ICOM 6005 – Database Management Systems Design

Dr. Manuel Rodríguez-Martínez

Electrical and Computer Engineering Department

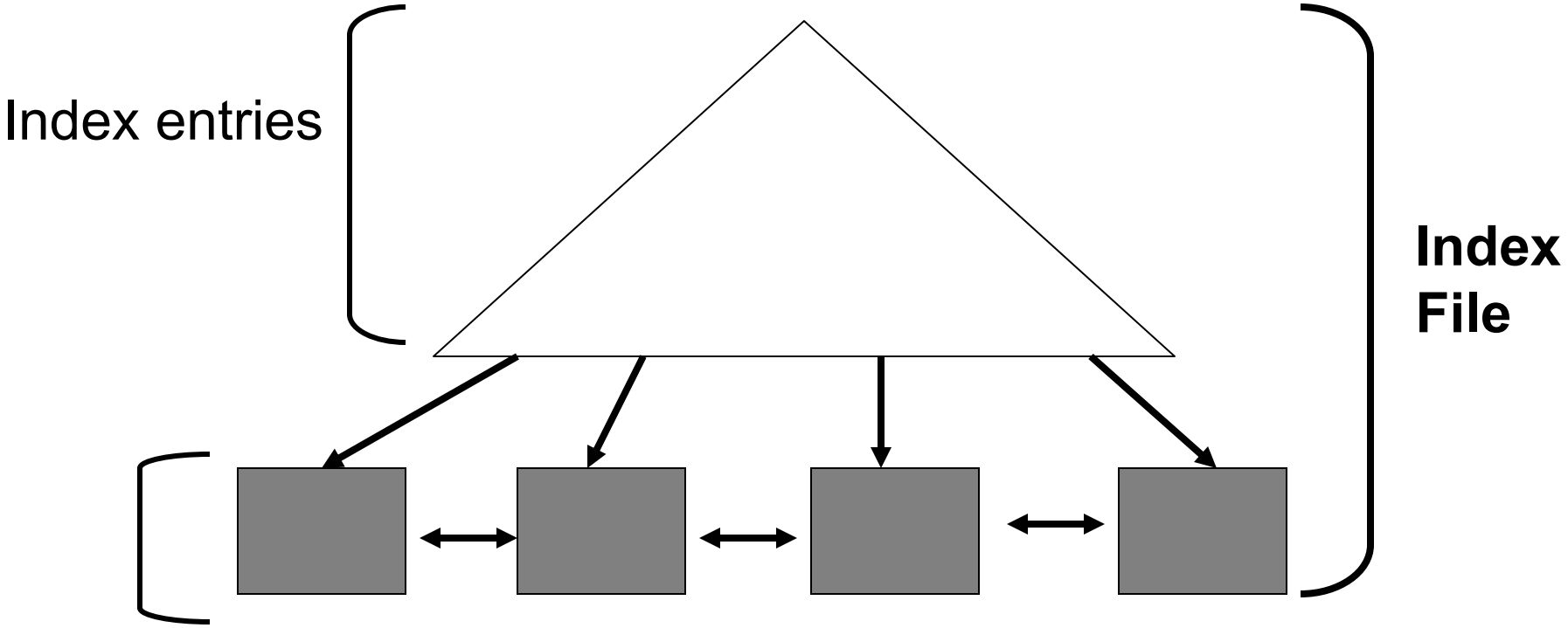
Tree-based Indexing

- Read Chapter 10.
- Idea:
  - Tree-based Data structure is used to order data entries
  - Index entries
    - Root and internal nodes in the tree
    - Guide “traffic” around to help locate records
  - Data entries
    - Leaves in the tree
    - Contain either
      - actual data
      - pairs of search key and rid
      - pairs of search key and rid-list
  - Good for range queries

Range queries

- Queries that retrieve group of records that lies inside a range of values
- Examples:
  - Find the name of all students with a gpa between 3.40 and 3.80
  - Find all the items with a prices greater than $50.
  - Find all the parts with an average stock amount less than 30.
  - Find all the galaxies that are within 10 light year from galaxy NC-1493.
  - Find all the images for regions that overlap the area of Puerto Rico.
- Note: Tree are also good for equality.

