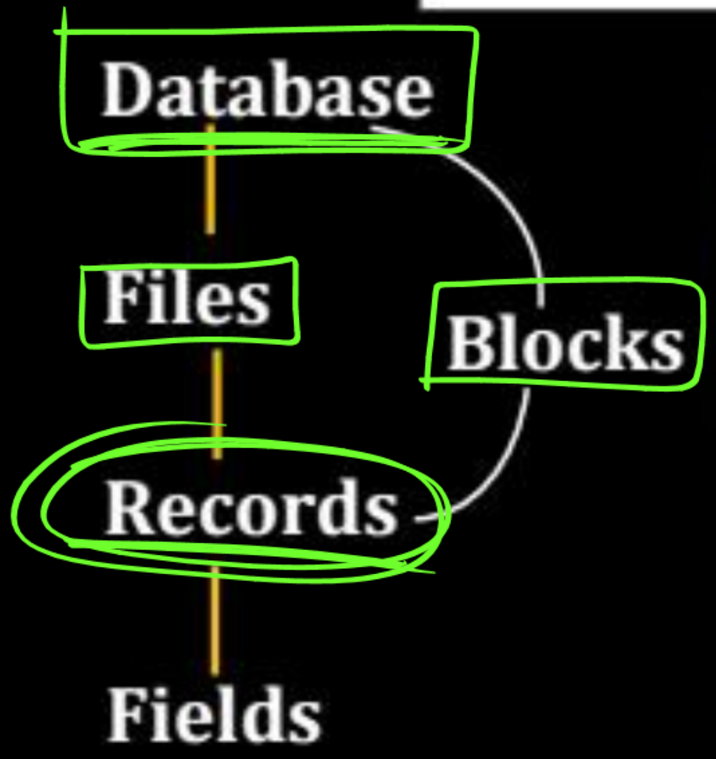
- File
  - Spanned & Unspanned
  - Ordered & Unordered

- Index
  - Dense
  - Sparse

P.I
C.I
S.I
& multilevel Indexing

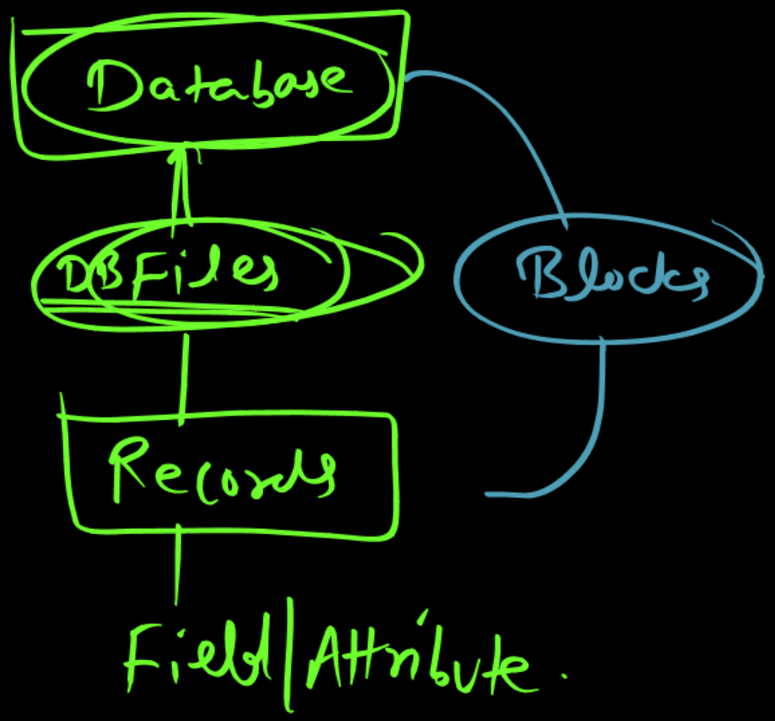
B Tree
B+ Tree

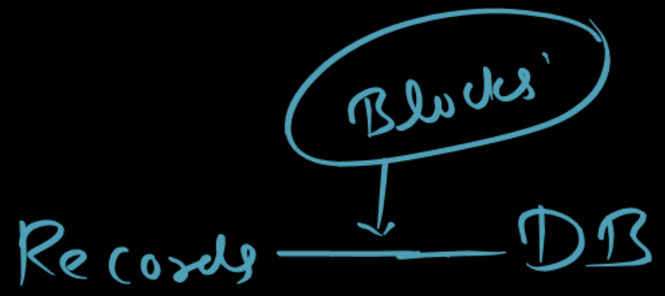
File Structures & Indexing



- Database is collection of files.
- Each file is a collection of Records.
- Each record is a sequence of fields.

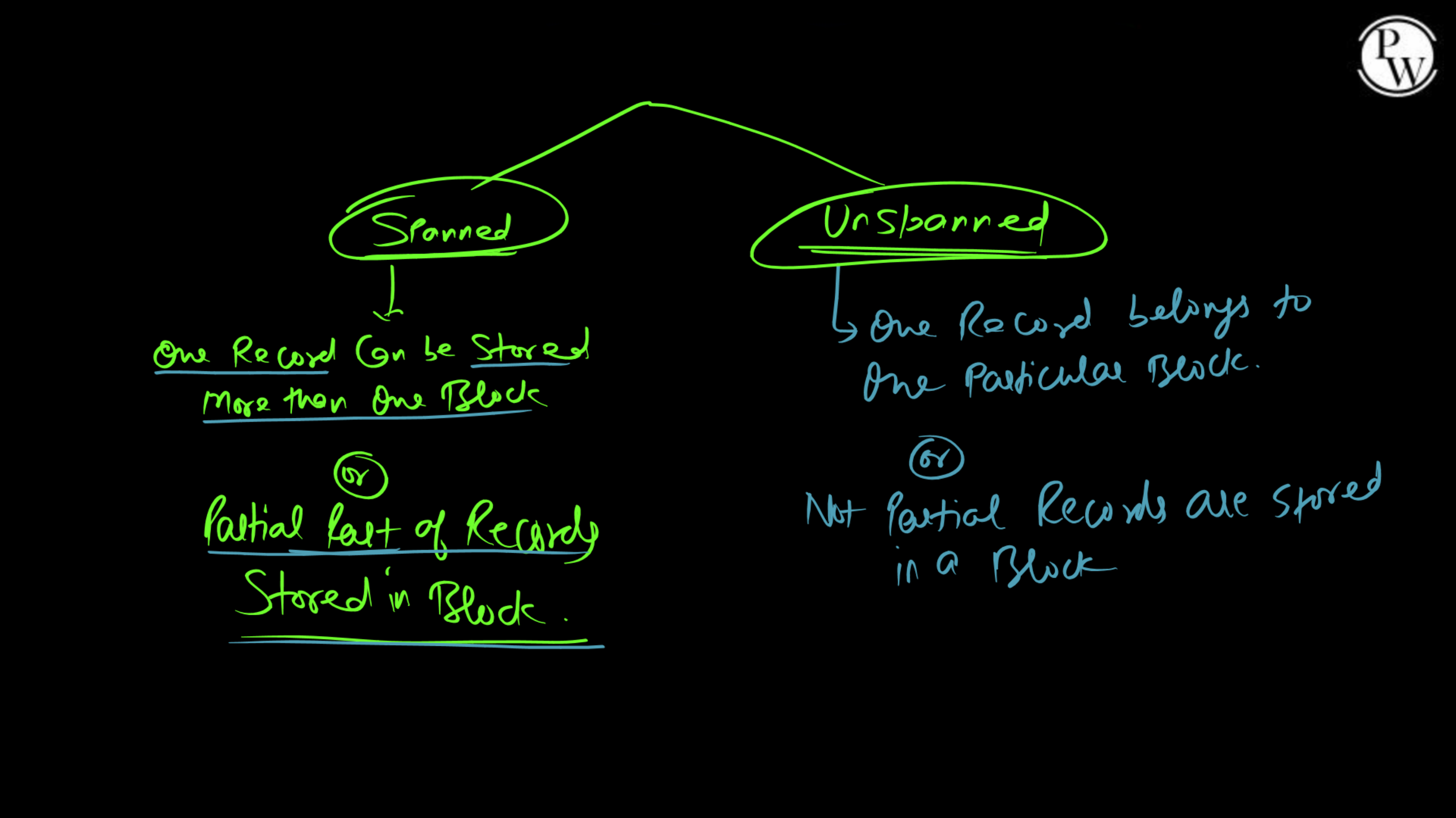
- DB is divided into number of blocks.
- Each block is divided into records.
- Record can be stored in a blocks.


Database
DB Files
Blocks
Records
Field | Attribute.



Blocking factor (B_F): Avg Number of Records Per Block.

$B_F = 4$ ⇒ 4 Records Stored in Per/One Block.


Spanned
Unspanned
One Record Can be Stored More than One Block
(or)
Partial part of Records Stored in Block.
One Record belongs to One Particular Block.
(or)
Not Partial Records are stored in a Block

Record Blocking and Spanned Versus Unspanned Records

- Blocking factor
  - Average number of records per block for the file

BF = 3 ⇒ 3 Records per Blocks.

R_3 → B_1 & B_2

Spanned organization : No Memory Waste.

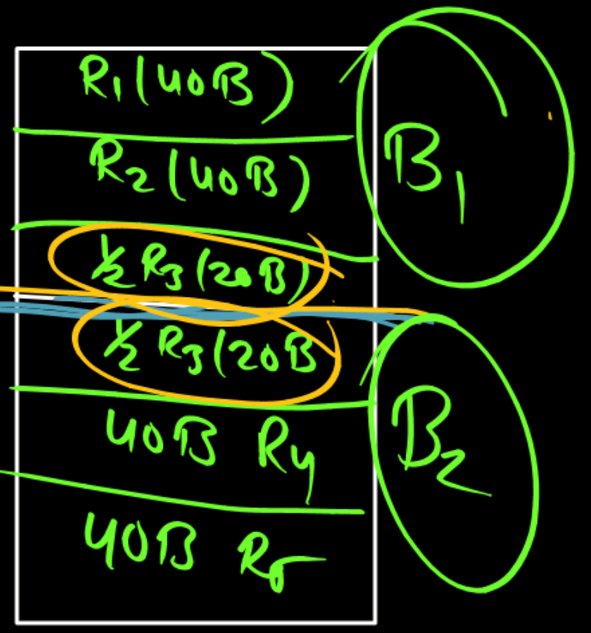
↳ I/O Cost is More

Eg

Block Size = 100 Byte

Record Size = 40 Byte

Blocking factor = $\frac{\text{Block Size}}{\text{Record Size}} = \frac{100B}{40B} = 2.5$



Record Blocking and Spanned Versus Unspanned Records

- Blocking factor
  - Average number of records per block for the file.

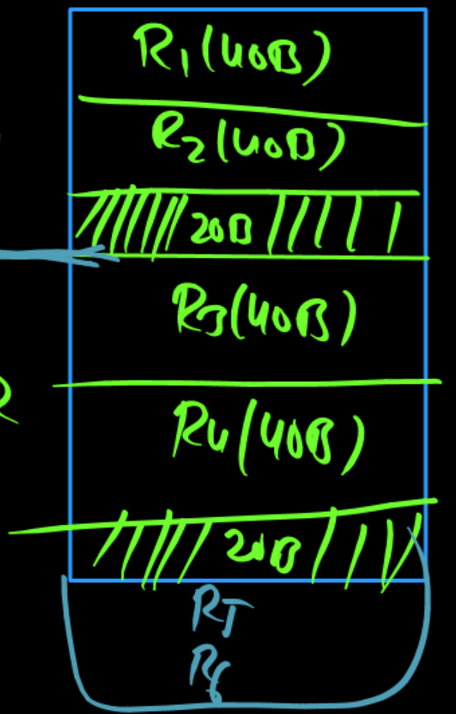
Unspanned organization : Memory Waste

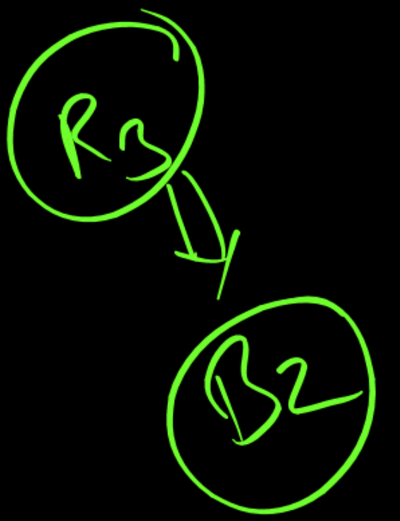
→ I/O Cost is less

Block Size = 100 B

Record Size = 40 B

$$B_F = \left\lfloor \frac{100B}{40B} \right\rfloor \Rightarrow \lfloor 2.5 \rfloor = 2$$

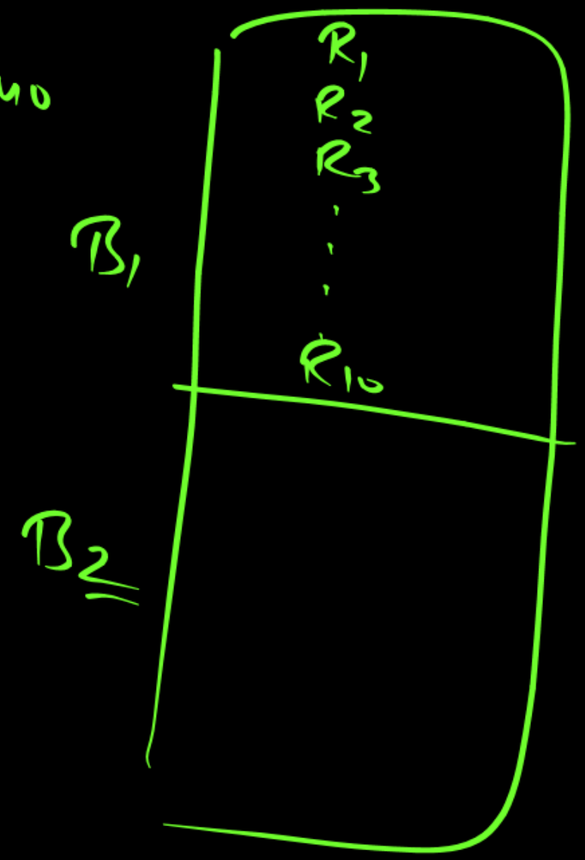




Block Size = 400

Record Size = 40

$B_F = \frac{400}{40} = 10$



Block Size = 400B

Record Size = 70B

Spanned

$B_F = \frac{400}{70}$

$= 5.6$

Unspanned

$B_F = \left\lfloor \frac{400}{70} \right\rfloor$

$= 5$

Note:

In spanned roganization No memory is waste but I/O cost is more (Block Access Increase).

But in unspanned organization memory is waste but input output cost is less compared to spanned organization.

Default organization is unspanned organization.

Block Size = 100B

512B

1024B

Record Size = 100B

1 - 2 : 1

12 Byte

24 Byte

Waste

Note:

I/O Cost: Input Output cost means Number of Blocks transferred from secondary memory to Main memory in order to access some records.

