CSE 142

Searching

Outline for Today

- Review – sequential (linear) search of a list
- Binary search
- Comparing algorithms

Searching a List

- For this lecture, assume that we've got a list, and some collection of strings has been added to the list

```
ArrayList names = new ArrayList( );
names.add("frog");
names.add("rabbit");
names.add("aardvark");
```

- Problem: Look for a name in the list
  - If found, report its position
  - If not found, report something to indicate "not found"

(Note: normally this would be implemented as a method in some class. For now, we'll focus on just the search and ignore surrounding context.)

Linear Search

- Locate a string in the list

```
/** Return position of str in the list, or –1 if not present */
public int find(String str) {




}
```


Can we do better?

- How much work does linear search do?
- Can we do it faster?
  - No, if we don't know anything about the order of elements in the list
  - Yes, if the list is sorted

Binary Search – Informal

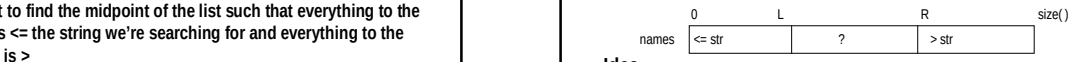
- Idea
  - Look in the middle of the list
  - If we haven't found what we're looking for, we can ignore half of the list and look at the other half
- Precondition: The list must be sorted for this to work
  - We'll assume names.get(0) <= names.get(1) <= ... <= names.get(names.size( )-1)

Binary Search – Goal

- Goal (more formally)
  - Want to find the midpoint of the list such that everything to the left is <= the string we're searching for and everything to the right is >
- Picture:

Binary Search – Strategy

- On a typical iteration, we have



- Idea:
  - Let mid = (L+R)/2
  - If names.get(mid) <= str, move L
  - If names.get(mid) > str, move R

(Note: In the book, Nino & Hosch use a slightly different invariant. For them, names.get(low) to names.get(high) is the *unexamined* region. In these slides, the unexamined region is names.get(L+1) to names.get(R-1). Either can be made to work correctly.)

String Comparisons

- We need to compare Strings to determine ordering, not just equality
- Can't use <, <=, etc. on objects
- Solution: method compareTo in class String

  s.compareTo(t)

  returns

  negative integer if s < t

  zero if s == t

  positive integer if s > t

Binary Search – Code

```
/** Return location of str in the list, or –1 if not present */
public int find(String str) {

    while ( _____ ) {



    }

}
```


Binary Search – Test

- Invent some data, try the algorithm

Binary Search – Test

Binary Search – Performance

- Is the extra complexity worth it?
- How much work is done to search a list of a given size?
- or, How big a list can be searched with *n* comparisons?

Binary & Linear Search Compared

- Linear search: work ~ size
- Binary search: work ~ $\log_2$ size
  - This is a fundamental difference – not just a constant speedup
  - But it requires a sorted list
- Graph: