log n Searching

- From last time:
  - Concordance example
- Binary search
- What is log n time?
- Balanced search trees

Announcements

- Claire's office hours today changed to: 12:30 – 1:30
- This week's lab is based on Concordance example we'll do today.
- PA4 will be published soon.
- Midterm 2 is Tue. 4/6
  - Online exam: same platform as last time (you can use blank paper for scratch work)
  - Closed book, closed note…
  - extends to no other online-resources, including in browser, other apps or other devices.
  - possible code handout (check your email Mon night 4/5)

POLL: Map review

Respond at pollev.com/cbono

Asynchronous participation: Link to Map poll

Review of some Java Map properties

- Restrictions on KeyType in a Map

  (Same issue as with ElmtType of Set)
  - restrictions specific to kind of map: HashMap vs TreeMap
  - best if it's an immutable type (e.g., String, Integer)
  - Unsafe to "mutate" keys that are in a Map.
  - Entry's location in Map data structure depends on its key.
  - No restrictions on ValueType
- No iterator on Maps directly: have to use keySet() or entrySet() and iterate over resulting Set.

`Map` seen as an array

- Map abstract data type is sometimes called an *associative array*

```
System.out.println(scores.get("Joe"));
```

- ArrayList index syntax, but it's not random access
- But it's fast:
  - TreeMap: get, put, remove O(log n) each.
  - HashMap: get, put, remove O(1) each (!)
- E.g., Need an "array" indexed by a String?

. . . use a Map

Example: concordance

Problem: find the number of occurrences of each word in a text document.

- Why?
- (Variation also finds the page numbers or line numbers where those words occur in the document.)

Concord.java

- We'll work on code on Vocareum . . .

Review: Searching

- Use to answer questions such as:
  - Is Joe in the class?
  - What's Joe's score in the class?
  - What is Joe's array index? (e.g., so I can remove him)

- Previously discussed linear search (**`Names`** class and big-O lecture)

Binary search

- Binary search is an algorithm for searching in an ***ordered*** array or ArrayList.
- Example of *divide and conquer* algorithm.
- Idea:
  - compare target value with middle element in array.
  - if target is less, eliminate half the array from consideration (if greater, eliminate the other half).
  - Repeat this process for the half that could have the target.

Alex Bob Cat Dan Ed Fran Gary Hal Jan Ken Lou Mary Ned Opie

Binary search method specification

- binSearch returns the index of target, or -1 if not found. PRECONDITION: values in nums are in increasing order (i.e., nums[0] <= nums[1] <= nums[2]…)

```
public static int binSearch(int[] nums, int target)
```

Binary search details (iterative version)

```
public static int binSearch(int[] nums, int target)
{
    int low = 0;
    int high = nums.length-1;
    while (low <= high) {
      int mid = (low + high) / 2;
      if (target == nums[mid])
        return mid;
      else if (target < nums[mid])
        high = mid - 1;
      else
        low = mid + 1;
    }
    return -1;
}
```

Binary search code + example

```
public static int binSearch(int[] nums, int target)
{
     int low = 0;
     int high = nums.length-1;
     while (low <= high) {
       int mid = (low + high) / 2;
       if (target == nums[mid])
         return mid;
       else if (target < nums[mid])
         high = mid - 1;
       else
         low = mid + 1;
     }
     return -1;
}
```

0	1	2	3	4	5	6	7	8	9	10	11
3	5	8	10	15	25	26	30	32	37	50	100

Asynchronous participation: Link to Binary search poll

Binary search example

0	1	2	3	4	5	6	7	8	9	10	11
3	5	8	10	15	25	26	30	32	37	50	100

Big-O

- What's the worst case performance?
- It's the number of times we can successively divide n by 2.
- e.g., if n = 16 . . .
- That is, $2^{\#steps} = n$
- $\#steps = \log_2 n$

