Maps (cont.) / log n Searching

- Review **`Map`** interface
- Concordance example
- Binary search
- What is log n time?
- Balanced search trees

Announcements

- This week's lab is based on Concordance example from today.
- PA4 will be published by tomorrow.
- Midterm 2 is Tue. 4/4
  - Location: THH 101
  - Closed book, closed note, bring USC ID card
- Sample exams available.

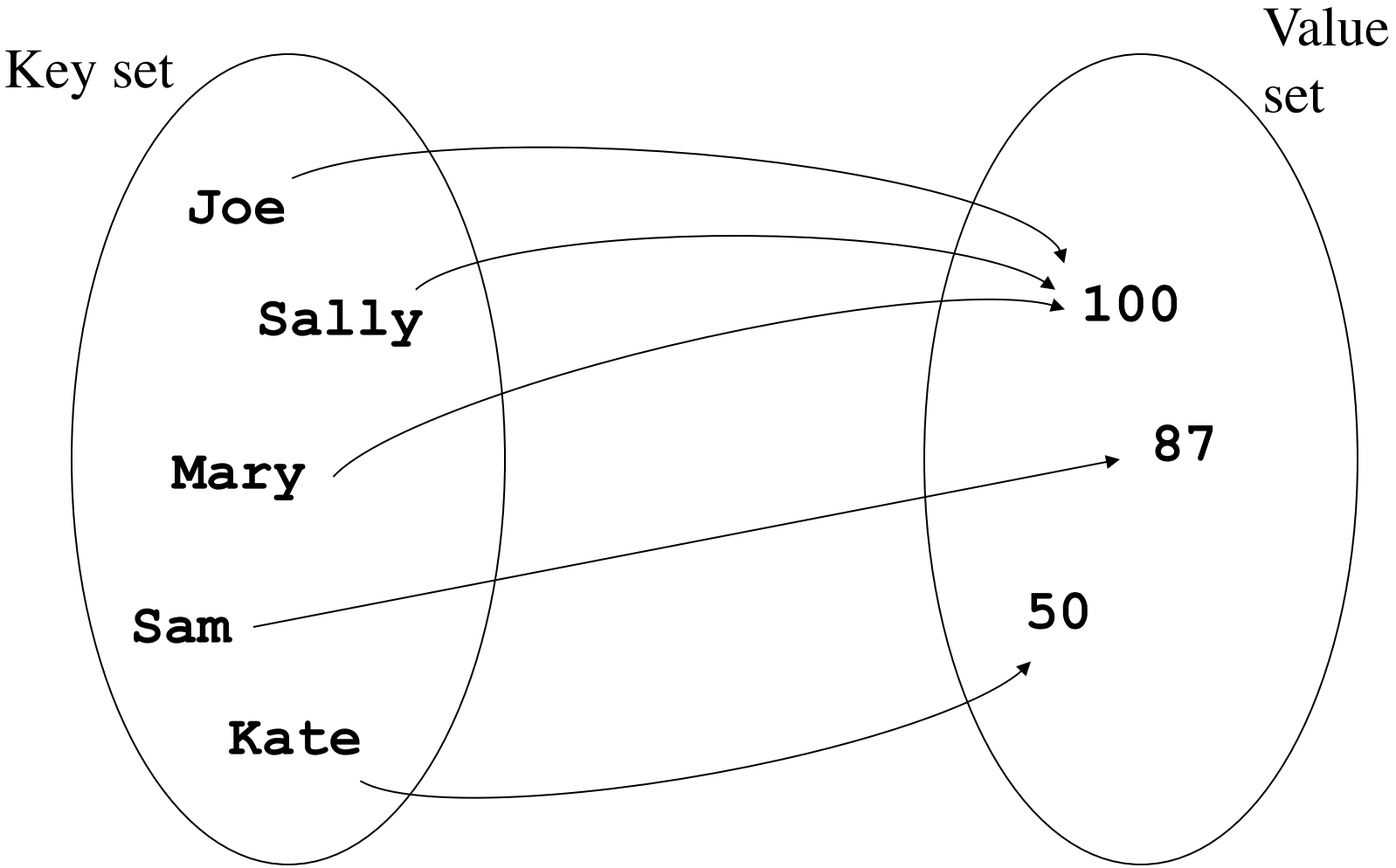
Review: Java Collections

- Collection is an interface in Java
- Linear collections:

  ArrayList, LinkedList, Stack, Queue

  – ordering of elements depended on order of insertion
- Two others: Set and Map
  – ordering is determined internally by the class based on value of the element

Example: map of students and their scores



Review: creating a **`Map`**

- Two kinds of maps:

```
Map<KeyType, ValueType> map =
      new HashMap<KeyType,ValueType>();
```

