Indexing

Indexing is a way to organize records in a file

- **index file** - is file containing indexes containing a **key** and a **pointer** to the corresponding record in the data file

```
struct Index{
    KEY         key;
    long int    record;
};
```

- **data file** - contains data records

Files in C

C supports two types of files:

- character files (streams)
- **binary files**

Operations on files:

- open a file:

FILE* fopen (char * name, char mode)

- close a file

void fclose (FILE* file)

- read a file

size_t fread (void* b, size_t s, size_t n, FILE* f)

- write a file

size_t fwrite (void* b, size_t s, size_t n, FILE* f)

- position a file

```
int fseek (FILE* f, long offset, int base)
```

base can be:

SEEK_SET - from the beginning
SEEK_CUR - from current position
SEEK_END - from the end

- test end of file

```
int feof (FILE* f)
```

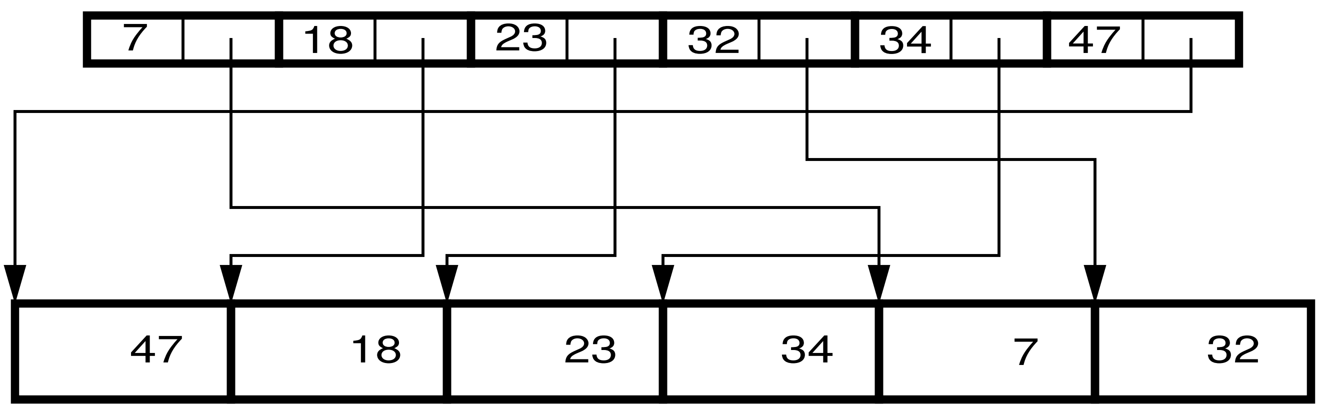
Example Create a file containing records of the following form:

```
struct Records{
    char    first[20];
    char    last[20];
    int     id_num;    // this is the key
    int     age;
}
```

