Dr. Vishwanath Karad
MIT WORLD PEACE UNIVERSITY | PUNE
TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

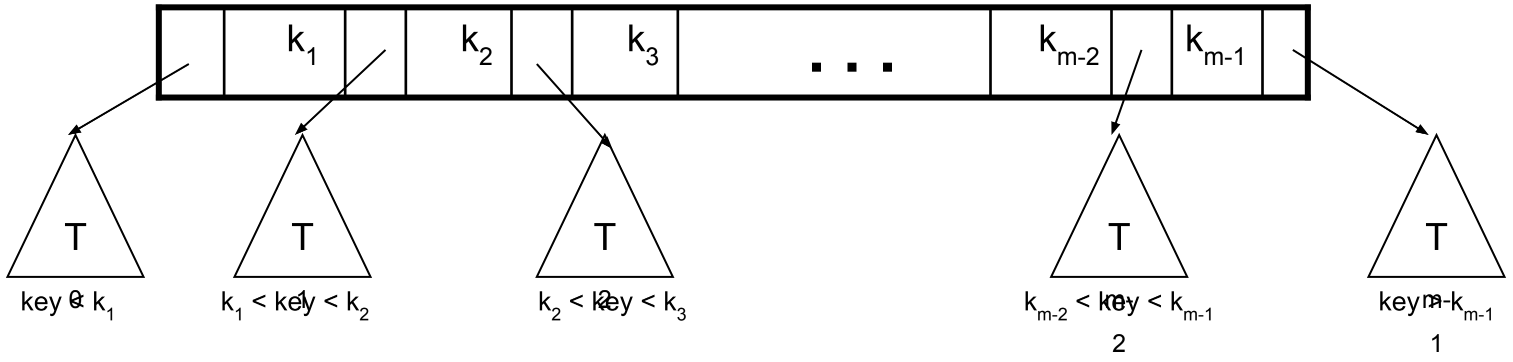
CET2001B Advanced Data Structure

School of Computer Engineering and Technology

S. Y. B. Tech

What is a Multi-way tree?

- A multi-way (or m-way) search tree of order m is a tree in which
  - Each node has at-most **m** subtrees, where the subtrees **may be empty**.
  - Each node consists of at least **1** and at most **m-1** distinct keys
  - The keys in each node are sorted.



- The keys and subtrees of a non-leaf node are ordered as:
- T0, k1, T1, k2, T2, . . . , km-1, Tm-1 such that:
  - All keys in subtree T0 are less than k1.
  - All keys in subtree Ti , 1 <= i <= m - 2, are greater than ki but less than ki+1.
  - All keys in subtree Tm-1 are greater than km-1

Definition of a B-tree

- A B-tree of order *m* is an *m*-way tree (i.e., a tree where each node may have up to *m* children) in which:
  - The number of keys in each non-leaf node is one less than the number of its children and these keys partition the keys in the children in the fashion of a search tree
  - All leaves are on the same level
  - All non-leaf nodes except the root have at least $\lceil m / 2 \rceil$ children
  - The root is either a leaf node, or it has from two to *m* children
  - A leaf node contains no more than $m – 1$ keys

Use of B-tree

- To understand use of B-Trees, we must think of huge amount of data that cannot fit in main memory.
- When the number of keys is high, the data is read from disk in the form of blocks. Disk access time is very high compared to main memory access time.
- The main idea of using B-Trees is to reduce the number of disk accesses.
- Most of the tree operations (search, insert, delete, max, min, ..etc) require O(h) disk accesses where h is height of the tree.

Create B-tree of order 3 for the data values: 78, 21, 14, 11, 97, 85, 74, 63, 45, 42, 57, 20, 16, 19, 52, 30, 21

Insertion

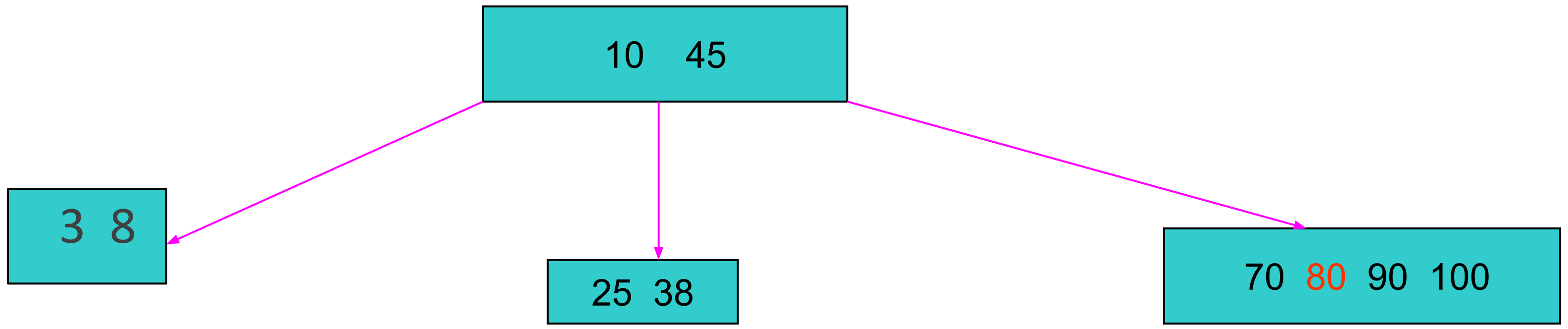
Insertion:

Find the appropriate leaf. If there is only one or two items, just add to leaf.

If no room, move middle item to parent and split remaining two items among two children.

Insertion

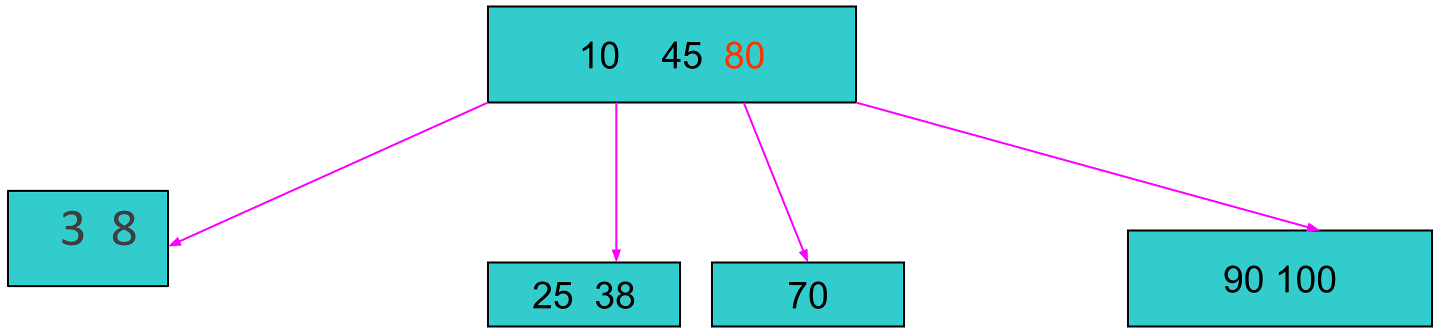
insert 80



Overflow!

