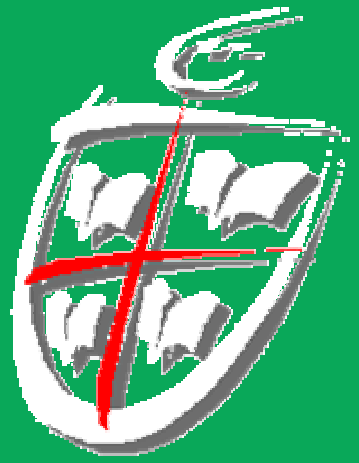
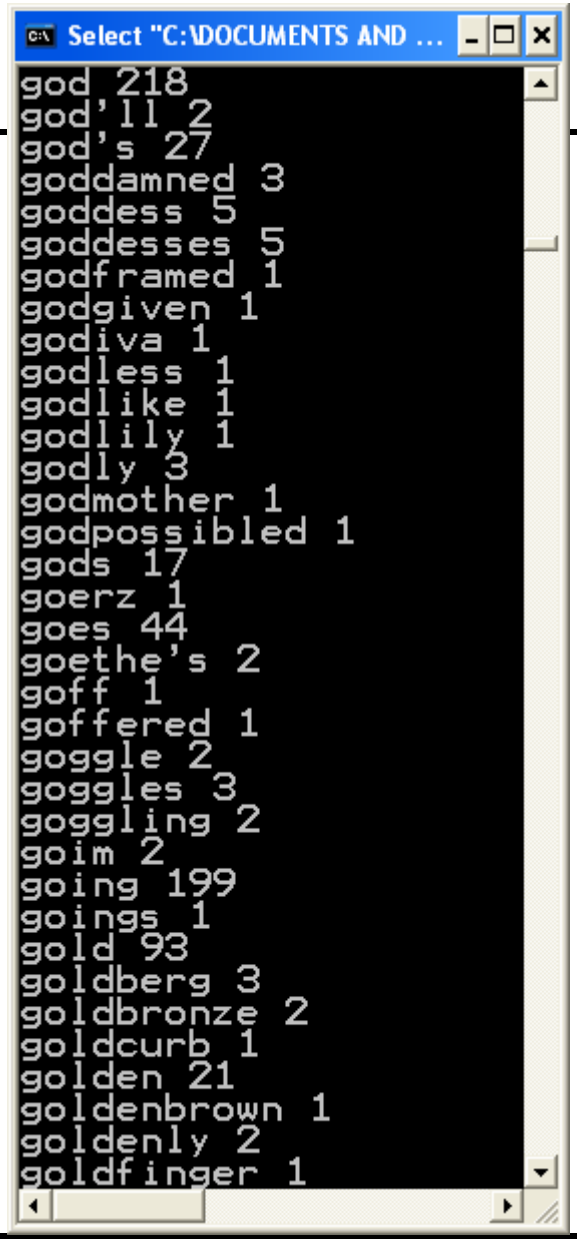


CS900-3a
Computer Science –
A Multifaceted Introduction

Lecture #13
A Computer Learns English
(even better)

Using the WordMap

```
Word token;
WordMap wm;
do {
  in >> token;
  if (in.fail()) break;
  token.clean();
  token.lowercase();
  wm += token;
} while (1);
wm.print();
```



Random Word Sampling

- Now we can generate new English text by randomized sampling of words according to their relative frequencies in the training text.
- This corresponds to the following procedure:
  - open the text at a random page and pick a word at random.
  - output this word
  - repeat as many times as you want

Implementing Word Sampling

- Here is a way to implement this as a method in our WordMap class:

```
string WordMap::random() const
{
   int rnd = lrand48() % total();
   WordMap::const_iterator i;
   for (i=begin(); i!=end(); i++) {
     rnd -= (*i).second;
     if (rnd<0) return (*i).first;
   }
   return ""; //-- should never happen
}
```

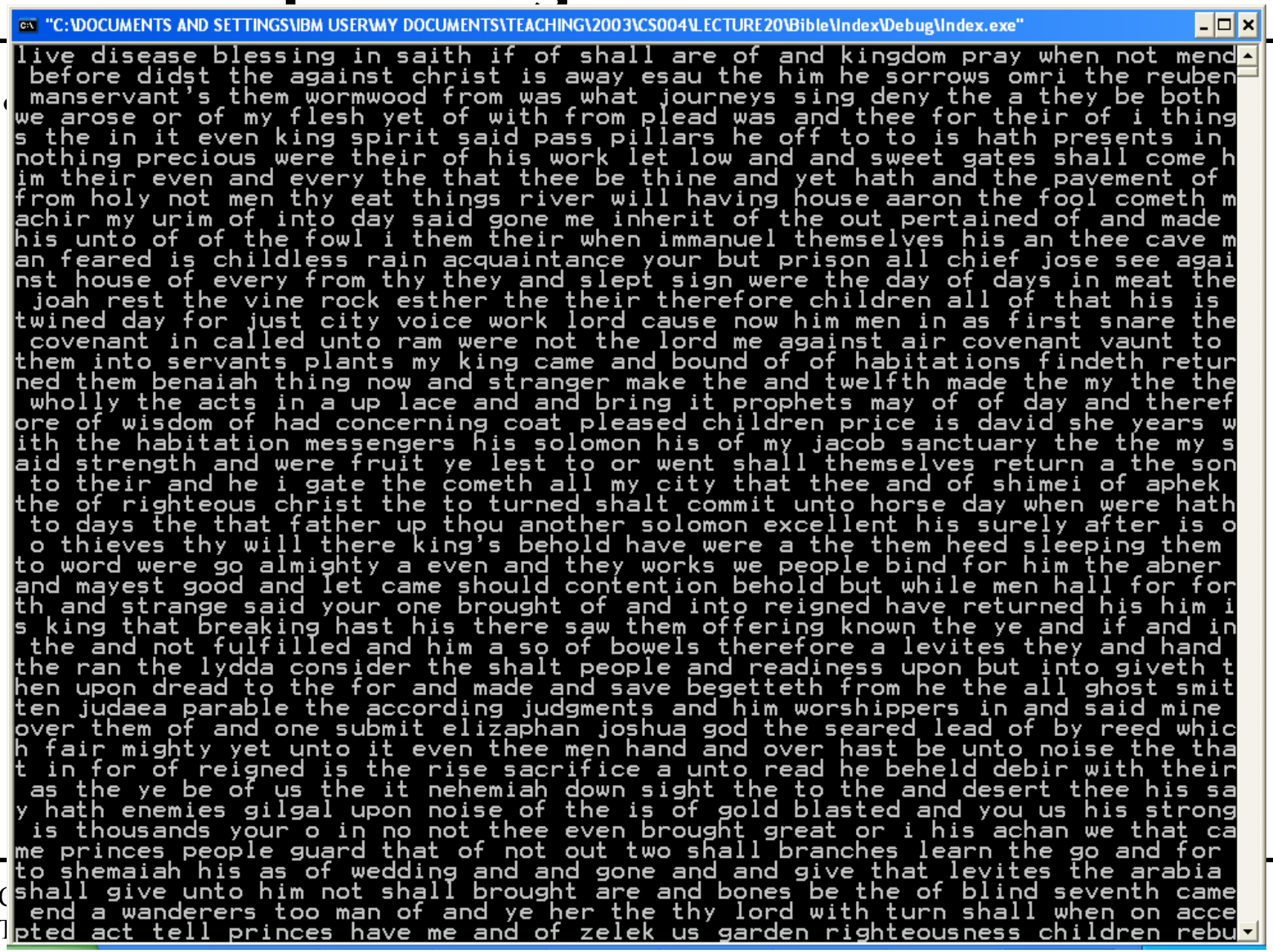
creates a large (long) integer random number

total()

