Data Management and File Organization

File Operations using Hashing

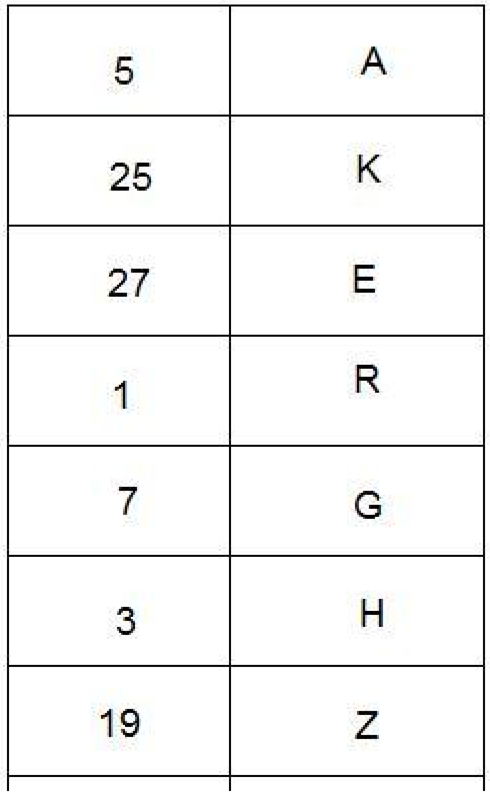
Multiple Indexes

Hashing

- Motivation: The number of file access in an indexed file is as many as the tree height (3 or 4 for example)
- Hashing method provides a quick access to the records (1 or 2 file access)

Definitions

- Hash function: A function that returns the location of a record given its key value.
  - Example: f(25)=1, f(1)=3



5	A
25	K
27	E
1	R
7	G
3	H
19	Z

Definition

- Hash table: The data file having the records is called the hash table.

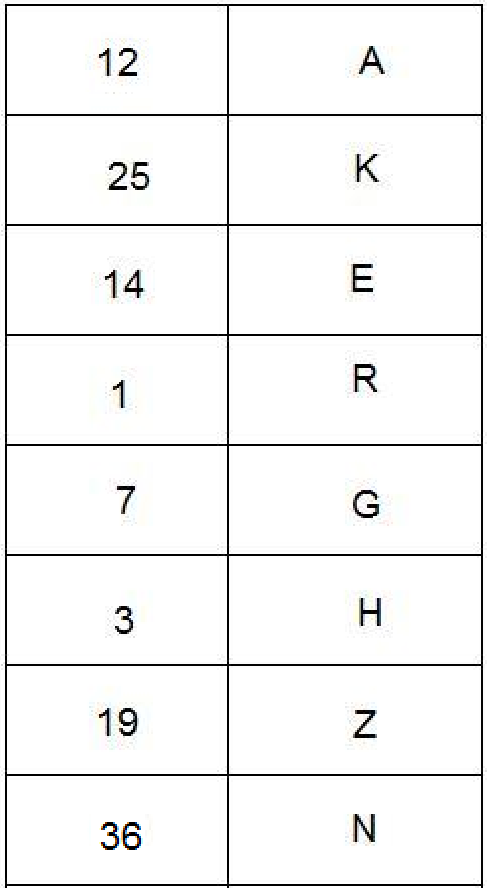
- Hash table is created using the order returned from the hash function.

Creating Hash Table

- Compute the location of the record using hash function.
- Put the record at the position returned from the hash function.