Insertion

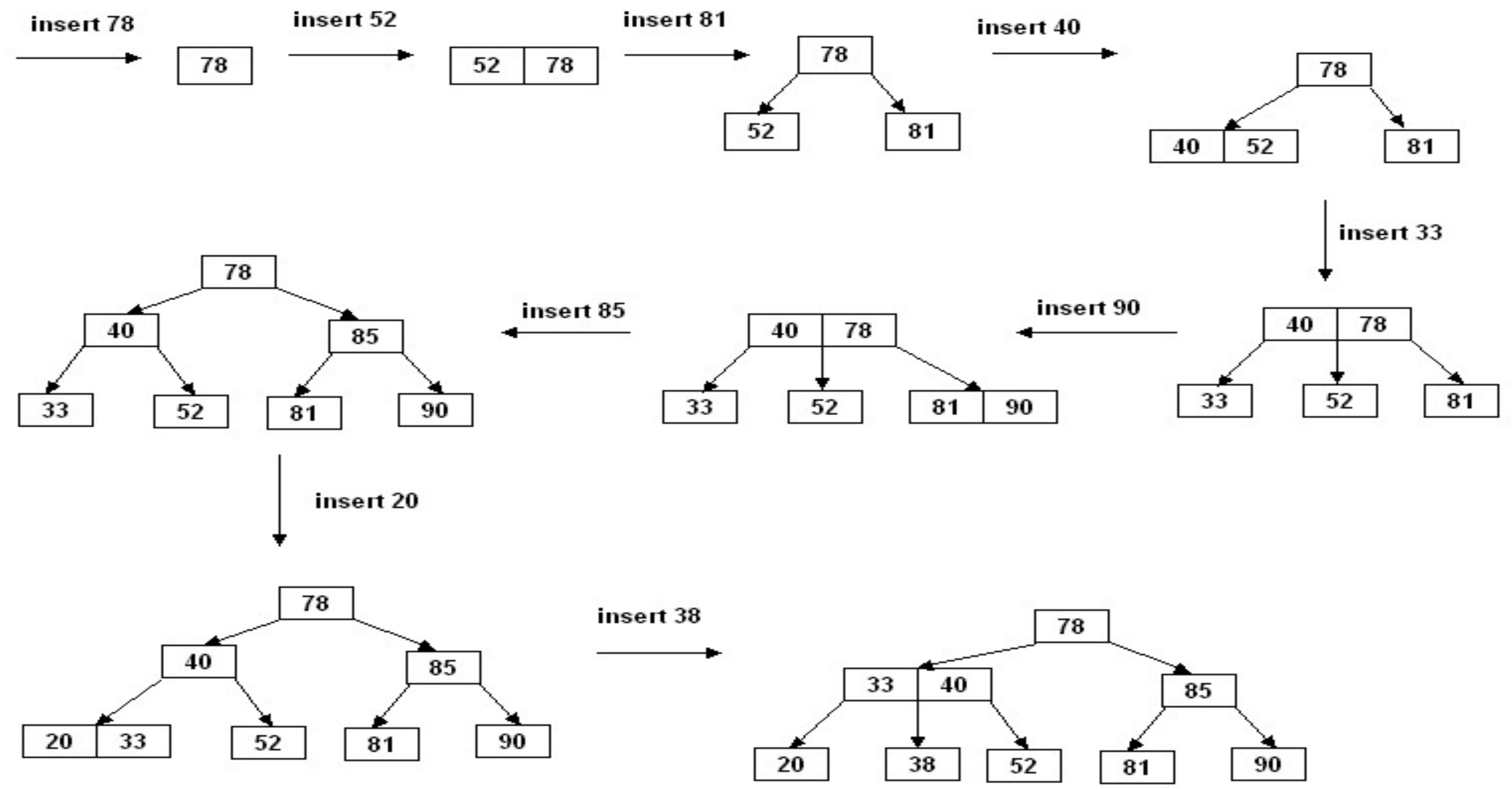
Split & move middle element to parent



Insertion in B-Trees

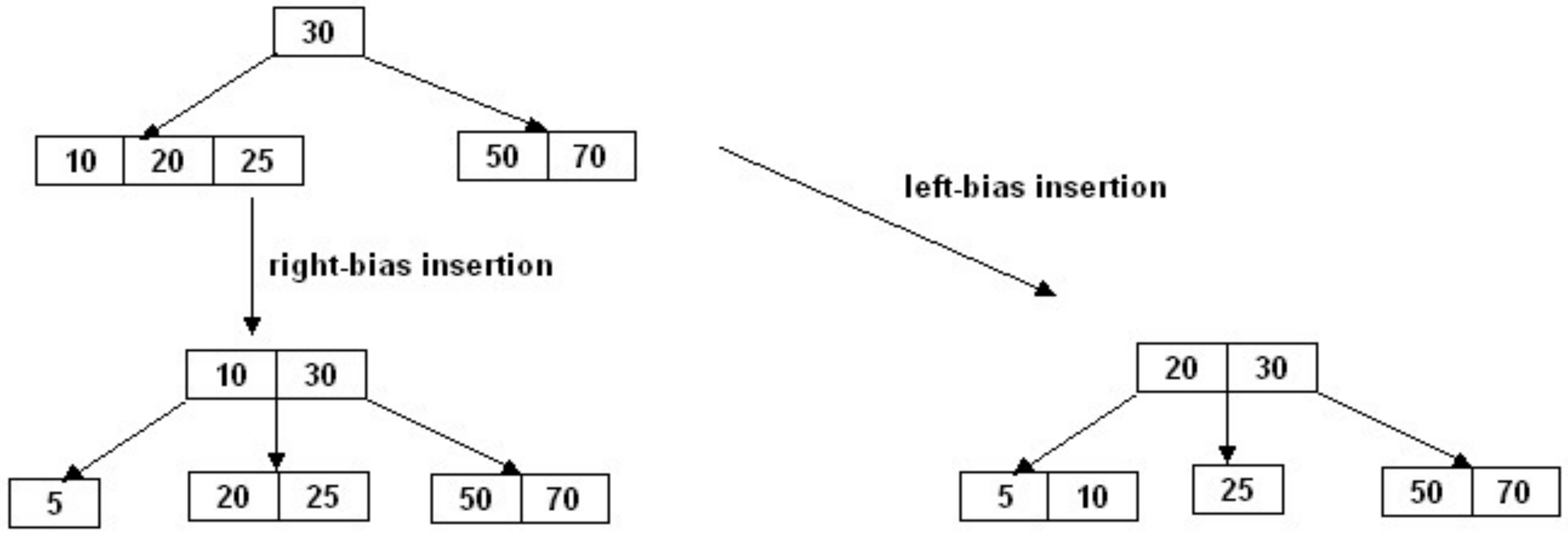
Insertion in a B-tree of <u>odd</u> order

Example: Insert the keys 78, 52, 81, 40, 33, 90, 85, 20, and 38 in this order in an initially empty B-tree of order 3



Insertion in B-Trees

- **Insertion in a B-tree of <u>even</u> order**
- At each node the insertion can be done in two different ways:
- **right-bias:** The node is split such that its right subtree has more keys than the left subtree.
- **left-bias:** The node is split such that its left subtree has more keys than the right subtree.
- **Example**: Insert the key **5** in the following B-tree of order **4**:



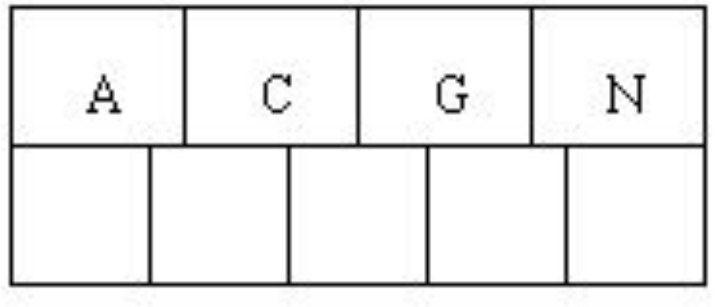
Insertion

- B-tree of order 5:
- C N G A H E K Q M F W L T Z D P R X Y S
- Order 5 means that a node can have a maximum of 5 children and 4 keys.
- All nodes other than the root must have a minimum of 2 keys.