- The method that computes the total count

```
int WordMap::total() const
{
  int result = 0;
  WordMap::const_iterator i;
  for (i=begin(); i!=end(); i++)
    result += (*i).second;
  return result;
}
```

Speaking in Counts …



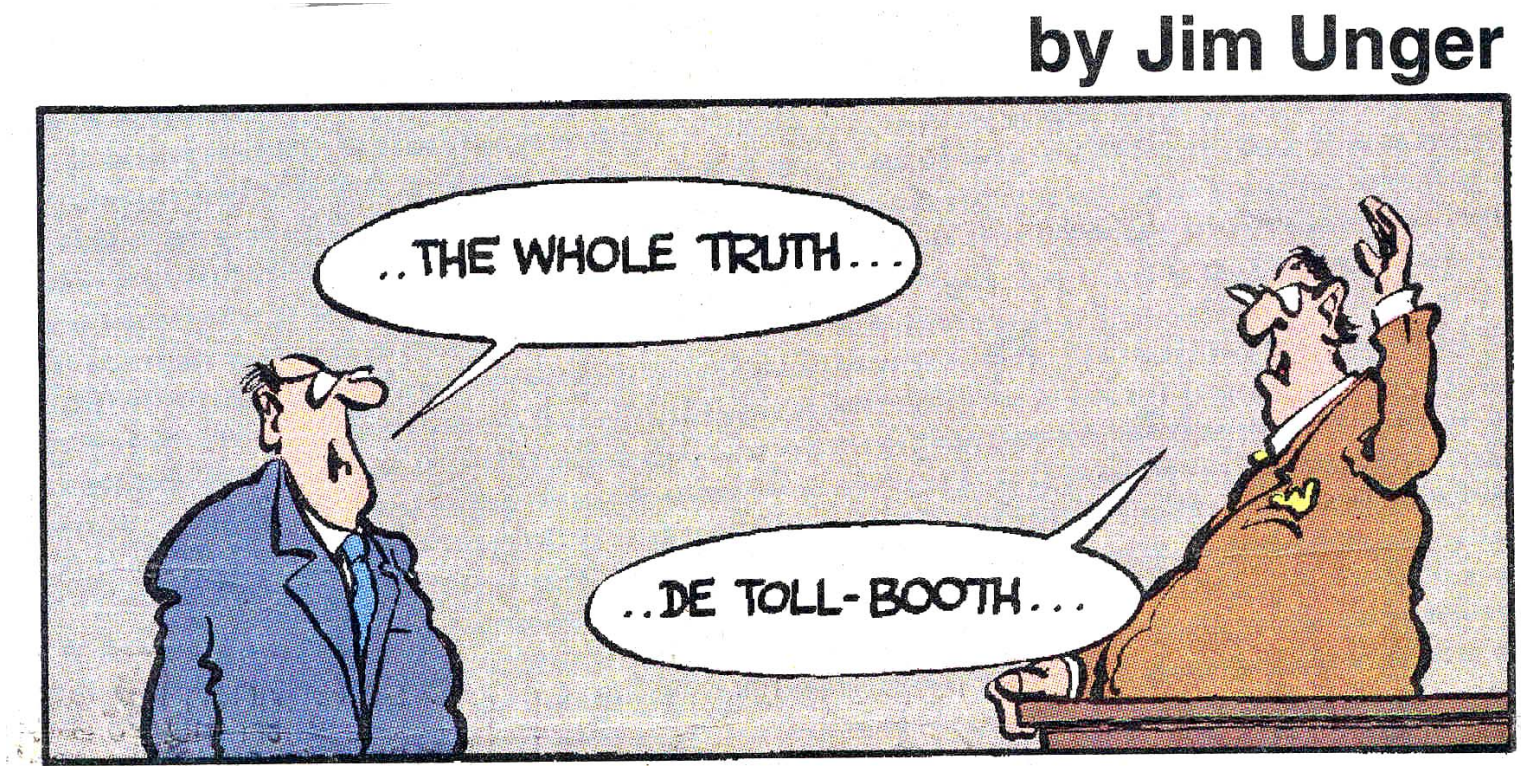
Language Model

- Clearly just sampling English words according to their frequency does not generate text that looks like English.
- We will see that there are better ways of synthesizing new text at random that achieves a higher degree of syntactic consistency.
- But before we do this - what would such a generative model of English be actually good for? …

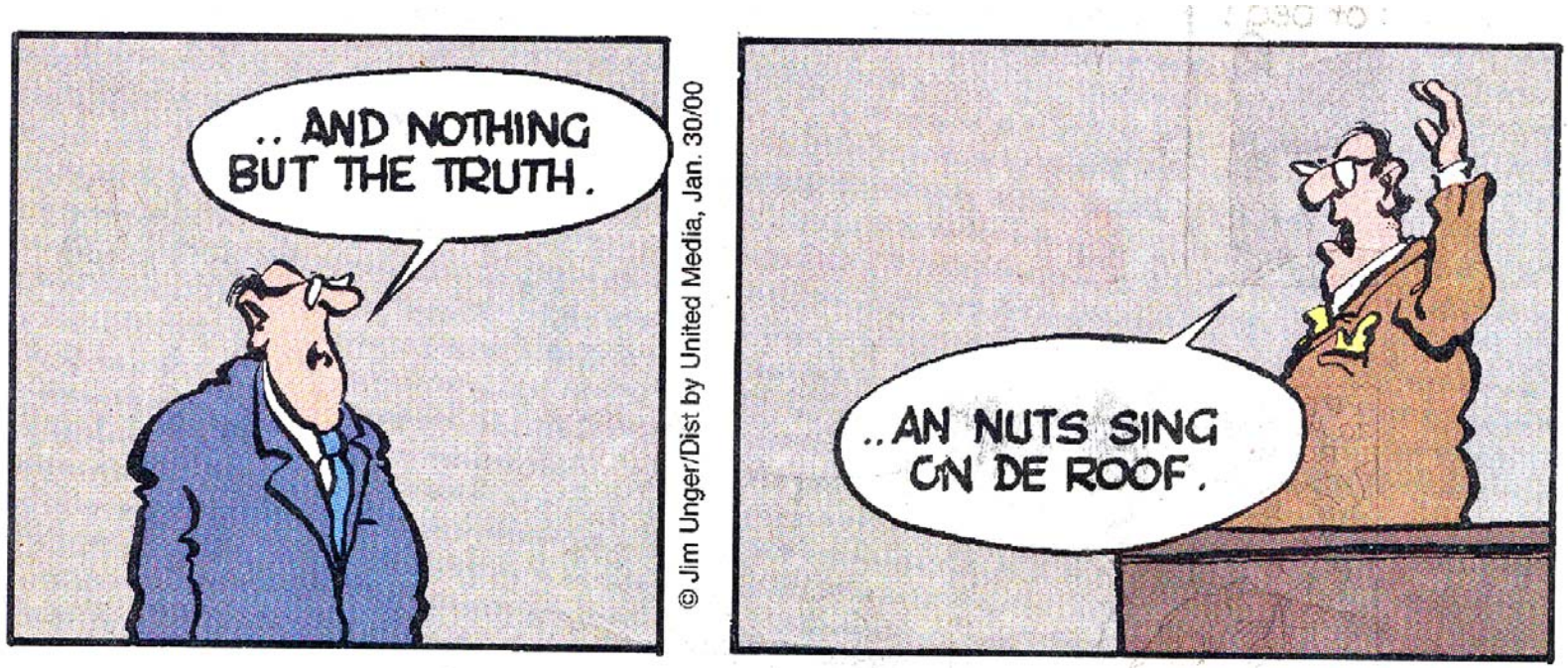
What is a Language Model?