Big-O

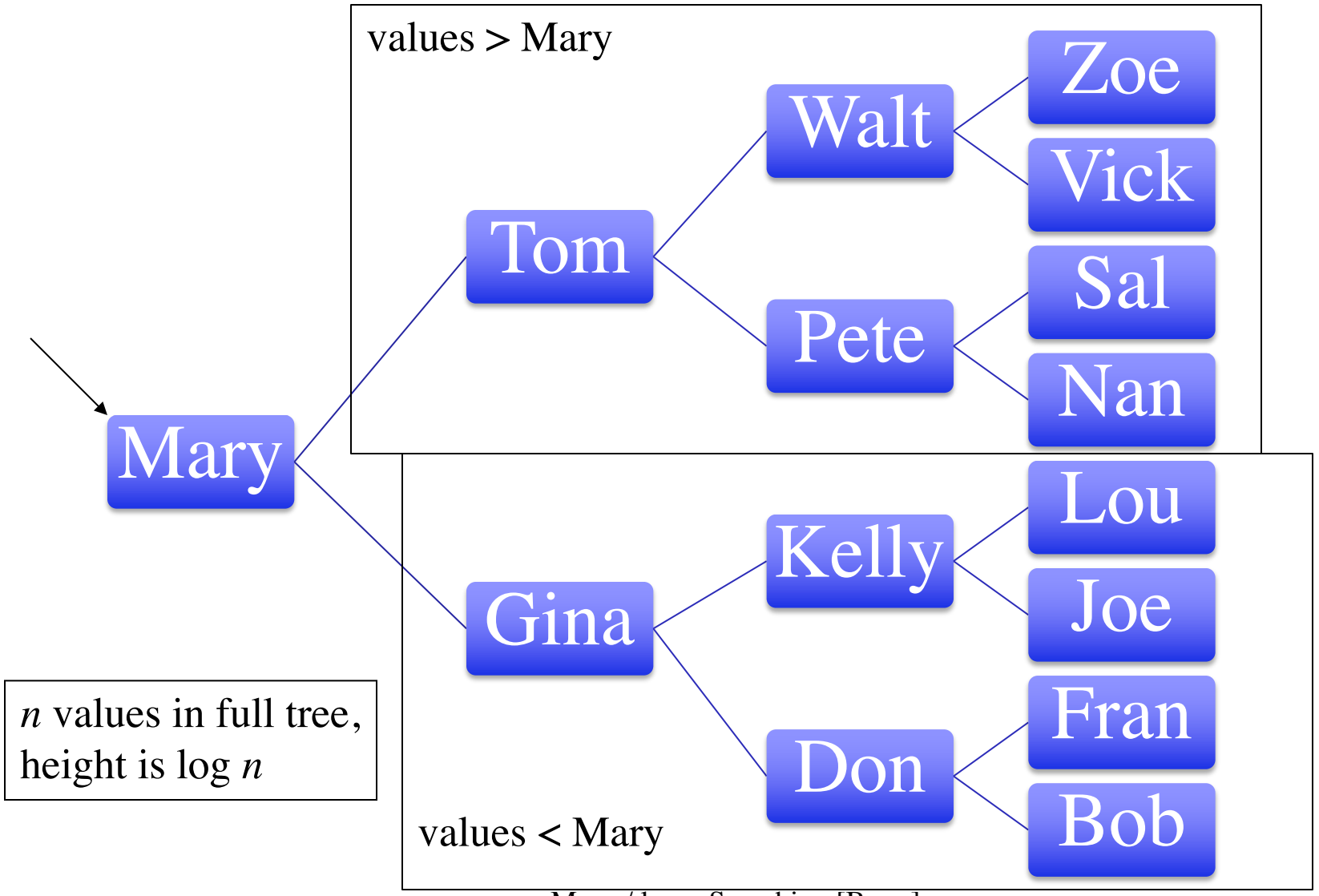
- What is $\log_2 n$?
  - number of bits to store the number n
    - e.g., 32767 takes 15 bits.
  - very fast: much faster than O(n)

Another log n example…

- Balanced search tree (what's in a **`TreeMap`** and **`TreeSet`**)
- Search is log n
- We'll do overview of the idea
  - related to binary search
- You are not responsible for detailed "balancing" algorithms