Tree index structure



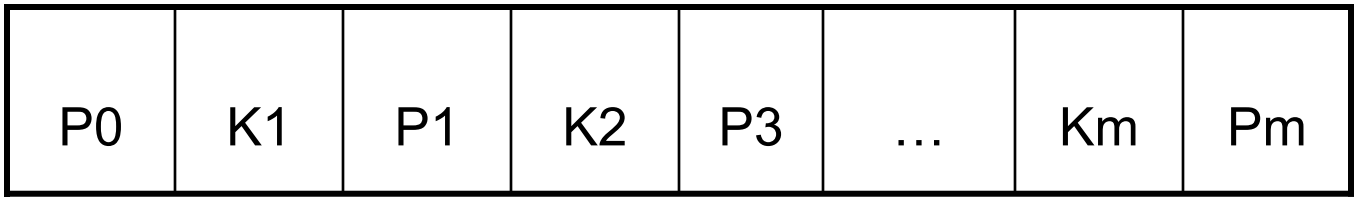
Records are stored at data entries

Three major styles

- ISAM
  - Static tree index
  - Good for alphanumeric data sets
- B+-tree
  - Dynamic tree index
  - Good for alphanumeric data sets
- R-tree
  - Dynamic tree index
  - Good for alphanumeric and spatial data sets
    - Polygons, maps, galaxies
    - Dimensions in a data warehouse
      - Parts, sales, date,