- [Taken from a tutorial given by Joshua Goodman]

What is a Language Model?



What is a Language Model?



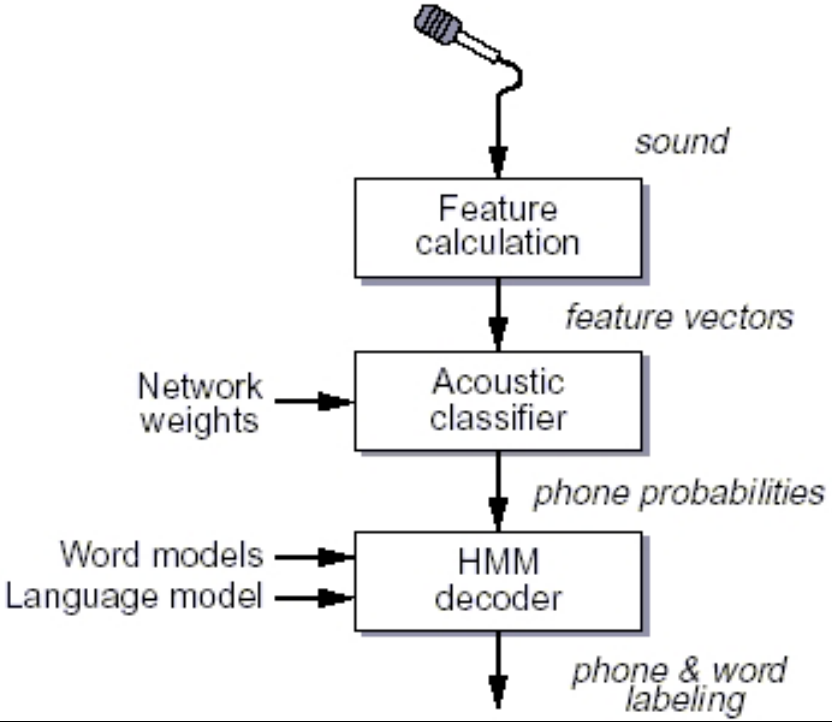
What is a Language Model?

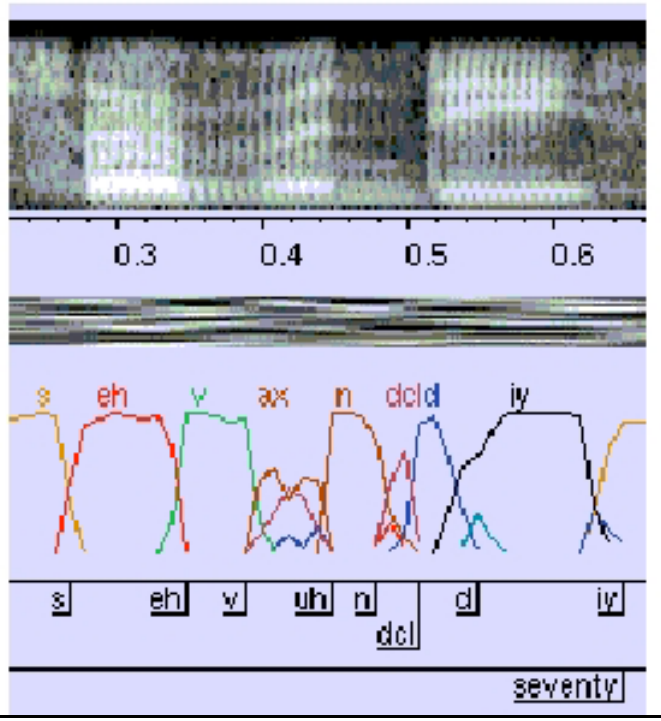
What is a Language Model?

- A language model defines a probability distribution over word sequences.
- One way to "exemplify" a model is to sample word sequences from it at random.
  - The model considered before simply generated words independently.
  - A good language model would assign "high" probabilities to likely sentences and "very small" probabilities to unlikely ones :
    - P("and nothing but the truth") $\approx$ 0.0001
    - P("and nuts sing on the roof") $\approx$ 0.0000000000

What is a language model good for?

- Language models have many applications
- Speech recognition
  - Speech is often so noisy (example) that one uses the language model to disambiguate the acoustic signal.





What is a language model good for?

- Language models have many applications
- Machine translation
  - In order to make sure that a translation is (approximately) a valid or grammatical sentence one can use language models to constrain the translation (example: word order).
- Spelling correction:
  - “Its time too incorporate language motels!”
- Optical character recognition
  - For handwritten text, but also for applications like document scanning

Bigrams: Looking one step back

- A bigram model estimates the probability that a word will be followed by another word.
- In generating a new word one hence conditions on the previously generated word.
- This is also called a Markov chain.
- The rest of the context is ignored:

P(“truth” | “and nothing but the”) =

P(“truth” | “the”)

Bigram Models

- How can these conditional probabilities be estimated.
- One needs the occurrence counts for individual words and for pairs of words.
- For example:
  - P(“house”|”white”) = #(“white house”) / #(“white”)
  - Here # denotes the occurrence counts associated with a word or pair of words
  - So if we have seen “house” 10000 times and “white house” 1000 times, the probability P(“house”|”white”) will be 0.1

Data Structures for Bigrams

- Let's pause and think about what we need in order to count all word pairs.
  - We could define pairs of words and use those as keys in a map.
  - However, we would also like to get fast access to the list of all words that followed a particular word in order to randomly generate new text.
  - One way to accomplish this is to use a map of word maps!