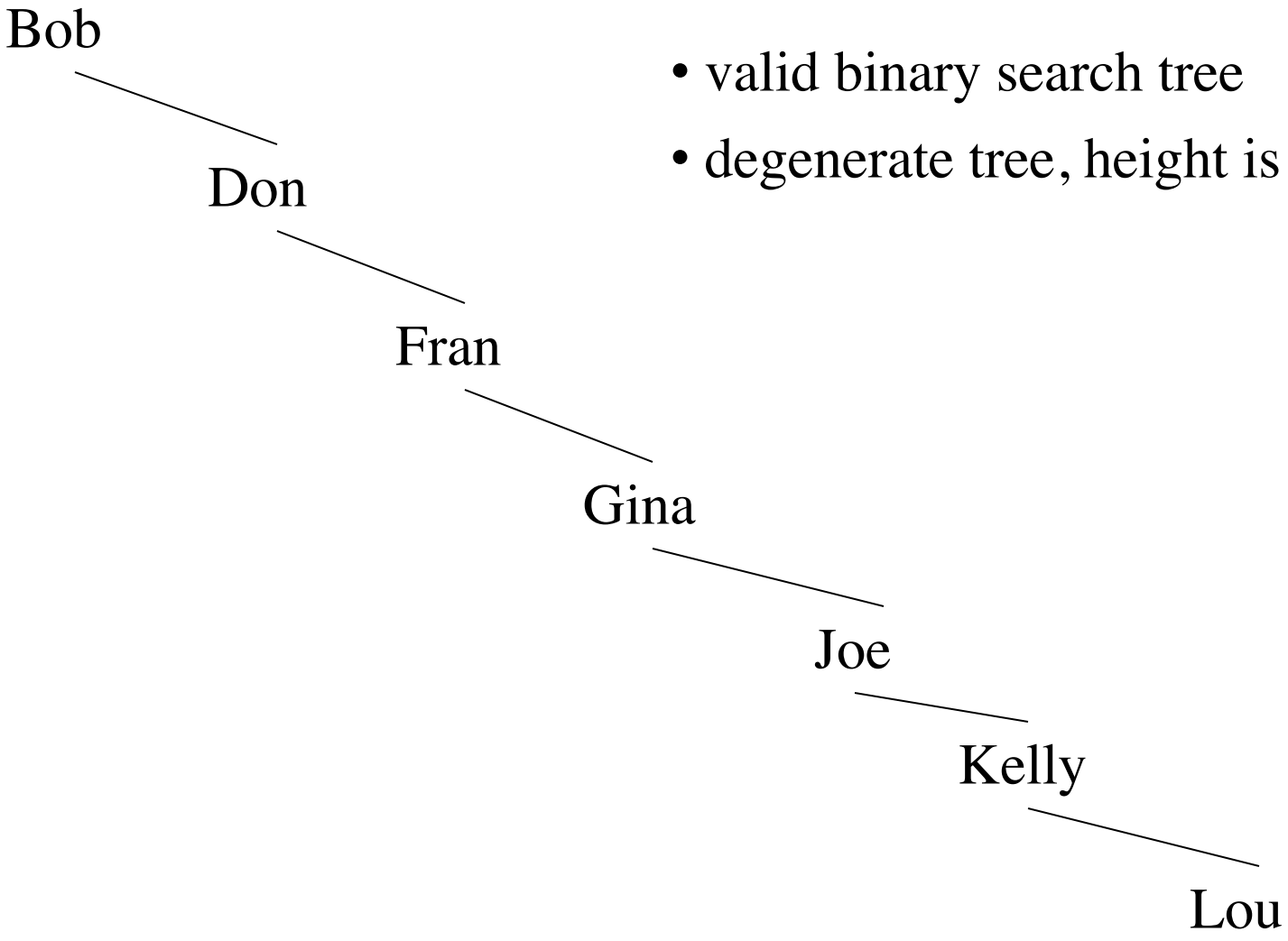
  (not enough time in the course)

Example of binary search tree



n values in full tree,
height is log *n*

Unbalanced binary search tree



- valid binary search tree
- degenerate tree, height is n

Balanced search trees

- Several variants: e.g., AVL trees, Red-Black trees, B-Trees
- Main idea
  - balanced tree: height is log n
  - **search** – uses binary search on a balanced tree
  - **insert** – inserts, rearranging to maintain the balance property in log n time
  - **remove** – removes, rearranging to maintain the balance property in log n time
  - **traverse** – O(n) total to visit n nodes

Traversing a binary search tree in sorted order

```
// inorder recursive tree traversal
void traverseInOrder(TreeType tree)
{
   if (tree is not empty) {
      traverseInOrder(tree.left);
      visit(tree.data);
      traverseInOrder(tree.right);
   }
}
```