Search Key: Attribute used to access the Data from DB

Organization of records in a file: (1) ORDERED file organization

(2) Unordered file organization

Blocking factor

(Avg # Records per Block)

- Spanned
- Unspanned
  - Default
    - I/O Cost less.

B: Data Block

File

- ORDERED
  - Searching Easy
  - Insertion expensive
  - Binary Search.
    - $\lceil \log_2 B \rceil$
- Unordered
  - Insertion Easy
  - Searching Expensive
    - Linear Search
      - Avg $B/2$

Files of Ordered Records (Sorted Files)

- Ordered (sequential) file
  - Records sorted by ordering field
    - Called ordering key if ordering field is a key field
- Advantages
- Reading records in order of ordering key value is extremely efficient
- Finding next record
- Binary search technique

NOTE:
To Access a record the average number of
Block Access = $\log_2 B$
(B: Data Blocks)

Files of Unordered Records (Heap Files)

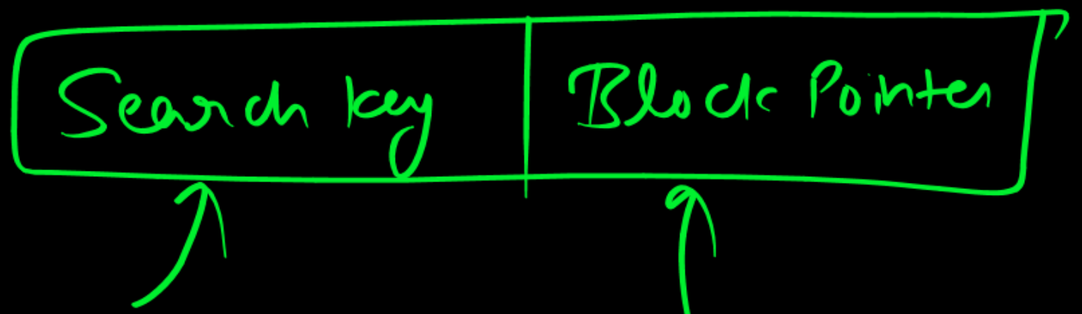
- Heap (or pile) file
  - Records placed in file in order of insertion
- Inserting a new record is very efficient
- Searching for a record requires linear search
- Deletion techniques
  - Rewrite the block
  - Use deletion marker

NOTE:
To Access a record the average number of
Block Access $= \frac{B}{2}$
(B: Data Blocks)

Access Times for Various File Organizations

Type of Organization	Access/Search Method	Average Blocks to Access a Specific Record
Heap (unordered)	Sequential scan (linear search)	$b/2$
Ordered	Sequential scan	$b/2$
Ordered	Binary search	$\log_2 b$

Average access times for a file of b blocks under basic file organizations



Point to a block where key is available.

Indexing (Basic Concepts)

- Indexing mechanisms used to speed up access to desired data.
  - E.g., author catalog in library
- **Search Key** - attribute to set of attributes used to look up records in a file.
- An **index file** consists of records (called **index entries**) of the form

- Index files are typically much smaller than the original file.
- Two basic kinds of indices:
  - **Ordered indices:** search keys are stored in sorted order
  - **Hash indices:** search keys are distributed uniformly across "buckets" using a "hash function".

Block Size of Index File is Same as Block Size of DB file

Ikey	Pointer

One Index Record Size = Size of search key + Size of Bp

To Access a Record Using Index, Avg # Block Accesses $= \log_2 B_i + 1$

B_i: Index Block

Note

Index file block size is same as DB file Block Size ✓

Block Size = 500B
Record Size = 150Byte

Block Size of Index File = Block Size of DB file ✓

One Index Record Size = Size of Search Key + Size of Block Pointer
$(x+y)$ Byte

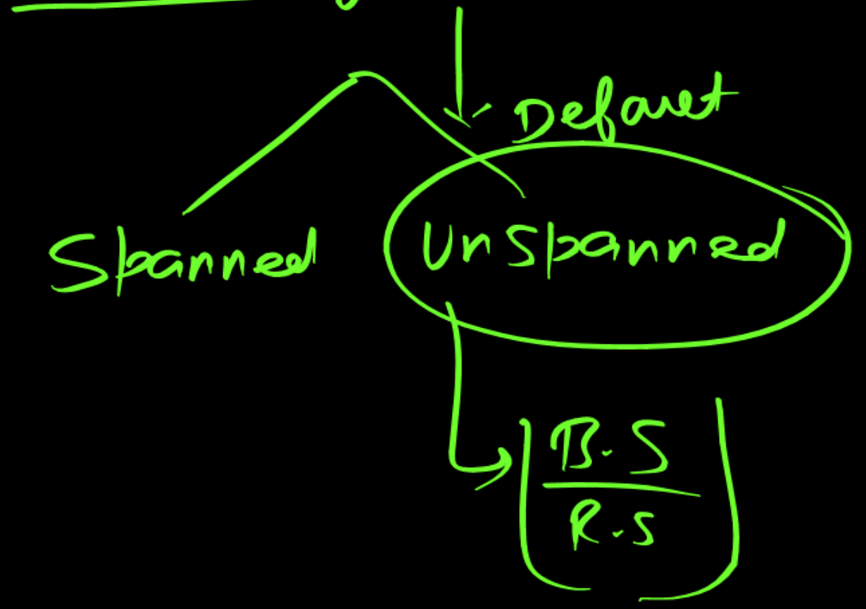
$B_{f_i} = \frac{\text{Block size}}{\text{Index Record}}$

NOTE: To Access a Record Average number of block access $= \log_2 B_i + 1$

- Index Block access
- Data Block access

[Bi : Index Block]

Blocking factor



To Access a Record Avg No. of Block Access

Ordered $\Rightarrow \log_2 B$

Unordered $= \frac{B}{2}$

B: Data Block

Q

Record Size = 150 Byte Block Size = 500 Byte #Records = 30000

Sol

Blocking factor (Unspanned org) = $\left\lfloor \frac{\text{Block Size}}{\text{Record Size}} \right\rfloor \Rightarrow \left\lfloor \frac{500\,B}{150\,B} \right\rfloor \Rightarrow \lfloor 3.3 \rfloor$

$2^{11} = 2048$
$2^{12} = 4096$
$2^{13} = 8192$
2^{14}

Blocking factor = 3 ⇒ 3 Records Per Block.

#Records = 30,000

#DATA Blocks = $\frac{30000}{3}$ = 10,000 Data Block

Ordered = $\lceil \log_2 10000 \rceil$ + 14 Block Access

Unordered = $\frac{B}{2} = \frac{10000}{2} = 5000$

B = 10,000

Example:

- Suppose that:
  - record size R = 150 bytes, block size B = 512 bytes,

    r = 30000 records
- Then, we get:
  - blocking factor Bfr = B div R = 512 div 150 = 3 records/block
  - number of file blocks $b = (r/\text{Bfr}) = (30000/3) = 10000$ blocks

$\log_2 B$ Ordered

$B/2$ Unordered

Example:

Given the following data file

EMPLOYEE (NAME, SSN, ADDRESS, JOB, SAL, ...)

Suppose that:

- record size R=150 bytes, block size B=512 bytes r=30000 records
- For an index on the SSN field, assume the field size V_{SSN} =9 bytes, assume the record pointer size P_R=7 bytes. Then:
  - index entry size $R_1=(V_{SSN}+P_R)=(9+7)=16$ bytes
  - index blocking factor $Bfr_1=B \text{ div } R_1=512 \text{ div } 16=32$ entries / block
  - number of index blocks $b=(r/Bfr_1)=(30000/32)=938$ blocks
  - binary search needs $\log_2 bI=\log_2 938=10$ block accesses

Indexing

Key = 9B , Bp = 7Byte

#Record = 30000

Record Size = 150Byte

One Index Record Size = 9+7 = 16 Byte & Block Size = 500B

Block factor of Index file (B_{fi}) [Unspanned] $= \left\lfloor \frac{500B}{16B} \right\rfloor$ = 32 Index Entries/Block

Assume

Total # Index Records = 30,000

$$\#\text{Index Block} = \frac{30000}{32} = 938 \text{ Index Block}$$

Univ: STUDENT DB

Assume Block factor = 10

#Record = 40000

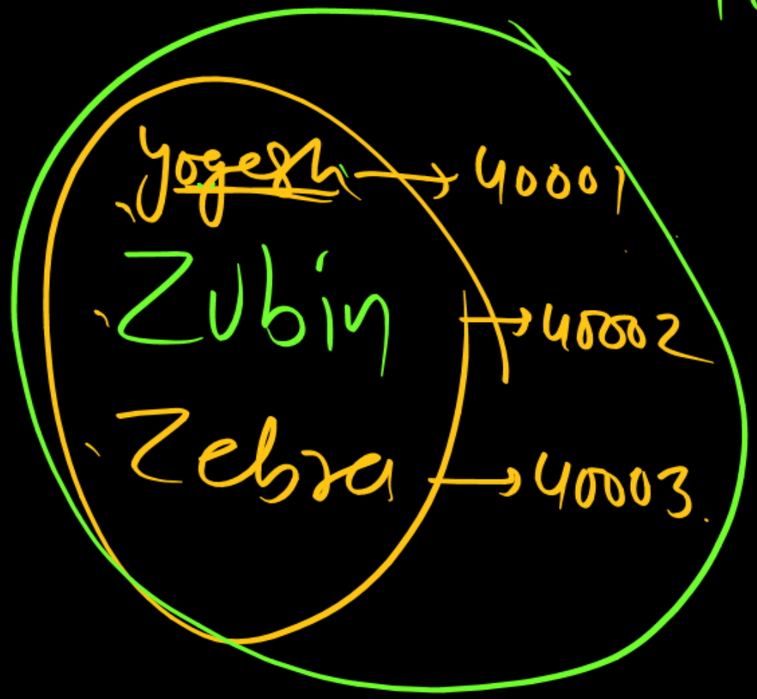
$$\#\text{Data Block} = \frac{40000}{10}$$