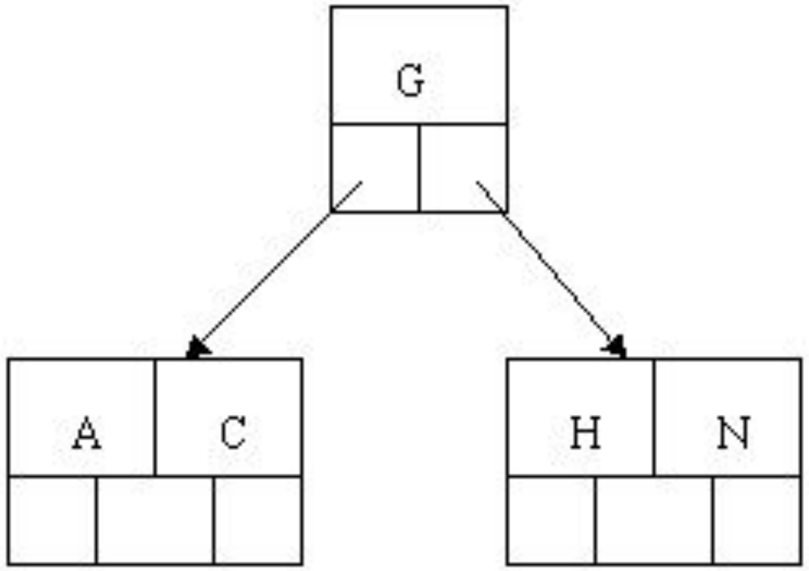
C N G A H E K Q M F W L T Z D P R X Y S

C N G A Order this A C G N

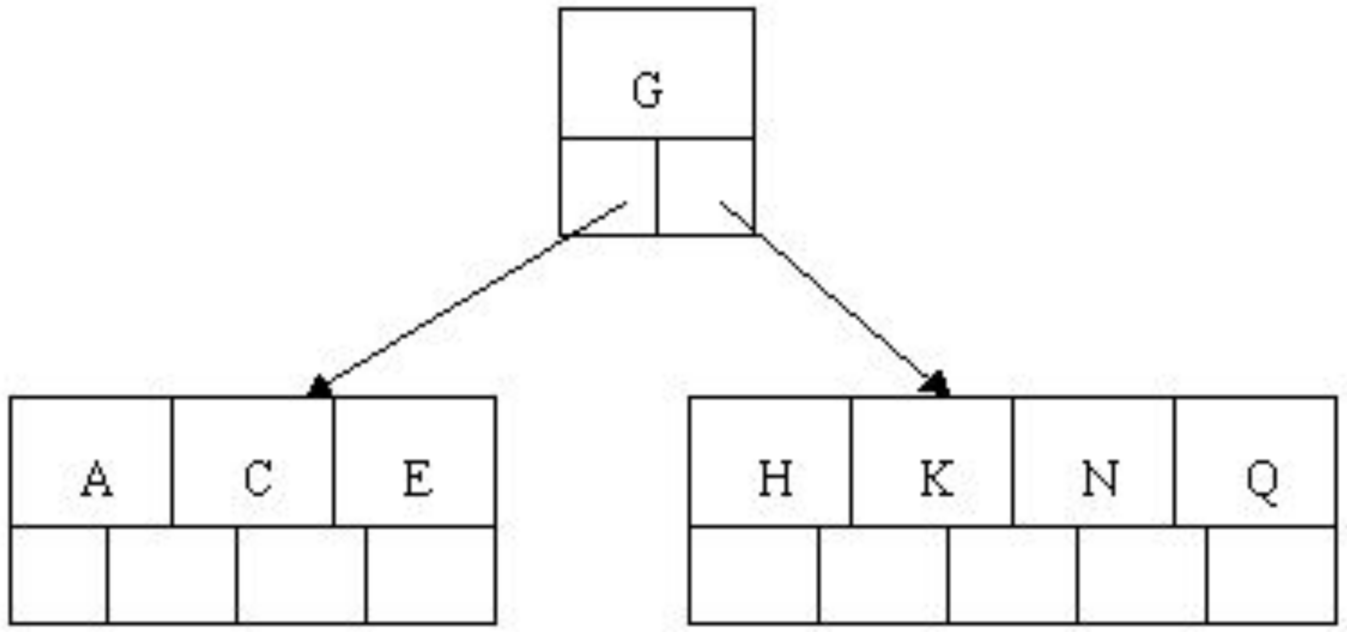
Inserting A C G N



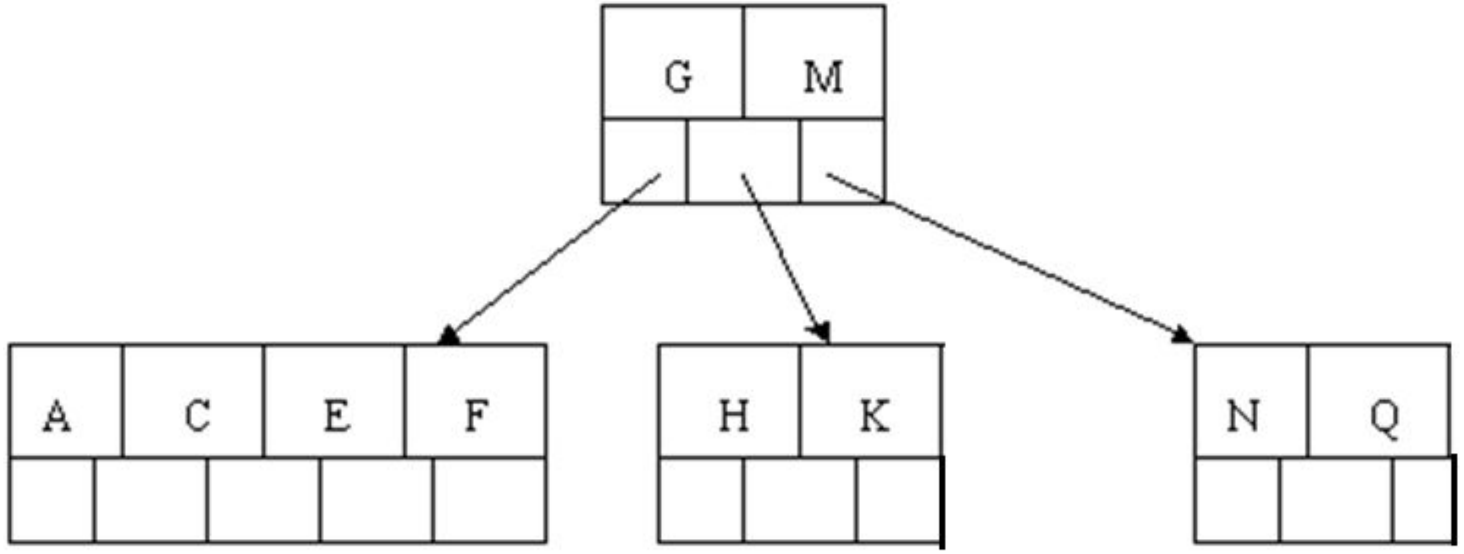
C N G A H E K Q M F W L T Z D P R X Y S



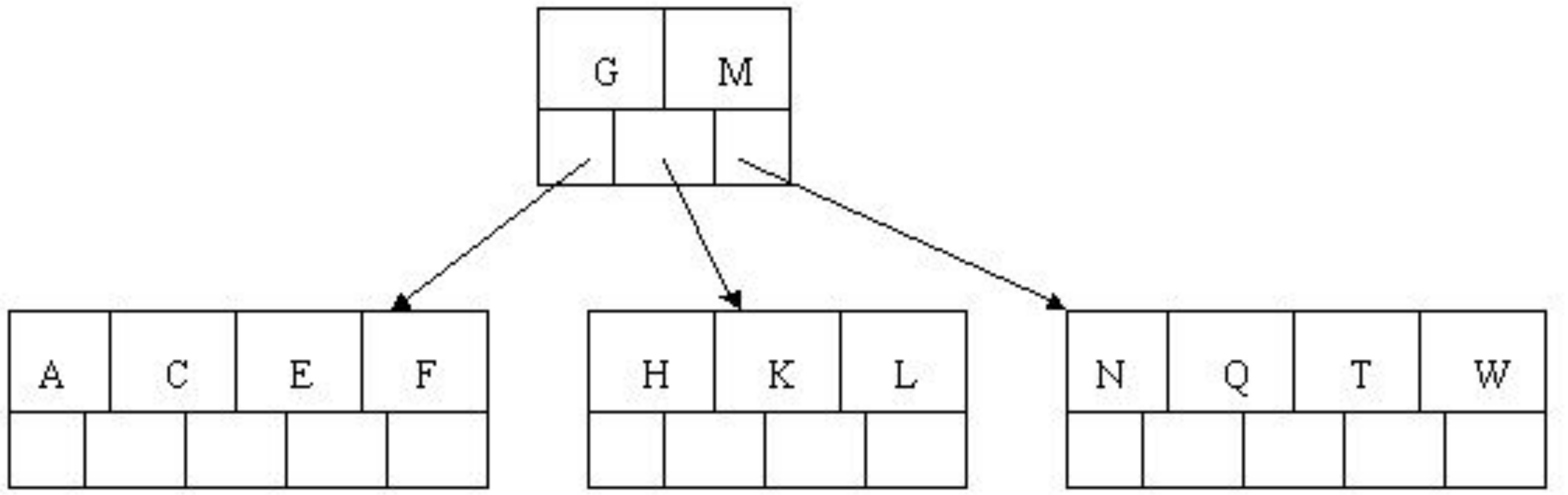
C N G A H E K Q M F W L T Z D P R X Y S



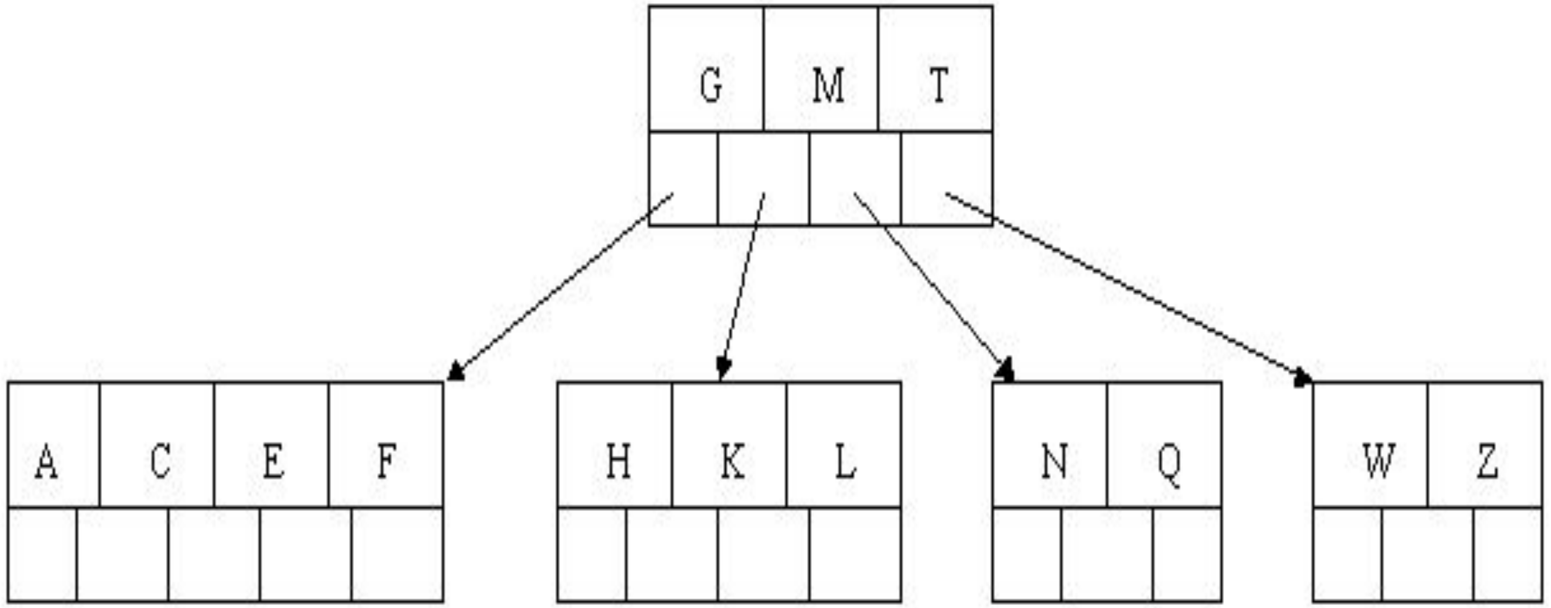
C N G A H E K Q M F W L T Z D P R X Y S



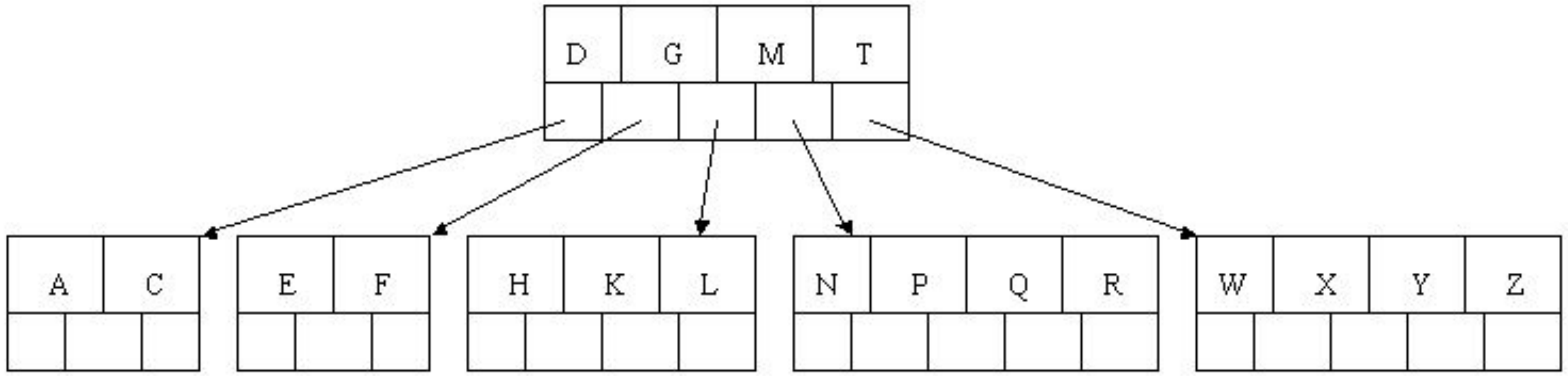
C N G A H E K Q M F W L T Z D P R X Y S



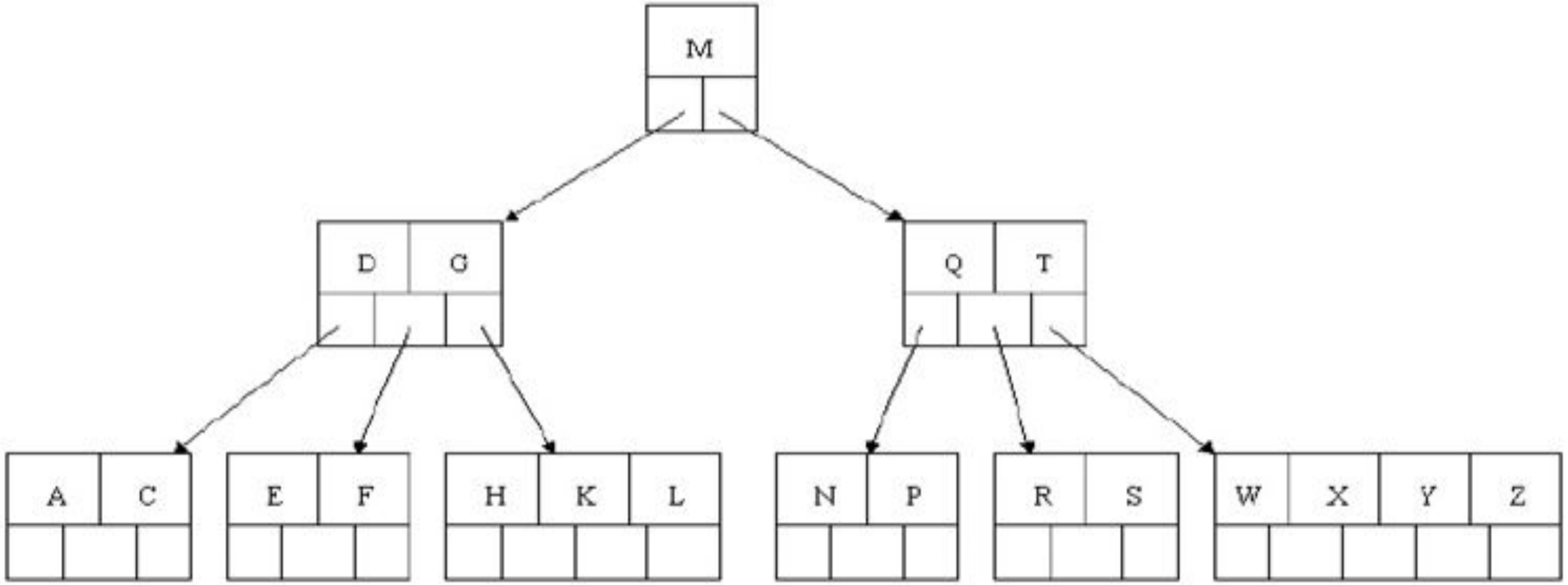
C N G A H E K Q M F W L T Z D P R X Y S



C N G A H E K Q M F W L T Z D P R X Y S



C N G A H E K Q M F W L T Z D P R X Y S



Removal from a B-tree

- During insertion, the key always goes *into* a *leaf*. For deletion we wish to remove *from* a leaf. There are three possible ways we can do this:

1 - If the key is already in a leaf node, and removing it doesn't cause that leaf node to have too few keys, then simply remove the key to be deleted.