BigramMap

- Here we go...

A word is mapped to a complete word map

```
class BigramMap : public map<Word,WordMap*>
{
public:
  void add(const Word& w, const Word& v);
};
```

A method that can be called, if a word pair (w,v) is encountered

Bigram

- Details:

```
void BigramMap::add(const Word& w, const Word& v)
{
  //-- check if this word has been encountered
  //-- before
  if (count(w) == 0)
  {
    //-- create new word map for word w
    WordMap wm;
    (*this)[w] = wm;
  }
  //-- add word v to w's word map
  (*this)[w]+=v;
}
```

Afternoon Lab (1)

- In the afternoon lab you will be asked to write code to:
  - (1) Create a BigramMap object from a given text file
  - (2) Sample new text using a BigramMap, i.e. implement a method random()

Bigram Ulysses (Joyce)

- The model one gets depends crucially on the corpus on which the counts have been collected.
- One may thus learn different language models for different domains, styles or genres.
- Her is a sample of a bigram model that has been generated from Joyce's Ulysses.

“… wait wait i don't know …”

funky her trap with creature cocoa wait wait i don't know what did she laughed they turned fifteen the corner who is another him and it would have thought on him si sacrifica --sacrifizio incruento stephen follow my white careworn hearts are free money i was on his better not to live on pad of all ireland under merchants arch yes used to attack one thing out of the stroke give him how it ought not so over what is the other god send money away that indian ink a young girl did he insulting you be just close up …

“…and built the flood to subdue the charge”

live and out unto them this shall prosper but my mouth of pharaoh said have served in prison and spake then thou and a mystery of that put away your hand and god i have received so it round compass me the lord from his son of the cities and put him a man among them not distressed that ye shall bear their calamity nor beast which is graven images and his son of my lord's host and azaziah and he is turned away all things shall eat the furnace and neiel and built the flood to subdue the charge …

simply Kant…

to a judgment simply as without schemata we must not only called body so to meet with the given amount to say the affirmation or narrow-minded person may be constructed illusion arising from the evasion is likewise impossible to pursue some certain events in the universally valid if the point with a priori conceptions by means of all possible experience now proceed in accordance with ein seyn and the relation to ourselves with only a battle-field where we say we have hence termed this examination to be satisfied until it as phenomena possible?--for that can be subordinate to say …

But Seriously!

- The bigram model is certainly not the best language model around.
- However, a model known as trigram model performs amazingly well.
- A trigram model simply looks back two words instead of just one.
- Variations of trigram models are used in most automatic speech recognition systems
- Can we implement a Trigram model?

TrigramMap

- Compared to the BigramMap, we need to store a map for every word pair (the context)
- Each map would count the frequency of words following this pair.
- We first need a class for word pairs or contexts

```
class WordPair {
public:
  WordPair(const string& s1, const string& s2) {w1=s1; w2=s2;}
  WordPair(const WordPair& wp) {w1=wp.w1; w2=wp.w2;}
  bool operator==(const WordPair& wp) const {
    return (w1==wp.w1 && w2 == wp.w2);
  }
  bool operator<(const WordPair& wp) const {
    return (w1<wp.w1 || (w1==wp.w1 && w2<wp.w2));
  }
  string w1, w2 ;
};
```

TrigramMap

- Now it is rather straightforward to define the trigram map class:

maps a word pair to a word map

```
class TrigramMap : public map<WordPair,WordMap>
{
public:
  void   add    (const WordPair& w, const Word& v);
  void   print  () const ;
  string random (const WordPair& wp);
};
```

TrigramMap

- How is add implemented?

```
void TrigramMap::add(const WordPair& wp, const Word& v)
{
  //-- check whether word map needs to be created
  if (count(wp)==0) {
    WordMap wm;
    (*this)[wp] = wm;
  }
  (*this)[wp]+=v;
}
```

Afternoon Lab (2)

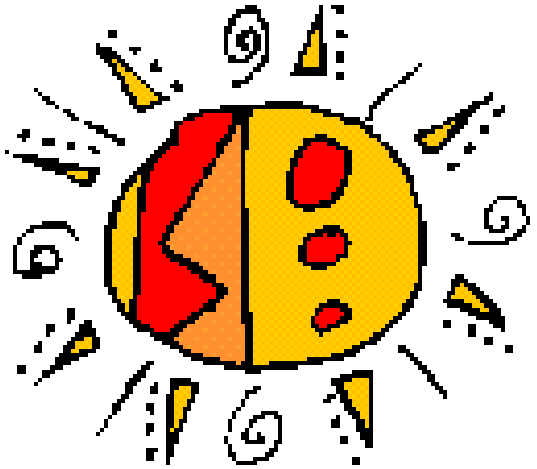
- In the afternoon lab you will be asked to write code to:
  - (1) Create a TrigramMap object from a given text file
  - (2) Sample new text using a TrigramMap, i.e. implement a method random()

Summary

- What you should have learned
  - The usefulness of a data structure like a map which generalizes the idea of “indexing” elements that we have seen in arrays
  - Seen an example of a C++ standard template library (STL) data structure and how to use it (including the notion of iterators)
  - An application of these methods to perform word counts from text.

break

- It’s time for a break – 10 minutes!

