4.2 Sorting and Searching

Sorting

Sorting problem. Rearrange N items in ascending order.

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

Insertion Sort

The Bridge Table
Insertion sort

- Brute-force sorting solution.
- Move left-to-right through array.
- Exchange next element with larger elements to its left, one-by-one.

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6</td>
<td>and had him his was you but</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>and had him his was you but</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>and had him his was you but</td>
</tr>
</tbody>
</table>

Inserting a[k] into position by exchanging with larger entries to its left

Insertion Sort: Java Implementation

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

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

Insertion Sort: Empirical Analysis

**Observation**. Number of compares depends on input family.
- **Descending**: \( \sim \frac{N^2}{2} \)
- **Random**: \( \sim \frac{N^2}{4} \)
- **Ascending**: \( \sim N \)

Insertion Sort: Mathematical Analysis

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

**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.

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>10^6</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 per $.

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 as problem may be 10x bigger.
- With quadratic algorithm, takes 10x as long!

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

Lesson: Need linear (or linearithmic) algorithm to keep pace with Moore’s law.

Mergesort

Mergesort algorithm:

- Divide array into two halves.
- Recursively sort each half.
- Merge two halves to make sorted whole.

```
input
was had him and you his the but
sort left
and had him was you his the but
sort right
and had him was but his the you
merge
and but had him his the was you
```

Mergesort: Example
Merging

Combine two pre-sorted lists into a sorted whole.

How to merge efficiently? Use an auxiliary array.

```java
String[] aux = new String[N];
// merge into auxiliary array
int i = lo, j = mid;
for (int k = 0; k < N; k++) {
    if (i == mid) aux[k] = a[j++];
    else if (j == hi) aux[k] = a[i++];
    else if (a[j].compareTo(a[i]) < 0) aux[k] = a[j++];
    else aux[k] = a[i++];
}
// copy back
for (int k = 0; k < N; k++)
    a[lo + k] = aux[k];
```

Mergesort: Java Implementation

```java
public class Merge {
    public static void sort(String[] a) {
        sort(a, 0, a.length);
    }
    // Sort a[lo, hi]
    public static void sort(String[] a, int lo, int hi) {
        if (hi - lo <= 1) return;
        // recursively sort left and right halves
        int mid = lo + (hi - lo) / 2;
        sort(a, lo, mid);
        sort(a, mid, hi);
        // merge sorted halves (see previous slide)
    }
}
```

Mergesort: Mathematical Analysis

Mathematical analysis.

<table>
<thead>
<tr>
<th>best</th>
<th>average</th>
<th>worst</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/2N</td>
<td>N log₂ N</td>
<td>N log₂ N</td>
</tr>
</tbody>
</table>

Validation. Theory agrees with observations.

<table>
<thead>
<tr>
<th>N</th>
<th>actual</th>
<th>predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>10k</td>
<td>110 thousand</td>
<td>133 thousand</td>
</tr>
<tr>
<td>20m</td>
<td>460 million</td>
<td>485 million</td>
</tr>
<tr>
<td>50m</td>
<td>1,216 million</td>
<td>1,279 million</td>
</tr>
</tbody>
</table>
Sorting Challenge 3

Q. What's the fastest way to sort 1 million 32-bit integers?

Mergesort: Lesson

Lesson. Great algorithms can be more powerful than supercomputers.

<table>
<thead>
<tr>
<th>Computer</th>
<th>Comparisons Per Second</th>
<th>Insertion</th>
<th>Mergesort</th>
</tr>
</thead>
<tbody>
<tr>
<td>laptop</td>
<td>$10^7$</td>
<td>3 centuries</td>
<td>3 hours</td>
</tr>
<tr>
<td>super</td>
<td>$10^{12}$</td>
<td>2 weeks</td>
<td>instant</td>
</tr>
</tbody>
</table>

$N = 1$ billion

Insertion Sort: Empirical Analysis

Data analysis. Plot # comparisons vs. input size on log-log scale.