2 - If the key is *not* in a leaf then it is guaranteed (by the nature of a B-tree) that its predecessor or successor will be in a leaf -- in this case can we delete the key and promote the predecessor or successor key to the non-leaf deleted key's position.

Removal from a B-tree

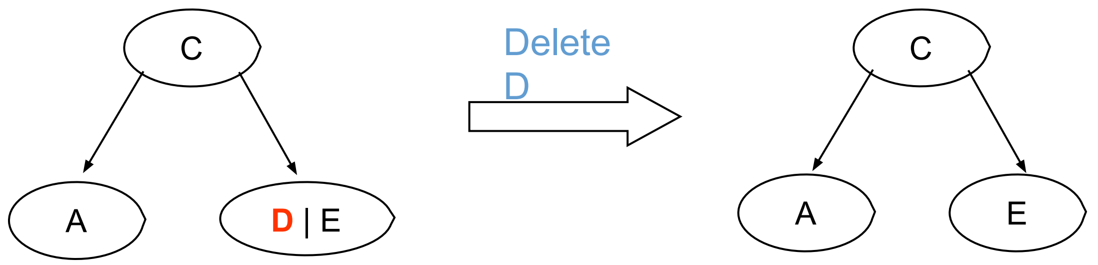
- If (1) or (2) lead to a leaf node containing less than the minimum number of keys then we have to look at the siblings immediately adjacent to the leaf in question:
  - 3: if one of them has more than the min number of keys then we can promote one of its keys to the parent and take the parent key into our lacking leaf
  - 4: if neither of them has more than the min’ number of keys then the lacking leaf and one of its neighbours can be combined with their shared parent (the opposite of promoting a key) and the new leaf will have the correct number of keys; if this step leaves the parent with too few keys then we repeat the process up to the root itself, if required

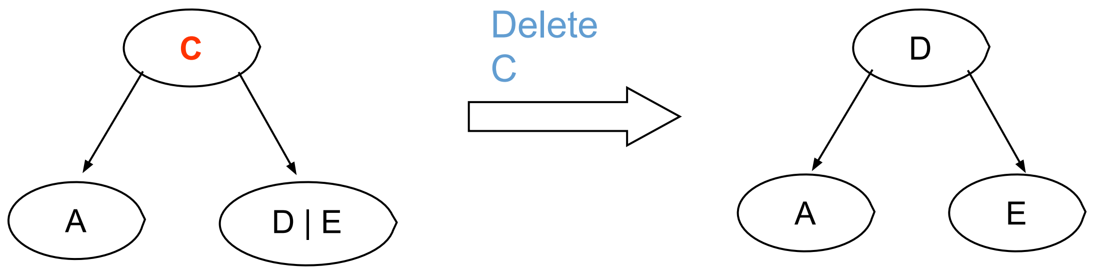
Deletion

- If the entry to be deleted is not in a leaf, swap it with its successor (or predecessor) under the natural order of the keys. Then delete the entry from the leaf.

- If leaf contains more than the minimum number of entries, then one can be deleted with no further action.

Deletion Example 1





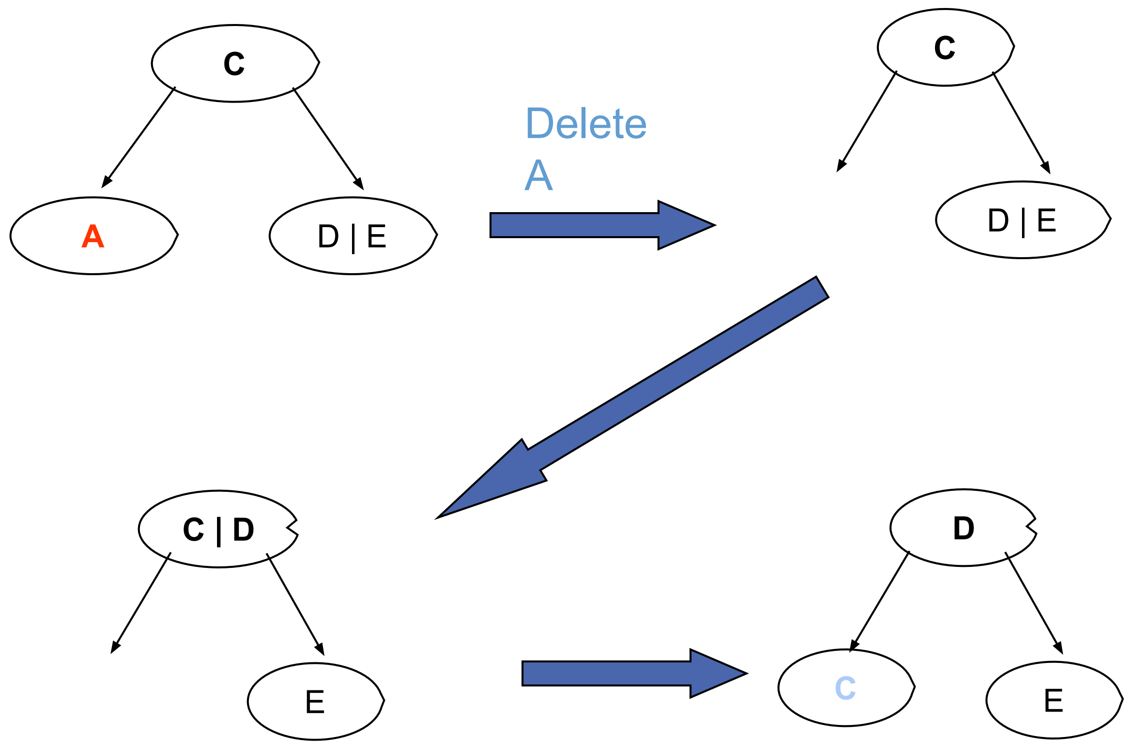
Successor is promoted, Element D

C is Deleted.

Deletion Cont...

- If the node contains the minimum number of entries, consider the two immediate siblings of the parent node:
- If one of these siblings has more than the minimum number of entries, then redistribute one entry from this sibling to the parent node, and one entry from the parent to the deficient node.
- This is a rotation which balances the nodes
- **Note: all nodes must comply with minimum entry restriction.**