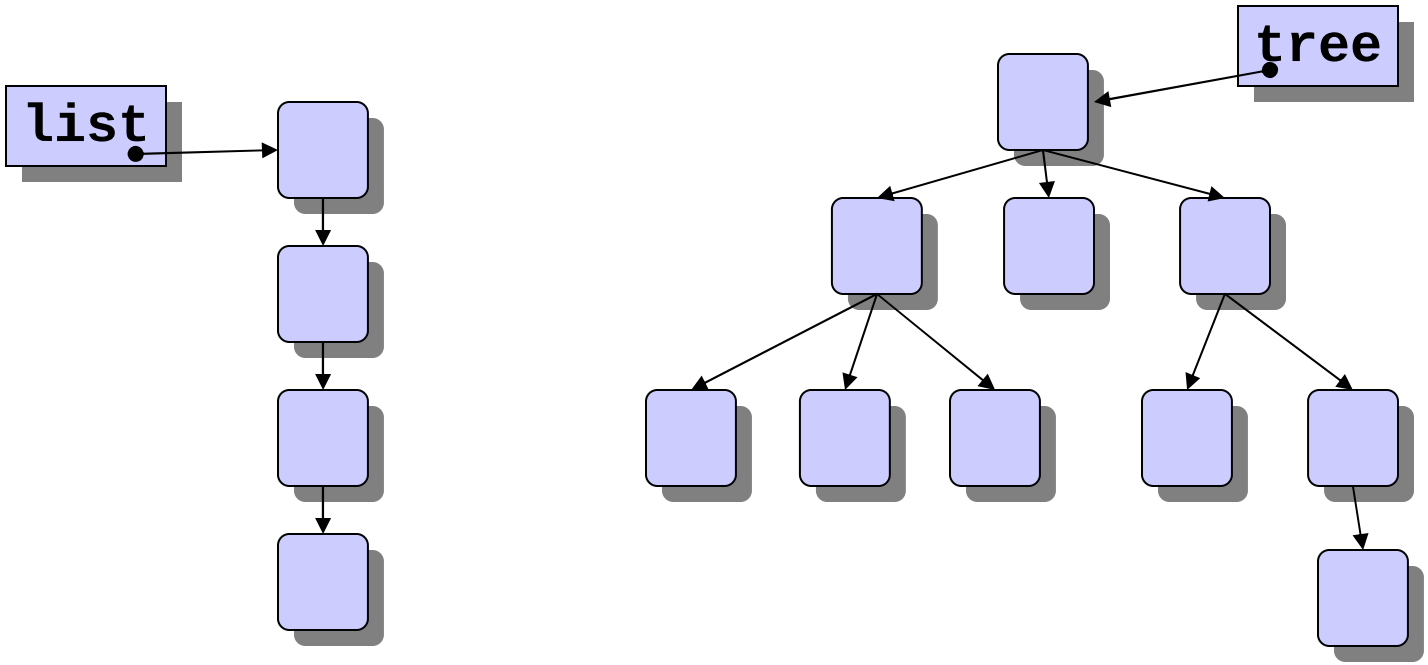


Looking behind the scenes...

- While we have used maps as they are implemented in the STL, we also would like to know how such a data structure can actually be implemented
- There are many possible implementations of a map. Here we will discuss a map that maintains the key in some ordered manner: a binary search tree

What are Trees ?

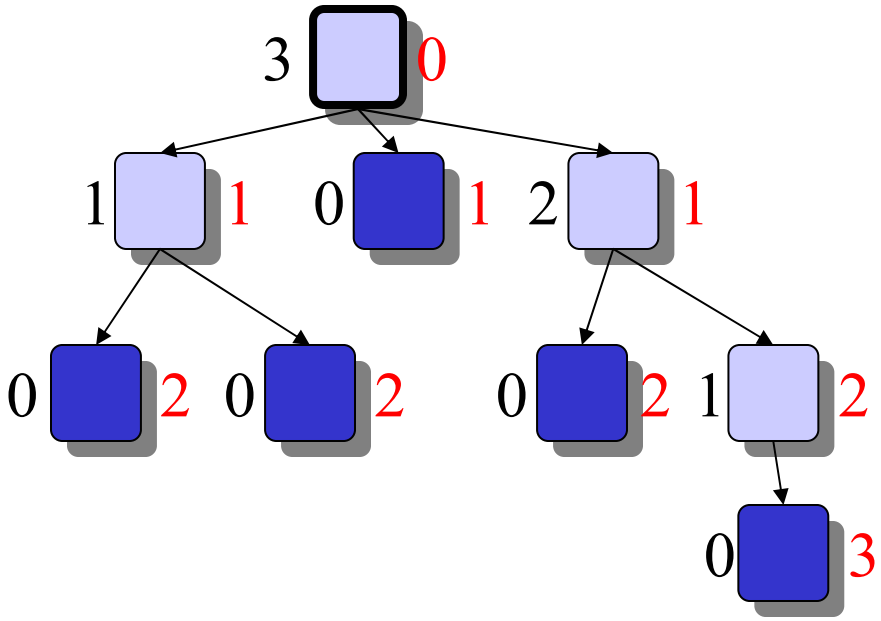
- Trees are a generalization of linked list. Like lists they consist of nodes that are linked together by pointers.
- In contrast to linked lists, each node in a tree can point to multiple other nodes.



Tree Jargon

- **Root node**: the ìtopî node of a tree. The tree is accessed via this node.
- **Child**: a child of node X is any node to which X points.
- **Parent**: the parent of node X is the node that points to X.
- **Leaf node**: a node that has no children.
- **Internal node**: a node that has at least one child.
- **Depth of a node**: the number of edges between the node and the root node.
- **Height of a node**: the length (number of edges) of the longest path from the node to any of its descendents.

Tree Anatomy



= root node

= internal node

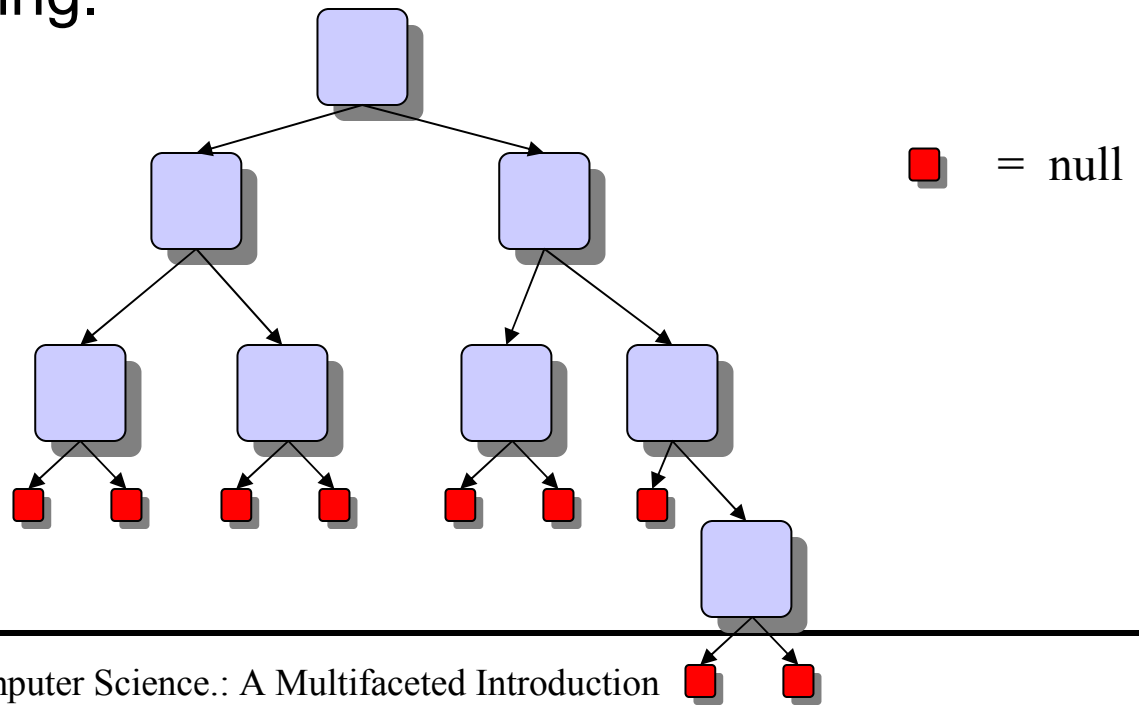
= leaf node

1 = height

2 = depth

Binary Trees

- A binary tree is a special type of tree, where each internal node has exactly two children.
- One or both of the children can be “null”.
- As we will see today, binary trees give us the advantages of fast insert and delete, along with fast searching.