  - fastest. for when you don't care about order when iterating, or if you don't need to iterate.
  - **`KeyType`** must support **`equals()`** and **`hashCode()`**

```
Map<KeyType, ValueType> map =
      new TreeMap<KeyType,ValueType>();
```

  - for when you need to visit element in sorted order by keys.
  - **`KeyType`** must implement **`Comparable`** (has **`compareTo`**)

Review: Java **`Map`** interface

```
Map<String, Integer> scores =
        new TreeMap<String, Integer>();
```
create an empty map

```
scores.put("Joe", 98); // inserts
```
if key wasn't there, adds it and returns null,
o.w., returns the old value that went with this key

```
scores.put("Joe", 100); // updates
```
changes Joe's score to 100. if "Joe" hadn't been there before, this would have added him.

```
scores.remove("Joe");
```
if key was there, removes it and returns the value that went with this key,
o.w., returns null and map is unchanged

```
Integer jScore = scores.get("Joe");
```
return the value that goes with "Joe", or null if "Joe" is not in the map

Review: Iterating over all entries in a Map

Example with **`Map<String, Integer> scores`**

```
Map<String, Integer> scores =
      new TreeMap<String, Integer>();
. . .
Iterator<Map.Entry<String, Integer>> iter =
                    scores.entrySet().iterator();
while (iter.hasNext()) {
       Map.Entry<String, Integer> curr =
iter.next();
       System.out.println(curr.getKey() + " ”
                              + curr.getValue());
}
```

Example: concordance

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

In code directory: **`Concord.java`**
`ConcordDriver.java`

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, 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 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

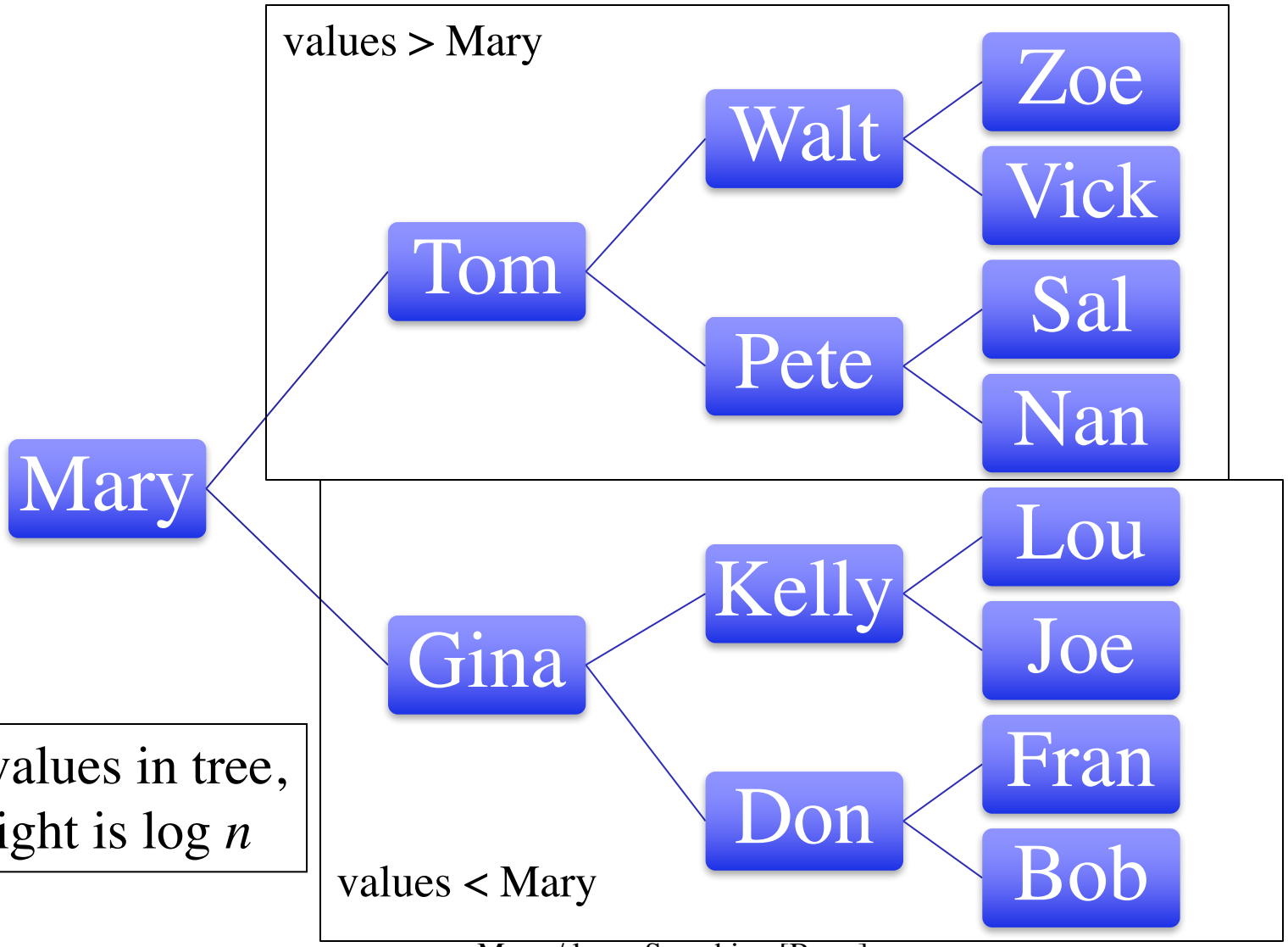
- What is $\log_2 n$?
  - number of bits to store the number n
    - e.g., 32767 takes 15 bits.
  - very fast: much faster than O(n)