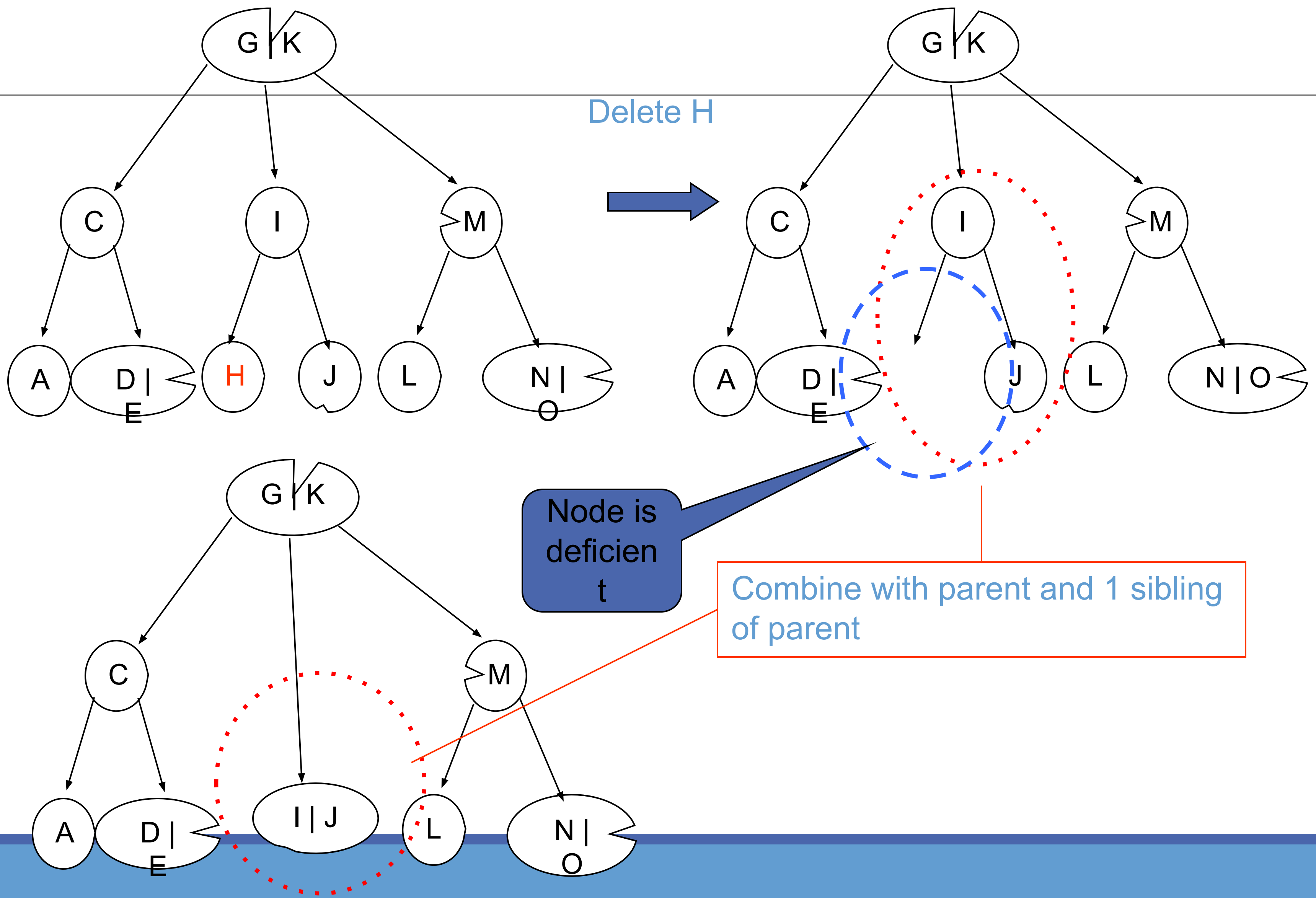
Deletion Example 2



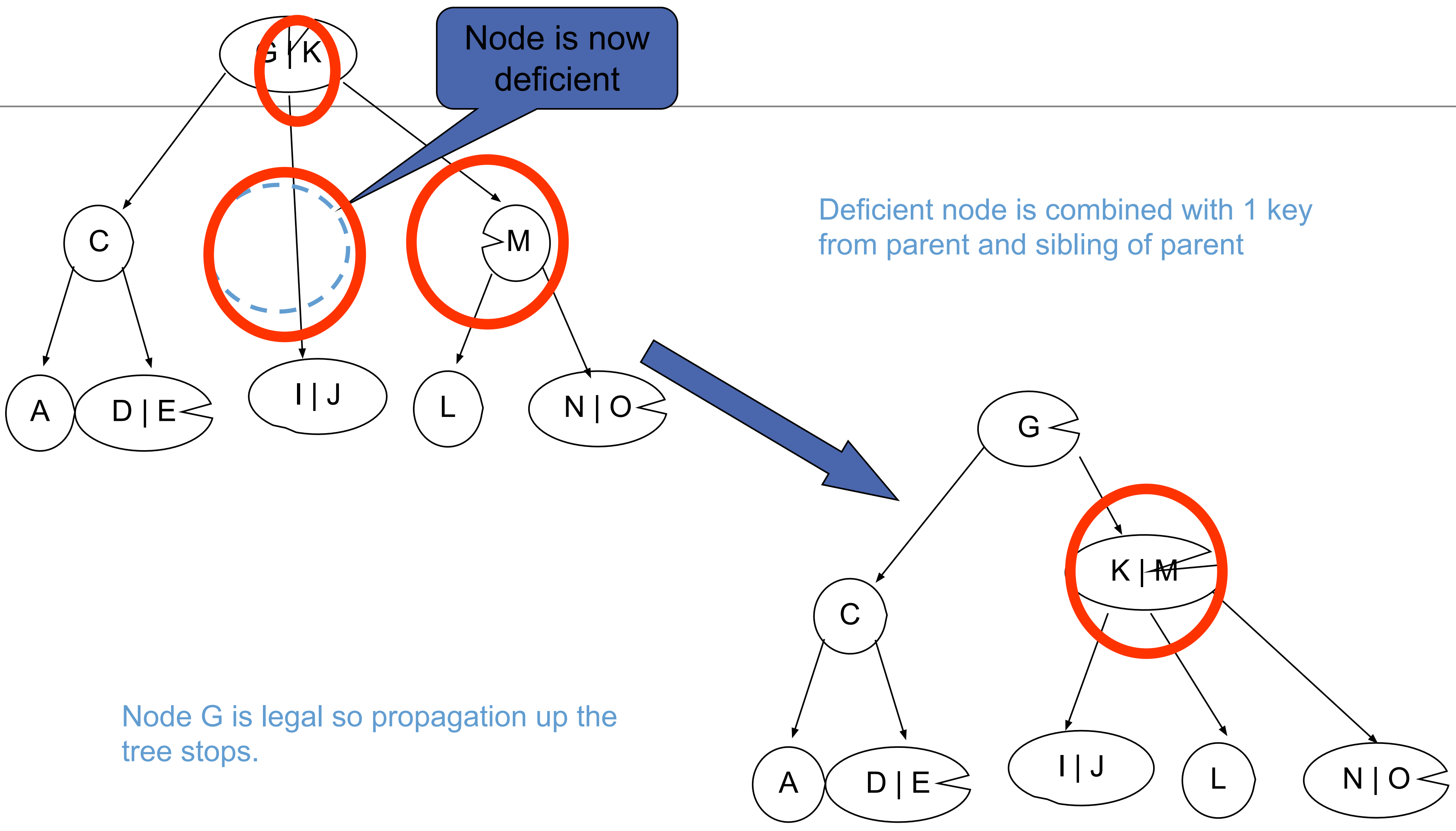
Deletion Cont...

- If both immediate siblings have exactly the minimum number of entries, then merge the deficient node with one of the immediate sibling node and one entry from the parent node.
- If this leaves the parent node with too few entries, then the process is propagated upward.

Deletion Example 3



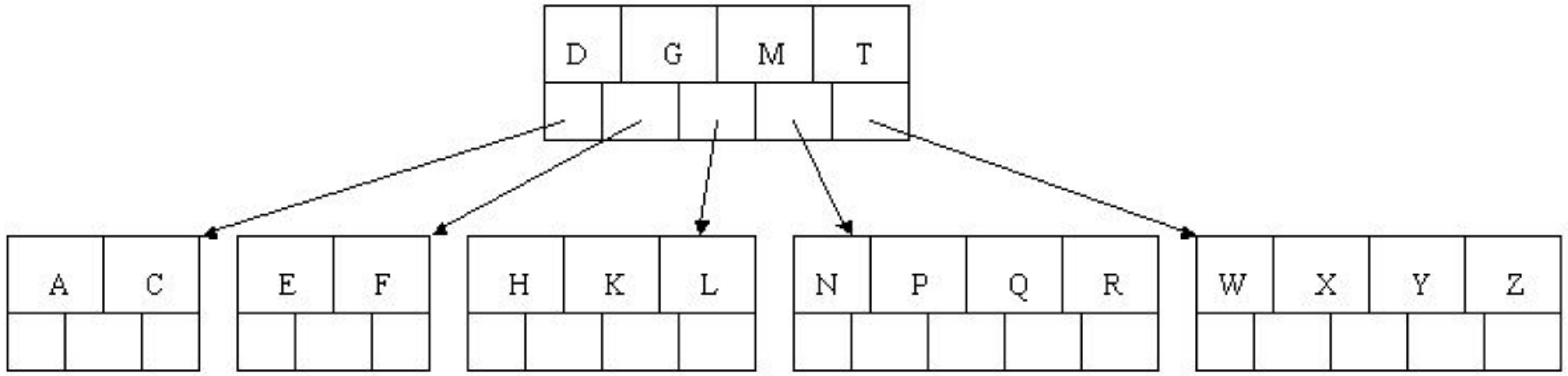
Deletion Example 3 Cont..



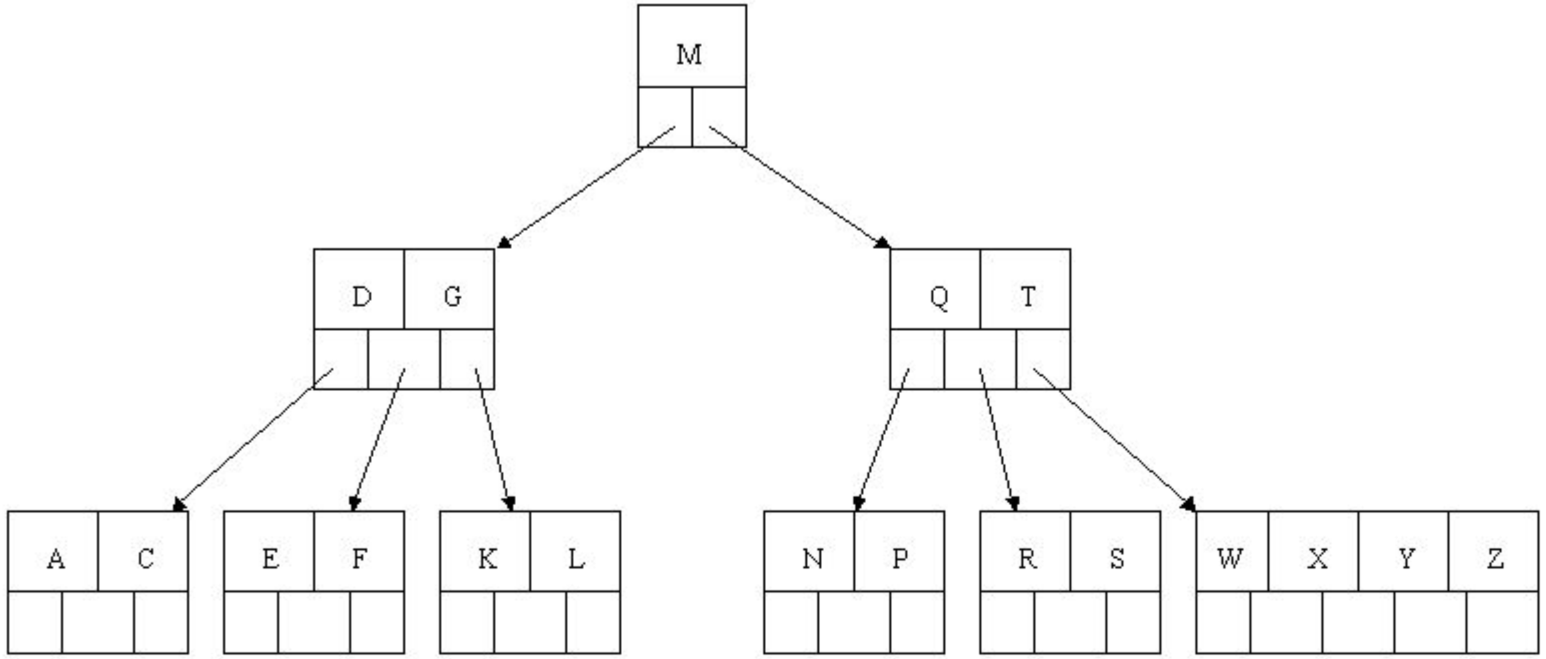
Deficient node is combined with 1 key from parent and sibling of parent

Node G is legal so propagation up the tree stops.