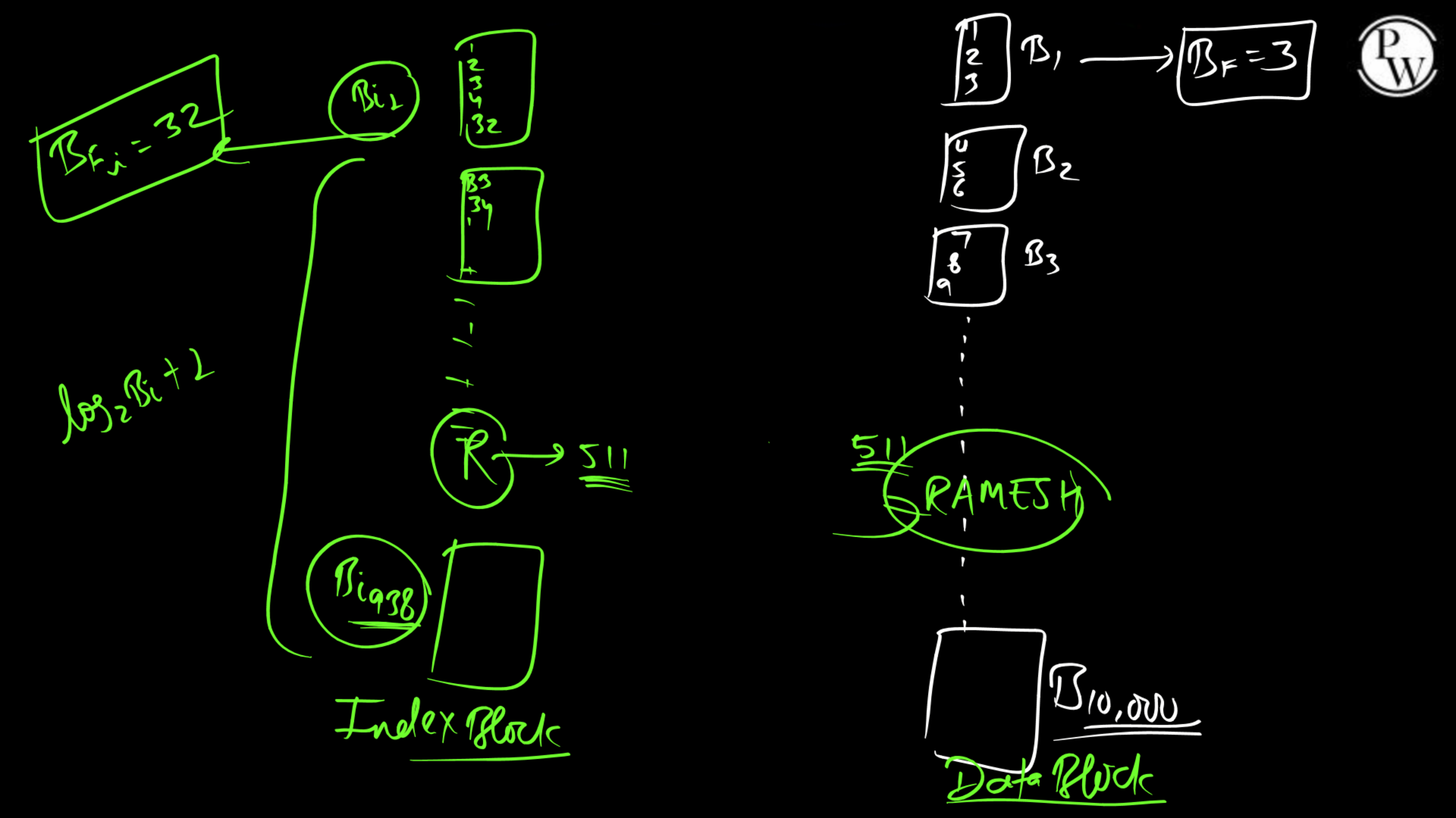
= 4000 Data Block

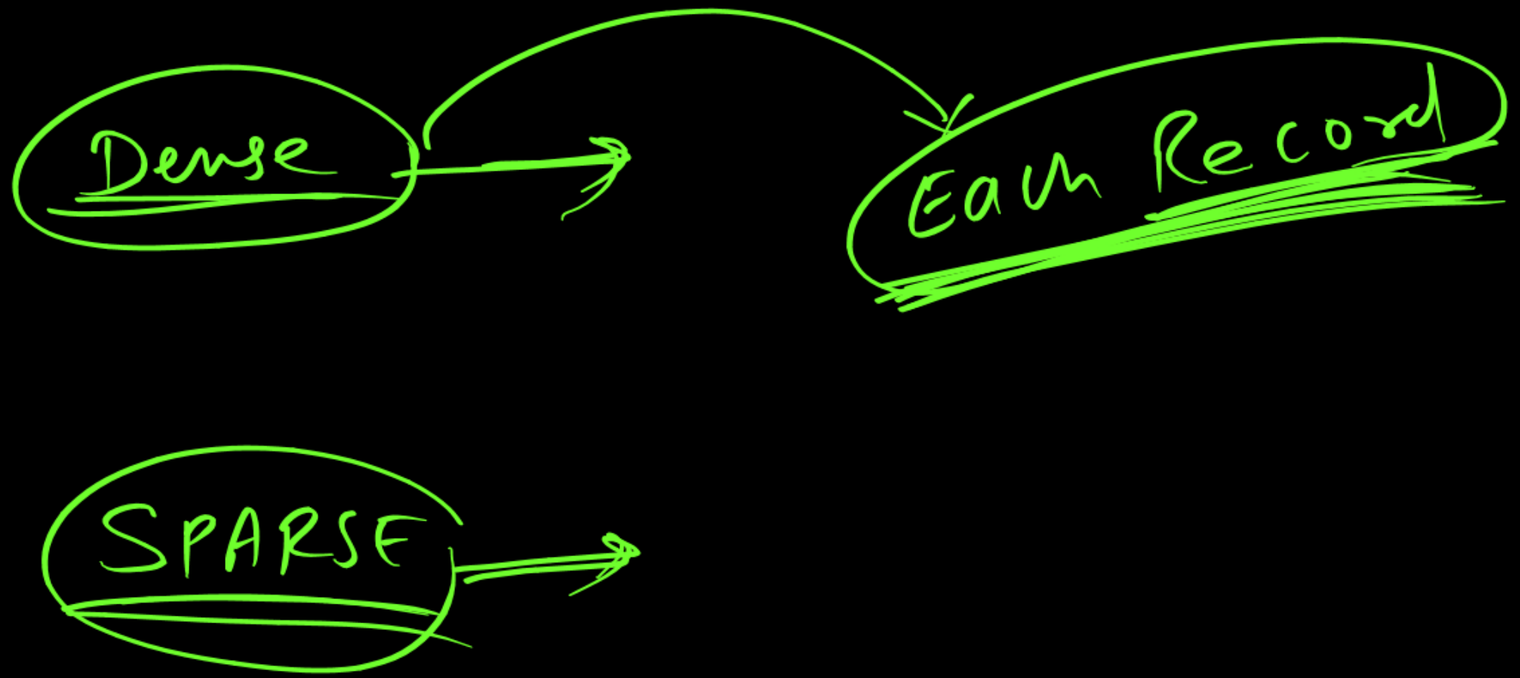
#Records = 40003

$$\#\text{Data Block} = \frac{40003}{10} = 4000$$

4001




$B_{F_i} = 32$
B_{i_1}
1
2
3
4
32
B3
34
$\log_2 B_i + 2$
R
511
$B_{i_{938}}$
Index Block
1
2
3
B_1
$B_F = 3$
4
5
6
B_2
7
8
9
B_3
511
RAMESH
$B_{10,000}$
Data Block


Dense
Each Record
SPARSE

Category of Index

1) Dense Index Files : for each DB Record there exist Index Entry

Number of Index entries = Number of DB Records

2) Sparse Index Files : → for a set of Records, One Entry in Index file.

Number of Index entries = Number of Blocks

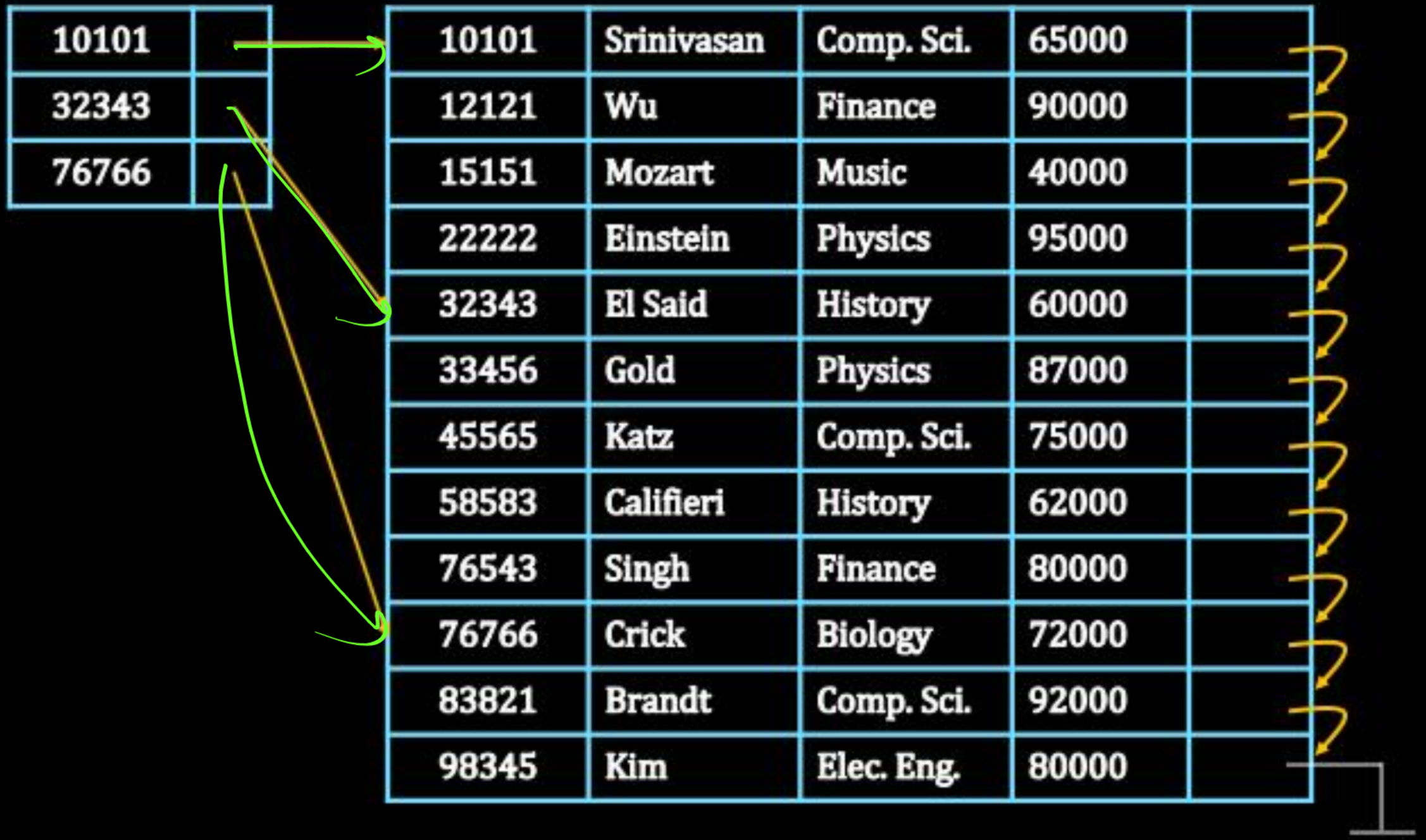
Dense Index Files

- Dense Index - Index record appears for every search-key values in the file.
- Example - index on *ID* attribute of *instructor* relation

10101	10101	Srinivasan	Comp. Sci.	65000	
12121	12121	Wu	Finance	90000	
15151	15151	Mozart	Music	40000	
22222	22222	Einstein	Physics	95000	
32343	32343	El Said	History	60000	
33456	33456	Gold	Physics	87000	
45565	45565	Katz	Comp. Sci.	75000	
58583	58583	Califieri	History	62000	
76543	76543	Singh	Finance	80000	
76766	76766	Crick	Biology	72000	
83821	83821	Brandt	Comp. Sci.	92000	
98345	98345	Kim	Elec. Eng.	80000	

Sparse Index Files

- Sparse Index: contains index records for only some search-key values.
  - Applicable when records are sequentially ordered on search-key
- To locate a record with search-key value K we:
  - Find index record with largest search-key value $< K$
  - Search file sequentially starting at the record to which the index record points

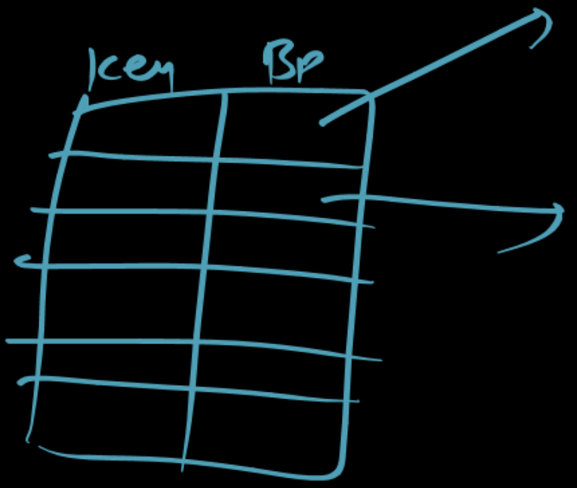

10101
32343
76766
10101 Srinivasan Comp. Sci. 65000
12121 Wu Finance 90000
15151 Mozart Music 40000
22222 Einstein Physics 95000
32343 El Said History 60000
33456 Gold Physics 87000
45565 Katz Comp. Sci. 75000
58583 Califieri History 62000
76543 Singh Finance 80000
76766 Crick Biology 72000
83821 Brandt Comp. Sci. 92000
98345 Kim Elec. Eng. 80000

Note
Index is a Ordered file

Types of Index

- Single-level Ordered Indexes
  - Primary indexes (P.K + DB File Ordered)
  - Clustering indexes (Nonkey + ordered)
  - Secondary indexes
    - (Nonkey / Candidate key + Unordered DB File)
- Multilevel Indexes
  - Dynamic multilevel indexes Using B-Tress and B+ Trees.
    - B+ Tree

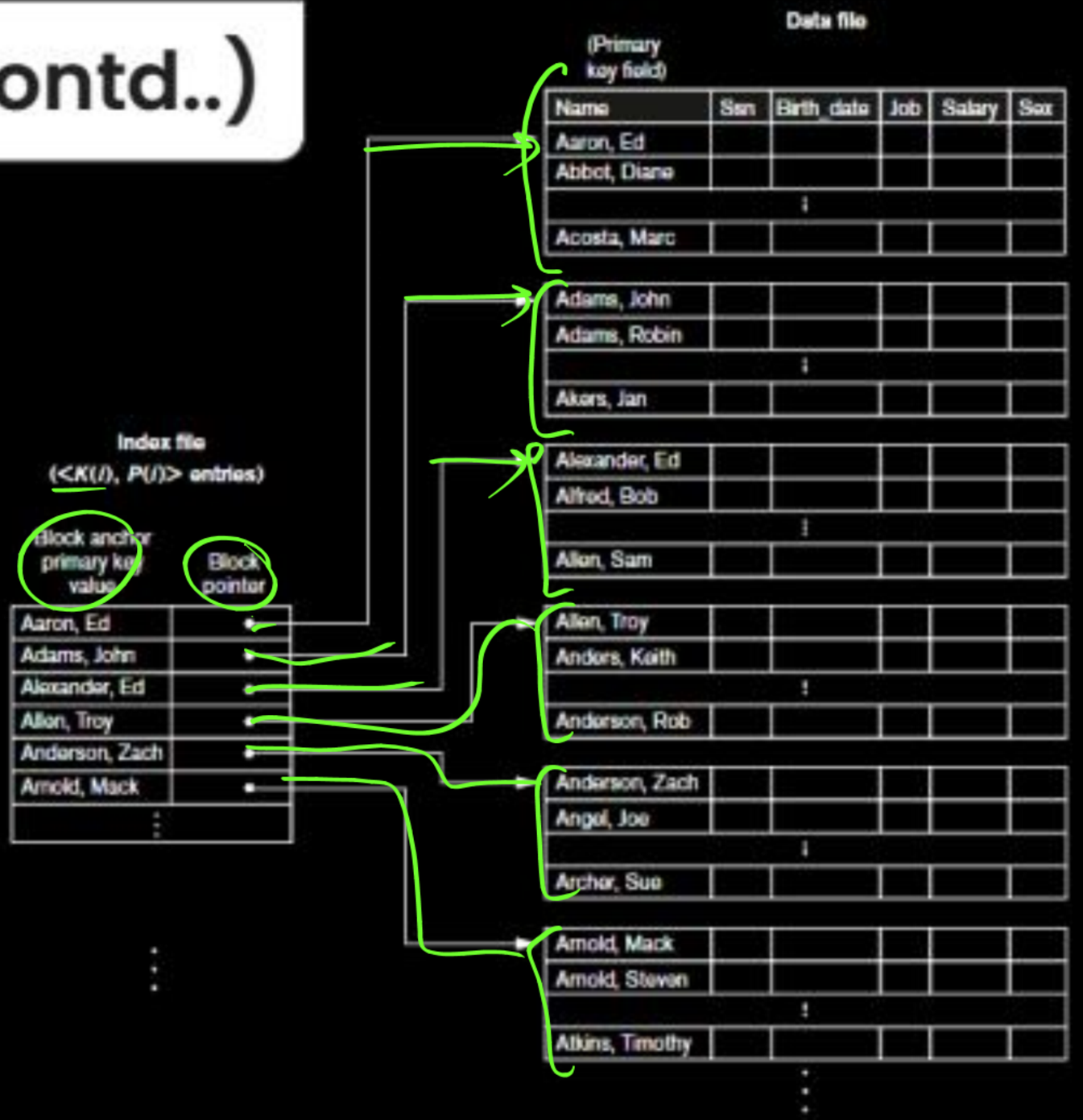
Primary Indexes



- Ordered file with two fields
  - Primary key, K(i)
  - Pointer to a disk block, P(i)
- One index entry in the index file for each block in the data file
- Indexes may be dense or sparse
  - Dense index has an index entry for every search key, value in the data file
  - Sparse index has entries for only some search values

Primary Index (Contd..)

Primary index on the ordering key field of the file shown in Figure



Unspanned

Ordered $\log_2 B$

Unordered $B/2$

With Index $= \log_2 B_i + 1$

Q.1 Suppose that we have Ordered file of 30,000 records, stored on a disk with Block Size 1024 Byte, file records are of fixed length & unspanned of size 100 Byte (Record size) and suppose that we Have created a primary index on the key field of the file of size 9 Byte and Block pointer of size 6 Byte then find the average number of Block Access to search for a record using with and Without Index ?

Soln

$\#Records = 30,000$ $Block\ Size = 1024B$, $Record\ Size = 100B$.

$Key = 9Byte$, $Bp = 6Byte$.

Without ⇒ 12

With Index = 7

#Records = 30000 Block Size = 1024 Byte, Record Size = 100B.