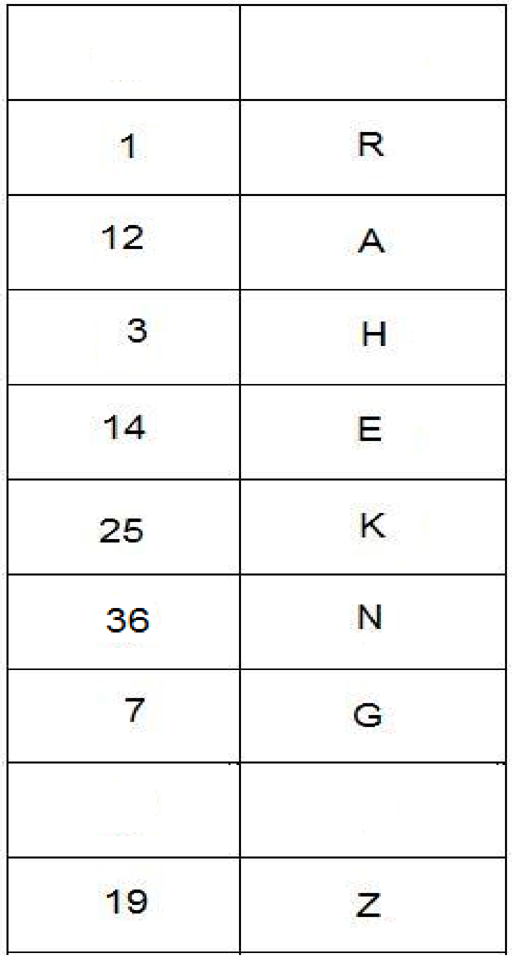
Example Hash Table

- Use Key Mod 10 to create the hash table.



12	A
25	K
14	E
1	R
7	G
3	H
19	Z
36	N

Data File



1	R
12	A
3	H
14	E
25	K
36	N
7	G
19	Z

Hash Table

Collision Problem

- The hash function may generate the same values for different keys.

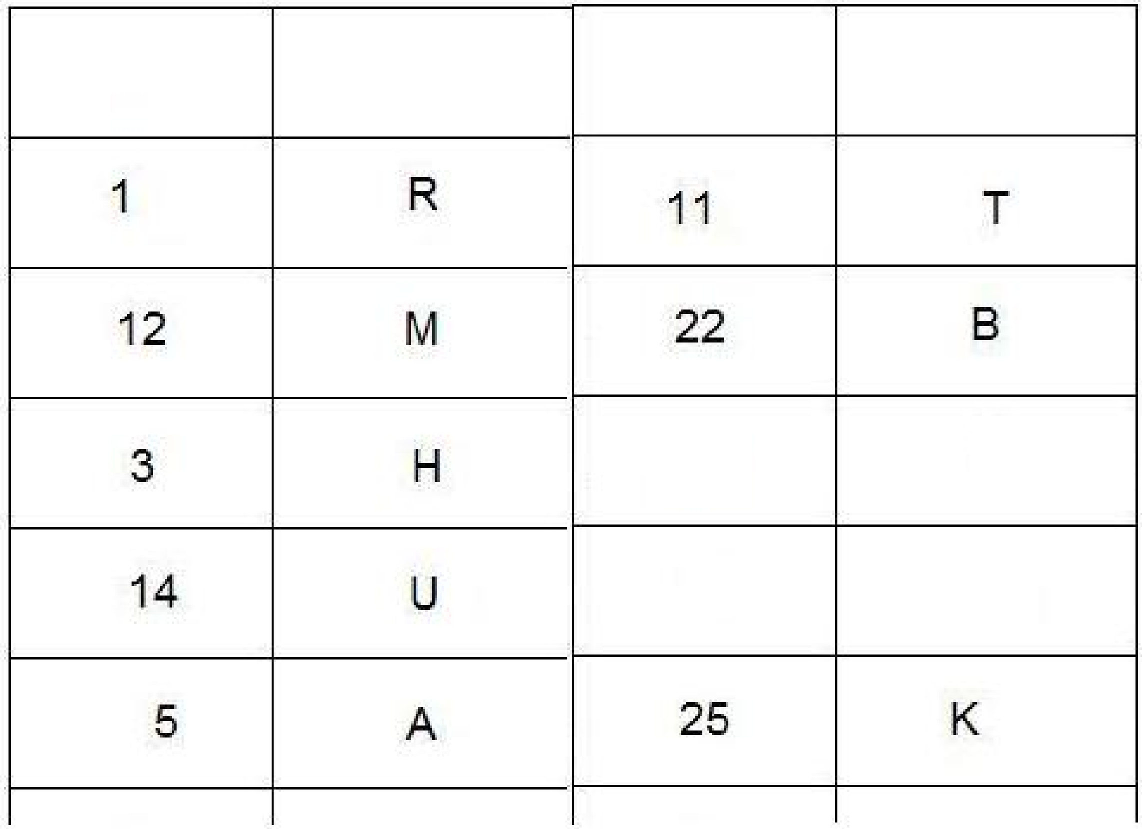
  Example: Keys 12 and 32 generate same results with hash function :: key mod 10