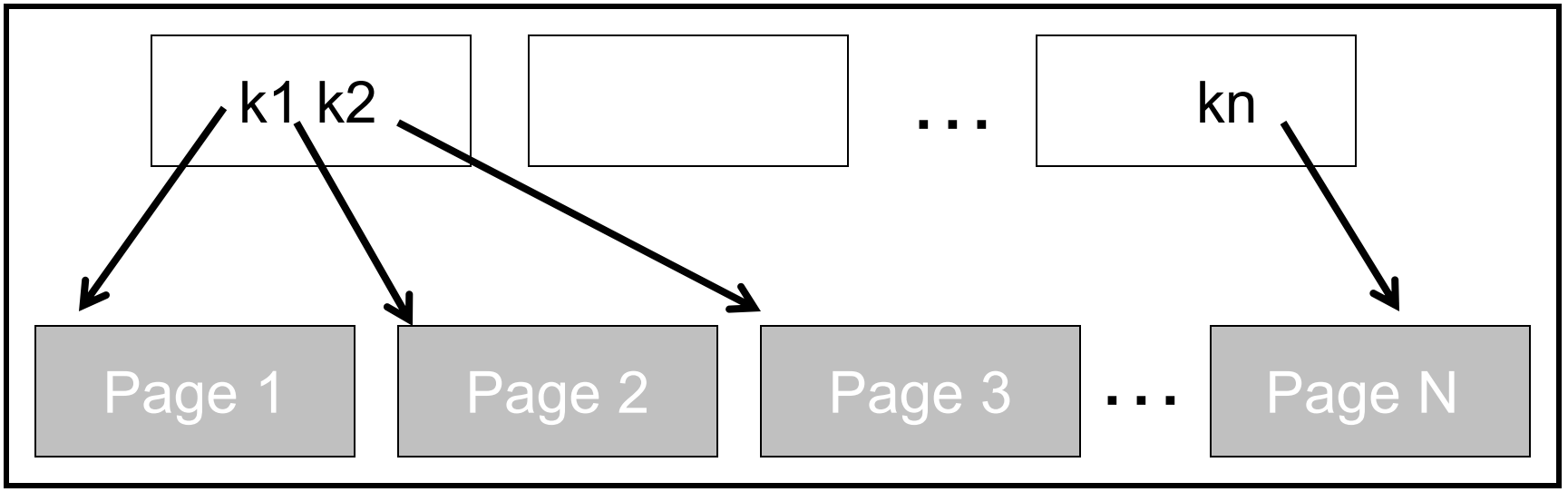
General form for index pages

- Index pages have
  - Key values – number, strings, rectangles (R-tree)
  - Pointers to child nodes
  - P0 leads to values less than K1
  - Pm leads to values greater than Km
  - For any other case, Pi points to values greater or equal than Ki, and values less than K_{i+1}
  - For R-tree is all about overlapping regions …



Some issues to keep in mind

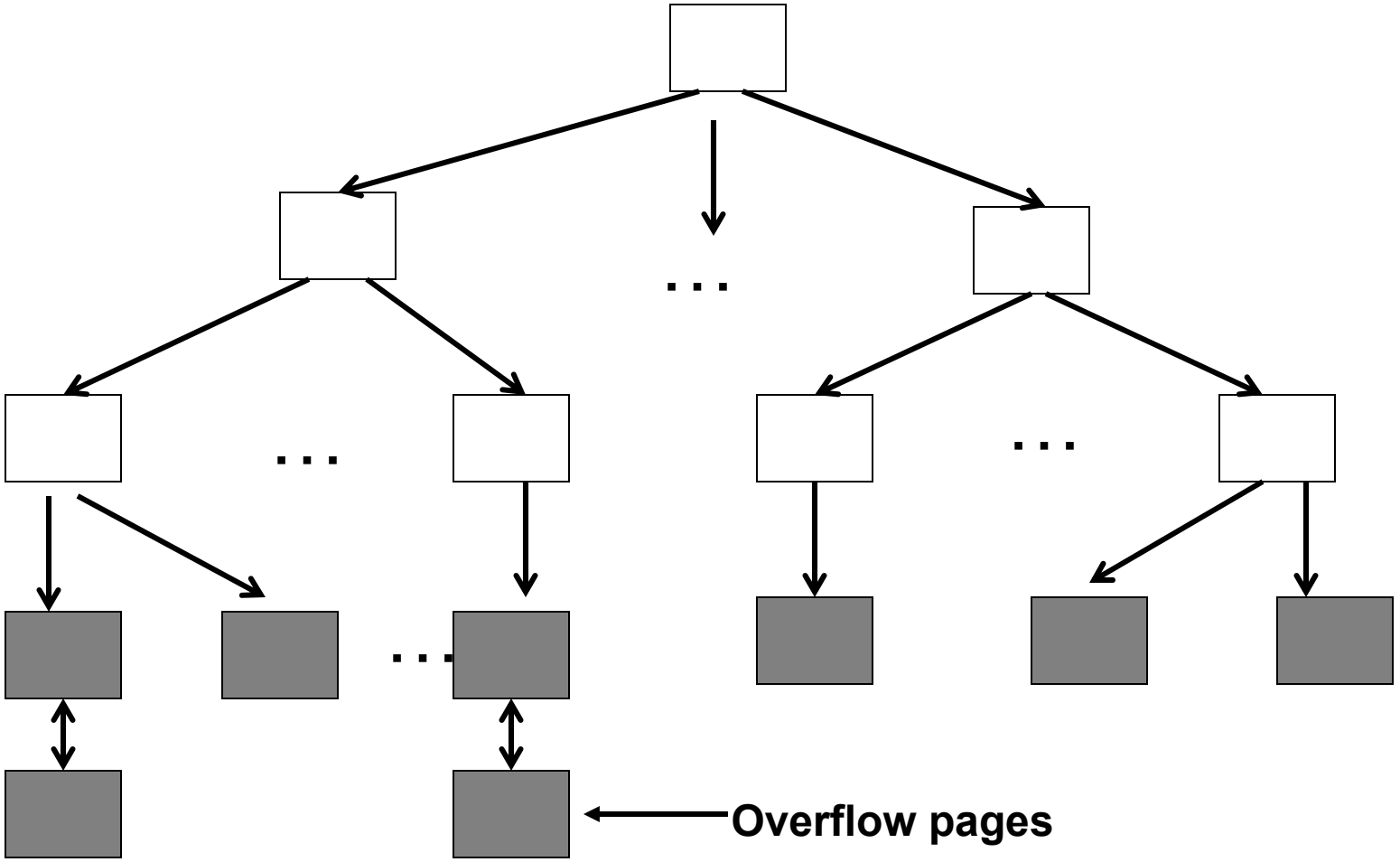
- Index entries are contained in pages
- Data entries are contained in pages
- We expect the root of the tree to stay around in the buffer pool
  - Often 3-4 I/Os are need to locate the first group of data items