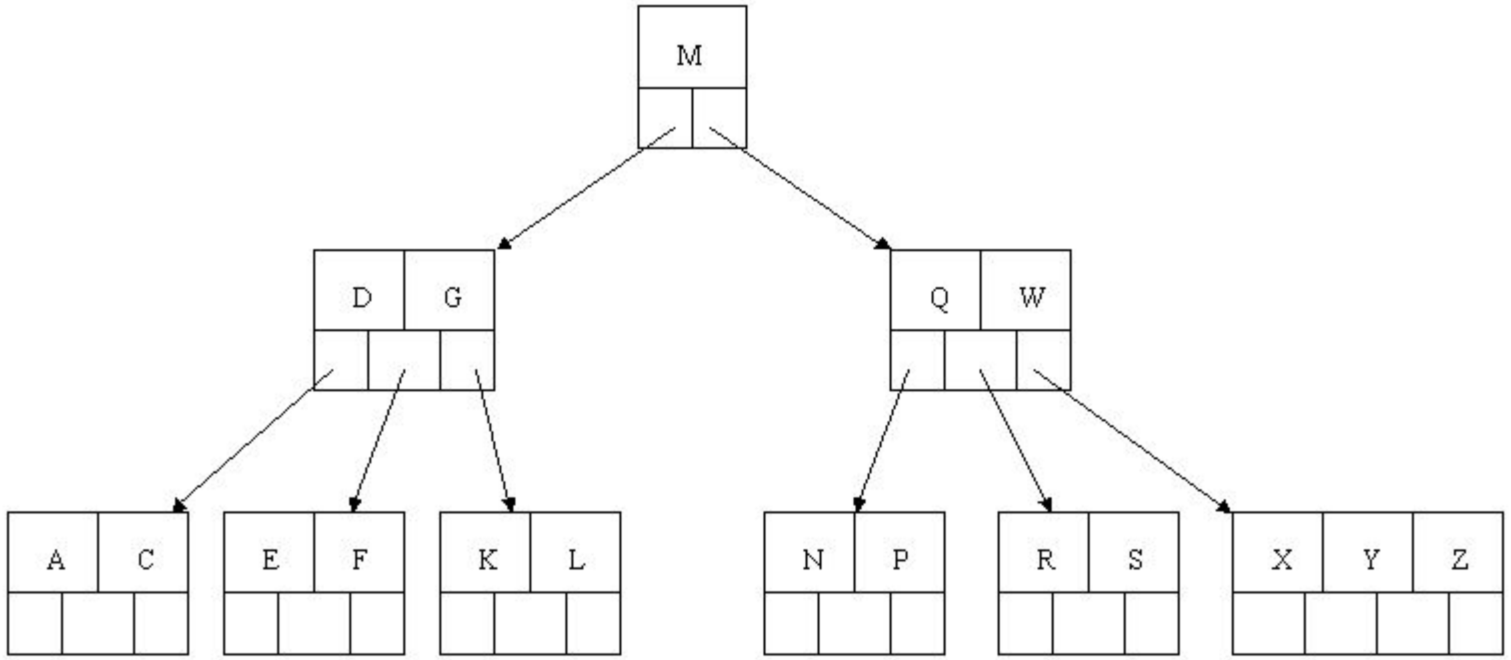
Inserting D



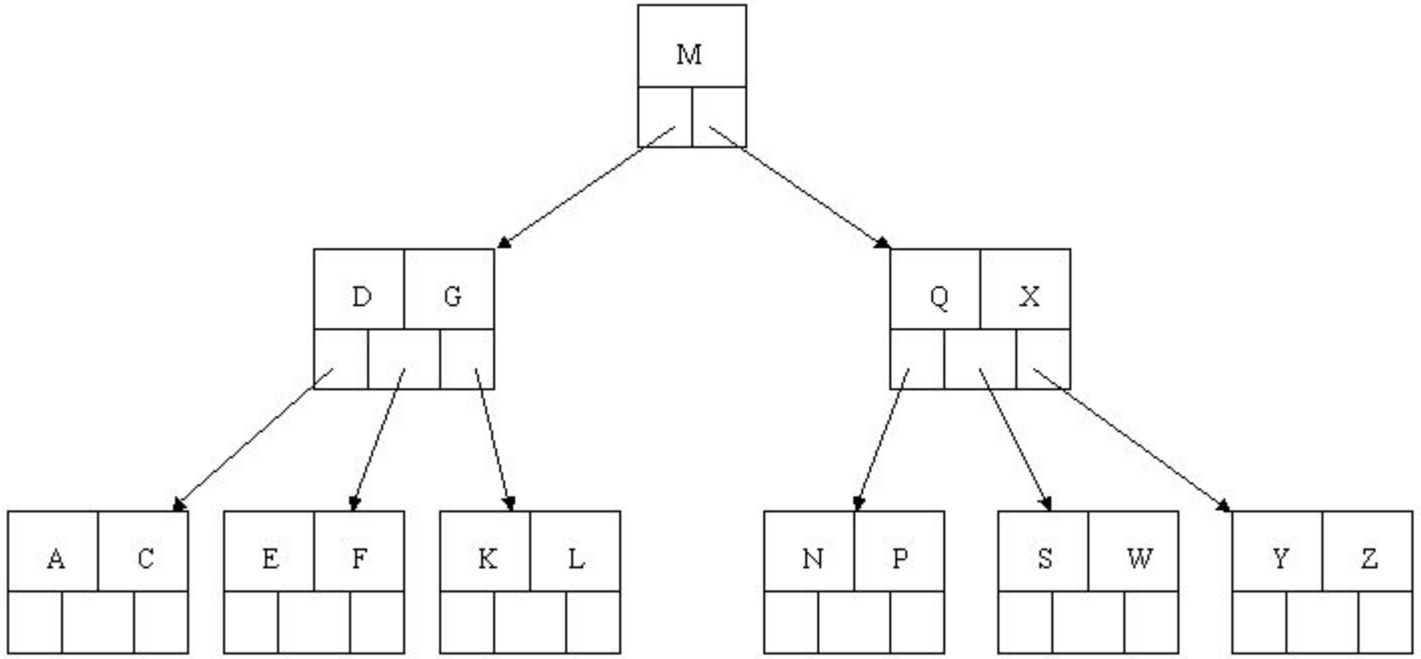
DELETION (H)



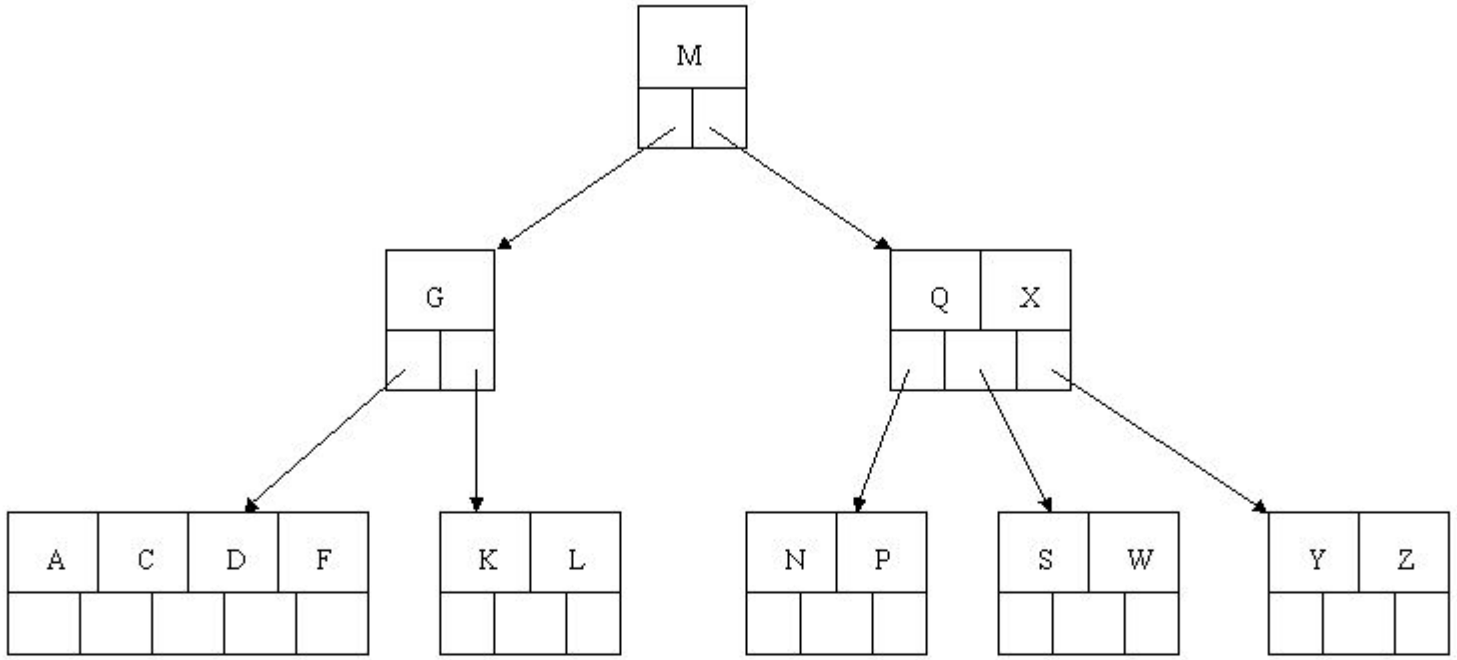
Delete T



Delete R



Delete E



Files

Introduction

- A file is a collection of records, each record having one or more fields
- The fields used to distinguish among the records are known as keys
- File organization describes the way where the records are stored in a file
- File organization is concerned with representing data records on an external storage media
- Mode of Retrieval:
  - Real time: Response time for any query should be minimized.
    - Reservation system
  - Batched: Response time is not very significant.
    - Bank account.

FILE ORGANIZATION

- Number of keys: Files having only one key & files with more than one key.
- One key-records may be stored on this key and stored sequentially either on tape or disk.(batch retrieval)
- For more than one key, sequential organization is not adequate.
- Several indices have to be maintained

Sequential File Organization

- ❖ A sequential file stores records in the order they are entered.
- ❖ The records are stored and sorted in physical contiguous blocks.
- ❖ Within each block records are in sequence.
- ❖ New records always appear at the end of file
- ❖ Records can only be read or written sequentially
- ❖ Records may be either fixed or variable in length
- ❖ Search time associated with sequential files is more because records are accessed sequentially from beginning of the file.
- ❖ Sequential files are compatible to the magnetic tape storage as shown in following figure

Record1	Record2	Record3	Record4	……	…	…	End

Primitive Operations

- Open – Opens the file and sets file pointer to the first record
- Read-next – returns next record to the user. If no record is present then EOF condition will be set.
- Close – Closes the file and terminates access to the file
- Write-next – File pointers are set to next of last record and this record is written to the file
- EOF – if EOF occurs this operation returns true otherwise it returns false
- Search – Searches for the record with a given key
- Update – The Current record is written at the same position with updated values

Add

- It is one operation algorithm
- The new record is appended at the end of the file
- One physical write is required for appending a record in a file.
- Also many records can be collected in the buffer and a block of records can be written at a time in the file
- The following steps involved in ‘Add a record’ operation
  - Open a file in append mode
  - Read a record from user
  - Write a record to the file
  - Close the file

Search

- A particular record is searched through the file using key sequentially by comparing with each record key.
- The Search starts from first record and continues till the EOF
- The following steps are involved in searching
  - Open a file in read mode
  - Read the value of the record key of the record to be searched
  - Read the next record from file
  - If record key=value, display record and go to 7
  - If not EOF, then go to 3
  - Display, ‘Record not found’
  - Close the file

Delete

- Deletion is done in two ways
  - Logical deletion
  - Physical deletion
- Logical Deletion –
  - Method 1
    - When disk files are used, records may be logically deleted by just flagging them as having been deleted.
    - This can be done by assigning a specific value to one of the attributes of the record
    - This method needs one extra field to be maintained with each record
    - The algorithm needs to modify and check the flag field during operations
  - Method 2
  - Keep a record of active and deleted records in a bit map file
  - A bit is a one dimensional array in which each bit represents a record in a file
  - The first bit refers to the first record & so on
  - Bit value ‘1’ indicates record is active and ‘0’ indicates record is deleted.