- This is called collision problem

Solutions for collision problem

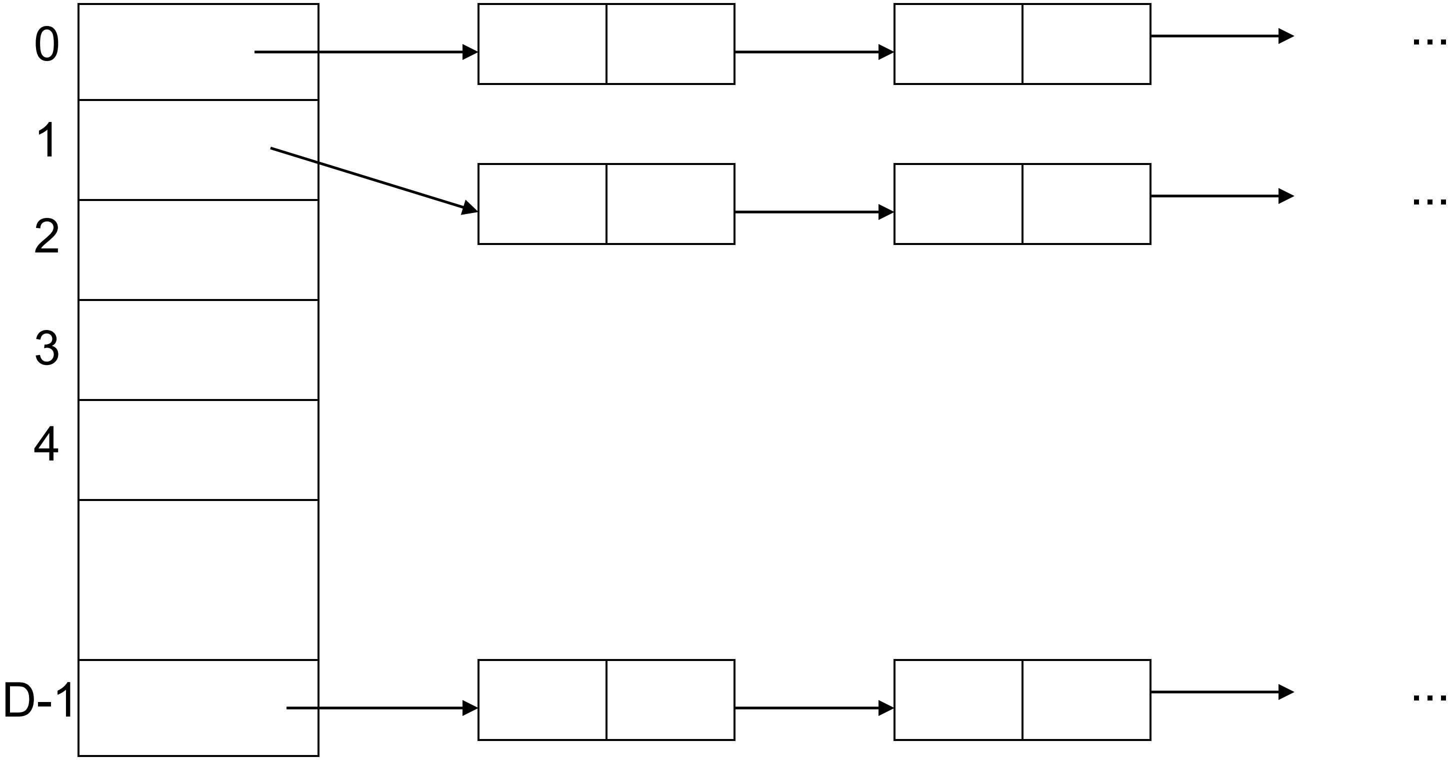
1. Bucketing: Use buckets as large as n records at each hash table entry
2. Chaining: Records with the same hash values are chained in a linked list using an overflow area or dynamic links