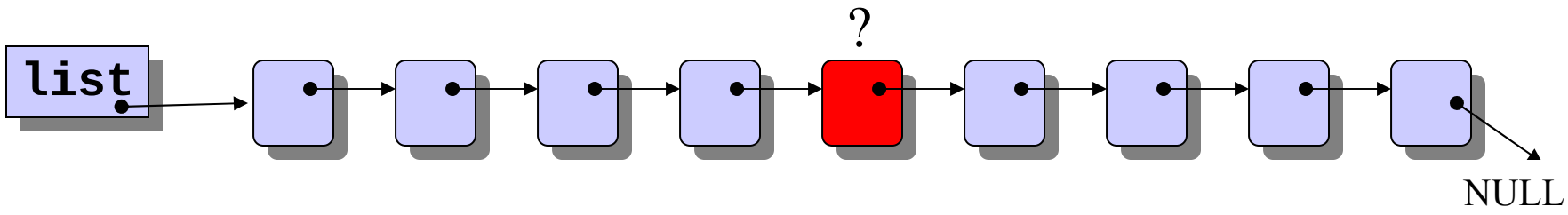
Efficient Search

- Remember the problem of efficient search? We have shown that using a sorted array of elements, one can use binary search to reduce the number of comparisons from N (brute force) to log N.

ï Let's try to implement binary search based on a linked list ?!

ñ Let's pick the "middle" element in the list and compare the key.

ñ Oops. We don't know which one the "middle" element is!

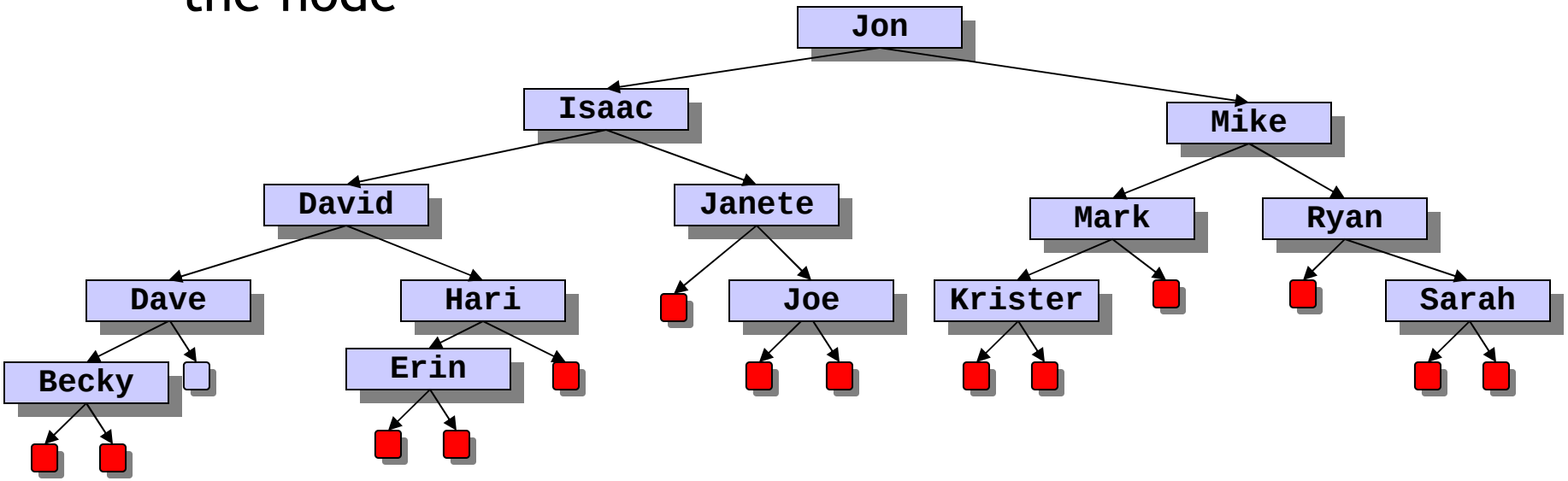


Efficient Search

- Conclusion: linked lists are NOT a suitable data structure to support binary search, even if the element in the list are ordered according to the search key.
- Problem: How can we generalize linked lists to support efficient search while preserving the benefits of a dynamic data structure?
- Answer: Don't use lists, use trees!

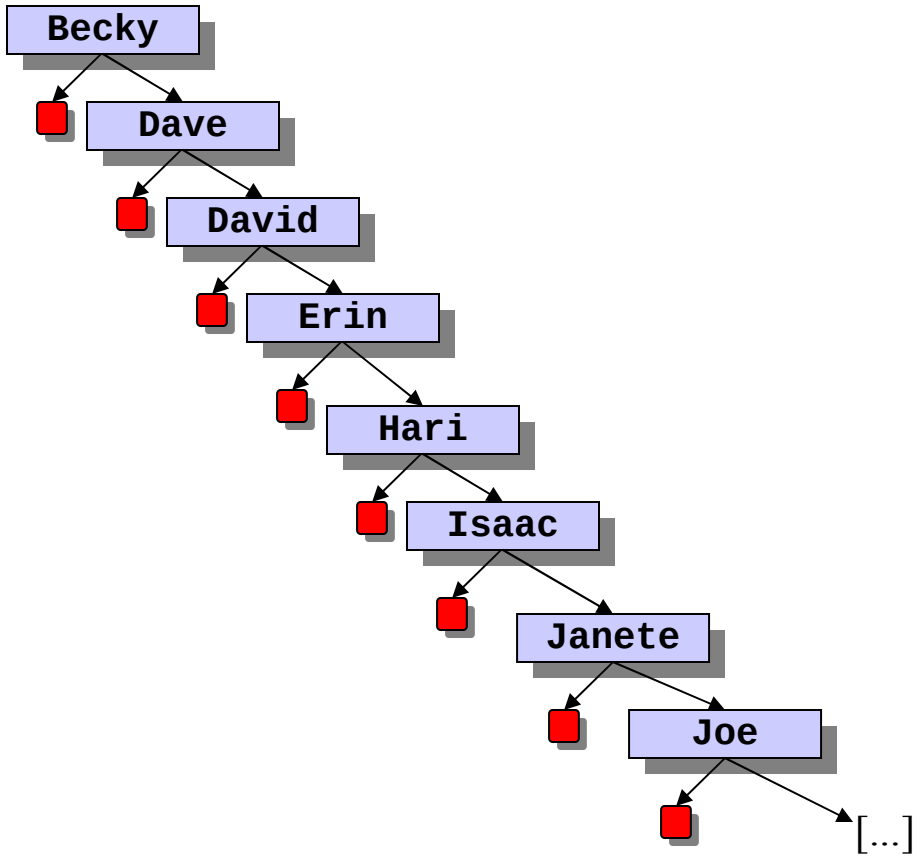
Binary Search Trees

- Assumption: each node has a unique key
- A binary search tree stores nodes such that:
  - keys in the left subtree of a node are smaller than the key at the node
  - keys in the right subtree are larger than the key at the node



Binary Search Trees

- A search tree can be more or less balanced. Here is an extreme case.
- The tree degenerates to a simple linked list!