Delete

- Method 2 -
  - 1. Open a file in read + write mode
  - 2. Read the record key of the record to be deleted
  - 3. Read the next record from the file
  - 4. If record key=value
    - Change status or deleted flag as 1
    - Write record back to the same position
    - Go to Step7
  - 5. If not EOF, then go to 3
  - 6. Display ‘Record not found’
  - 7. Close the file

Delete

- Physical Deletion(pack or reorganize)
  - Copy records to another file skipping the deleted records and rename the file
  - When number of logically deleted records is high, then it is advisable to delete them physically which is known as reorganization of file.
  - The steps involved are
    - 1. Open a file in read mode
    - 2. Open ‘temporary’ file in write mode
    - 3. Read the record key of the record to be deleted
    - 4. Read the next record from file
    - 5. If record key! Value, write the record to temporary file
    - 6. If not EOF, then go to 4
    - 7. Close both the files
    - 8. Delete the original file
    - 9. Rename temporary file as original file
    - 10. Close the file

Updation

❖ A record is updated when one or more fields is changed by modifying the information. The following steps are involved in updation:

```
1. Open a file in write mode
2. Read the record key of the record to be modified
3. Read the new attributes of the record to be modified
4. If record key=value, modify record and go to 7
5. If not EOF, then go to 4
6. Display ‘Record not found’
7. Close the file
```

Advantages

- ❖ Owing to its simplicity, it can be used with a variety of media including magnetic tapes and disks.
- ❖ It is compatible with variable length records, while most other file organizations are not.
- ❖ Security is ensured with ease
- ❖ For a run in which a high proportion of a block is hit, sequential file is efficient specially when processed in batches.

Drawbacks

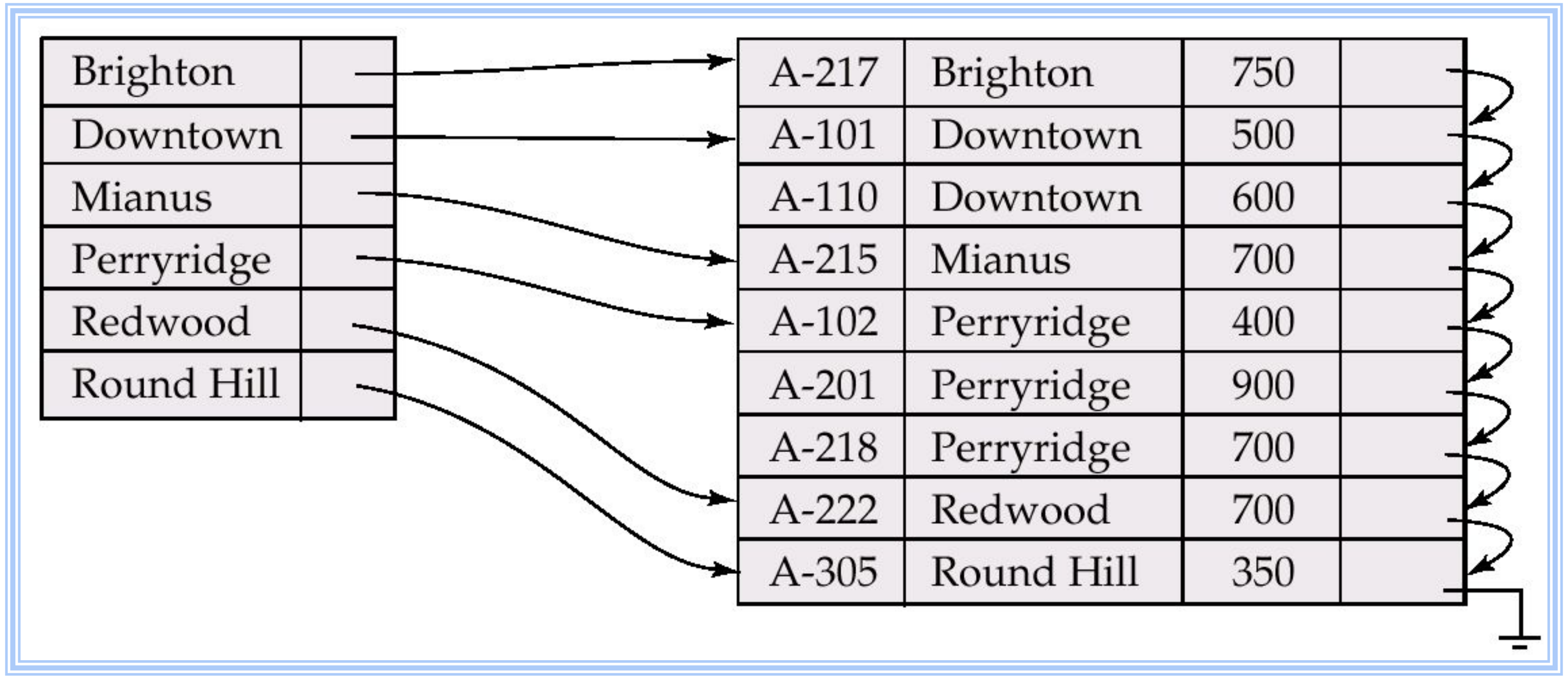
- Insertion and deletion of records in between positions cause huge data movements.
- Accessing any record requires a pass through all the preceding records which is time consuming. So, searching a record also takes more time
- Needs reorganization of the file from time to time. If too many records are deleted logically then the file must be reorganized to free the space occupied by unwanted records.

Indexing

- An index, whether it is a book or a data file index (in computer memory), is based on the basic concepts such as keys and reference fields

- The index to a book provides a way to find a topic quickly

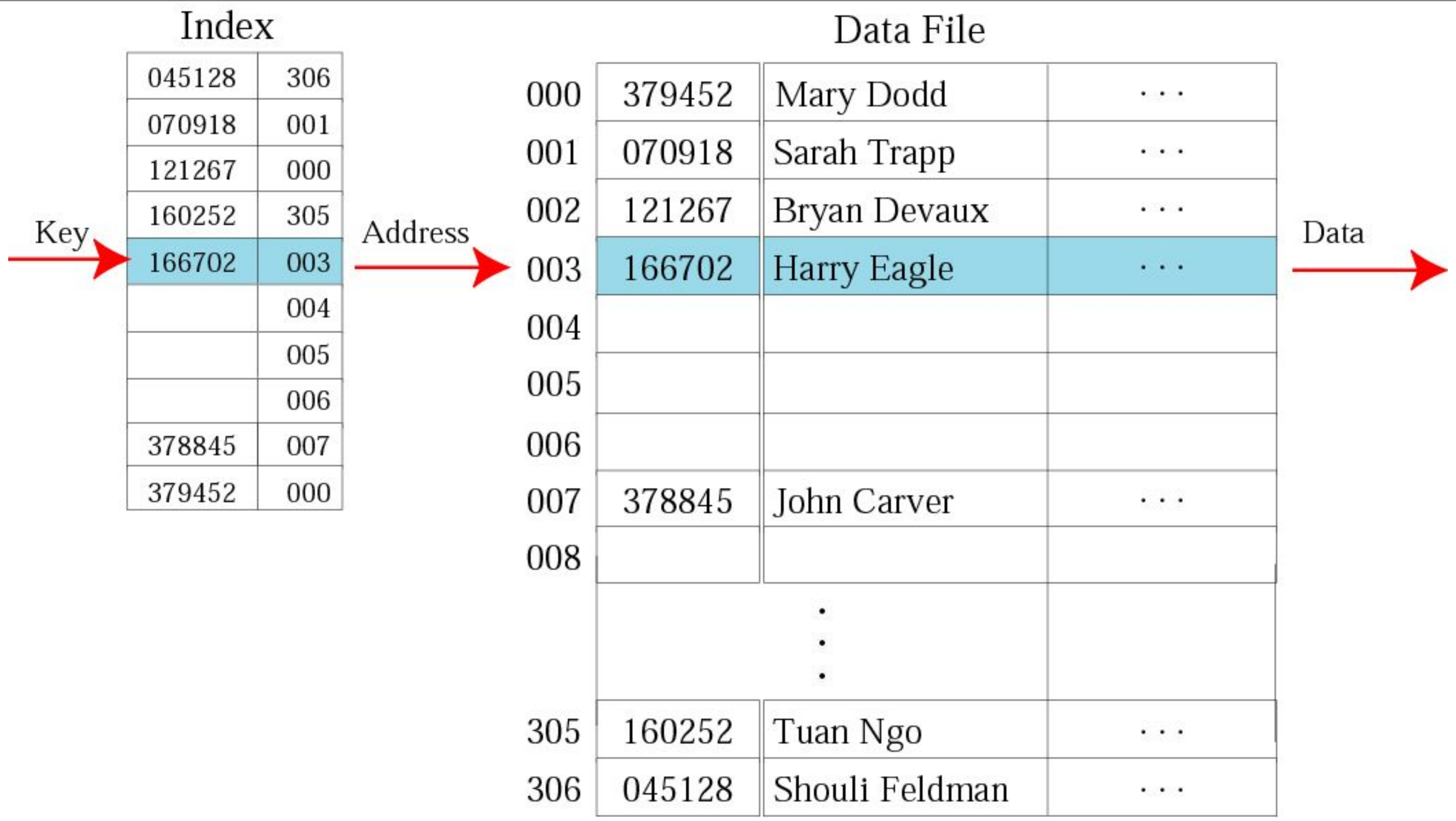
Index Files Example



Brighton	
Downtown	
Mianus	
Perryridge	
Redwood	
Round Hill	

A-217	Brighton	750	
A-101	Downtown	500	
A-110	Downtown	600	
A-215	Mianus	700	
A-102	Perryridge	400	
A-201	Perryridge	900	
A-218	Perryridge	700	
A-222	Redwood	700	
A-305	Round Hill	350	

Logical view of an indexed file



Key	Address
045128	306
070918	001
121267	000
160252	305
166702	003
	004
	005
	006
378845	007
379452	000

000	379452	Mary Dodd	· · ·
001	070918	Sarah Trapp	· · ·
002	121267	Bryan Devaux	· · ·
003	166702	Harry Eagle	· · ·
004			
005			
006			
007	378845	John Carver	· · ·
008			
	⋮		
305	160252	Tuan Ngo	· · ·
306	045128	Shouli Feldman	· · ·

Indexed Sequential File Organization

- An index file contains records ordered by record key
- Each record contains a field that contains record key
- Record key uniquely identifies the record and determines the sequence in which it is accessed with respect to other records.
- An index file can use alternate indices that is record keys that let you the file using a different logical arrangement of the records.
- For example: file can be accessed through employee dept instead of emp number
- Index file read & written sequentially. Index is a data structure that allows particular record in a file to be located more quickly.