Bucketing

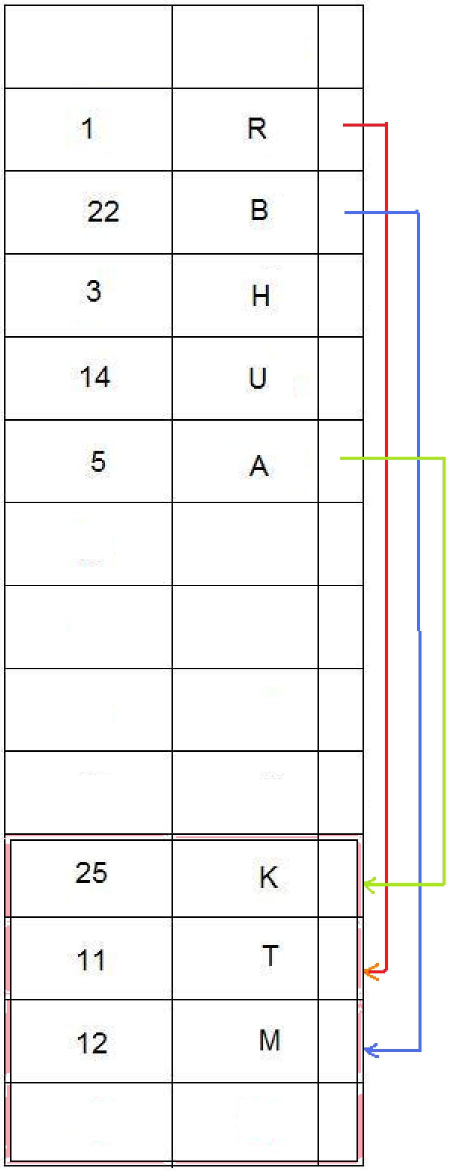


1	R	11	T
12	M	22	B
3	H		
14	U		
5	A	25	K

Dynamic Memory Allocation for Chaining



Chaining using Overflow Area



1	R	
22	B	
3	H	
14	U	
5	A	
25	K	
11	T	
12	M	

Combining Bucketing and Chaining

- Bucketing can be used with chaining for better performance.
- If a bucket is the same size of a block, file I/O operations will be more efficient (the unit of I/O operation is a block)
- The buckets are connected using linked lists if collisions happens.