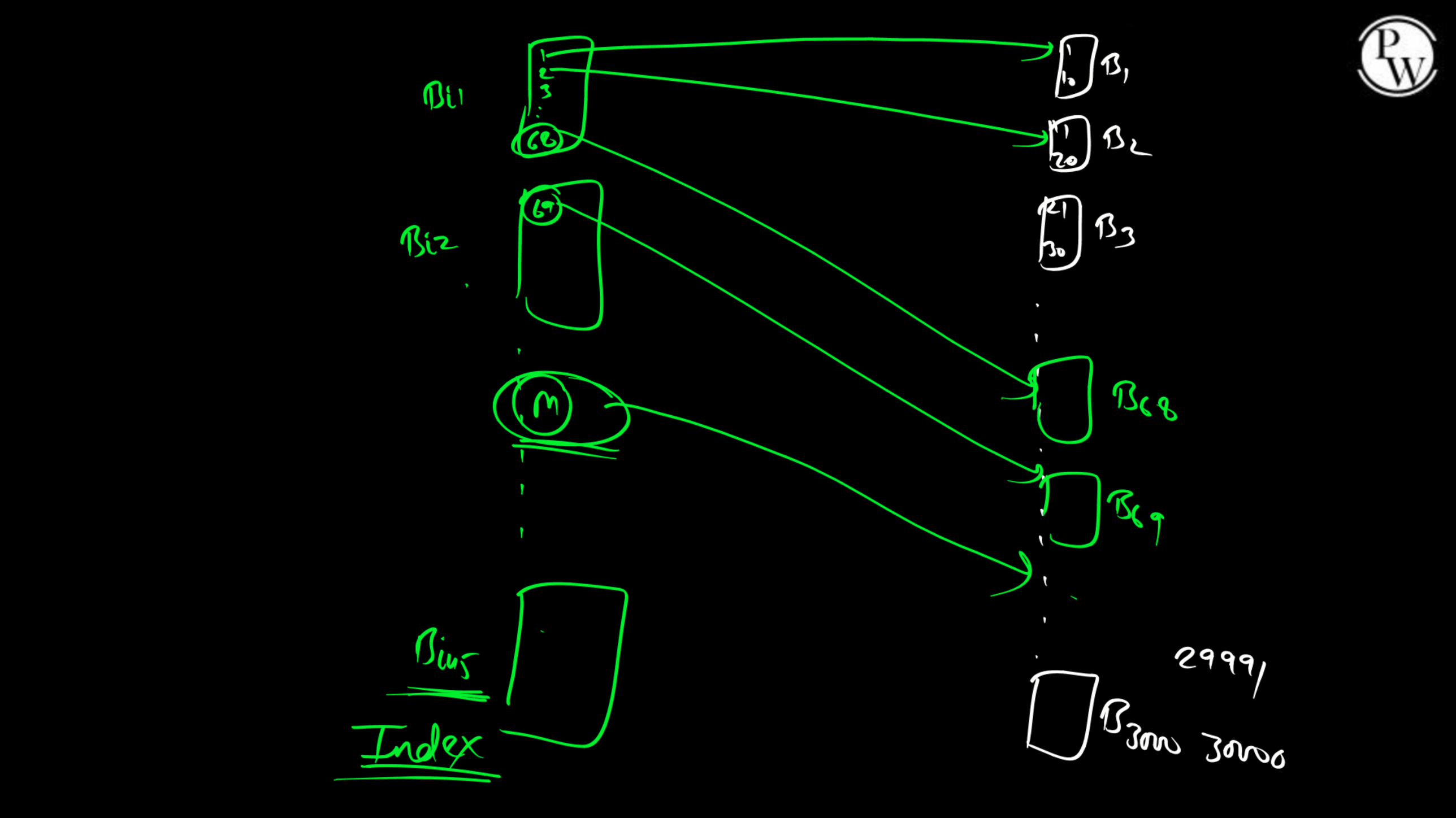
l_{key} = 9B, B_r = 6 Byte, Ordered

(I) Without Index

Block factor of DB File (Unspanned) $= \left\lfloor \frac{\text{Block Size}}{\text{Record Size}} \right\rfloor \Rightarrow \left\lfloor \frac{1024\,B}{100\,B} \right\rfloor \Rightarrow$ 10 Records Per Block.

#Records = 30,000

Total Number of DB Block $= \left\lceil \frac{30,000}{10} \right\rceil =$ 3000 DATA Block


Bi1
Biz
Bins
Index
1
2
3
60
67
M
1
10
B1
1
20
B2
21
30
B3
Bc8
Bc9
29991
B3000 30000

#Records = 30000 Block Size = 1024 Byte, Record Size = 100B.

Key = 9B, Br = 6 Byte, Ordered

$2^{10} = 1024$
$2^{11} = 2048$
$2^{12} = 4096$
$2^{13} = 8192$.

(I) Without Index

Block factor of DB File (Unspanned) $= \left\lfloor \frac{\text{Block Size}}{\text{Record Size}} \right\rfloor \Rightarrow \left\lfloor \frac{1024B}{100B} \right\rfloor \Rightarrow$ 10 Records Per Block.

#Records = 30,000

Total Number of DB Block (B) $= \left\lceil \frac{30,000}{10} \right\rceil =$ 3000 DATA Block

ORDERed file

To Access a Record Avg # Block Access $= \lceil \log_2 B \rceil \Rightarrow \lceil \log_2 3000 \rceil \Rightarrow$ 12 Block Access

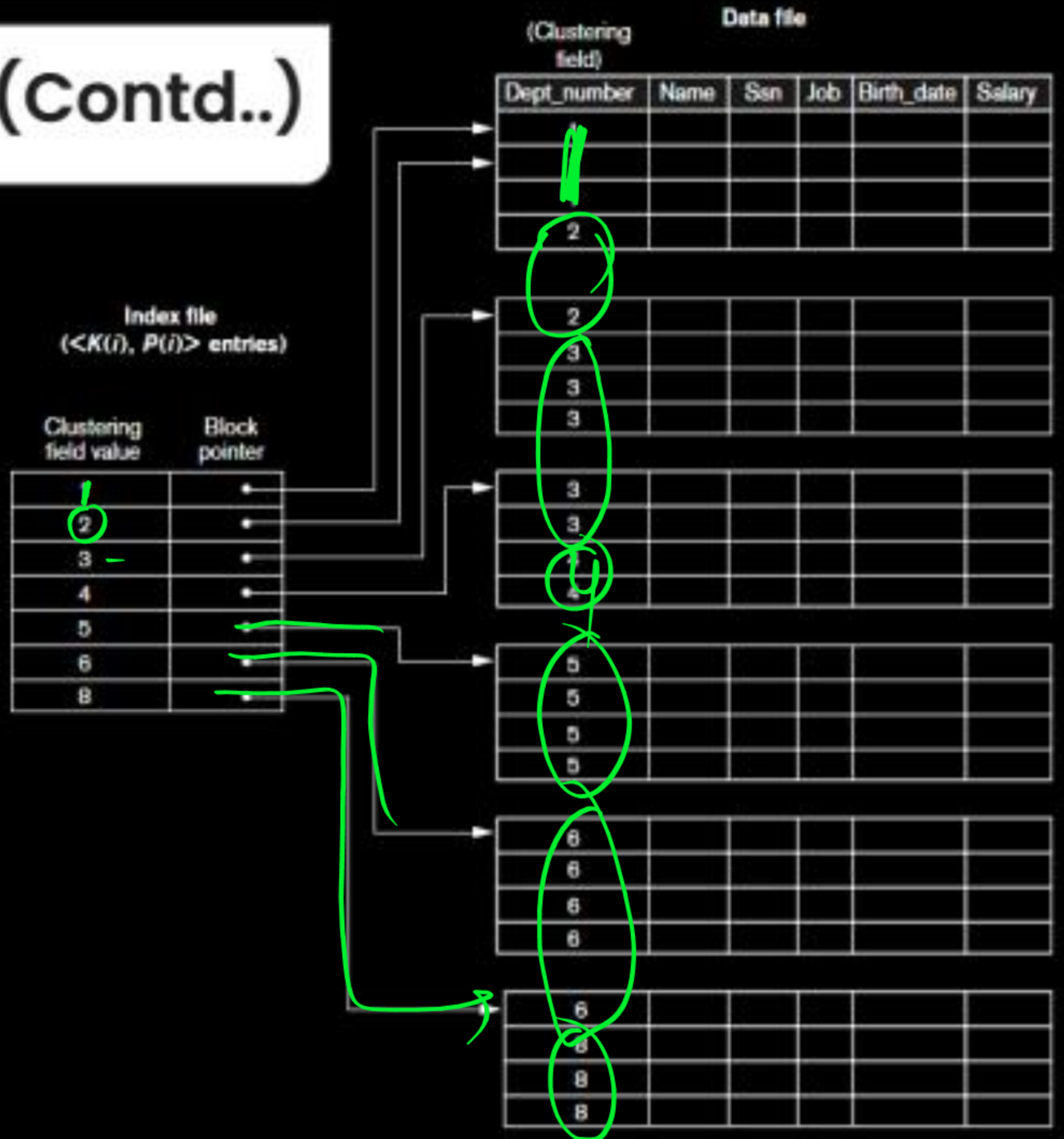
Ans

Clustering Indexes

- Clustering field
  - File records are physically ordered on a nonkey field without a distinct value for each record
- Ordered file with two fields
  - Same type as clustering field
  - Disk block pointer

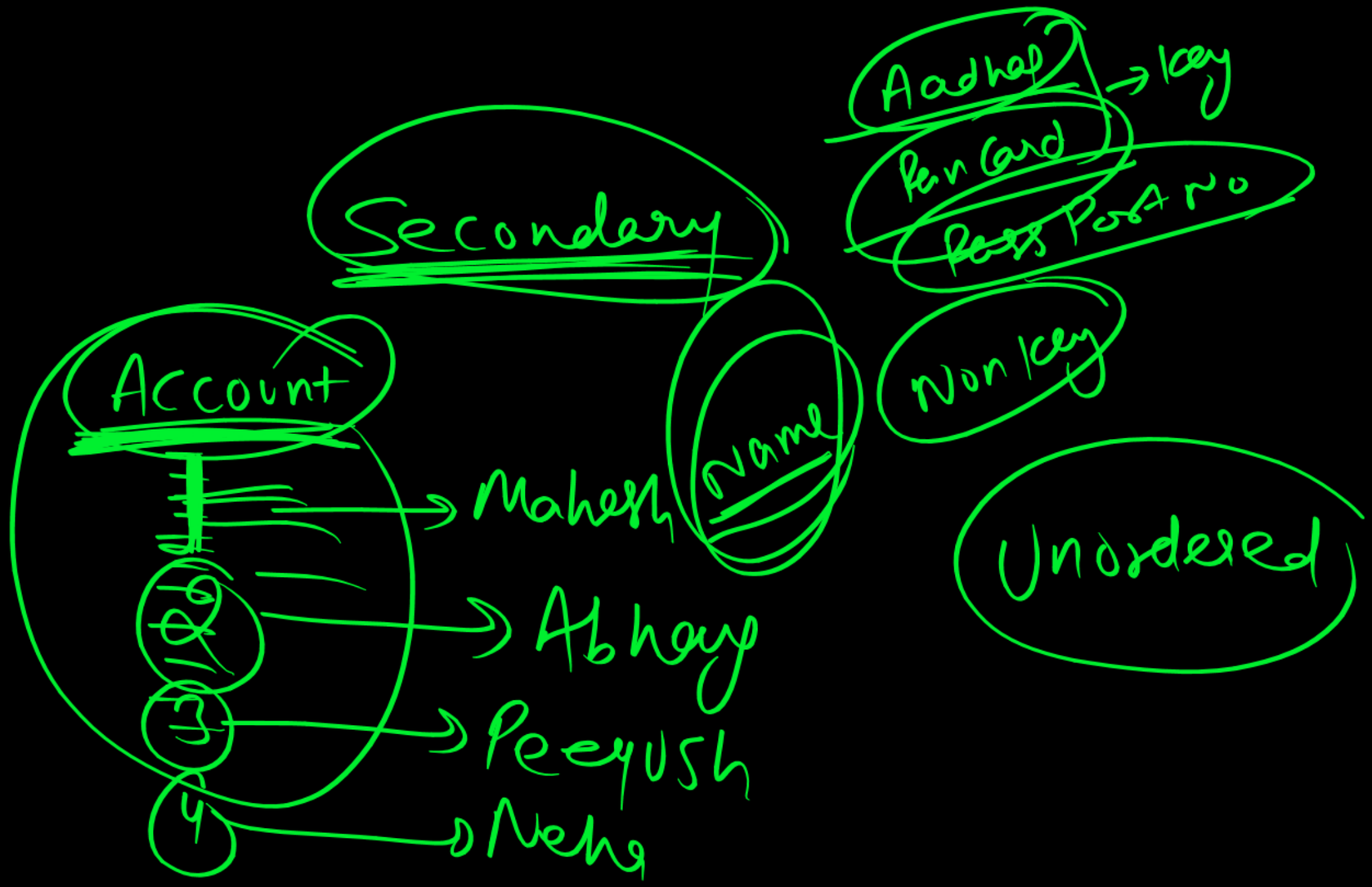
Clustering Indexes (Contd..)

A clustering index on the Dept_number ordering nonkey field of an EMPLOYEE file

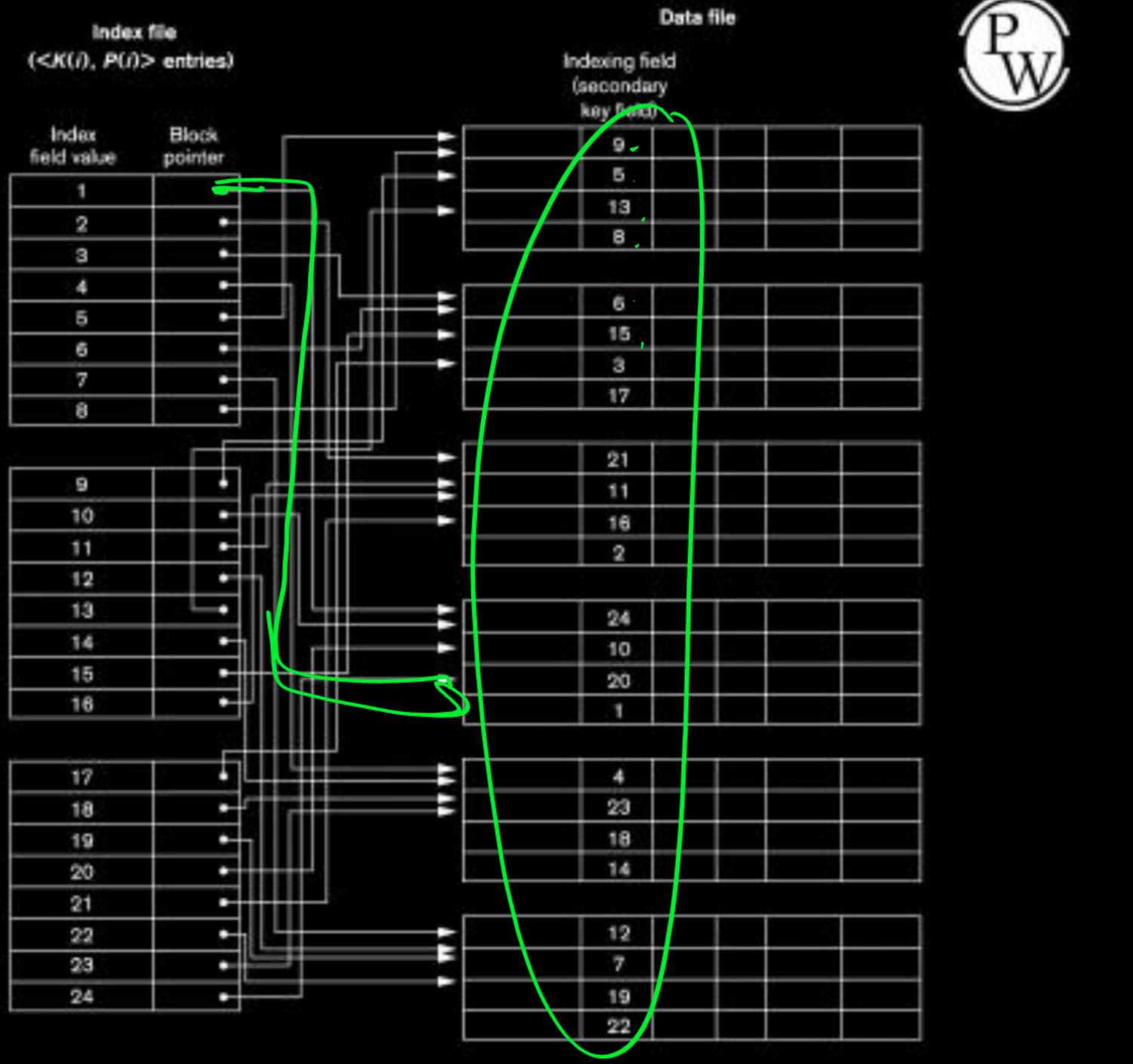


Secondary Indexes

- Secondary Index
  - A secondary index provides a secondary means of accessing a file for which some primary access already exists.
  - The secondary index may be on a field which is a candidate key and has a unique value in every record, or a non-key with duplicate values.
  - The index is an ordered file with two fields.
    - The first field is of the same data type as some non-ordering field of the data file that is an indexing field.
    - The second field is either a block pointer or a record pointer.
    - There can be many secondary indexes (and hence, indexing fields) for the same file.
- Includes one entry for each record in the data file; hence, it is a dense index


Aadhar
Pan Card
Passport No
Key
Secondary
Account
Name
Mahesh
Abhay
Peeyush
Neha
Non key
Unordered

Secondary Indexes (Contd..)