together with an index file

Linear indexing

- the index file is organized as a sequence of indexes sorted by the keys

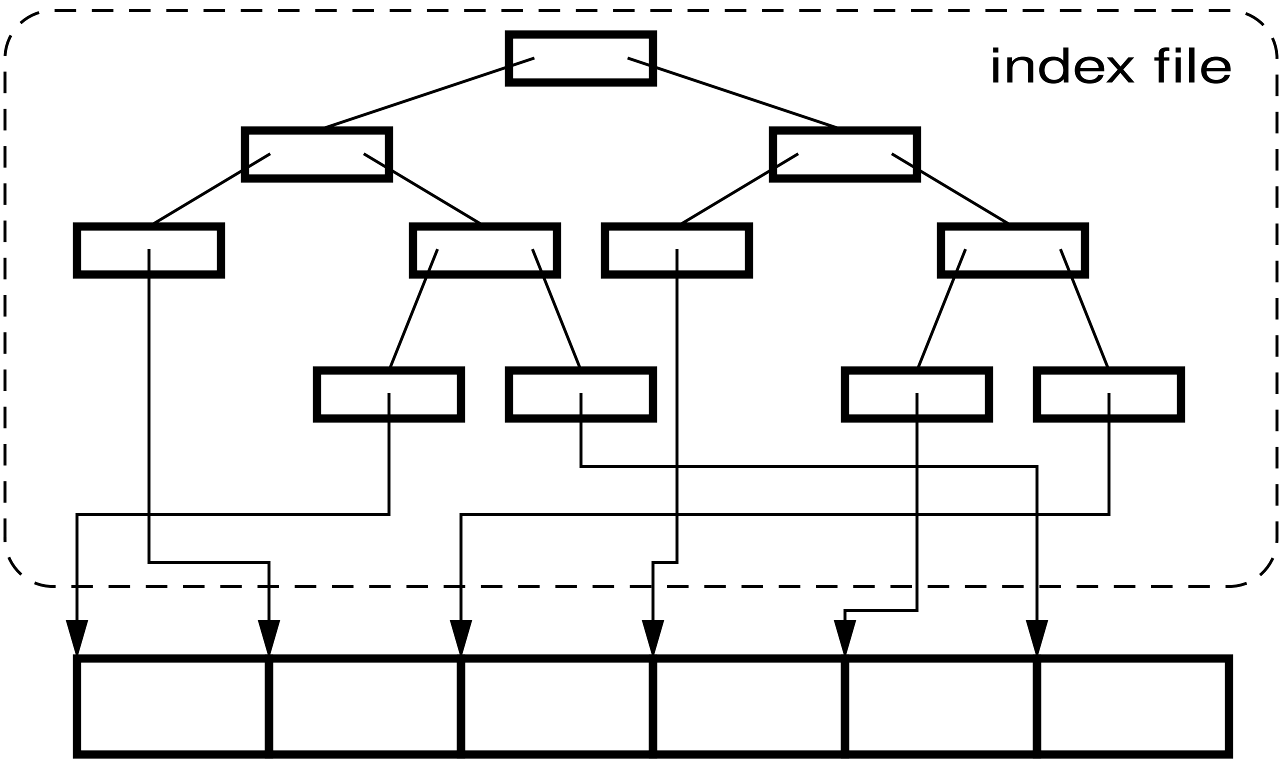


- the disadvantages are the same as for array-implementation of lists

- it is possible to extend this indexing scheme to multiple key

Tree Indexing

- use BST's to organize indexes to obtain efficient search, insertion and deletion



- **problem:** BST may become unbalanced ... for files this is a greater problems than for data structures stored in memory

2-3 Trees

Definition. 2-3 tree:

❑ a node contains one or two keys

❑ every internal node contains two or three children

❑ the leaves are all at the same level in the tree

❑ the tree has the search property:

- nodes in the left subtree < value of the first key
- nodes in the middle subtree >= value of the first key
- node in the right subtree >= value of the second key (if there is any)

```
structure TTNode{
    int        nkeys;
    KEY        lkey;
    KEY        rkey;
    TTNode* left;
    TTNode* center;
    TTNode* right;
};
```