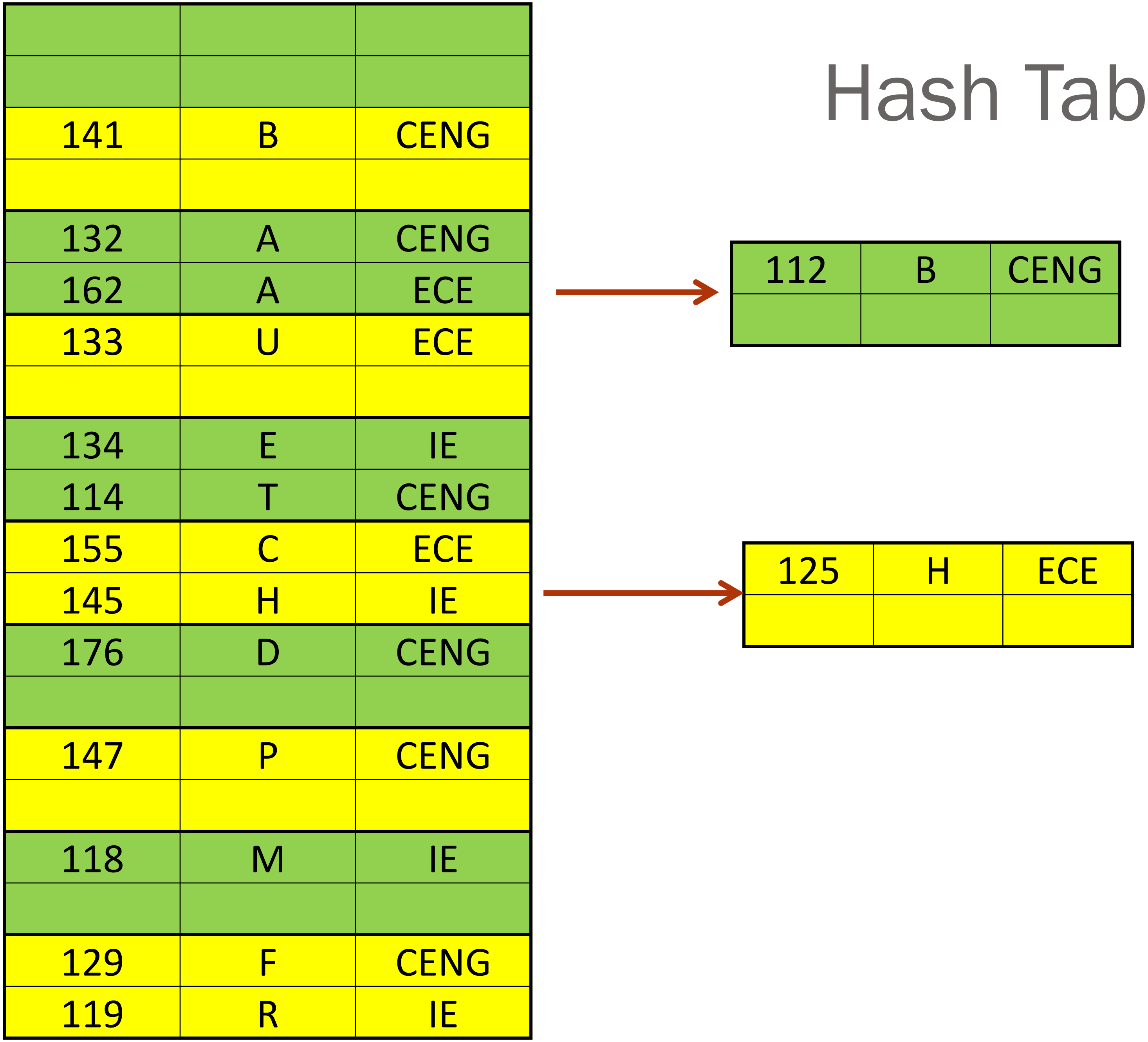
Sample Data

Student ID	Student Name	Department
132	A	CENG
141	B	CENG
155	C	ECE
176	D	CENG
162	A	ECE
134	E	IE
145	H	IE
112	B	CENG
114	T	CENG
125	H	ECE
133	U	ECE
147	P	CENG
118	M	IE
129	F	CENG
119	R	IE

Bucket Size and Hash Function

- For this example we used
  - Student ID as key value
  - Key MOD 10 as hash function
  - Bucket size = 2

Hash Table



141	B	CENG
132	A	CENG
162	A	ECE
133	U	ECE
134	E	IE
114	T	CENG
155	C	ECE
145	H	IE
176	D	CENG
147	P	CENG
118	M	IE
129	F	CENG
119	R	IE

112	B	CENG

125	H	ECE

File Operations using Hashing (1) Insert Operation

- The new record is added to the hash table by finding the location of the record using hash function.
- Then the chain is followed and the record is added to the end of the chain.
- Assuming the average chain length is L, insert operation timing is:
  - $T_I=(s+r+btt)*L+2r$
  - Where (s+r+btt)*L is the time to read until the last bucket of the chain, and 2r is the time needed to write the new record into the hash table.

File Operations using Hashing (2) Delete Operation

- The record is found in the hash table using hash function and following the chain.
- On average half of the chain is followed to find a record.
- Assuming the average chain length is L, delete operation timing is:
  - $T_D=(s+r+btt)*(L/2)+2r$
  - Where $(s+r+btt)*(L/2)$ is the time to read the buckets of the chain, and $2r$ is the time needed to mark the record as deleted in the hash table.

File Operations using Hashing (3) Update Operation

- The record is found in the hash table using hash function and following the chain.
- On average half of the chain is followed to find a record.
- Assuming the average chain length is L, update operation timing is:
  - $T_U=(s+r+btt)*(L/2)+2r$
  - Where $(s+r+btt)*(L/2)$ is the time to read the buckets of the chain, and $2r$ is the time needed to update the record and write it back in the hash table.

Main Issues in Hashing

- Two main problems with hashing are:
  - Choosing a hash function is very difficult
  - Hashing creates a hash table based on one key field only. Creating multiple hash functions is difficult.
    - E.g. The student data file is changed into a hash table. The hash function uses StudentID. If we want to search based on student name, hash table, and hash function should change.

Multiple Indexing

- If a data file is searched using two or more attributes, multiple indexes should be created for it.
- Multiple indexes can be created using:
  - Linear index
  - B-trees
  - B+trees

Multiple Indexes using Linear Indexing

- Data file is in the form of a pile file.
- Records are always added from the end of the data file.
- For each search attribute, a linear index is created.
- If the index files are large, we cannot load them into the memory together.

Sample Data

Student ID	Student Name	Department
132	K	CENG
141	B	CENG
155	C	ECE
176	D	CENG
162	A	ECE
134	E	IE
145	S	IE
112	W	CENG
114	T	CENG
125	H	ECE
133	U	ECE
147	P	CENG
118	M	IE
129	F	CENG
119	R	IE

Location	Key
7	112
8	114
12	118
14	119
9	125
13	129
0	132
10	133
5	134
1	141
6	145
11	147
2	155
4	162
3	176

