# File Organization

B-Trees I

## Reminder

- Final exam
  - The date for the Final has been decided:
  - Saturday, November 16th
  - 8am – 10am
  - 01-2000

## Project Notes

- Change your generic solver?
  - Don't forget to change your Clock and Farmer problem as well.
- Memory Management
  - Using purify
    - Add to Makefile:
      - `CCC = purify CC`
    - Or better yet, create a file `header.mak`
  - `Workshop` also has memory management tools.
- Parking Lot Problem: due Nov 11th

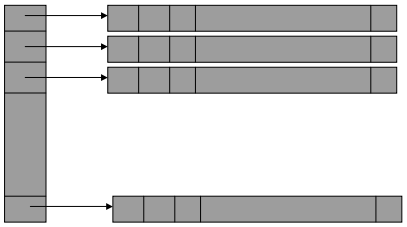
## New plan

- Today: Exam / Files 2
- Thursday: Files 3
- Monday: Ethics
- Tuesday: Final Review
- Then we are done!

## Before we start

- Multidimensional Arrays
  - C++ does support multidimensional arrays, however, using them can be troublesome.

## Multidimensional Arrays



## Multidimensional Arrays

- Ways to declare

```
int **foo;  // array of pointers to int
foo = new int * [20];
for (i = 0; i < 20; i++) foo[i] = new int
  [10];
```

`foo[3][5]` is the same as `(foo[3])[5]`

Must delete each `foo[i]` when done.

## Multidimensional Arrays

- C++ does support
  - `int [20][10]`
  - However, dimensions must be known at compile time.
  - Must pass at least 1 dimension when passing to functions.

## Multidimensional Arrays

- Using vectors or deques

```
using namespace std;
typedef deque<int> Decker ;
typedef deque<Decker> DoubleDecker ;


int main (int argc, char *argv[])
{

   Decker D (10);
   DoubleDecker DD (20, D);

   DD[3][4] = 10;

}
```

## Before we start

- Any questions

## File Organization

- How to find stuff in a file
  - Improve access time by imposing a defined structure on a file
  - Database applications
  - Advanced searching strategies

## Going up a level

- The OS must also:
  - Organize it's file system for maximum performance (I.e. low access time)
  - File System
- If the file is a searchable database
  - This optimization can be improved by imposing an additional structure on a file.

## Terminology

- Record
  - Collection of data pertaining to one entity
    - E.g. Employee
  - Each record consists of a number of data fields.
  - Files are composed of a collection of records.
- Key
  - Field within a record upon which a search is made

## Organization Strategies

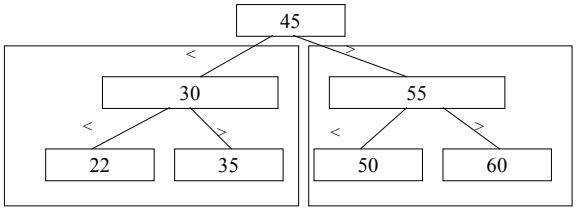
- Sequential
- Indexed
- Hashing
- Tree-based Organization

## Recall from CS2

- Binary Search Tree
  - Efficient storage for search / retrieval of "sortable" data.
  - Basic idea
    - Left subtree contains nodes with data less than data at node
    - Right subtree contains nodes with data greater than data at node

## Binary Search Trees

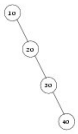
- Branching can also imply ordering of node data




All values less than 45 All values greater than 45

## Binary Search Trees

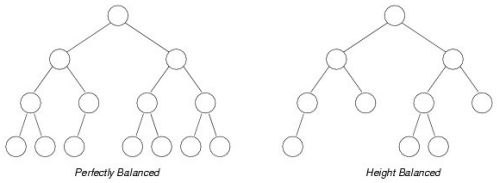
- Searches using binary trees can be done in O(log n) time.
- However, to achieve this, trees must be balanced.
  - Consider:



## Binary Search Trees

- Balanced trees
  - A binary tree is perfectly balanced if the number of nodes in the left and right subtrees is no more than 1 for each node.
    - Difficult to insert into a perfectly balanced tree and maintain it's "perfect" status
  - A binary tree is height balanced if the height of the left and right subtrees of every node (at each level) differs by no more than one.
    - Height balancing is easier to achieve than perfect balancing

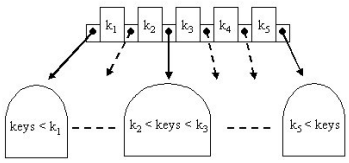
## Binary Search Trees




## Multiway Search Trees

- A generalization of binary search trees
- Each node has at most m children
- If a node has k children, it has k-1 keys
- The tree is ordered

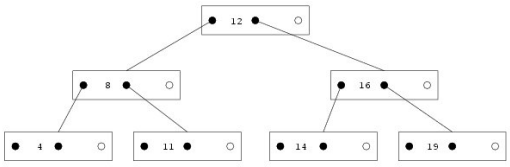
## Multiway Search Trees




## B-trees

- A B-Tree (of order m) is a multiway search tree with:
  - All leaves at the same level
  - Each interior node has between m and m/2 non-empty children
  - Each leaf node holds between m/2 to m - 1 keys
  - The root has between 2 and m non-empty children

## B-Trees (order 3)




## B-trees

- For simplicity we say nodes have pointers and keys.
- Since they are to used for file indexing:
  - Nodes contain key / file location pairs
  - Trees are sorted on key values.

## We'll stop here

- Thursday:
  - B-Tree insertion and deletion