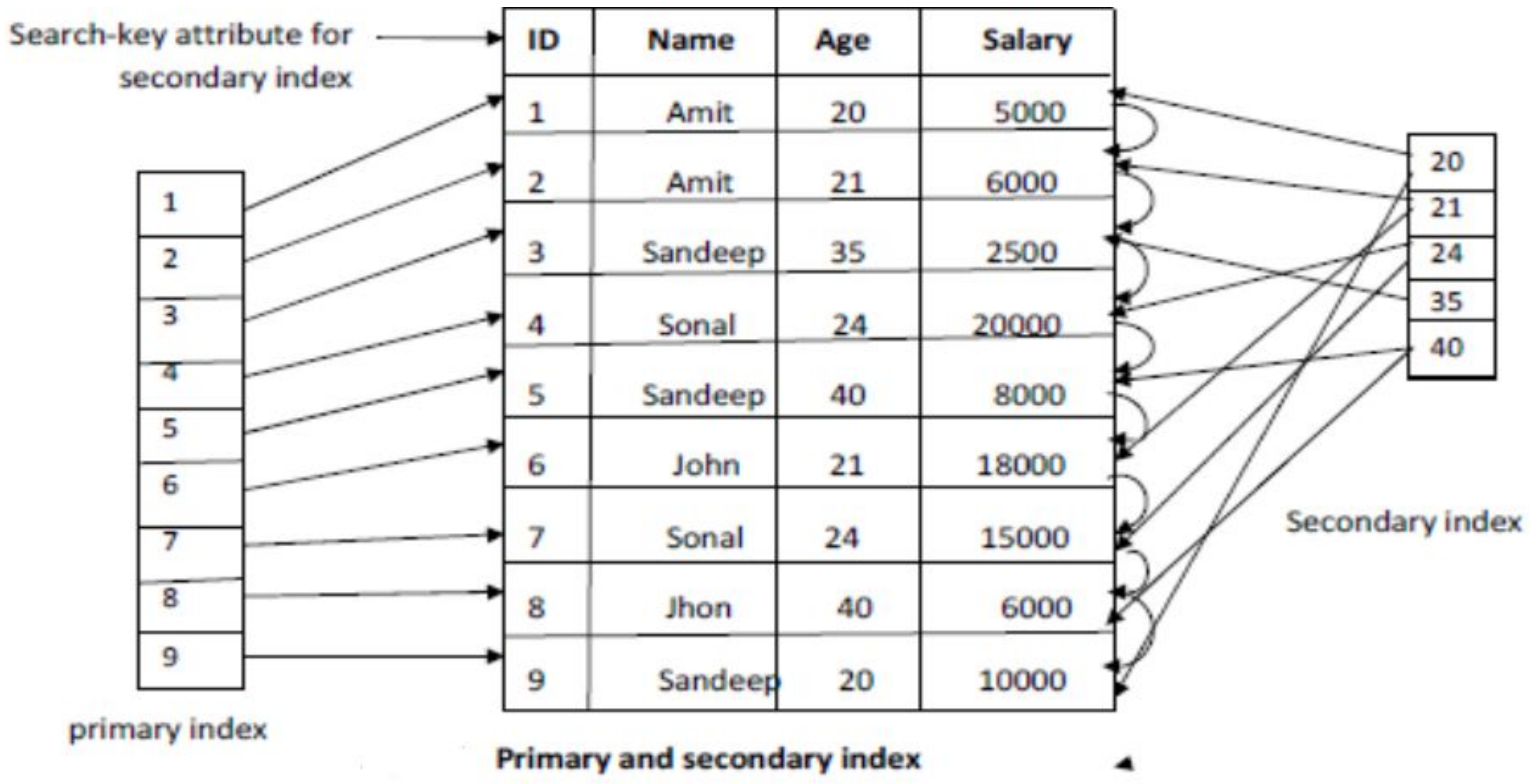
Types of Indices

- Primary Index- It is an index ordered in the same way as data file which is sequentially ordered according to a key. The indexing field is equal to this key

- Secondary Index- This is an index that is defined on a non ordering field of data file. In this case indexing field need not contain unique values

- Clustering Index- A data file can associate with utmost one primary index and several secondary indices. In this organization, key searches are improved. The single level indexing structure is simplest one where file, whose records are pairs, contains a key & pointer. This pointer is the position in the data file of the record with given key.

Types of Indices



ID	Name	Age	Salary
1	Amit	20	5000
2	Amit	21	6000
3	Sandeep	35	2500
4	Sonal	24	20000
5	Sandeep	40	8000
6	John	21	18000
7	Sonal	24	15000
8	Jhon	40	6000
9	Sandeep	20	10000

Primary and secondary index

Types of Indices contd…

- Key search is performed as below

- Search key is compared with index keys to find highest index key coming in front of search key while a linear search is performed from the record that the index key points to until the search key is matched or until the record pointed to by the next index entry is reached

- Hardware for indexed sequential file is usually disk based rather than tape.
- Records are physically ordered by primary key and index gives the physical location of the record.
- Records can be accessed sequentially or directly via index.

- Index is stored in file and read into memory at the point when the file is opened. The indices must also be maintained.

Structure of Indexed Sequential File

- In primary area, actual data records are stored. Data records are stored as sequential file
- Second area is an index area in which the index is stored and is automatically generated. An index file consist of three areas.
- **Primary Storage Area-** It includes unused space to allow for additions made in data
- **Separate Index or Indices-** Each query will reference this index first; it will redirect query to part of data file in which the target record is saved.
- **Overflow Area-** This is optional separate overflow area.

Track No.	Records
1	1, 2, 4, 7, 8, 9
2	12, 15, 17, (unused space)
3	.
.	.
.	.
.	.
10 (Overflow Area)	10

Structure of Indexed Sequential File

- A number of index levels may be involved in index sequential file
- The lowest level is track index which is written at track 0, i.e first track of cylinder
- The track index contains two entries for each prime track of the cylinders for index sequential file
- The normal entry is composed of address of prime track to which each entry is associated and highest value of the keys for the records is stored on that track.
- The index indicates how records are distributed over number of cylinders
- In index sequential file, records are organized in sequence of key field known as primary key
- For fast searching, it is supported by index
- Index is a pair of key & address where that record is stored in main file
- Number of records are same as number of blocks of main file

Characteristics of Indexed Sequential File

- Records are stored sequentially and a separate index file is maintained for accessing the record directly
- Records can be accessed randomly in constant time
- Magnetic tape is not suitable for indexed sequential storage
- Index is the address of physical storage of a record
- When a very few records are to be accesses, then indexed sequential file is better
- This is a faster access method
- Additional overhead is that the index is to be maintained
- Indexed sequential files are popularly used in many applications such as digital library.

Direct Access File

- Files that have been designed to make direct record retrieval as easy & efficient as possible are known as directly organized files.
- This is achieved by retrieving a record with a key by getting address of a record using the key.
- To achieve this, a suitable algorithm called as hashing is used to convert keys to addresses.
- They are often used in accessing large databases.

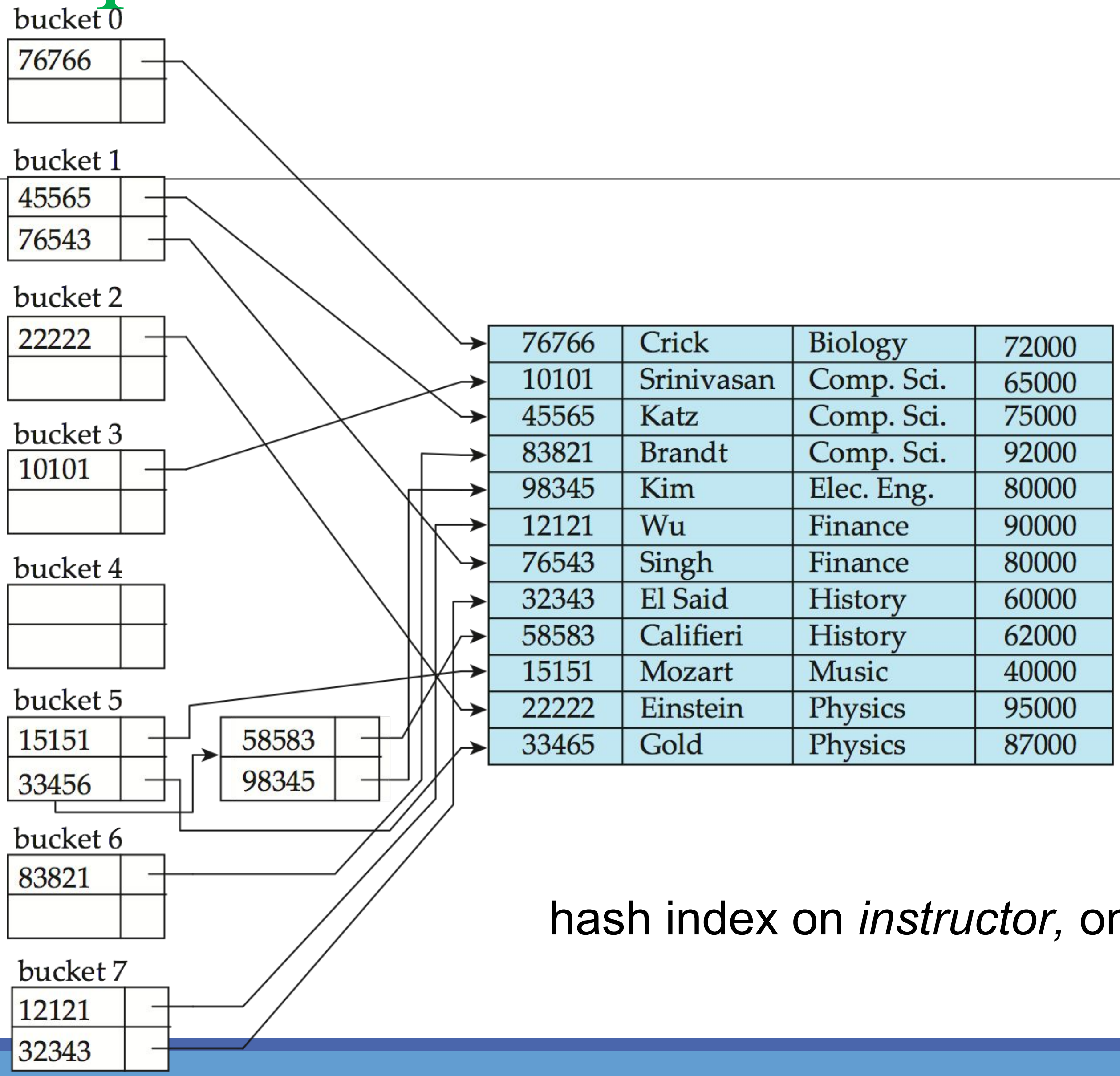
Operations on Direct Access File

- Open – Opens the file and sets the file pointer to the first record
- Read-next – It returns next record to user. If no record present then EOF will be set.
- Read-direct – Sets file pointer to specific position and gets the record for the user. if slot is empty or out of range, then it gives error
- Write-direct – It sets file pointer to specific position and write record to file at that position. If slot is out of range, then it gives error.
- Update – Current record is written at the same position with updated values
- Close – This will terminate the access to the file
- EOF – If EOF occurs, it returns true else returns false

Hash Indices

- Hashing can be used not only for file organization, but also for index-structure creation.
- A **hash index** organizes the search keys, with their associated record pointers, into a hash file structure.
- Strictly speaking, hash indices are always secondary indices
  - if the file itself is organized using hashing, a separate primary hash index on it using the same search-key is unnecessary.
  - However, we use the term hash index to refer to both secondary index structures and hash organized files.

Example of Hash Index



76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
45565	Katz	Comp. Sci.	75000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
33465	Gold	Physics	87000

hash index on *instructor,* on attribute *ID*

Implement an index sequential file

Open
Read
Write
Close