4.2 Sorting and Searching

Section 4.2
Searching

The problem:

Given a collection of data, determine if a query is contained in that collection.

This is a fundamentally important problem for a myriad of applications (from finding webpages to searching for fragments of DNA).
Motivating Example: Spellchecker!
Exhaustive (Linear) Search

– Systematically enumerate all possible values and compare to value being sought
– For an array, iterate from the beginning to the end, and test each item in the array

Find “awsome"

Array of all words in dictionary
Linear Search

Scan through array, looking for key.

- **Search hit:** return array index.
- **Search miss:** return -1.

```java
public static int search(String key, String[] a) {
}
```
Linear Search

Scan through array, looking for key.

- **Search hit:** return array index.
- **Search miss:** return -1.

```java
public static int search(String key, String[] a) {
    int N = a.length;
    for (int i = 0; i < N; i++) {
        if (a[i] == key) {
            return i;
        }
    }
    return -1;
}
```
How Fast is Linear Search?

• Let’s say we have an array of 1,000 Strings.

• What’s the *absolute* fastest it can run?

• What if we make it an array of 2,000 Strings?
Binary Search

Quickly find an item (query) in a sorted list.

Examples: Dictionary, phone book, index, credit card numbers, ...

Binary Search:
• Examine the middle key
• If it matches, return its index
• Otherwise, search either the left or right half

Binary search in a sorted array (one step)
Binary Search

Find "J"
Binary Search Example

Find “awesome”

aardvark ... macabre ... Zyzzogeton
Binary Search Example

Find "awsome"

aardvark ... fable ... macabre
Binary Search Example

Find “awsome”

aardvark ... catfish ... fable
Binary Search Example

Find “awsome"

aardvark ... beetle ...

... catfish
Binary Search Example

Find “awsome”

aardvark ... awake ... beetle
Binary Search Example

Find “awsome”

awaken ... banjo ... beetle

Repeat a few more times…
Binary Search Example

Find “awsome"

awry

Return false
public static int search(String key, String[] a) {
    int lo = 0;
    int hi = a.length - 1;
    while (lo <= hi) {
        // Key is in a[lo...hi] or not present.
        int mid = lo + (hi - lo) / 2;
        int cmp = a[mid].compareTo(key);
        if (cmp > 0) hi = mid - 1;
        else if (cmp < 0) lo = mid + 1;
        else return mid;
    }
    return -1;
}
The `compareTo()` method

- really handy for comparing two Strings
  - returns how two Strings differ “lexicographically”

```java
str1.compareTo(str2);
```

- if the strings are **equal**, it returns 0
- if `str1` is “less than” `str2`, it returns a negative int
- if `str1` is “greater than” `str2`, it returns a positive int
The `compareTo()` method

```java
String str1 = "hello";
String str2 = "hey";
String str3 = "hi";

str1.compareTo(str1); // returns 0
str1.compareTo(str2); // returns -13
str3.compareTo(str2); // returns 4
```
How Fast is Binary Search?

• Let’s say we have an array of 1,000 Strings.

• What’s the *absolute* fastest it can run?

• What if we make it an array of 2,000 Strings?
Binary Search

**Analysis:** Binary search in an array of size $N$
- One compare
- Binary search in array of size $N/2$

$$
N \rightarrow N/2 \rightarrow N/4 \rightarrow N/8 \rightarrow \ldots \rightarrow 1
$$

**Q.** How many times can you divide by 2 until you reach 1?

**A.** $\log_2 N$

1

2 → 1
4 → 2 → 1
8 → 4 → 2 → 1
16 → 8 → 4 → 2 → 1
32 → 16 → 8 → 4 → 2 → 1
64 → 32 → 16 → 8 → 4 → 2 → 1
128 → 64 → 32 → 16 → 8 → 4 → 2 → 1
256 → 128 → 64 → 32 → 16 → 8 → 4 → 2 → 1
512 → 256 → 128 → 64 → 32 → 16 → 8 → 4 → 2 → 1
1024 → 512 → 256 → 128 → 64 → 32 → 16 → 8 → 4 → 2 → 1
Spell-Checking
Midsummer Night’s Dream

• Linear Search: **385,554 milliseconds** to check the entire text

• Binary Search: **104 milliseconds** to check the entire text

• Speedup: **3,707 times!!!**
"Can I Guess?"
Given the numbers 1 to 1,000, what is the minimum number of guesses needed to find a specific number if you are given the hint "higher" or "lower" for each guess you make?
Asked at Facebook
Akinator: The Web Genie

http://en.akinator.com/
Sorting

• Sorting problem. Rearrange N items in ascending order.

• Applications. Statistics, databases, data compression, bioinformatics, computer graphics, scientific computing, (too numerous to list), ...
Section 4.2
Sorting

pentrust.org

shanghaiscrap.org

Section 4.2
Section 4.2
Selection Sort

• Idea:
  – Find the smallest element in the array
  – Exchange it with the element in the first position
  – Find the second smallest element and exchange it with the element in the second position
  – Continue until the array is sorted
Example
To insert 12, we need to make room for it by moving first 36 and then 24.
Insertion Sort
Insertion Sort
Insertion Sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange value with larger ones to left, one-by-one.

