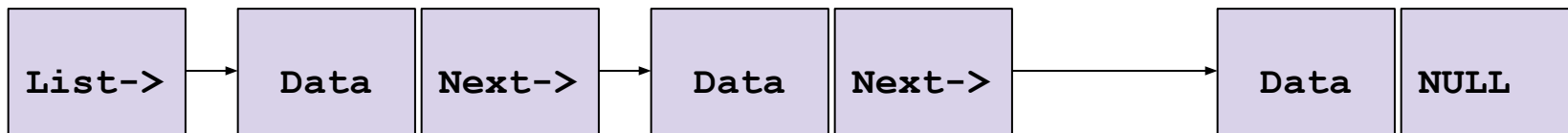


CSE 374 Lecture 15

More Data Structures



Linked Lists



Points to
the List

Last node doesn't
point to next

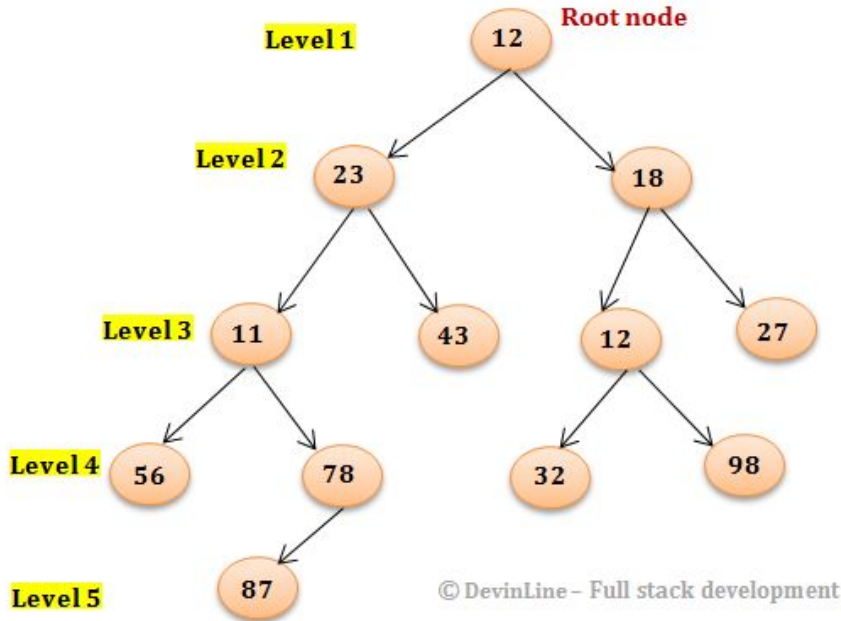
// A single list node that stores an integer as data.

```
typedef struct IntListNode {  
    int data;  
    struct IntListNode* next;  
} IntListNode;
```

```
IntListNode* makeNode(int data, IntListNode* next) {  
    IntListNode* n = (IntListNode*) malloc(sizeof(IntListNode));  
    if (n) { // malloc might return null  
        n->data = data;  
        n->next = next;  
    }  
    return n;  
}
```

Binary Trees

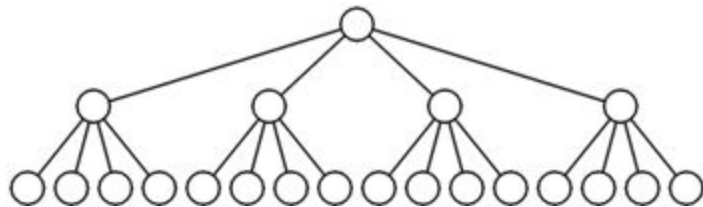
Binary tree



```
struct BinaryTreeNode {  
    int data;  
    struct BinaryTreeNode* left;  
    struct BinaryTreeNode* right;  
}
```

```
struct BinaryTree {  
    Struct BinaryTreeNode* root;  
}
```

N-ary Trees



```
struct TrinaryTreeNode {  
    char* data;  
    struct TrinaryTreeNode* left;  
    struct TrinaryTreeNode* middle;  
    struct TrinaryTreeNode* right;  
}
```

```
struct QuadTreeNode {  
    char* data;  
    struct QuadTreeNode* children[4];  
}
```