ISAM

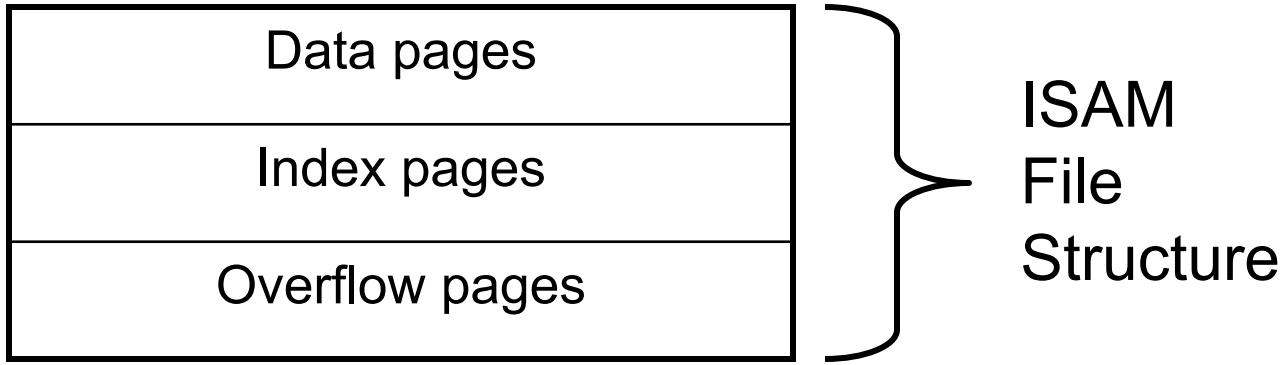
- Indexed sequential access method (ISAM)
- Support insert, delete, search operations
- Static index structure based on tree
  - Balanced tree
- Number of leaves and internal nodes is fixed at file creation time
- More space is allocated as overflow pages
  - Chained with appropriate leaf
  - Long overflow chains are no good.