Q.2 Consider a secondary Index on the key field of the file of question number 1, then find the Average number of Block Access to Access a record using with & without Index?

Number of records = 30000 Block size = 1024 B

record size = 100 B

Key = 9B Bp = 6 Byte

Unspanned & Unordered.

#Records = 30000 Block Size = 1024 Byte, Record Size = 100B.

Ikey = 9B, Bp = 6 Byte, Ordered

$2^{10} = 1024$
$2^{11} = 2048$
$2^{12} = 4096$
$2^{13} = 8192$.

(I) Without Index

Block factor of DB File (Unspanned) $= \left\lfloor \frac{\text{Block Size}}{\text{Record Size}} \right\rfloor \Rightarrow \left\lfloor \frac{1024B}{100B} \right\rfloor \Rightarrow$ 10 Records Per Block.

#Records = 30,000

Total Number of DB Block (B) $= \left\lceil \frac{30,000}{10} \right\rceil =$ 3000 DATA Block

ORDERed file

To Access a Record Avg # Block Access $= \frac{B}{2} = \frac{3000}{2} =$ 1500 Ans

#Records = 30000 Block Size = 1024 Byte, Record Size = 100 B.

$V_{key} = 9B$, $B_p = 7B$.

$2^5 = 32$
$2^6 = 64$
$2^8 = 256$
$2^9 = 512$

With Index

Size of key + Bp Size

One Index Record Size = 9 + 6 = 15 Byte

Block Size = 1024 Byte

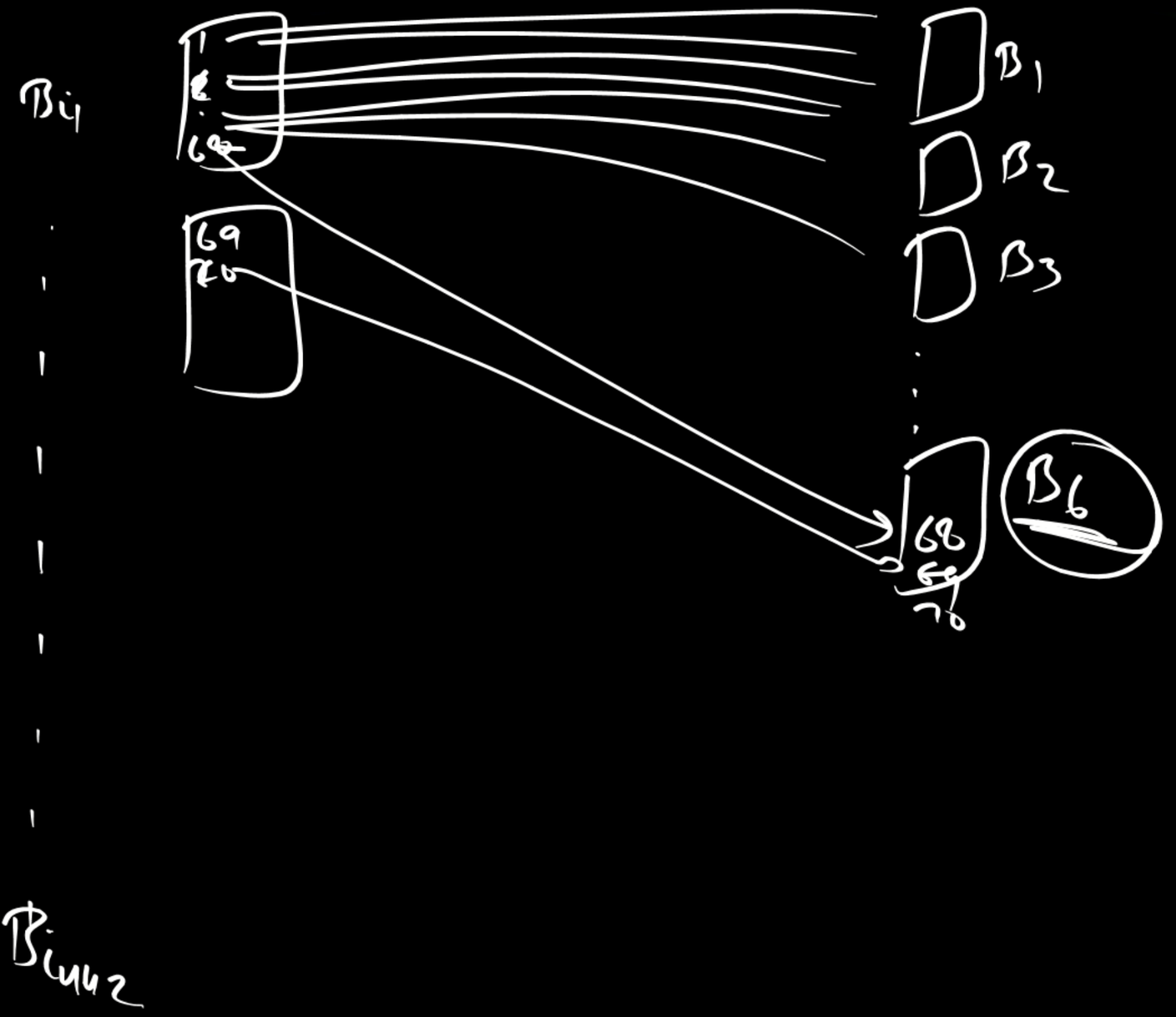
Block factor of Index file (unspanned) = $\left\lfloor \frac{1024B}{15B} \right\rfloor \Rightarrow$ 68 Index Entries/Block

Secondary (Dense) # Index Entries = # of DB Records = 30000

Index Block (B_i) = $\lceil \frac{30000}{68} \rceil$ = 442 Index Block

To Access a Record Avg # Block Access = $\log_2 B_i + 1 \Rightarrow \log_2 442 + 1 \Rightarrow$

$9 + 1 = 10$

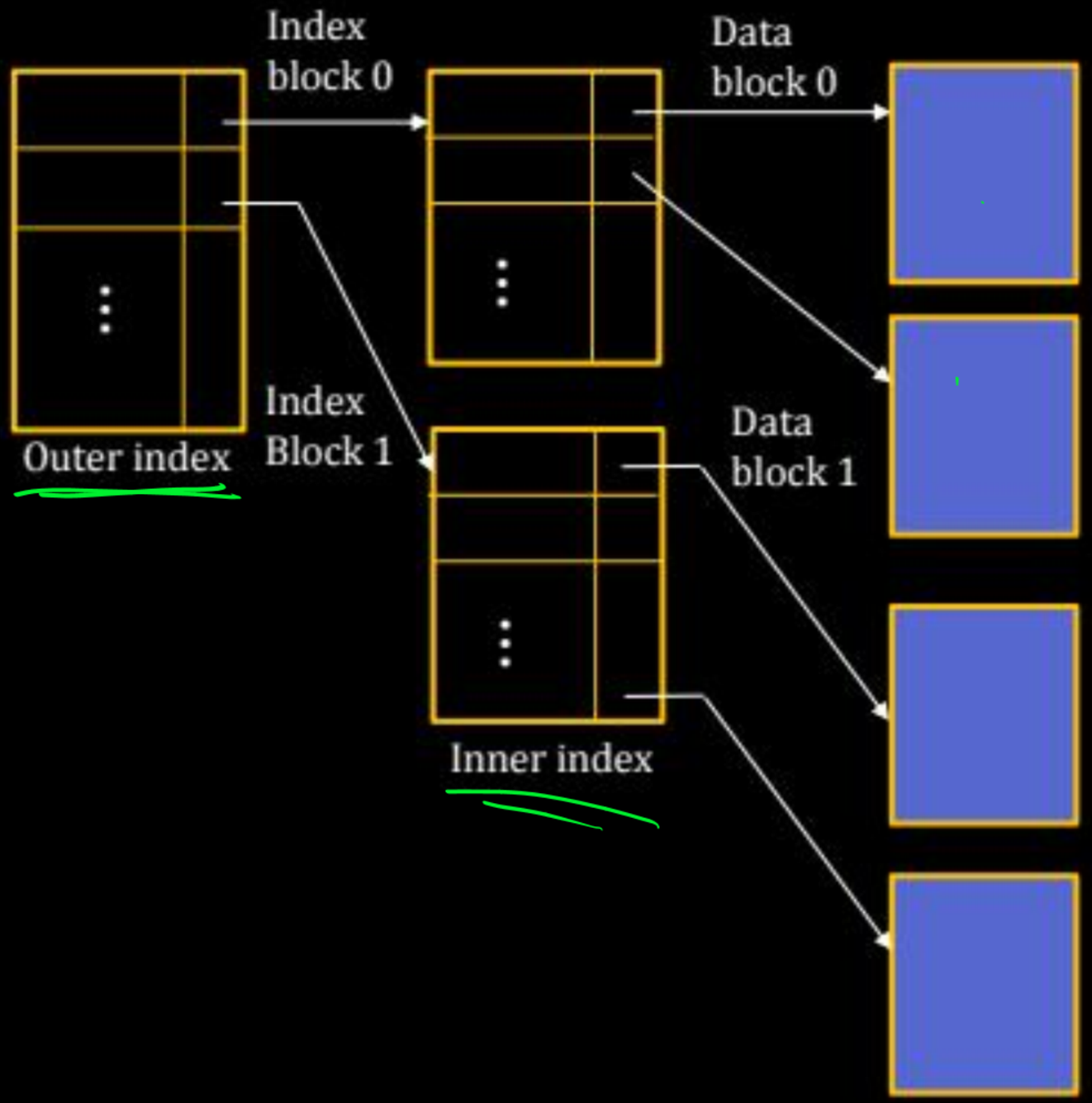

B_1
B_2
B_3
B_6

Multilevel Indexing :

Multilevel Index

- If index does not fit in memory, access becomes expensive.
- Solution: treat index kept on disk as a sequential file and construct a sparse index on it.
  - outer index - a sparse index of the basic index
  - inner index - the basic index file (442)
- If even outer index is too large to fit in main memory, yet another level of index can be created, and so on.
- Indices at all levels must be updated on insertion or deletion from the file.

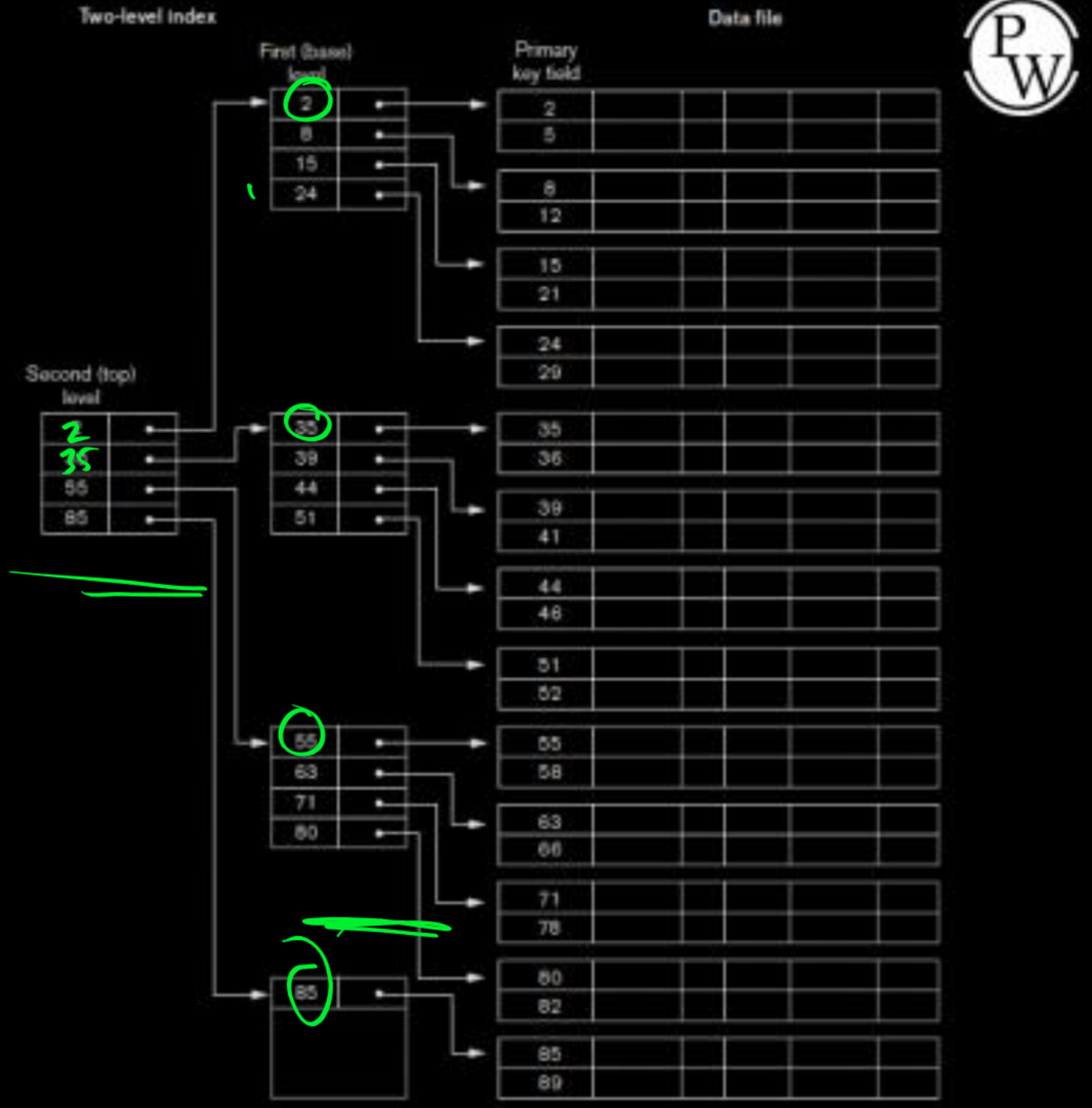
Multilevel Index (Contd..)