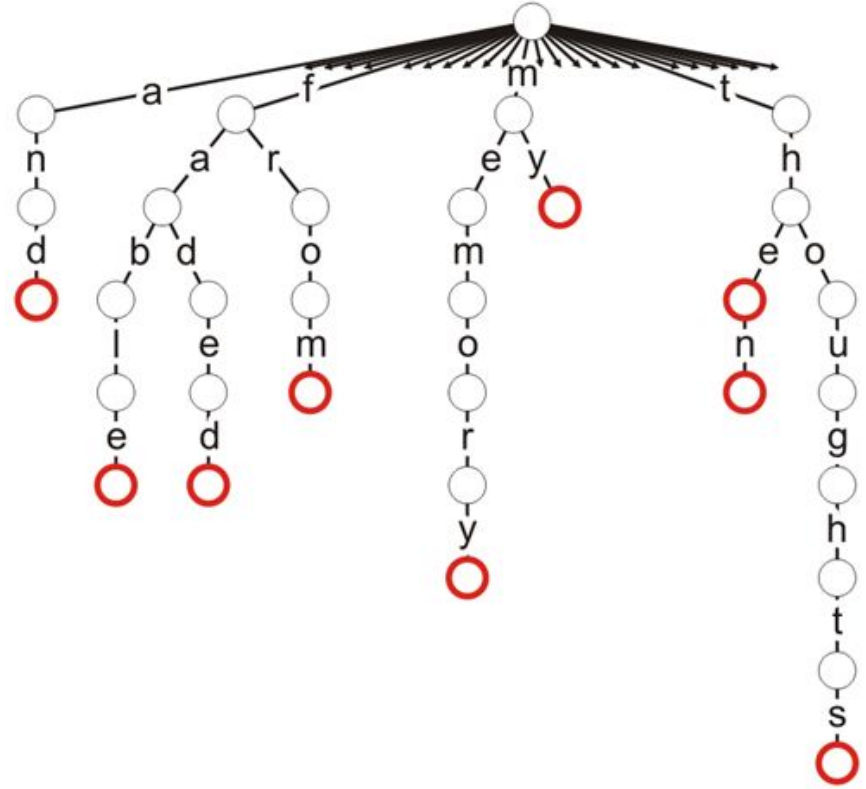
Binary trees just one form; can have any “branching number”.

Trinary trees have branching number of three.

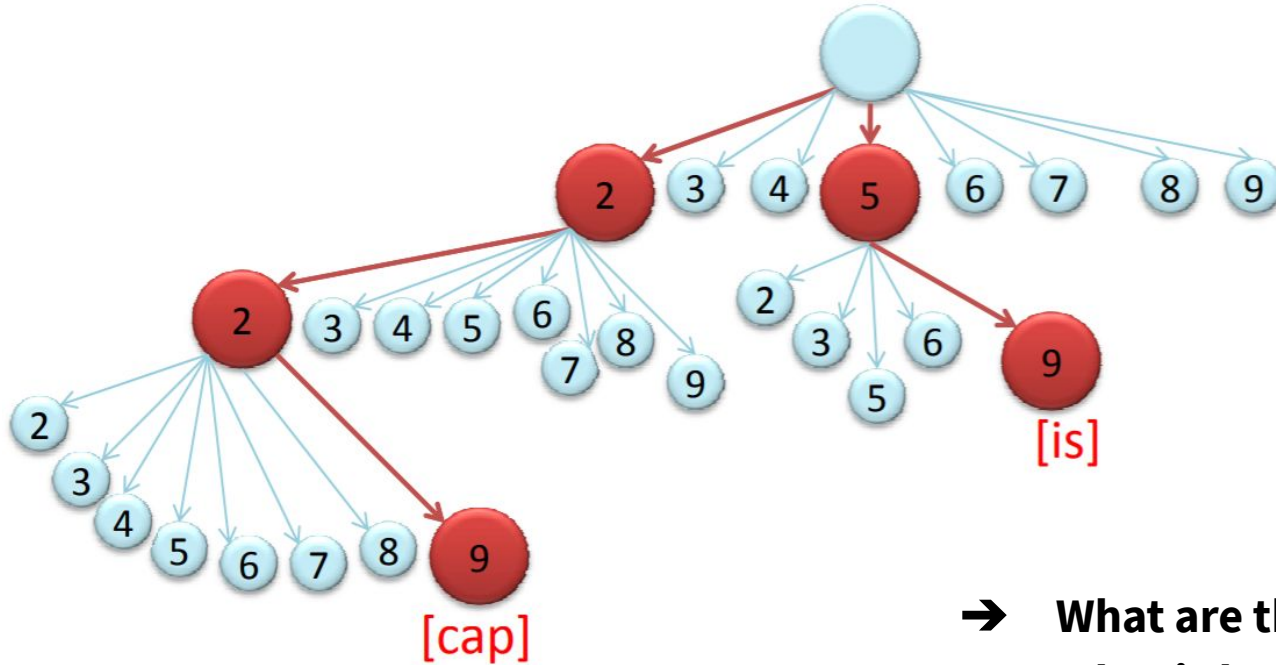
For arbitrarily large branching numbers, arrays can make more sense than lists of named pointers.

Prefix tree (Trie)

- Compact storage
 - Or generative automaton
- Key of each node defined entirely by position
- Compact data storage
- Efficient worst-case searching
- Strings often use 26-ary tree
 - Predictive text
 - Spell check



T9 Trie



- What are the branches labeled?
- What is branching factor?
- What data is stored in each node?

Linked List Continued

- One set of code to define linked list:
 - `Linkedlist.h`
 - `Linkedlist.c`
- Another piece of code uses it:
 - `Linkedlistclient.c`
 - `#include "linkedlist.h"`

Compile with

```
$gcc -o lldemo linkedlist.c  
linkedlistclient.c
```