ISAM Structure

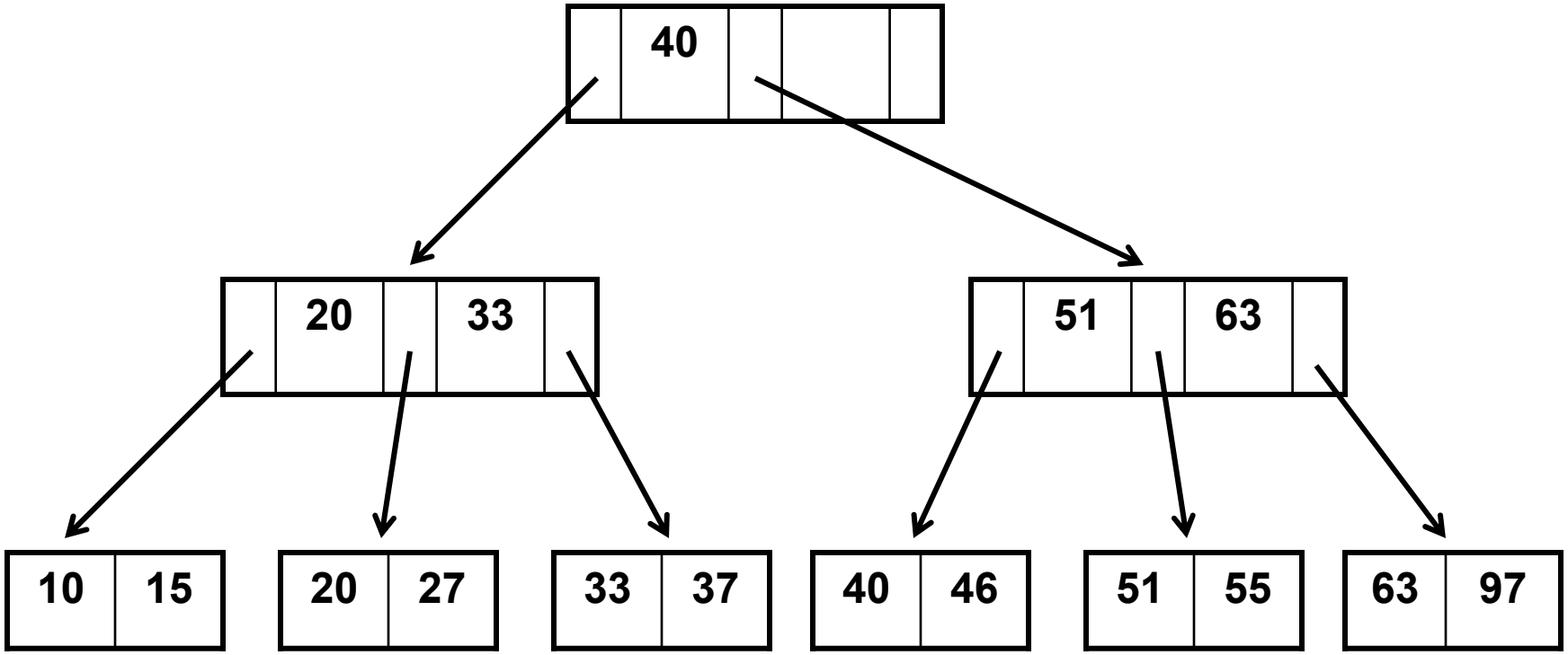


ISAM Disk Organization

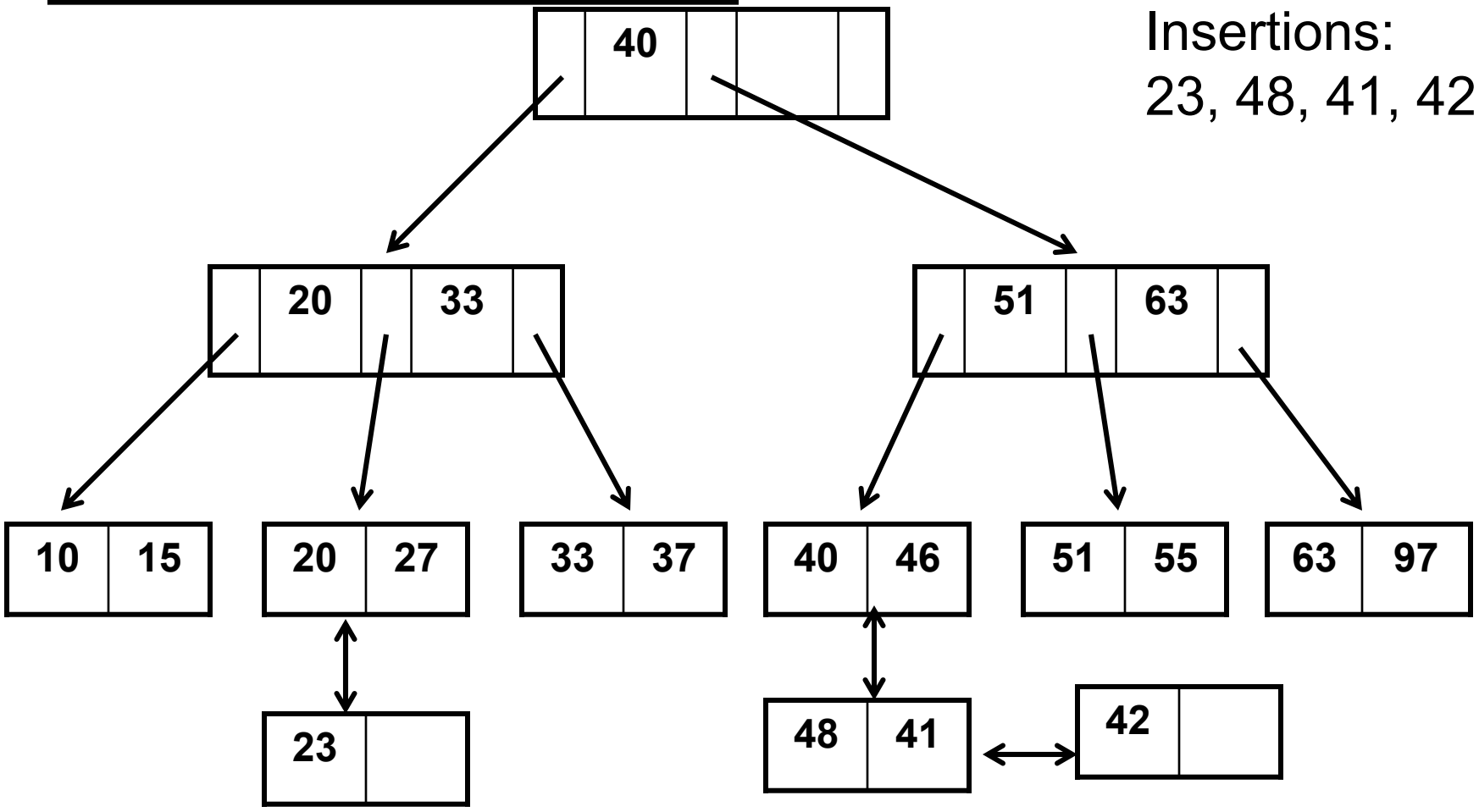
- Data pages are allocated sequentially
  - Fixed number of pages at file creation
- Index pages are then allocated
  - Fixed number of pages at file creation
- Overflow pages go at the end
  - Variable number
  - Must be chained with the base data pages



Sample ISAM Tree



ISAM Tree After a few insertions



Insertions:
23, 48, 41, 42

Search Algorithms

```
nodeptr find(search key K){
    return find_aux(root, K);
}

nodeptr find_aux(nodeptr P, key K){
    if P is a leaf then return P
    else {
        if (k < K1) then return find_aux(node_ptr.P0, K);
        else if (k >= Km) then return find_aux(node_ptr.Pn, k);
        else {
            find Ki such that Ki < K <= Ki+1
            return find_aux(node_ptr.Pi, k);
        }
    }
}
```

Search Algorithm

- Above algorithms just finds a pointer to the page where record **might be**
- Once we get the pointer, need to search the value inside the page
- If overflow pages exists, need to traverse them
  - Lots of overflow pages mean more I/Os
- Here need to understand the format of the page
  - Determine the how to locate the record
- If a range query is issued need to travel adjacent pages to get the appropriate values

Insertion and Deletion

- Use search algorithm to find the page where the record(s) should go
- Then within this page
  - Insert the record
  - Delete the record
- If not found, then if there are overflow pages,
  - Repeat this process on the overflow page

Some Issues

- Fan out
  - Number of entries in the data pages
  - Fixed at file creation
  - Often used in the hundreds
- Each node has
  - N keys
  - N + 1 pointers
- Oftern, ISAM is built on an existing group of records
  - That's how you determine number of pages and so forth

B+-trees

- Dynamic index structure
- Adapts its size and height to the pattern of insertion and deletions.
- No overflow pages
- Each internal node has an order
  - Capacity of node