Multilevel Indexes

- Designed to greatly reduce remaining search space as search is conducted
- Index file
  - Considered first (or base level) of a multilevel index
- Second level
  - Primary index to the first level
- Third level
  - Primary index to the second level

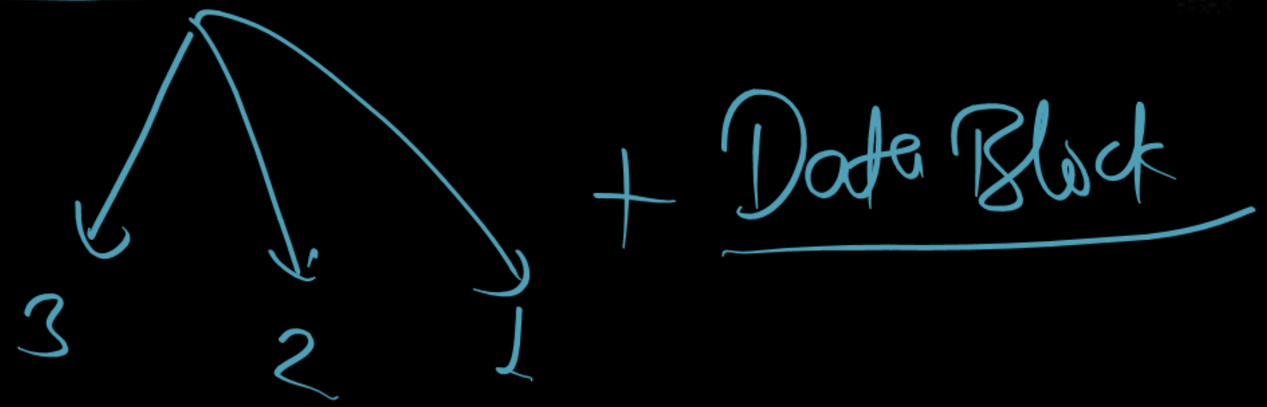
A two-level primary index resembling ISAM (indexed sequential access method) organization



NOTE: We can Repeat the above process until index entries fit into One Block.

NOTE: If there are n level in multilevel index then the number of Block Access to search for a record = n + 1

(at each level One Index Block + 1 Data Block)



Q.3 Find the average number of block access required to search for a record if multilevel Index is created on the Data file of Question 2.

$\#\ \text{Index Block} = 442$
B_i

$B_{fi} = 68\ \text{Index entry/Block}$

Q.3 Find the average number of block access required to search for a record if multilevel Index is created on the Data file of Question 2.

Block factor of Index file = 68 Index entries per Block

Ist Level: Total number of Index Block = 442 Index Block

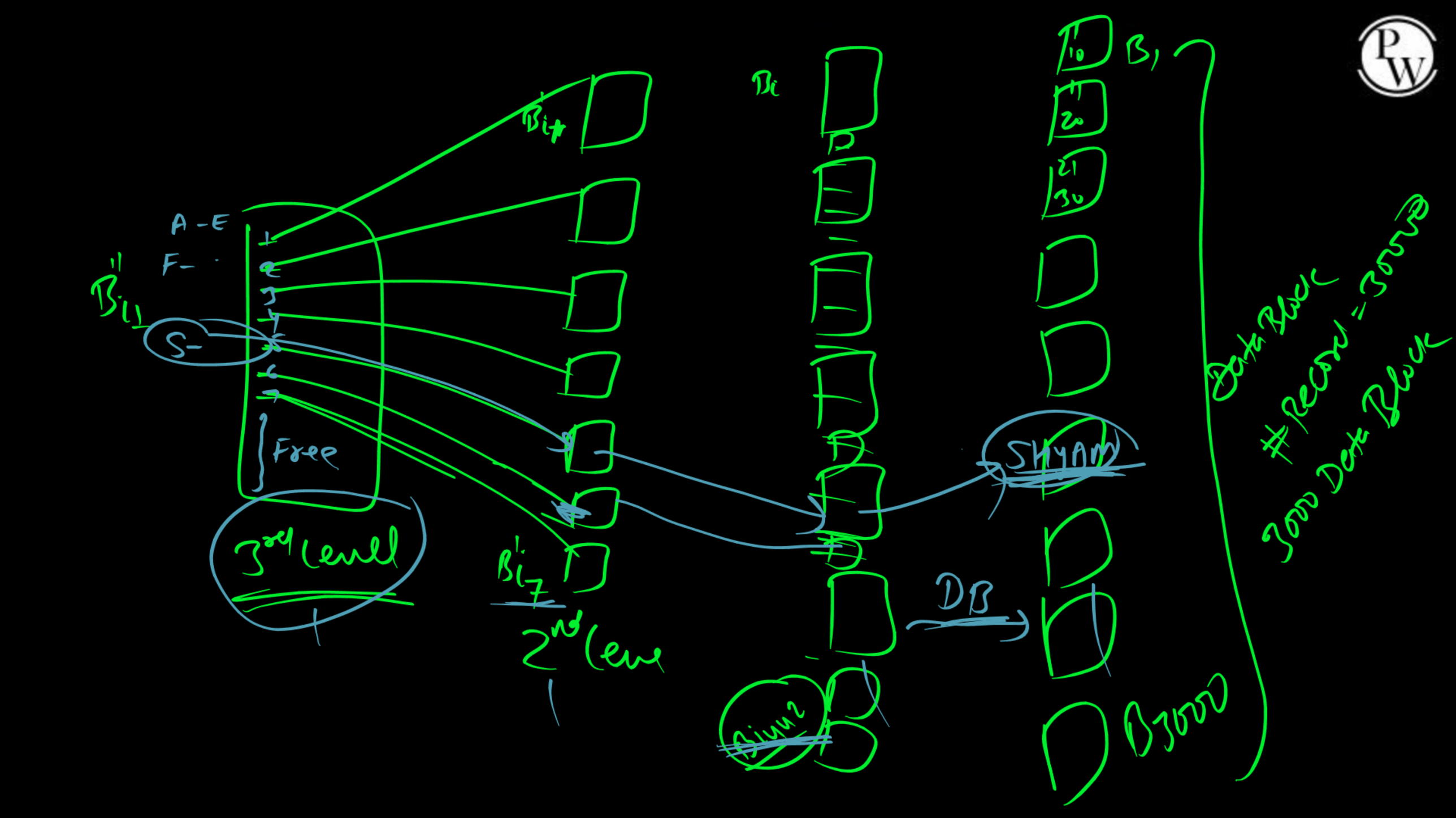
IInd Level: number of Index Records (entries) = 442 (SPARSE number of Ist level block) & Block factor = 68 Index entries for block

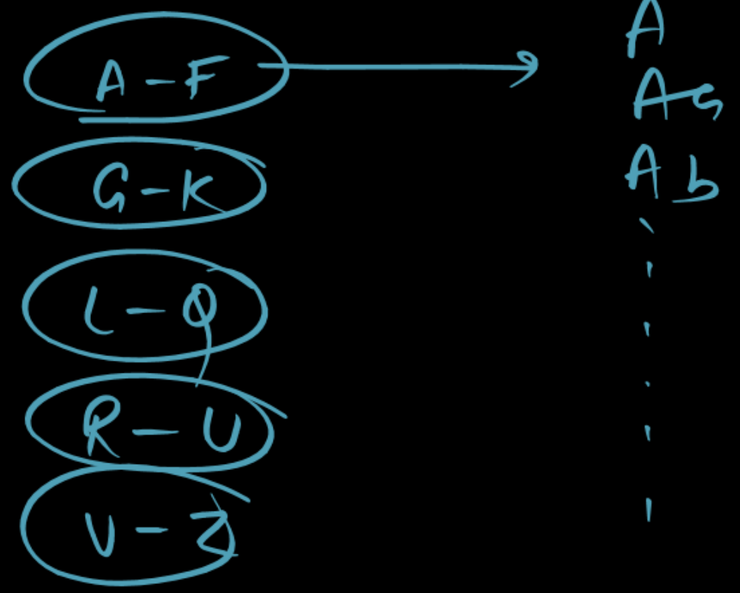
$$\text{Total number Index Block} = \left\lceil \frac{442}{68} \right\rceil = 7 \text{ Index Block}$$

IIIrd Level: Number of Index Record = 7 (number of 2nd level block)

$$\text{Total number of index Block} = \left\lceil \frac{7}{68} \right\rceil = 1$$

Average Number of block Access = 1 + 1 + 1 + 1 = 4


A - E
F -
Free
3rd Level
2nd Level
SHYAM
DB
Data Block
Record = 3000
3000 Data Block


A-F
G-K
L-Q
R-U
V-Z
A
Aa
Ab

Problem with Static Multilevel Index

$\frac{1024}{100} = 10$

$B_F = 10$

Charges 1st level

1000 Randomly

2nd level

Next level

30,000 Record

2900

3100

3000 Data Block

3100

$\frac{100}{10} =$ 100 DATA Block

(Balanced Tree)

B Tree :

ORDER = "P"

Max element = (P-1)

Min element = $\lceil \frac{P}{2} \rceil - 1$

ORDER : 5

Max element (key) = 5-1 = 4

min key = $\lceil \frac{5}{2} \rceil - 1$

$\Rightarrow 3-1 = (2)$

B Tree

ORDER: P

B_1	$K_1 R_1$	B_2	$K_2 R_2$	B_3	$K_3 R_3$ . . .	B_{P-1}	$K_{P-1} R_{P-1}$	B_P

ORDER : P

B_P : P

key = P-1

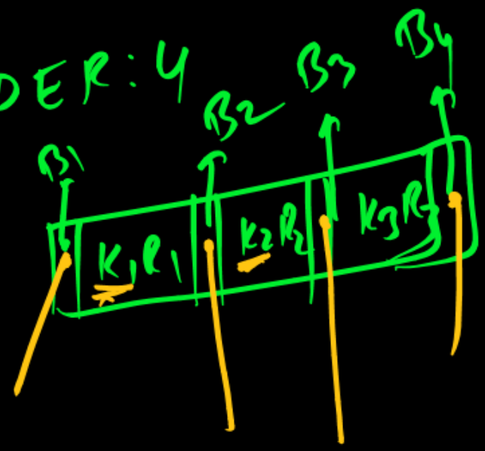
R_P = P-1

(3) ORDER : 4

B_P : 4

key : 3

RP : 3

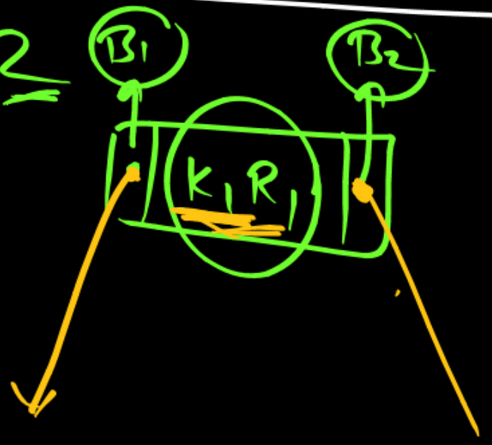


(1) Order = 2

B_P = 2

key = 1

R_P = 1

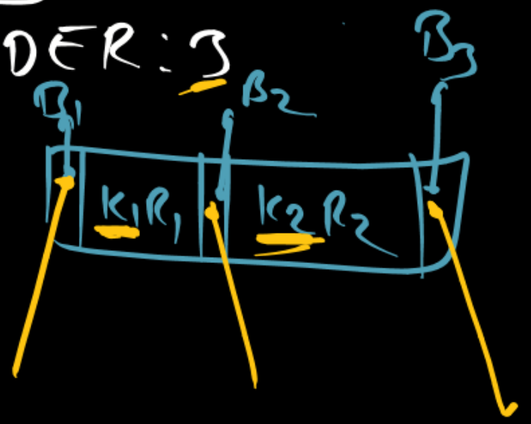


(2) ORDER : 3

B_P : 3

key : 2

RP : 2



ORDER : 5

B_P = 5

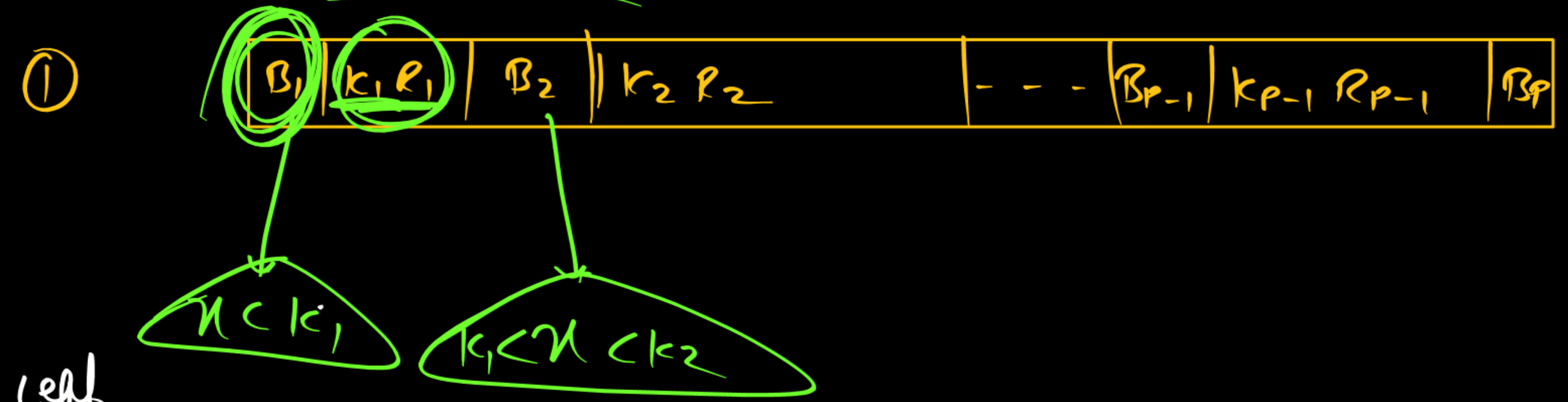
key = 4

R_P = 4

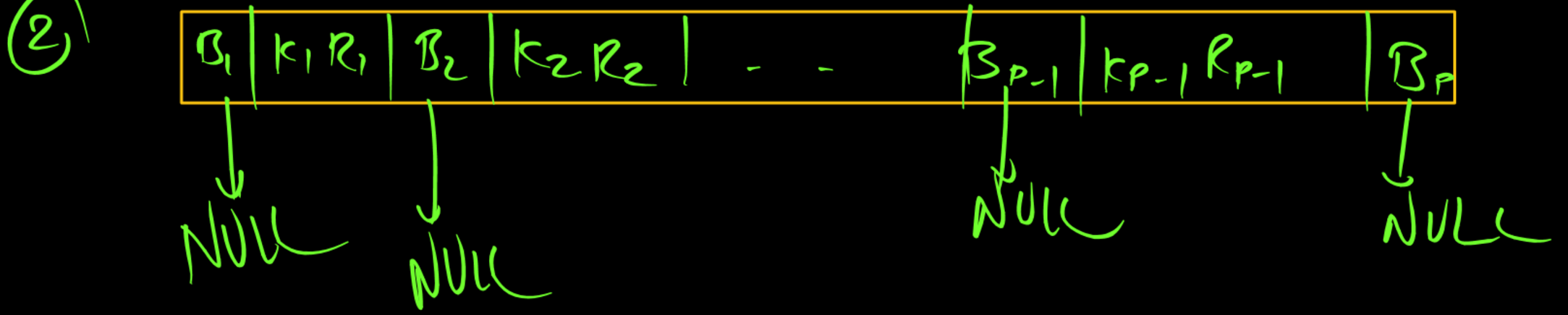
P_1	$K_1 R_1$	P_2	$K_2 R_2$	. . . . .	(P_{p-1})	$K_{p-1} R_{p-1}$	P_p

$$P * B_p + (P-1) \, key + (P-1) R_p \leq \text{Block Size}$$

B Tree : Internal Node



Leaf



ORDER

(3) Root : min 2 Block Pointer to max P Block Pointer.

