

---

## CSE 142

### Searching

3/2/2004

(c) 2001-4, University of Washington

O-1

---

## Outline for Today

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

3/2/2004

(c) 2001-4, University of Washington

O-2

---

## 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.)

(Note: the same ideas apply to searching an array)

3/2/2004

(c) 2001-4, University of Washington

O-3

---

## 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) {
```

```
}
```

3/2/2004

(c) 2001-4, University of Washington

O-4

## 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

3/2/2004

(c) 2001-4, University of Washington

O-5

## 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  $\text{names.get}(0) \leq \text{names.get}(1) \leq \dots \leq \text{names.get}(\text{names.size}()-1)$   
(where  $s \leq t$  really means  $s.\text{compareTo}(t) \leq 0$ , but the simpler notation is easier to follow)

3/2/2004

(c) 2001-4, University of Washington

O-6

## Binary Search – Goal

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

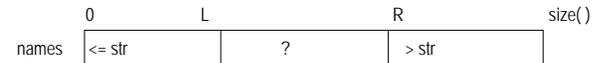
3/2/2004

(c) 2001-4, University of Washington

O-7

## Binary Search – Strategy

- On a typical iteration, we have



- Idea:
  - Let  $\text{mid} = (L+R)/2$
  - If  $\text{names.get}(\text{mid}) \leq \text{str}$ , move L
  - If  $\text{names.get}(\text{mid}) > \text{str}$ , move R

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

3/2/2004

(c) 2001-4, University of Washington

O-8

## Reminder: 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  $s > t$

---

3/2/2004

(c) 2001-4, University of Washington

O-9

## Binary Search – Code

---

```
/** Return location of str in the list, or -1 if not present */
```

```
public int find(String str) {
```

```
    while ( _____ ) {
```

```
    }
```

```
}
```

---

3/2/2004

(c) 2001-4, University of Washington

O-10

## Binary Search – Test

---

- Invent some data, try the algorithm

---

3/2/2004

(c) 2001-4, University of Washington

O-11

## Binary Search – Test

---

---

3/2/2004

(c) 2001-4, University of Washington

O-12

## 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?
- Work it out:

List Size                      # comparisons needed for binary search

---

3/2/2004

(c) 2001-4, University of Washington

O-13

## Binary & Linear Search Compared

---

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

---

3/2/2004

(c) 2001-4, University of Washington

O-14