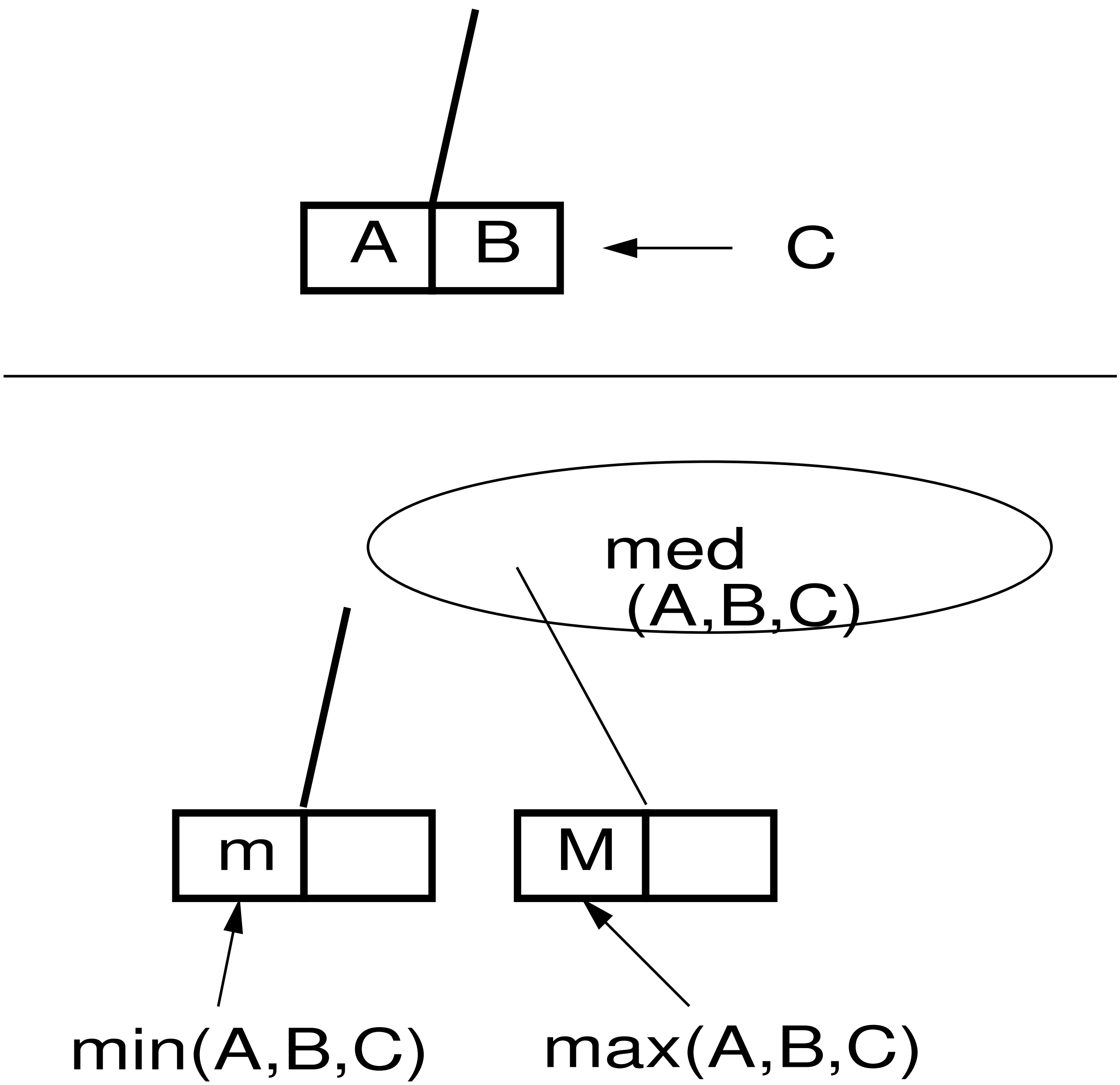
Searching in a 2-3 tree

- similar to binary search, except for the three-way branching

Inserting and deleting in a 2-3 tree

- do not create new nodes at the bottom of the tree - they use split or merge and have effect higher in the tree

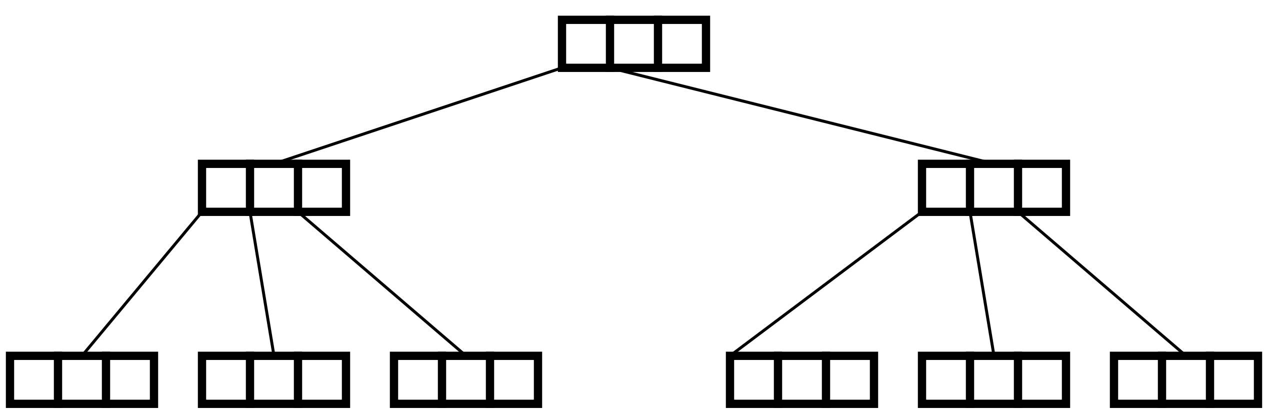
Insertion



B-Trees

Definition. A B-Tree of order *m* is a tree with the following properties:

1. The root is either a leaf, or it has at least two children
2. Each node, except for the root has between *m/2* and *m* children.
3. All leaves are at the same level in the tree, so that the tree is always height balanced.



B$^+$-Trees

- a B-Tree in which only the leaf nodes contain data (all the other nodes only contain keys)