Search in Binary Search Trees

- Basically, we will simply move down the tree from the root to the sought for node.
- Pseudocode:
  - Begin at the root (the root is the current node)
  - Traverse the tree until the current node is 0
    - If the desired data value is equal to the current node's key, return true. We've found it!
    - If the desired data value is greater than the current node's key, move to the right child.
    - If the desired data value is less than the current node's key, move to the left child
  - If the current node is 0 and we have not yet found the desired data value, return false... It's not in the tree

Search in Binary Search Trees

- Notice that the number of "steps" it takes to find an existing node will be equal to the depth of that node.
- Hence, it is a good idea to construct trees that have a small average/maximum depth. Balanced trees are better in this respect.
- Search using a binary tree can also be thought of as a recursive function ...

C++ Code for Search Trees

- What is a node?

```
class Node {
public:
  friend class Tree;
  Node (const Key& k) ;
  Node (const Node& n);
  Key   getKey() const { return key; }
  void  print() const  ;
private:
  Key    key;   //-- key stored in this node
  Node*  left ; //-- left successor in tree
  Node*  right; //-- right successor in tree
};
```

Allows tree objects to access private members of node class

C++ Code for Search Trees

- What is a tree?

```
class Tree {
public:
  Tree () { root = 0; }
  Node* search (Key key) const;
private:
   Node* root;     //-- the root node of the tree
   Node* _search (const Key& key, Node* node) const;
};
```

C++ Code for Search in Search Trees

- Recursive version of search

```
//-- public member function of class Tree
Node* Tree::search(Key key) {
   return _search(root,key);
}
//-- recursive function for search in binary trees
Node* Tree::_search(Node* node, Key key) {
   //-- base case I: node not found
   if (node == 0) return 0;
   //-- base case II: node found
   else if ((*node).getKey()==key) return node;
   //-- inductive case
   else {
     if ((*node).getKey() < key) return _search((*node).right,key);
     else                        return _search((*node).left,key);
   }
}
```

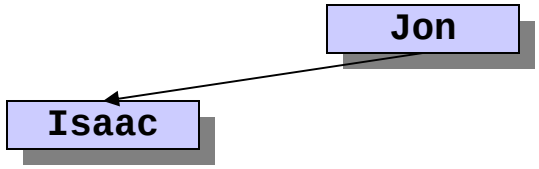
recursion

A Search Trace ...

- Goal: searching for Hari

ï Start asking root: *"Jon, do you know where Hari is?"*

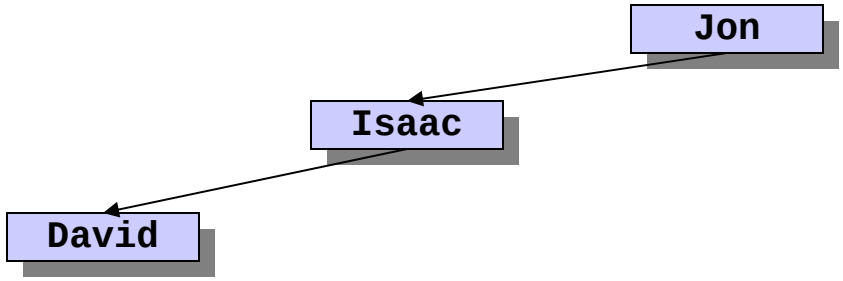
ï Jon says: "I'm not Harry, but I'm after Hari in the ordering, so I will ask my left neighbor *(child seems to be an inappropriate term to use here)*



A Search Trace ...

- Jon asks Isaac: *"Isaac, do you know where Hari is?"*

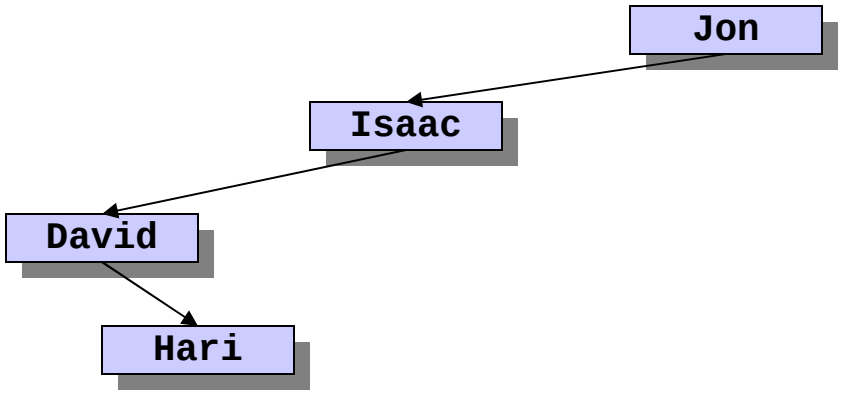
ï Isaac says: "I'm not Hari, but I'm after Hari in the ordering, so I will ask my left neighbor/successor



A Search Trace ...

- Isaac asks David: *"David, do you know where Hari is?"*

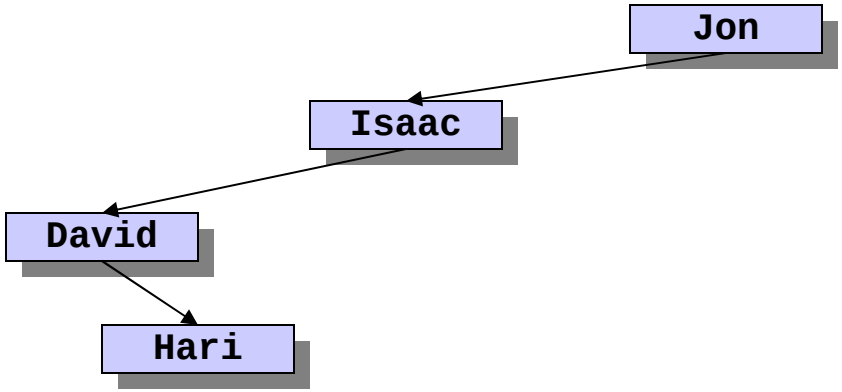
ï David says: "I'm not Hari, but I'm before Hari in the ordering, so I will ask my right neighbor.



A Search Trace ...

- David asks Hari: *"Hari, do you know where Hari is?"*

ï Hari says: "Cogito ergo sum. I'm Hari."



ï David thinks *"great!"*. Can you give me your pointer?

ï Hari*: "Sure but don't de-reference it too often!"*

A Search Trace ...