<table>
<thead>
<tr>
<th>5</th>
<th>2</th>
<th>4</th>
<th>6</th>
<th>1</th>
<th>3</th>
</tr>
</thead>
</table>

At each iteration, the array is divided into two sub-arrays:

- **Sorted**
- **Unsorted**
Insertion Sort
public class Insertion {

    public static void sort(int[] a) {
        int N = a.length;
        for (int i = 1; i < N; i++)
            for (int j = i; j > 0; j--)
                if (a[j-1] > a[j])
                    exch(a, j-1, j);
                else break;
    }

    private static void exch(int[] a, int i, int j) {
        int swap = a[i];
        a[i] = a[j];
        a[j] = swap;
    }
}
Insertion Sort: Call By Reference

```java
class Insertion {
    public static void sort(int[] a) {
        int N = a.length;
        for (int i = 1; i < N; i++)
            for (int j = i; j > 0; j--)
                if (a[j-1] > a[j])
                    exch(a, j-1, j);
                else break;
    }

    private static void exch(int[] b, int i, int j) {
        int swap = b[i];
        b[i] = b[j];
        b[j] = swap;
    }
}
```

b contains a copy of the address in a. Both point to the same contents.
Insertion Sort: Observation

Observe and tabulate running time for various values of N.

- Data source: N random numbers between 0 and 1.
- Machine: iMac Core i5 2.7GH, 12GB RAM.
- Timing: `System.currentTimeMillis()`.

<table>
<thead>
<tr>
<th>N</th>
<th>Comparisons</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,000</td>
<td>6.2 million</td>
<td>0.016 s</td>
</tr>
<tr>
<td>10,000</td>
<td>25 million</td>
<td>0.063 s</td>
</tr>
<tr>
<td>20,000</td>
<td>99 million</td>
<td>0.211 s</td>
</tr>
<tr>
<td>40,000</td>
<td>400 million</td>
<td>0.79 s</td>
</tr>
<tr>
<td>80,000</td>
<td>1600 million</td>
<td>3.125 s</td>
</tr>
</tbody>
</table>
Empirical Analysis

Observation. Number of compares depends on input family.

- Descending: $\sim N^2 / 2$
- Random: $\sim N^2 / 4$
- Ascending: $\sim N$
Mathematical Analysis

Worst Case. (descending)

- Iteration $i$ requires $i$ comparisons.
- Total $= (0 + 1 + 2 + \ldots + N-1) \sim \frac{N^2}{2}$ compares.

<table>
<thead>
<tr>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>D</th>
<th>C</th>
<th>B</th>
<th>A</th>
</tr>
</thead>
</table>

Average Case. (random)

- Iteration $i$ requires $i / 2$ comparisons on average.
- Total $= (0 + 1 + 2 + \ldots + N-1) / 2 \sim \frac{N^2}{4}$ compares

<table>
<thead>
<tr>
<th>A</th>
<th>C</th>
<th>D</th>
<th>F</th>
<th>H</th>
<th>J</th>
<th>E</th>
<th>B</th>
<th>I</th>
<th>G</th>
</tr>
</thead>
</table>

$i$
Mathematical Analysis

Worst Case. (descending)

- Iteration $i$ requires $i$ comparisons.
- Total = $(0 + 1 + 2 + \ldots + N-1) \sim N^2 / 2$ compares.

Average Case. (random)

- Iteration $i$ requires $i / 2$ comparisons on average.
- Total = $(0 + 1 + 2 + \ldots + N-1) / 2 \sim N^2 / 4$ compares
Sorting Challenge 1

• Q. A credit card company sorts 10 million customer account numbers, for use with binary search.

• Using **insertion sort**, what kind of computer is needed?

   A. Toaster
   B. Cell phone
   C. Your laptop
   D. Supercomputer
   E. Google server farm
**Insertion Sort: Lesson**

**Lesson.** Supercomputer can't rescue a bad algorithm.

<table>
<thead>
<tr>
<th>Computer</th>
<th>Comparisons Per Second</th>
<th>Thousand</th>
<th>Million</th>
<th>Billion</th>
</tr>
</thead>
<tbody>
<tr>
<td>laptop</td>
<td>$10^7$</td>
<td>instant</td>
<td>1 day</td>
<td>3 centuries</td>
</tr>
<tr>
<td>super</td>
<td>$10^{12}$</td>
<td>instant</td>
<td>1 second</td>
<td>2 weeks</td>
</tr>
</tbody>
</table>
Moore's Law

Moore's Law. Transistor density on a chip doubles every 2 years.

Variants. Memory, disk space, bandwidth, computing power/$.

http://en.wikipedia.org/wiki/Moore's_law
Moore's Law and Algorithms

Quadratic algorithms do not scale with technology.

• New computer may be 10x as fast.
• But, has 10x as much memory so problem may be 10x bigger.
• With quadratic algorithm, takes 10x as long!

Lesson. Need linear or $N \log N$ algorithms to keep pace with Moore's law.

“Software inefficiency can always outpace Moore's Law. Moore's Law isn't a match for our bad coding.” – Jaron Lanier