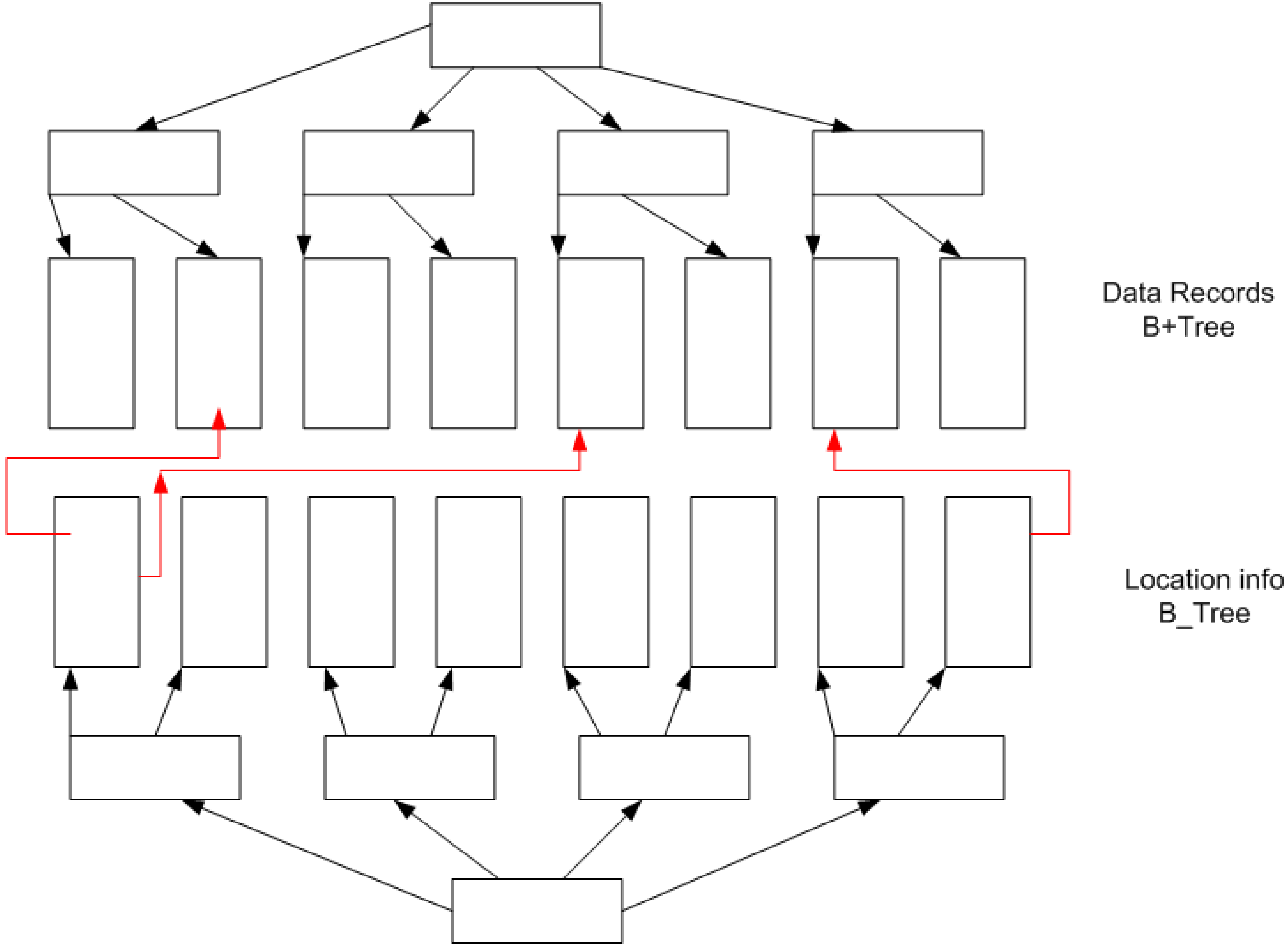
Location	Key
4	A
1	B
2	C
3	D
5	E
13	F
9	H
0	K
12	M
11	P
14	R
6	S
8	T
10	U
7	W

Multiple Indexes using B-Trees

- Data is in a pile file.
- The record locations are at the leaf nodes of the index files.
- For each search attribute a B-tree is created.
- B-trees can be large. Only first two levels of the B-trees are loaded into the memory and the rest are read from files.

Multiple Indexes using B+Trees

- A B+tree is created for the first (most important) search attribute.
- The records are in the leaf nodes of the B+tree.
- For the second and third, .. search attributes, B-trees are created.
- B-trees have the location of the records in the B+tree


Data Records
B+Tree
Location info
B_Tree

Summary

- Multiple indexes are necessary in many data files.
- In sorted sequential files, search using two attributes requires two copies of the data file (each one sorted according to one of the attributes)
- Hash tables are created using hash functions and multiple search in them is difficult.
- Multiple index files (linear, B-tree, B+tree) can be created for multiple search attributes.

Questions?