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
- Not responsible for detailed "balancing" algorithms

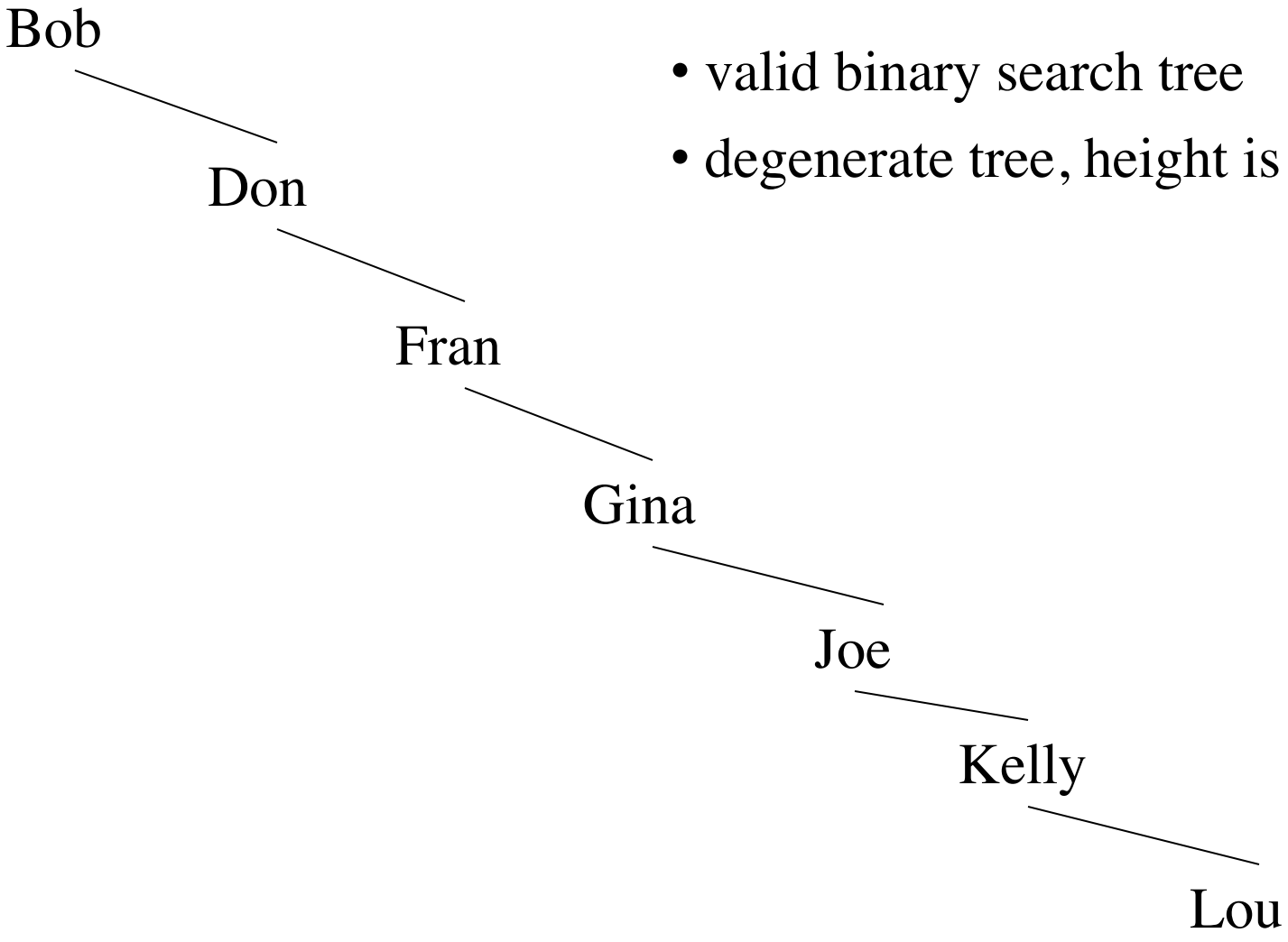
  (not enough time this semester)

Example of binary search tree



n values in 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);
   }
}
```