(4) Internal $\Rightarrow$ $\lceil P/2 \rceil$ B_P to P Block Pointer

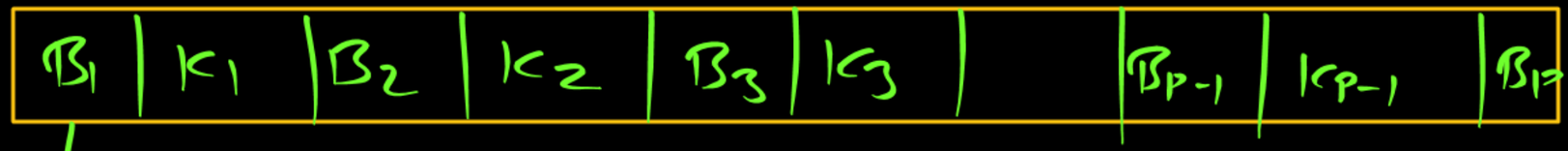
(5) All keys in Assending Order.

(6) All Leaf Node Same Level

B^+ Tree

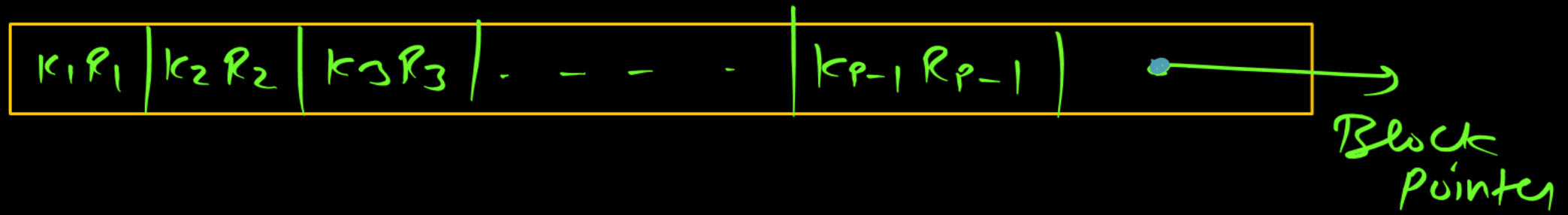
1. Internal Node : No Read Pointer
2. Key Copied at Leaf Node.
3. In leaf Node Only 1 Block Pointer

B+ Tree ① Internal Node



OR

Structure of Leaf Node



V. Imp

Internal Node

$$P * B_p + (P-1)\,|key| \leq Block\ size.$$

V. Imp.

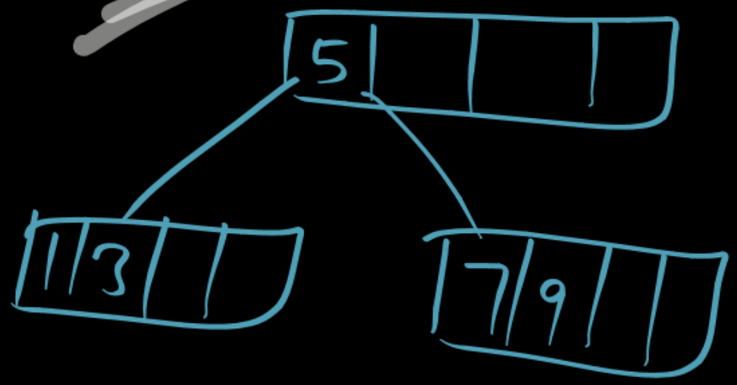
Leaf Node

$$(P-1)\,|key| + (P-1)\,P_p + 1\,B_p \leq Block\ size.$$

ORDER : 5

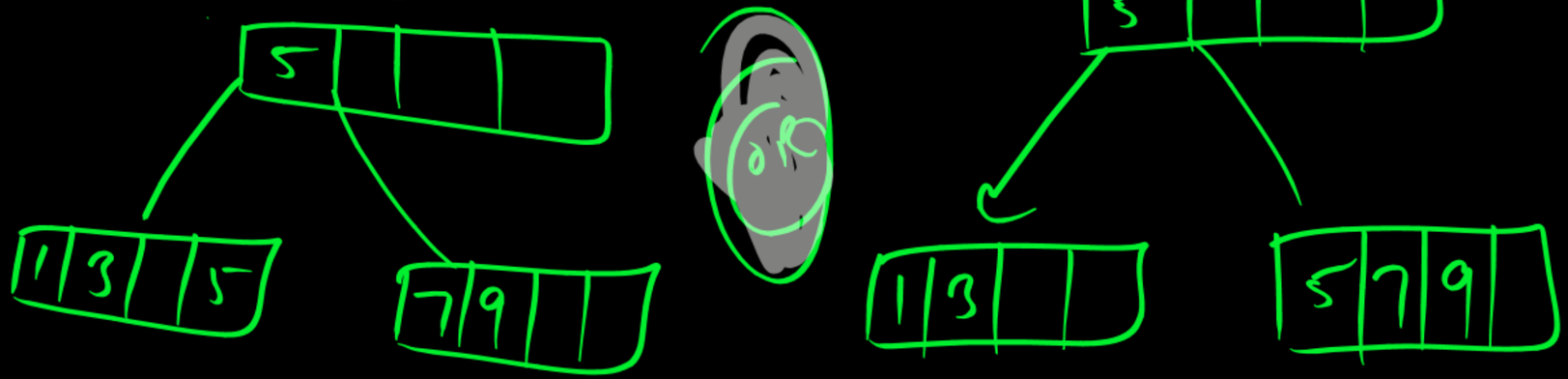
1, 3, 5, 7, 9, 11

B Tree



B Tree

B+ Tree

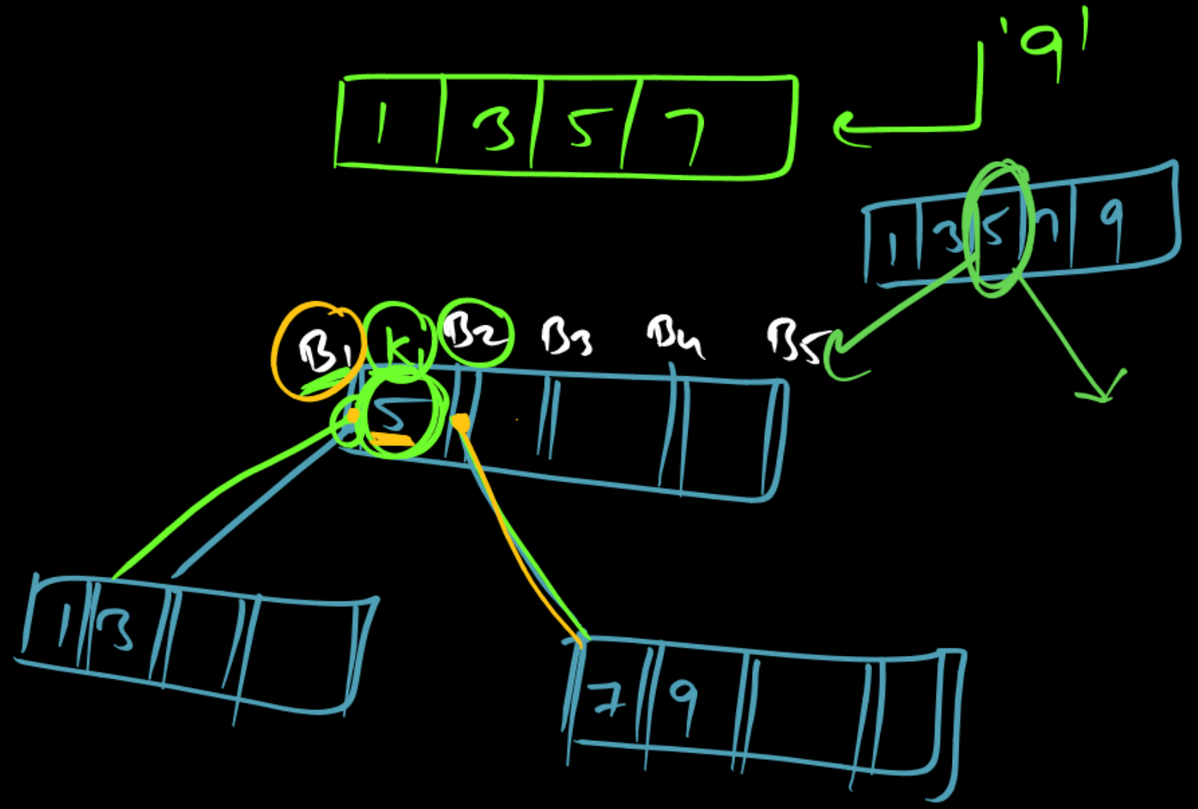


1, 3, 7, 5, 9, 11

(9) ORDER: 5

max keys = 5 - 1 = 4

Min keys = $\lceil 5/2 \rceil - 1 = 2$



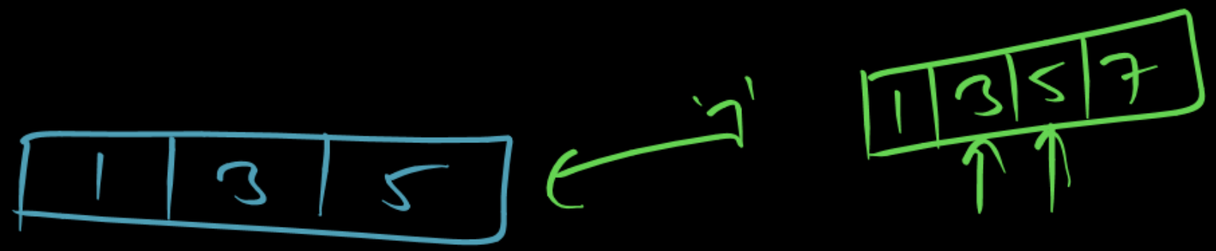
ORDER : 4

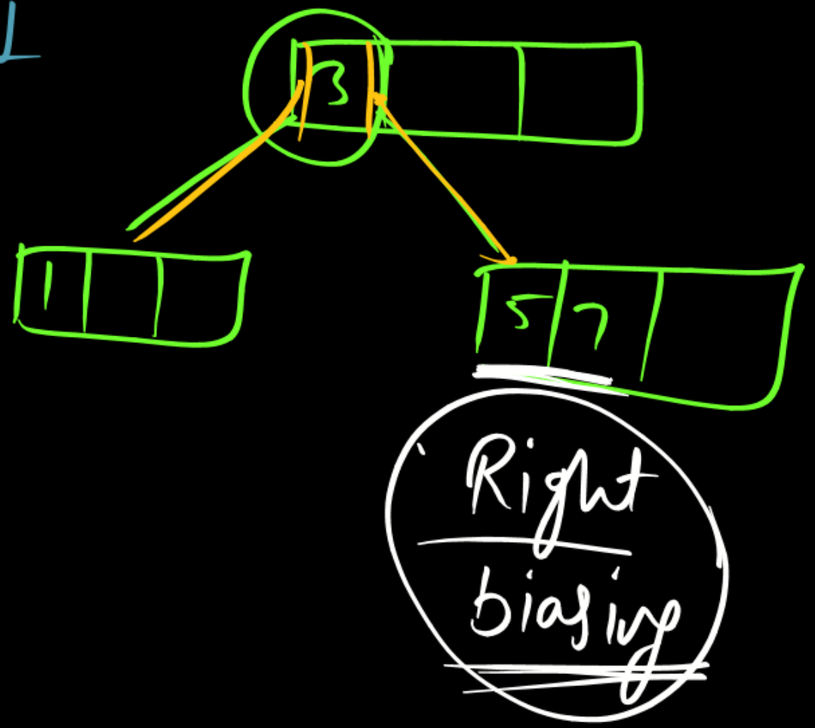
1, 3, 5, 7, 9, 11

max key = 3

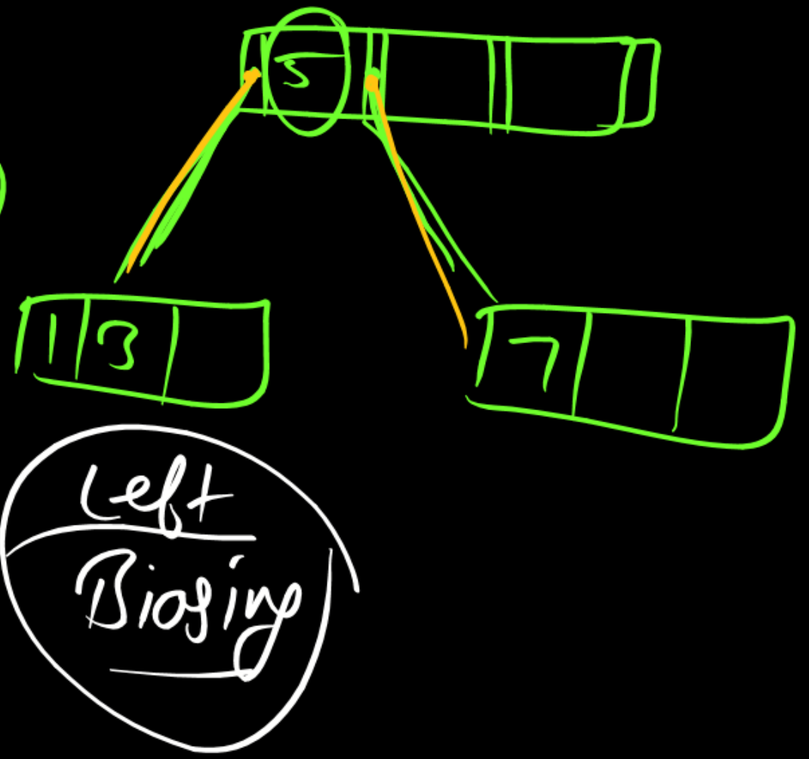
min key = $\lceil \frac{4}{2} \rceil - 1$