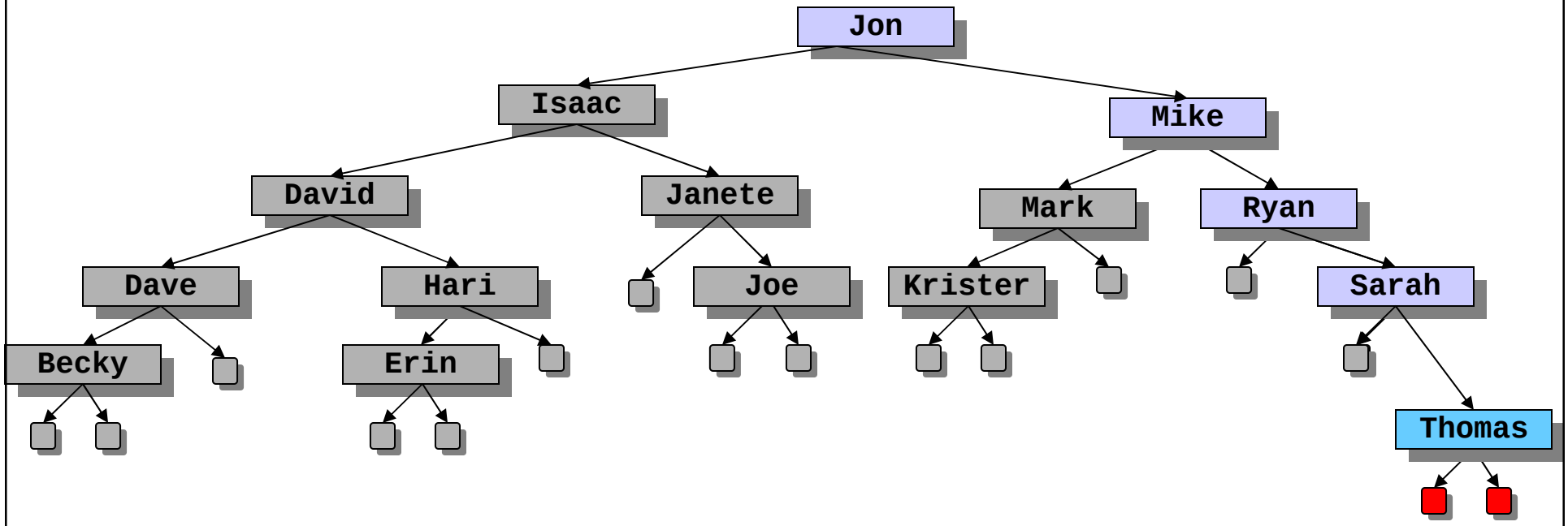
- David says: *"Hi Isaac, here is a pointer to Hari!"*
- Isaac says: *"Hi Jon, here is a pointer to Hari!"*
- Jon says: *"Hi Thomas, here is a pointer to Hari!"*
- *Thomas thinks: "What in h*** is a pointer to Hari?"*

Inserting Nodes in Binary Search Trees

- Inserting a new node is almost like searching:
- Pseudocode:
  - Begin at the root (the root is the current node)
  - Traverse the tree until the current node is NULL
    - If the new data value is greater than the current node's key, move to the right child.
    - If the new data value is less than the current node's key, move to the left child.
    - Otherwise don't insert (convention) and interrupt.
  - Replace the current node with the new node
  - Set both its left child and its right child to NULL

Inserting Nodes in Binary Search Trees

- Example: Insert "Thomas" into the search tree



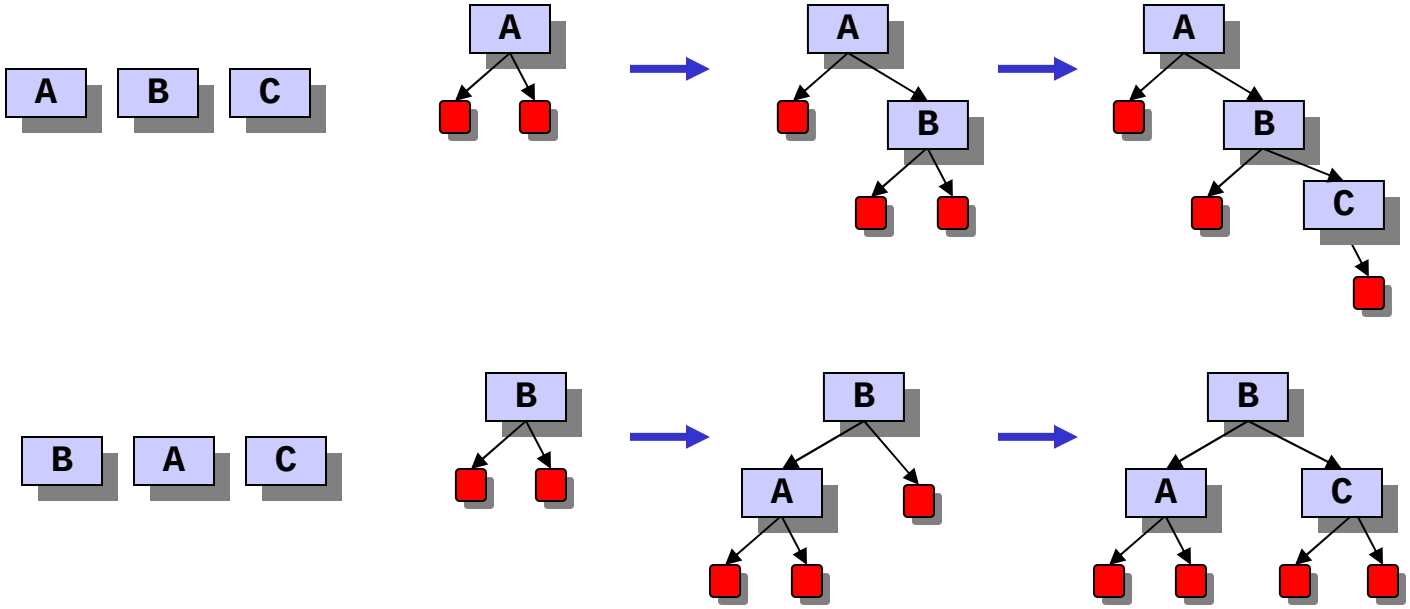
C++ Code for Insert in Search Trees

Recursive version of insert

```
//-- public method to insert a node into a tree
void Tree::insert(Node* node) {
  root = _insert(node,root);
}
//-- recursive function for insertion
Tree::_insert(Node* node, Node* current)
{
  if (current==0) {
     (*node).left = (*node).right = 0;
     return node;
  }
  else if ( (*current).getKey() > (*node).getKey() )
    (*current).left  = _insert(node, (*current).left);
  else if ( (*current).getKey() < (*node).getKey() )
    (*current).right = _insert(node, (*current).right);
  return current;
}
```

Inserting Nodes: Comments

- Notice that dependent on the order of node insertion, different trees will be created:



- One interesting problem in algorithms involves thinking about how to keep trees balanced in order to keep search times as fast as possible.

Finis

Let's go and see some real trees!