Hypothesis. # comparisons grows quadratically with input size ~ $N^2/4$.

Insertion Sort: Observation

Observe and tabulate running time for various values of $N$.
- Data source: $N$ random numbers between 0 and 1.
- Machine: Apple G5 1.8GHz with 1.5GB memory running OS X.
- Timing: Skagen wristwatch.

<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>0.13 seconds</td>
</tr>
<tr>
<td>10,000</td>
<td>25 million</td>
<td>0.43 seconds</td>
</tr>
<tr>
<td>20,000</td>
<td>99 million</td>
<td>1.5 seconds</td>
</tr>
<tr>
<td>40,000</td>
<td>400 million</td>
<td>5.6 seconds</td>
</tr>
<tr>
<td>80,000</td>
<td>1600 million</td>
<td>23 seconds</td>
</tr>
</tbody>
</table>

Insertion Sort: Prediction and Verification

Experimental hypothesis. # comparisons ~ $N^2/4$.

Prediction. 400 million comparisons for $N = 40,000$.

Observations.

<table>
<thead>
<tr>
<th>$N$</th>
<th>Comparisons</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>40,000</td>
<td>401.3 million</td>
<td>5.595 sec</td>
</tr>
<tr>
<td>40,000</td>
<td>399.7 million</td>
<td>5.573 sec</td>
</tr>
<tr>
<td>40,000</td>
<td>401.6 million</td>
<td>5.648 sec</td>
</tr>
<tr>
<td>40,000</td>
<td>400.0 million</td>
<td>5.632 sec</td>
</tr>
</tbody>
</table>

Agrees.

Prediction. 10 billion comparisons for $N = 200,000$.

Observation.

<table>
<thead>
<tr>
<th>$N$</th>
<th>Comparisons</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>200,000</td>
<td>9.997 billion</td>
<td>145 seconds</td>
</tr>
</tbody>
</table>

Agrees.

Insertion Sort: Mathematical Analysis

Mathematical analysis.

<table>
<thead>
<tr>
<th>Analysis</th>
<th>Comparisons</th>
<th>Stddev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worst</td>
<td>$N^2/2$</td>
<td>-</td>
</tr>
<tr>
<td>Average</td>
<td>$N^2/4$</td>
<td>1.6 $N^{1/2}$</td>
</tr>
<tr>
<td>Best</td>
<td>$N$</td>
<td>-</td>
</tr>
</tbody>
</table>

Validation. Theory agrees with observations.

<table>
<thead>
<tr>
<th>$N$</th>
<th>Actual</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>40,000</td>
<td>401.3 million</td>
<td>400 million</td>
</tr>
<tr>
<td>200,000</td>
<td>9.997 billion</td>
<td>10.000 billion</td>
</tr>
</tbody>
</table>
Mergesort: Preliminary Hypothesis

Experimental hypothesis. Number of comparisons \( \approx 20N \).

Mergesort: Prediction and Verification

Experimental hypothesis. Number of comparisons \( \approx 20N \).

Prediction. 80 million comparisons for \( N = 4 \) million.

Observations.

<table>
<thead>
<tr>
<th>( N )</th>
<th>Comparisons</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 mill</td>
<td>82.7 million</td>
<td>3.13 sec</td>
</tr>
<tr>
<td>4 mill</td>
<td>82.7 million</td>
<td>3.25 sec</td>
</tr>
<tr>
<td>4 mill</td>
<td>82.7 million</td>
<td>3.13 sec</td>
</tr>
</tbody>
</table>

Prediction. 400 million comparisons for \( N = 20 \) million.

Observations.

<table>
<thead>
<tr>
<th>( N )</th>
<th>Comparisons</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>20 mill</td>
<td>460 million</td>
<td>17.5 sec</td>
</tr>
<tr>
<td>50 mill</td>
<td>1216 million</td>
<td>45.9 sec</td>
</tr>
</tbody>
</table>

Agrees. Not quite.