$\Rightarrow 1$





OR



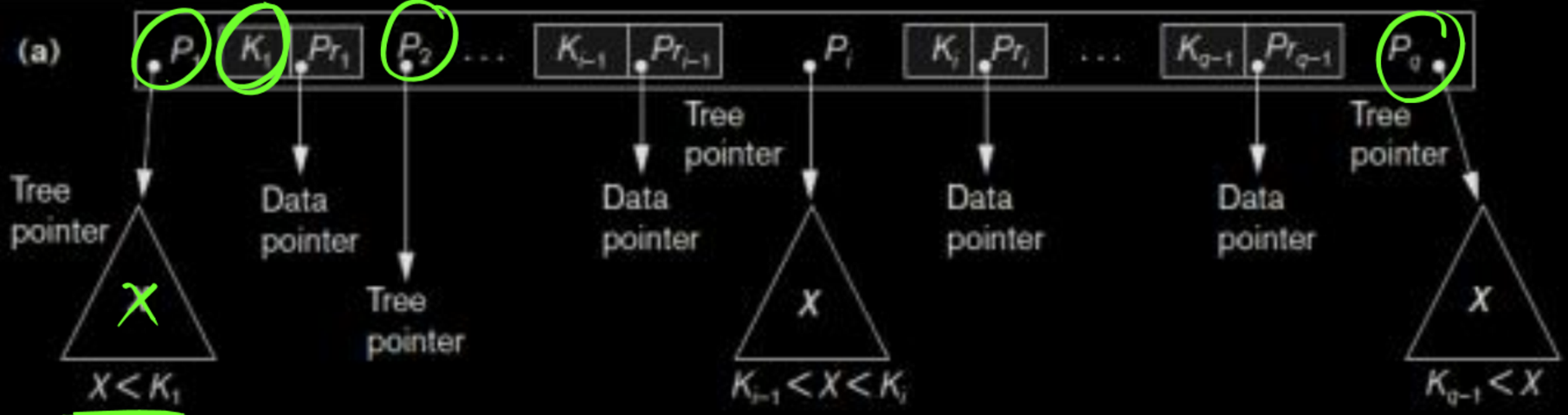
B Trees

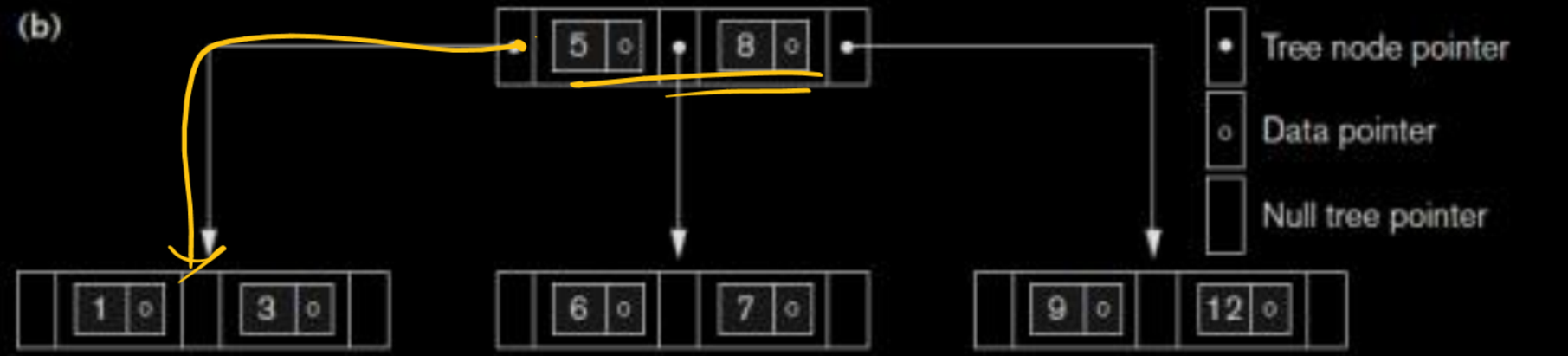
- Provide multi-level access structure
- Tree is always balanced
- Space wasted by deletion never becomes excessive
- Each node is at least half-full
- Each node in a B-tree of order p can have at most p-1 search values

B Tree Structure

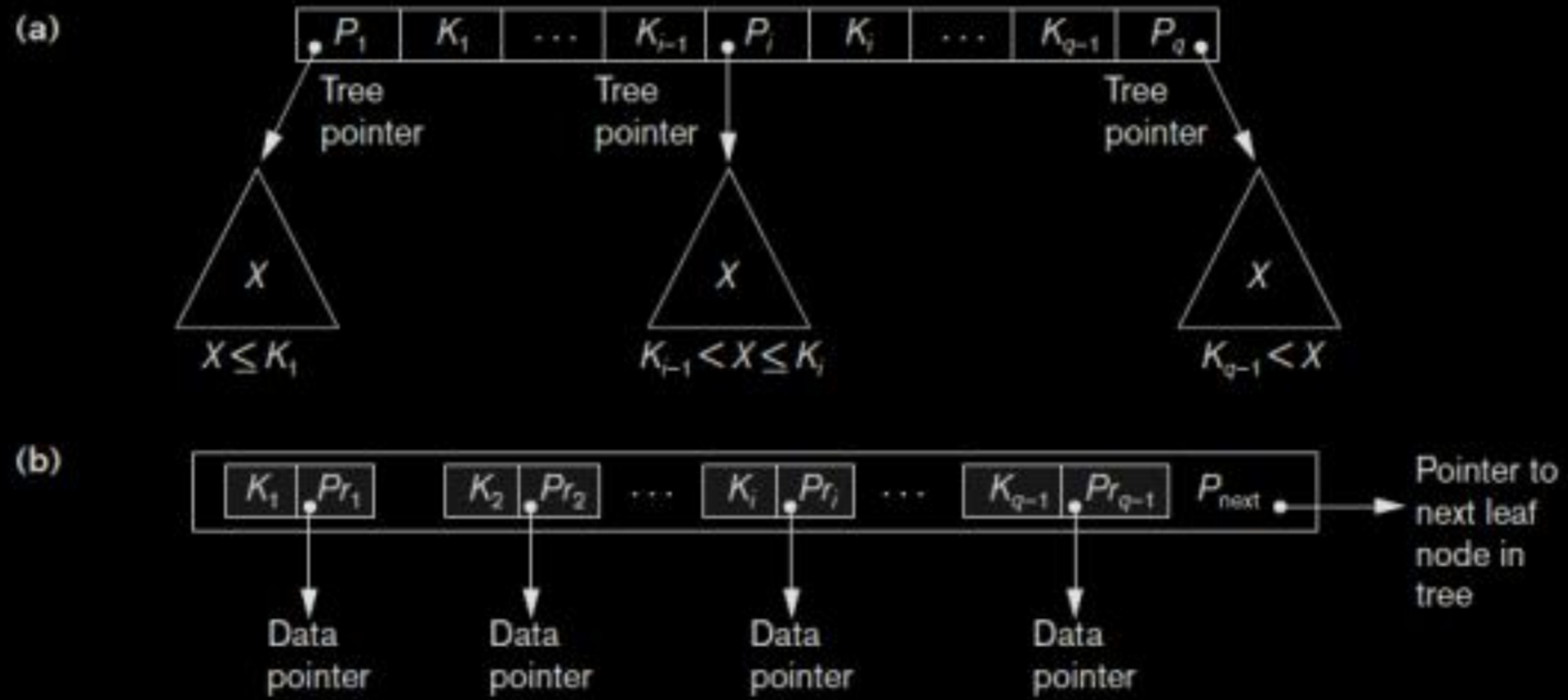
ORDER : P

B-tree structures (a) A node in a B-tree with q−1 search values (b) A B-tree of order p=3. The values were inserted in the order 8, 5, 1, 7, 3, 12, 9, 6



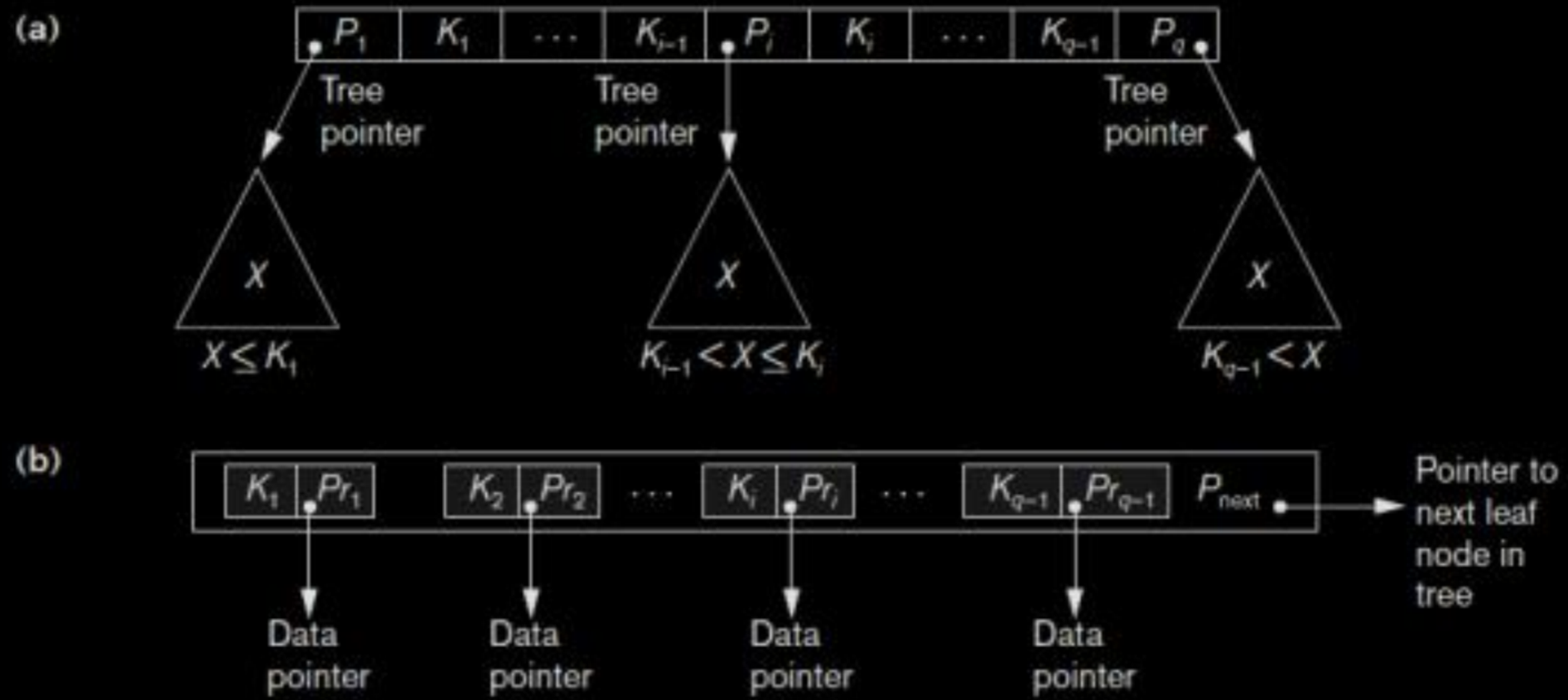


B Trees (contd...)



The nodes of a B+-tree (a) Internal node of a B+-tree with q−1 search values (b) Leaf node of a B+-tree with q−1 search values and q−1 data pointers

B Trees (contd...)



The nodes of a B+-tree (a) Internal node of a B+-tree with q−1 search values (b) Leaf node of a B+-tree with q−1 search values and q−1 data pointers

Q.1 A clustering index is defined on the fields which are of type

[GATE-2008 : 1 Mark]

A. Non-key and ordering

B. Non-key and non-ordering

C. key and ordering

D. key and non-ordering

Q.2 Consider a file of 16384 records. Each record is 32 bytes long and its key field is of size 6 bytes. The file is orderd on a non-key field, and the file organization is unspanned. The file is stored in a file system with block size 1024 bytes, and the size of block pointer is 10bytes. If the secondary index is built on the key field of the file, and a multilevel index scheme is used to store the secondary index, the number of first-level and second-level block in the multilevel index are respectively [GATE-2008 : 2 Marks]

(A) 8 and 0

(B) 128 and 6

(C) 256 and 4

(D) 512 and 5

#Records = 2^{14} , Block Size = 1024 (2^{10}) Key = 6B, B_P = 10B

One Index Record Size = 6 + 10 = (16 Byte) ⇒ 2^4 B

Block factor of Index file (B_{fi}) = $\frac{2^{10}\,B}{2^4\,B} = 2^6$ ⇒ 64 Index Entry per Block

Secondary → Dense : Total # Index Entries = # Records = 2^{14} = (16384)

Index Block = $\frac{2^{14}}{2^6} = (2^8)$ ⇒ 256 Index Block

1st Level

2nd Level

$BFi = 2^6$

Index Entries = $256 (2^8)$

Index Block = $\frac{2^8}{2^6} = 2^2 \Rightarrow$ 4 Index Block

3rd Level

$BF_i = 2^6$

Index entries = 4

→ # Index Block = $\left\lceil \frac{4}{64} \right\rceil$ ⇒ 1 Index Block

---

Total 3 level Required.

To Access a Record Avg # Block Access = $n+1$
$= 3+1 = 4$ Avg

Q.3 Consider a table T in a relational database with a key field K. A B-tree of order p is used as an access structure on K, where p denotes the maximum number of tree pointers in B-tree index node. Assume that K is 10 bytes long; disk block size is 512 bytes; each data pointer P_D is 8 bytes long and each block pointer P_B is 5 bytes long. In order for each B-tree node to fit in a single disk block, the maximum value of p is [GATE-2004 : 2 Marks]

(A) 20

(B) 22

(C) 23

(D) 32

Ikey = 10 Byte

Rp = 8 Byte

Bp = 5 Byte

BS = 512 Byte

B Tree ORDER : P

Key = 10

$R_P = 8$

$B_P = 5$

$BS = 512\ B$

$$P * B_P + (P-1)\,Key + (P-1)\,R_P \leq Block\ Size$$

$$P * 5 + (P-1)\,10 + (P-1)\,8 \leq 512$$

$$5P + 10P - 10 + 8P - 8 \leq 512$$

$$23P - 18 \leq 512$$

$$23P \leq 530$$

$P =$

$$P = \frac{530}{23} = (23.04)$$

$\Downarrow$

23 Ans

Q.4 The order of an internal node in a B^+tree index is the maximum number of children it can have. Suppose that a child pointer takes 6 bytes, the search field value takes 14 bytes, and the block size is 512 bytes. What is the order of the internal node?

(A) 24
(B) 25
(C) 26
(D) 27

$P * B_P + (P-1) key < Block\ Size$

$P * 6 + (P-1) 14 \leq 512$

$6P + 14P - 14 \leq 512$

$20P \leq 526$

$P = \frac{526}{20} = \lfloor 26.3 \rfloor = 26$

Q.5 Consider a B^+-tree in which the maximum number of keys in a node is 5. What is the minimum number of keys in any non-root node?

(A) 1
(B) 2
(C) 3
(D) 4

max lkey = 5

ORDER = 6

ORDER = lkey + 1

lkey = order − 1

Min #lkeys = $\lceil \frac{P}{2} \rceil - 1$

$\Rightarrow \lceil \frac{6}{2} \rceil - 1 \Rightarrow 3 - 1 = 2$ Ans

THANK YOU!